

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.



untuk Robotika:

Menuju Kecerdasan Terwujud



YAYASAN PRIMA AGUS TEKNIK



Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

AI untuk Robotika:

Menuju Kecerdasan Terwujud



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-634-7695-40-6 (PDF)



9

786347

695406

AI untuk Robotika : Menuju Kecerdasan Terwujud

Penulis :

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

ISBN : 978-634-7695-40-6 (PDF)

Editor :

Dr. Agus Wibowo, M.Kom, M.Si, MM.

Penyunting :

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :

Irdha Yuniarto, S.Ds., M.Kom

Penebit :

Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Anggota IKAPI No: 279 / ALB / JTE / 2023

Redaksi :

Jl. Majapahit no 605 Semarang

Telp. 08122925000

Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :

Universitas STEKOM

Jl. Majapahit no 605 Semarang

Telp. 08122925000

Fax. 024-6710144

Email : info@stekom.ac.id

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara
apapun tanpa ijin dari penulis

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya sehingga buku ini, berjudul "*AI untuk Robotika: Menuju Kecerdasan Terwujud*", dapat diselesaikan. Buku ini disusun sebagai buku ajar yang komprehensif dan terstruktur untuk memberikan landasan konseptual, teknik, dan praktis tentang penerapan kecerdasan buatan—terutama pembelajaran mendalam dan model-model dasar—dalam domain robotika modern.

Motivasi penyusunan buku ini berakar pada kebutuhan akademik dan industri akan sumber belajar yang mengintegrasikan teori fundamental robotika dengan perkembangan terkini dalam kecerdasan buatan. Perkembangan pesat dalam arsitektur jaringan neural, teknik fusi sensor, pemodelan bahasa besar, serta metode pembelajaran penguatan telah membuka peluang besar bagi realisasi kemampuan otonom dan adaptif pada robot. Di sisi lain, kompleksitas tantangan seperti Sim2Real, keselamatan sistem, dan skala data menuntut pendekatan lintas-disiplin yang sistematis. Buku ini hadir untuk menjembatani kebutuhan tersebut: menawarkan pengantar konseptual yang solid, pembahasan teknis yang mendalam, dan wawasan praktis yang relevan untuk penelitian maupun pengajaran.

Struktur buku disusun sedemikian rupa agar sesuai dengan alur pembelajaran dari dasar menuju aplikasi lanjut. Bab pertama menyajikan pengantar robotika serbaguna, termasuk komponen dasar, siklus kerja, serta konsep desain yang menjadi fondasi bagi bab-bab berikutnya. Bab kedua dan ketiga membahas aspek persepsi robot—mulai dari sensor dan pemrosesan citra hingga data tiga dimensi dan fusi multimodal—dengan penekanan pada metode berbasis *convolutional neural networks* (CNN) dan transformer yang kini banyak digunakan dalam tugas deteksi, segmentasi, dan pemahaman lingkungan.

Bab keempat mengeksplorasi peran model dasar (*foundation models*), khususnya model bahasa besar, dalam konteks robotika. Pembahasan meliputi proses pelatihan, penskalaan, integrasi bahasa sebagai jaringan penghubung fungsional, serta pendekatan kontrol ujung-ke-ujung dan pembelajaran dari demonstrasi. Isu keamanan dan evaluasi model turut dibahas untuk memberikan perspektif kritis terhadap penerapan model besar pada sistem fisik. Untuk mendukung praktik dan pengembangan, Bab lima mengulas peran simulasi dalam pelatihan robotika, komponen utama simulator, serta tantangan dan solusi pada transisi dari simulasi ke dunia nyata (Sim2Real). Bab enam sampai tujuh menguraikan masalah pemetaan, lokalisasi, navigasi, serta kerangka pembelajaran penguatan dan kontrol yang menjadi inti banyak aplikasi robot otonom.

Aplikasi nyata diarena kendaraan tanpa pengemudi, robot industri, dan robot humanoid dikaji pada Bab delapan hingga sepuluh. Setiap bab menyajikan tinjauan teknologi, arsitektur sistem, aspek keselamatan, serta prospek perkembangan industri dan riset. Bab sebelas menutup buku dengan pembahasan mengenai praktik robotika berbasis data, mencakup pengelolaan operasional, infrastruktur data dan pelatihan, serta tantangan pengumpulan data skala besar yang relevan bagi pengembangan model dasar untuk robotika.

Secara metodologis, buku ini mengombinasikan pendekatan konseptual, tinjauan pustaka terpilih, dan contoh penerapan teknis yang relevan. Penekanan diberikan pada prinsip-prinsip inti dan trade-off desain, lengkap dengan referensi yang dapat digunakan pembaca untuk pendalaman lebih lanjut. Materi dirancang agar berguna bagi mahasiswa pascasarjana dan sarjana tingkat akhir, peneliti yang berkecimpung di bidang robotika dan kecerdasan buatan, serta praktisi industri yang ingin memahami landasan ilmiah dan teknis di balik solusi robotika modern.

Penulis berharap buku ini tidak hanya menjadi sumber rujukan, tetapi juga memicu diskusi akademik dan inovasi praktis. Kami mendorong pembaca untuk menggunakan buku ini sebagai pijakan untuk eksperimen, pengembangan prototipe, serta penelitian lanjutan—dengan memperhatikan aspek keselamatan, etika, dan dampak sosial dari penerapan teknologi robotika cerdas.

Akhir kata, kami mengucapkan terima kasih kepada rekan-rekan sejawat yang telah memberikan masukan teknis dan substansial selama proses penyusunan, serta kepada keluarga dan institusi yang mendukung terealisasinya karya ini. Segala kekurangan dalam buku ini menjadi tanggung jawab penulis; kritik dan saran konstruktif sangat kami harapkan untuk perbaikan edisi selanjutnya.

Semangat & Selamat Membaca...!!!!

Semarang, Juli 2026

Penulis

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI.....	iii
BAB 1 PENGANTAR ROBOTIKA SERBAGUNA.....	1
1.1 Komponen Utama Dan Siklus Kerja Robot	3
1.2 Jenis Robot Umum	4
1.3 Konsep Umum Dalam Desain Robot	7
1.4 Pembelajaran Mendalam Untuk Robotika	9
1.5 Konsep Dasar AI, ML, Dan DL	12
1.6 Paradigma Pembelajaran Dan Penentuan Tujuan Robot	14
1.7 Menuju Kecerdasan Umum Yang Terwujud	15
1.8 Ringkasan.....	16
BAB 2 PERSEPSI ROBOT: SENSOR DAN PEMROSESAN CITRA	18
2.1 Sensor Dan Persepsi Robot.....	19
2.2 Masalah Dalam Persepsi.....	30
2.3 Gambaran Umum Jaringan Neural Konvolusional	35
2.4 CNN Untuk Persepsi.....	38
2.5 Resnet Dan U-Net Untuk Persepsi Robot	43
2.6 EfficientNet Dan Penskalaan CNN	45
2.7 YOLO Dan SSD Untuk Deteksi Objek.....	46
2.8 Evaluasi Kinerja Model Deteksi Objek	48
2.9 Transformer Untuk Persepsi	49
2.10 Ringkasan.....	60
BAB 3 PERSEPSI ROBOT: DATA 3D DAN FUSI SENSOR.....	62
3.1 Pemrosesan Data 3D	62
3.2 Persepsi Multimodal Dan Fusi Sensor	68
3.3 Ringkasan.....	79
BAB 4 MODEL DASAR DALAM ROBOTIKA	81
4.1 Model Bahasa Besar: Pelatihan, Penskalaan, Dan Evaluasi.....	81
4.2 Bahasa Sebagai Jaringan Penghubung Dalam Robotika	88
4.3 Kontrol Robot Ujung-Ke-Ujung.....	99
4.4 Pembelajaran Dari Demonstrasi Video	115
4.5 Keamanan AI Untuk Robotika.....	117
4.6 Ringkasan.....	119
BAB 5 SIMULASI ROBOT	121
5.1 Simulasi Dalam Pelatihan Robotika	121
5.2 Komponen Simulator Robot	122
5.3 Konsep Dalam Sim2Real	127
5.4 Menutup Kesenjangan Sim2Real Untuk RL	131

5.5	Pembelajaran Dari Simulasi	135
5.6	Ringkasan.....	148
BAB 6	PEMETAAN, LOKALISASI, DAN NAVIGASI	152
6.1	Metode Lokalisasi Dan Pemetaan Robot.....	152
6.2	Pengaturan Robot Seluler Khas	154
6.3	Pemetaan Berbasis Pembelajaran Mendalam	155
6.4	Lokalisasi Berbasis Pembelajaran Mendalam.....	161
6.5	Navigasi Dan Lokomosi Cerdas Pada Robot	166
6.6	Ringkasan.....	176
BAB 7	PEMBELAJARAN PENGUATAN DAN KONTROL	178
7.1	Dasar-Dasar Pembelajaran Penguatan	180
7.2	RL Tanpa Model Vs Berbasis Model.....	182
7.3	Pembelajaran Penguatan Offline.....	192
7.4	Aplikasi Dan Tantangan.....	193
7.5	Tantangan Dalam RL Untuk Robotika	199
7.6	Tren Yang Muncul Dalam RL Untuk Robotika	200
7.7	Masa Depan Pembelajaran Penguatan Dalam Robotika	201
7.8	Ringkasan.....	201
BAB 8	KENDARAAN TANPA PENGEMUDI	203
8.1	Industri Dan Model Bisnis Kendaraan Otonom	203
8.2	Arsitektur Sistem Kendaraan Otonom.....	204
8.3	Sistem Persepsi Pada Kendaraan Otonom	206
8.4	Prediksi Niat Pada Kendaraan Otonom	208
8.5	Perencanaan Jalur Pada Kendaraan Otonom	210
8.6	Keselamatan Dan Keandalan Kendaraan Otonom	211
8.7	Ringkasan.....	213
BAB 9	PEMBELAJARAN MENDALAM DALAM ROBOT INDUSTRI.....	215
9.1	Tugas Utama Robot Industri	215
9.2	Jenis Dan Platform Robot Industri	218
9.3	Tren Dan Pertumbuhan Industri Robotika.....	220
9.4	Desain Sistem Untuk Robot Pengambil Dan Penempatan	222
9.5	Aspek Keselamatan Dalam Robotika Industri Pembelajaran Mendalam	229
9.6	Ringkasan.....	229
BAB 10	ROBOTIKA HUMANOID.....	231
10.1	Alasan Pengembangan Robot Humanoid	231
10.2	Pendekatan Humanoid Dalam Robotika	231
10.3	Ekosistem Dan Pasar Robot Humanoid	232
10.4	Desain Dan Pengendalian Robot Humanoid	234
10.5	Prospek Robot Humanoid	240
10.6	Ringkasan.....	240
BAB 11	ROBOTIKA BERBASIS DATA DALAM PRAKTIK	242

11.1	Operasi Dan Keselamatan Robot.....	242
11.2	Infrastruktur Data Robotika.....	243
11.3	Infrastruktur Pelatihan Robot.....	244
11.4	Roda Penggerak Data Robot.....	245
11.5	Pengumpulan Data Robot Skala Besar	246
11.6	Masa Depan Robotika Berbasis Model Dasar	249
DAFTAR PUSTAKA		250

BAB 1

PENGANTAR ROBOTIKA SERBAGUNA

Manusia telah bermimpi membuat mesin cerdas yang berperilaku dan berpikir seperti manusia selama berabad-abad. Dari revolusi industri hingga "I, Robot" karya Asimov dan humanoid pertama di dunia yang dibangun seabad yang lalu, robot telah lama mengisi imajinasi kolektif kita. Robot telah melampaui sekadar khayalan fiksi ilmiah menjadi kenyataan di masa kini, dengan kemampuan yang semakin meningkat. Apa yang menyebabkan transformasi ini? Kemajuan dalam kecerdasan buatan telah mengganggu berbagai industri dalam dekade terakhir dengan membuka kemampuan baru melalui pembelajaran mesin. Robotika telah keluar dari aplikasi yang terbatas dan sempit di lingkungan industri dan studi yang terstruktur dengan baik dan sekarang terintegrasi ke dalam kehidupan sehari-hari kita. Robot mendorong persaingan melalui otomatisasi dalam manufaktur skala besar, eksplorasi ruang angkasa dan bawah laut, pertanian, dan perawatan kesehatan, di antara industri lainnya. Di masa depan, kita berharap melihat robot menangani tugas manipulasi halus di industri, melakukan pekerjaan rumah tangga di rumah, dan beroperasi secara otonom di jalan umum dan di rumah sakit.

Seiring dengan peningkatan kemampuan dalam beberapa dekade terakhir, biayanya pun menurun. Selama 30 tahun terakhir, biaya rata-rata robotika telah turun setengahnya dalam istilah yang disesuaikan dengan Indeks Harga Konsumen (CPI) (setelah memperhitungkan inflasi), menurut laporan McKinsey & Company baru-baru ini. Biaya bahkan turun lebih jauh dibandingkan dengan kemampuannya karena hukum Moore, meluasnya penggunaan GPU, dan penurunan biaya namun peningkatan kapasitas baterai dan komputer onboard. Adopsi robot secara luas didorong oleh peningkatan ekspansi ekonomi, meningkatnya biaya tenaga kerja manusia, menurunnya biaya robot, dan peningkatan kemampuannya.

Namun, seiring dengan perpindahan robot dari laboratorium studi dan lingkungan industri yang terbatas ke dunia nyata, mereka menghadapi tantangan baru. Pertimbangkan, misalnya, robot pembersih rumah tangga. Robot ini harus melakukan banyak tugas, termasuk membersihkan lantai, membersihkan debu di meja dapur, dan mencuci piring. Untuk mencapai hal ini, robot harus tahu cara:

- Melintasi lingkungan dalam ruangan sambil merasakan dan menghindari rintangan.
- Menangani benda-benda rapuh, lunak, dan terkadang berat dengan bentuk tidak beraturan, termasuk benda-benda yang mungkin belum pernah ditemui sebelumnya.
- Memanipulasi pemandangan yang mungkin belum pernah dialami sebelumnya, karena setiap rumah terlihat berbeda, memiliki pencahayaan, tata letak, dan sebagainya yang berbeda.
- Bernalar tentang interaksi dengan benda-benda rumah tangga, manusia, dan hewan peliharaan serta konfigurasi ruang di masa lalu.

Tantangannya adalah merancang pendekatan yang dapat beradaptasi dengan perubahan di dunia nyata dan berbagai situasi yang akan dihadapinya. Sebelum munculnya pembelajaran mendalam, tumpukan perangkat lunak untuk menyelesaikan tugas-tugas ini akan ditulis sebagai mesin keadaan dengan primitif gerakan "yang sudah terpasang" yang menyerupai kontrol tradisional untuk tugas tertentu tersebut. Pendekatan ini tidak dapat menangani situasi yang belum pernah dilihat sebelumnya dengan baik, tidak dapat diskalakan, dan membatasi kegunaan robot yang diprogram secara kaku. Selain itu, bahkan untuk pengambilan dan penempatan sederhana, menerjemahkan berbagai intuisi manusia ke dalam kontrol transisi adalah hal yang menantang, jika tidak hampir mustahil.

Luasnya cakupan dan universalitas persepsi, penalaran, dan kontrol yang dibutuhkan untuk robotika tujuan umum paling baik ditangani oleh aproksimator fungsi universal: jaringan saraf. Alih-alih membuat kode sistem kontrol secara manual, kita menggunakan pembelajaran mesin untuk memungkinkan robot mempelajari fitur-fitur yang relevan dan hubungannya dari data pelatihan.

Pendekatan ini telah menghasilkan hasil di banyak bidang lain. Keberhasilan ChatGPT dan model bahasa secara umum baru-baru ini telah menghasilkan perusahaan AI bernilai miliaran dolar. Studi di balik produk-produk ini menunjukkan bahwa penskalaan data, komputasi, dan model secara terprogram mengarah pada kemampuan umum di bidang bahasa/visi/audio. Dengan mengubah data menjadi token, kesamaan di antara mereka dapat diidentifikasi, dan kesamaan tersebut dapat ditransfer ke domain lain. Berbagai kemampuan telah dibuka sebagai hasilnya:

1. Pembuatan gambar dan video kustom dan realistis dan sesuai permintaan, yang menjanjikan transformasi industri film, pemasaran, dan periklanan.
2. Generasi bahasa, termasuk terjemahan, penulisan kreatif, penulisan iklan, pembuatan kode, dan transkripsi.
3. Generasi audio. Membuat podcast sesuai permintaan dan musik.
4. Penalaran multimodal. Memecahkan masalah matematika, masalah sains tingkat pascasarjana, dan masalah hukum dan medis.

Kemampuan ini telah menyebabkan munculnya AI spring, dengan perusahaan AI generatif mengumpulkan pendanaan sebesar Rp25,9 triliun pada tahun 2023 saja. Tren ini, serta terobosan terbaru dalam kecerdasan spasial dan model dasar robot, menunjukkan bahwa robotika juga dapat ditingkatkan dan dipecahkan sebagai masalah AI. Di era di mana kita memiliki kecerdasan generik, bahan-bahan yang dibutuhkan untuk membangun robot yang berguna secara umum sebagian besar sudah ada.

Keberhasilan AI dan janji kemampuan yang muncul telah menyebabkan peningkatan pesat dalam robotika yang didukung pembelajaran mesin dan meningkatnya permintaan akan talenta di pasar tenaga kerja. Menurut studi dari Mordor Intelligence, Pasar Robotika Global bernilai Rp27,73 triliun pada tahun 2020 dan diperkirakan akan mencapai Rp74,1 triliun pada tahun 2026, dengan mencatat Tingkat Pertumbuhan Tahunan Majemuk (CAGR) sebesar 17,45 persen. Meskipun peningkatan ini sebagian besar disebabkan oleh peningkatan pesat robot industri, pasar robotika AI diperkirakan akan tumbuh dengan CAGR sebesar 38,6 persen dari

6,9 triliun rupiah pada tahun 2021 menjadi 35,5 triliun rupiah pada tahun 2026 menurut laporan ini. Untuk memanfaatkan peluang ini, perusahaan teknologi besar, perusahaan rintisan, dan laboratorium studi semakin mencari insinyur AI dan robotika yang berkualitas untuk divisi R&D robotika, R&D mobil otonom, dan manufaktur mereka. Untuk mulai berkontribusi pada upaya pembelajaran mesin perusahaan-perusahaan ini, Anda perlu memahami:

- Bagaimana merumuskan masalah robotika dalam konteks pembelajaran mesin
- Metode pembelajaran mesin mana yang dapat digunakan untuk menyelesaikan berbagai masalah dalam robotika dan pertukaran di antara metode-metode tersebut
- Pada titik mana dalam tumpukan robotika Anda harus menggunakan pembelajaran mesin

Paradoks Moravec adalah salah satu tantangan utama pembelajaran mesin untuk robotika. Seperti yang dijelaskan Steven Pinker pada tahun 1994, “Pelajaran utama dari 35 tahun studi AI adalah bahwa masalah yang sulit menjadi mudah dan masalah yang mudah menjadi sulit.” Kecerdasan buatan, terutama jaringan saraf, adalah bentuk kecerdasan yang cukup berbeda dari otak manusia dan, karenanya, memiliki kekuatan yang berbeda.

Hal-hal yang mungkin tampak sangat sulit bagi manusia—seperti pencitraan generatif, kompresi bahasa, dan proyeksi sekuensial seperti analitik saham—cukup mudah bagi AI. Namun, tugas-tugas yang bahkan dapat dilakukan dengan mudah oleh anak berusia empat tahun melalui penalaran sensorimotor dan perseptual—seperti berjalan dan mengangkat pensil—jauh lebih sulit. Dalam sejarah inovasi ilmiah, semua masalah tampak sulit sebelum dipecahkan, dan para penulis buku ini optimis bahwa pemetaan dan perbaikan tantangan nyata dalam pembelajaran robotika dapat memberikan dampak pada kemajuan kecerdasan fisik.

Bab ini dimulai dengan mendefinisikan dua premis buku ini: robot dan AI. Kami menyajikan motivasi umum mengapa seseorang perlu menggunakan AI untuk robotika dan tantangan dalam melakukannya. Bab-bab selanjutnya memetakan area-area kunci dalam pengembangan AI untuk robotika, seperti persepsi pembelajaran mesin, bahasa dalam robotika, pelatihan robot dalam simulasi, dan membangun infrastruktur untuk pembelajaran robot yang terukur. Kemudian kami menjelaskan bagaimana cara praktis merancang dan mengimplementasikan prinsip-prinsip ini dalam beberapa aplikasi pilihan—mobil otonom, robot industri, dan humanoid.

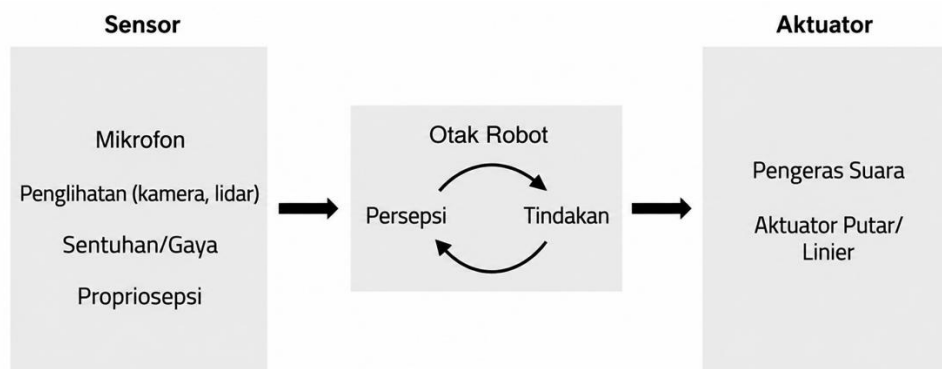
Mari kita mulai!

1.1 KOMPONEN UTAMA DAN SIKLUS KERJA ROBOT

Robot didefinisikan sebagai mesin interaktif yang menerima model dunia dan menghasilkan tindakan. Tidak seperti banyak aplikasi pembelajaran mesin, robot dicirikan oleh agensi dan umpan balik loop tertutup di dunia nyata atau simulasi. Robot biasanya merasakan dunia melalui serangkaian sensornya, termasuk kamera, LiDAR, unit pengukuran inersia (IMU), detektor suara, dan/atau radar, sebagai beberapa contoh. Otak robot, yang biasanya dijalankan pada mikrokontroler onboard, memproses input dari sensor dan menghitung

tindakan, yang dikirim sebagai sinyal ke aktuator robot. Aktuator ini dapat berupa motor arus searah yang menyebabkan persendiannya bergerak atau material yang lentur dalam kasus robotika lunak.

Ruang aksi robot ditentukan oleh aplikasinya. Misalnya, ruang aksi untuk mobil otonom dan robot navigasi adalah percepatan dan sudut kemudi. Lengan robot dapat dirancang sebagai posisi atau kecepatan persendian pada lengan. Ruang aksi tambahan untuk robot yang berinteraksi dengan manusia dapat mencakup bahasa alami melalui antarmuka obrolan, gerakan, dan ekspresi wajah. Gambar 1.1. menunjukkan diagram tingkat tinggi dari sistem robot.



Gambar 1.1. Komponen utama robot: sensor (mikrofon, sistem penglihatan seperti kamera dan LiDAR, sensor sentuh/gaya, dan proprioepsi), yang digunakan untuk merasakan lingkungan; otak robot tempat data persepsi diproses dalam siklus persepsi-aksi yang berkelanjutan; dan aktuator (speaker dan aktuator putar/linier), yang menjalankan tindakan robot. Siklus persepsi-aksi ini sangat penting untuk pembelajaran robot.

1.2 JENIS ROBOT UMUM

Robot hadir dalam berbagai ukuran dan bentuk. Kita dapat memisahnya berdasarkan vertikal (atau sektor tempat mereka ditempatkan), seperti yang ditunjukkan pada Tabel 1.1 (disusun dengan bantuan AI).

Tabel 1.1. Jenis Robot berdasarkan Vertikal

Jenis	Definisi	Contoh Robot industri
1. Robot industri	Robot yang digunakan dalam proses manufaktur seperti perakitan, pengecatan, pengelasan, dan pengemasan	Lengan robot, robot gantry, robot layanan
2. Robot layanan	Robot yang melakukan tugas untuk membantu manusia di berbagai lingkungan seperti rumah sakit, hotel, dan restoran	Robot pengiriman, robot pembersih, robot telepresence

3. Robot medis	Robot yang digunakan di lingkungan perawatan kesehatan untuk membantu operasi, diagnosis, dan perawatan pasien	Sistem bedah da Vinci, robot rehabilitasi, sistem otomatisasi farmasi
4. Robot militer dan pertahanan	Robot yang dirancang untuk digunakan dalam aplikasi militer, seperti pengintaian, pengawasan, dan dukungan tempur	Kendaraan udara tak berawak (UAV), kendaraan darat tak berawak (UGV), robot penjinak bom
5. Robot pertanian	Robot yang digunakan dalam pertanian untuk mengotomatiskan tugas-tugas seperti penanaman, panen, dan pemantauan kesehatan tanaman	Traktor otonom, drone untuk pemantauan tanaman, robot pemetik buah
6. Robot rumah tangga	Robot yang dirancang untuk digunakan di rumah untuk membantu pekerjaan rumah tangga dan tugas-tugas lainnya	Robot penyedot debu Roomba, robot pemotong rumput, robot asisten pribadi seperti Jibo
7. Robot pendidikan	Robot yang digunakan dalam lingkungan pendidikan untuk membantu mengajarkan berbagai mata pelajaran atau keterampilan	LEGO Mindstorms, robot sosial seperti Pepper, Sphero
8. Robot studi	Robot yang digunakan dalam studi ilmiah, termasuk menjelajahi lingkungan terpencil atau berbahaya dan mengembangkan teknologi robot baru	Robot bawah air, Mars Rover, robot humanoid seperti ASIMO
9. Robot hiburan	Robot yang dirancang untuk hiburan atau teman	Hewan peliharaan robot seperti Aibo, mainan interaktif seperti Furby, robot yang digunakan di taman hiburan atau film
10. Robot kawanan	Robot yang bekerja bersama dalam kelompok besar, mengoordinasikan tindakan mereka untuk menyelesaikan tugas dengan lebih efisien.	Robot kawanan digunakan dalam studi, pertanian, pencarian dan penyelamatan, serta pemantauan lingkungan.

Cara kedua untuk mengelompokkan robot adalah berdasarkan sifat perwujudannya, seperti yang ditunjukkan pada Tabel 1.2 (disusun dengan bantuan AI).

Tabel 1.2. Jenis Robot berdasarkan Perwujudan

Nomor seri	Perwujudan	Penjelasan	Contoh
1.	Robot beroda	Robot yang menggunakan roda untuk bergerak, sering digunakan di permukaan datar.	Roomba, TurtleBot, mobil otonom seperti Waymo dan Cruise
2.	Robot beroda rantai	Robot yang menggunakan trek untuk bergerak, memberikan traksi dan stabilitas yang lebih	Robot penjelajah Mars, robot penjinak bom

		besar di medan yang kasar atau tidak rata.	
3.	Robot berkaki	Robot yang menggunakan kaki untuk bergerak, menavigasi lingkungan yang kompleks seperti tangga dan medan yang tidak rata.	Spot dari Boston Dynamics, ASIMO
4.	Robot terbang	Robot yang mampu terbang, biasanya menggunakan rotor atau sayap, untuk pengawasan udara, inspeksi, dan fotografi.	Drone quadcopter, UAV sayap tetap
5.	Robot bawah air	Robot yang dirancang untuk operasi bawah air, digunakan untuk tugas eksplorasi, inspeksi, dan pemantauan.	AUV Bluefin Robotics, SeaBED
6.	Robot ular	Robot dengan badan yang panjang dan fleksibel, untuk bergerak melalui ruang sempit dan menavigasi di sekitar rintangan.	Robot ular dari Laboratorium Biorobotika CMU, robot lengan ular dari OC Robotics
7.	Lengan robot	Robot yang terdiri dari serangkaian sendi dan penghubung, menyerupai lengan manusia, digunakan dalam lingkungan industri.	Lengan robot KUKA, Fanuc, dan ABB
8.	Robot humanoid	Robot dengan bentuk seperti manusia, digunakan dalam studi, hiburan, dan aplikasi layanan.	Pepper dari SoftBank Robotics, Sophia dari Hanson Robotics
9.	Robot lunak	Robot yang meniru mekanisme pergerakan materi yang dapat berubah bentuk seperti cairan, gel, dan elastomer untuk fleksibilitas yang lebih besar. Umumnya digunakan dalam aplikasi biomedis seperti alat lunak untuk pembedahan, perangkat rehabilitasi, dan pengiriman obat.	Robot lunak dari Wyss Institute Harvard, Octobot

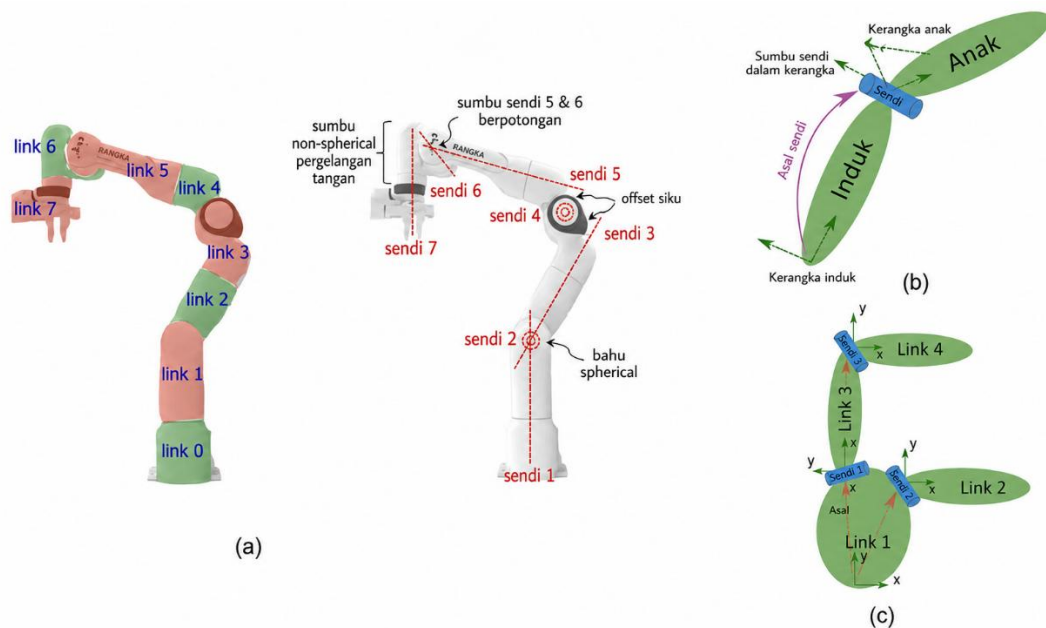
Meskipun terdapat beragam jenis robot, mereka memiliki banyak kesamaan yang dapat digunakan untuk membangun kerangka kerja dan ilmu robotika yang umum, yang dapat diperluas dengan modifikasi agar sesuai dengan kondisi penerapan robot.

1.3 KONSEP UMUM DALAM DESAIN ROBOT

Bagian ini menjelaskan beberapa konsep umum yang digunakan dalam desain robot.

Manipulator Robot

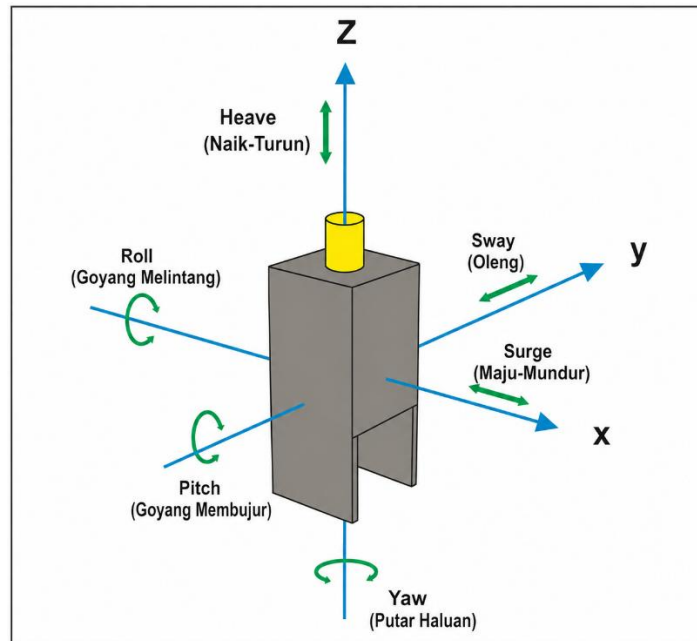
Salah satu bentuk robot yang umum adalah lengan/manipulator robot. Robot ini dapat ditemukan dalam bentuk lengan robot industri, robot bantu, dan robot medis, yang digunakan untuk menyelesaikan berbagai tugas di lingkungannya. Lengan robot adalah serangkaian sendi dan penghubung, seperti yang digambarkan pada Gambar 1.2. Di sini, penghubung menghubungkan sendi dan merupakan benda kaku. Sendi menghubungkan dua atau lebih penghubung dan memungkinkan gerakan relatif antar penghubung.



Gambar 1.2. Sambungan dan sendi pada lengan/manipulator robot. (a) Lengan robot tujuh sambungan yang diberi label dengan sendi-sendinya. (b) Representasi hubungan induk-anak antara sambungan melalui sebuah sendi. (c) Visualisasi sambungan yang terhubung melalui sendi dengan sumbu rotasi.

Derajat Kebebasan

Derajat kebebasan (DoF) adalah ukuran dari berbagai komponen gerakan yang dapat dilakukan oleh robot. Misalnya, benda kaku di ruang angkasa memiliki enam DoF: gerakan translasi sepanjang sumbu X, Y, dan Z, serta gerakan rotasi di sekitar sumbu X, Y, dan Z, seperti yang ditunjukkan pada Gambar 1.3. Setiap tautan tambahan menambah derajat kebebasan robot dan setiap sendi mengurangi derajat kebebasan dengan membatasi gerakan dalam satu atau lebih arah. Setiap derajat kebebasan dapat dimodelkan sebagai variabel independen dan terbatas yang perlu diprediksi oleh otak robot sebagai target tindakan untuk mengendalikan robot.



Gambar 1.3. Setiap benda kaku memiliki enam derajat kebebasan (DoF): tiga translasi (gerak maju, gerakan menyamping, gerakan naik turun) dan tiga rotasi (gerak berguling, gerakan menukik, gerakan berputar)

Efektor Ujung dan Ruang Kerja

Efektor ujung adalah perangkat yang terpasang di ujung lengan. Gripper, atau tangan yang cekatan, adalah jenis efektor ujung yang paling umum dan merupakan faktor bentuk yang dapat digeneralisasikan ke berbagai macam tugas. Terkadang robot juga memiliki efektor ujung khusus tugas seperti penyapu, pemeras, dan sebagainya. Gabungan ruang tiga dimensi yang ditempati oleh robot didefinisikan sebagai ruang kerjanya. Himpunan bagian dari ruang ini, yaitu ruang yang dapat dijangkau oleh ujung efektor, didefinisikan sebagai ruang jangkauan robot. Ruang jangkauan dan batasan ruang kerja mendorong pertimbangan desain robot karena keduanya menentukan kegunaan robot dan kemampuannya untuk memanipulasi objek dalam penempatannya di ruang angkasa.

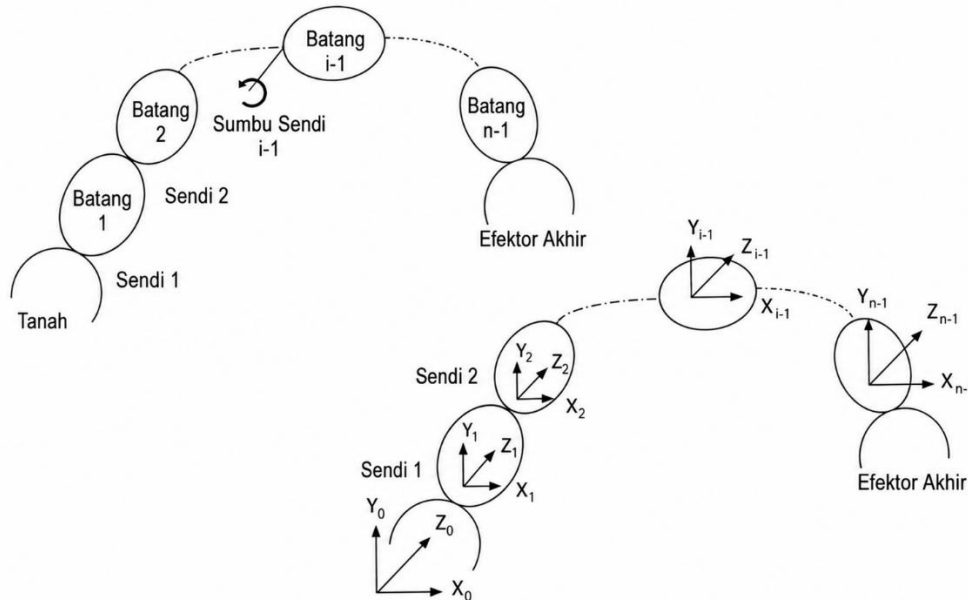
Kinematika

Konsep kunci dalam kontrol robot adalah kinematika robot. Secara etimologis, istilah ini merujuk pada studi tentang gerakan suatu benda atau sistem benda. Dalam hal ini, yaitu gerakan sendi dari sendi dan sambungan robot. Berkaitan dengan kontrol robot, dua jenis kinematika yang relevan adalah:

1. **Kinematika maju:** Di sini, dengan sudut sendi masukan yang diberikan, kita menentukan posisi dan orientasi ujung efektor ketika semua parameter sendi lainnya diketahui dalam konstelasi sambungan dan sendi.
2. **Kinematika invers:** Di sini, dengan diberikan posisi atau kecepatan spesifik yang ingin dicapai oleh ujung lengan robot, kita menghitung gerakan/orientasi yang diperlukan dari sendi-sendi sebelumnya untuk mencapai gerakan tersebut.

Urutan tautan dalam tubuh fisik robot, sifat-sifatnya (misalnya, massa, momen inersia, panjang), dan sifat-sifat sendi (misalnya, batasan yang dikenakan, torsi sendi) menentukan

sistem/rantai kinematik yang mewakili robot. Gambar 1.4 menunjukkan penggambaran rantai kinematik dari lengan robot tipikal. Gambar 1.5 mewakili transformasi antara tautan i dan tautan $i + 1$.



Gambar 1.4. Rantai kinematik lengan robot yang menunjukkan beberapa tautan yang dihubungkan oleh sendi, dengan sumbu sendi dan kerangka koordinat.

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & 0 \\ 0 & \sin \alpha & 1 \end{bmatrix} \begin{bmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{bmatrix}$$

α, β, γ adalah *yaw, pitch, dan roll* dari *link i+1* mengenai *link i*

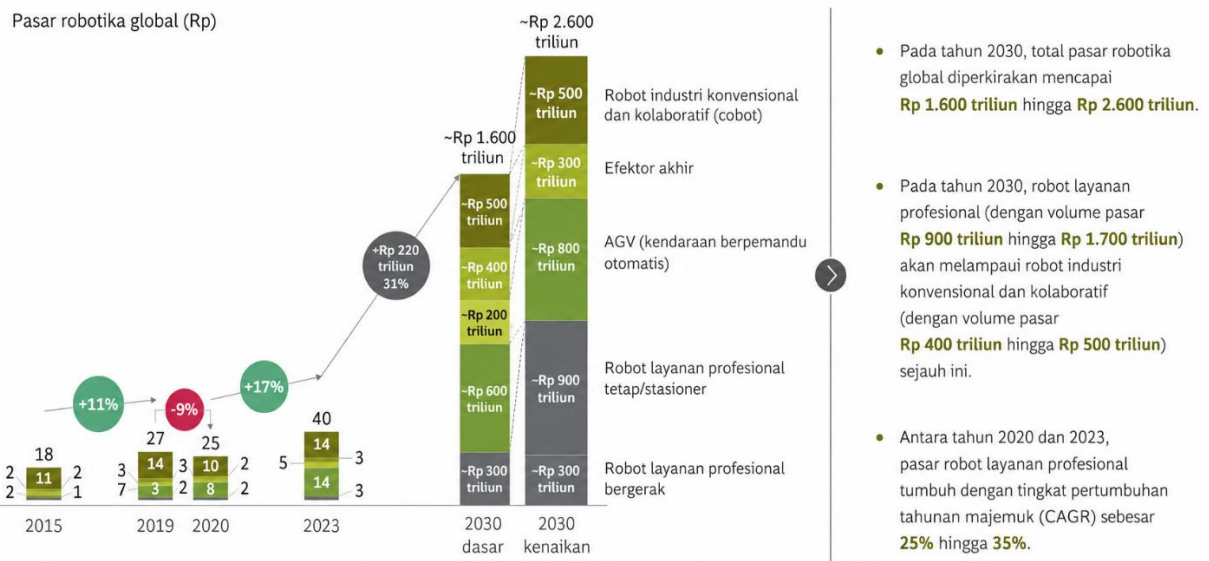
$$x_i = R x_{i+1}$$

$$x_{i+1} = R^{-1} x_i$$

Gambar 1.5. Matriks transformasi antara tautan i dan tautan $i+1$ menggunakan sudut yaw, pitch, dan roll α, β , dan γ , masing-masing.

1.4 PEMBELAJARAN MENDALAM UNTUK ROBOTIKA

Dalam dekade terakhir, beberapa perusahaan robotika telah berkembang dan gulung tikar di pasar karena berbagai alasan. Alasan-alasan ini berkisar dari kesesuaian produk dengan pasar hingga masalah pendapatan/pembiayaan dan teknologi. Terlepas dari kegagalan ini, prospek robotika untuk tahun 2030 tetap positif, sebagian karena sebagian besar terobosan dalam studi pembelajaran mendalam terjadi pada dekade ini dan belum diproduksi/diterapkan pada robot.



Gambar 1.6. Prospek pasar robotika untuk tahun 2030, memproyeksikan total volume pasar sebesar Rp1.600 triliun hingga Rp2.600 triliun, dari BCG.

Memperbaiki kegagalan robot klasik generasi terakhir dan pemikiran ulang mendasar tentang paradigma pembelajaran robot adalah kunci untuk menghadirkan robot yang lebih kuat dan dapat digeneralisasikan ke pasar. Pembelajaran mendalam adalah prospek yang menjanjikan menuju tujuan tersebut, seperti yang ditunjukkan pada Gambar 1.6. Beberapa manfaat yang diberikan oleh pembelajaran mendalam meliputi:

- **Generalisasi:** Jaringan saraf dalam dapat mempelajari fungsi nonlinier dengan parameter yang cukup dalam model dan metode pelatihan yang kuat untuk menghindari overfitting. Fungsi-fungsi ini tidak mungkin dimodelkan dengan rekayasa manual. Ini berguna untuk menangani keragaman di dunia nyata, seperti menangani berbagai objek dan adegan, mempelajari dinamika invers, dan perencanaan dalam berbagai situasi.
- **Pembelajaran fitur:** Propagasi balik dan fungsi kerugian yang dibangun dengan cermat memungkinkan jaringan saraf dalam untuk mempelajari apa yang penting dari data, tanpa perlu secara eksplisit memodelkan representasi atau fitur rekayasa. Ini juga memungkinkan jaringan untuk mempelajari beberapa representasi data input serupa berdasarkan persyaratan aplikasi. Ini berarti mempelajari distribusi yang benar untuk menghasilkan tindakan untuk berbagai tugas dengan satu jaringan.
- **Paralelisme:** Robotika dunia nyata membutuhkan respons dengan kecepatan inferensi yang sangat cepat, sekitar 10Hz atau lebih. Meskipun metode klasik yang lebih sederhana lebih cepat daripada jaringan saraf, ketika ruang keputusan menjadi lebih kompleks, metode berbasis pencarian/grafik menjadi tidak efisien dari perspektif kecepatan dan tidak ramah terhadap paralelisasi. Deep learning memungkinkan paralelisasi masif pada akselerator perangkat keras seperti *Tensor Processing Units* (TPUs) dan *Graphics Processing Units* (GPUs), yang memungkinkan jutaan perkalian

matriks per detik secara optimal. Optimasi arsitektur jaringan seperti YOLO dan EfficientNet, yang dibahas dalam Bab 2, memberikan kompromi antara akurasi dan kecepatan berdasarkan aplikasi.

Untuk menerapkan deep learning untuk robotika, sejumlah paradigma infrastruktur sangat penting:

- **Komputasi:** Peningkatan kinerja dari penerapan model AI yang semakin besar didukung oleh inovasi dalam arsitektur komputasi melalui paralelisasi masif pada TPU dan GPU. Meskipun teori deep learning telah ada sejak tahun 1950-an, lonjakan studi dan aplikasi saat ini dipicu oleh ketersediaan dan penurunan biaya GPU dalam dekade terakhir. Selain itu, kemampuan komputasi onboard robot telah meningkat berkat platform seperti NVIDIA Jetson, yang memungkinkan inferensi real-time yang cepat pada robot mobile. Pembaruan over-the-air memungkinkan penyebaran versi perangkat lunak yang lebih baru untuk robot di lapangan, memungkinkan iterasi perangkat lunak robot terjadi dengan kecepatan yang sama dengan yang dilakukan oleh perusahaan perangkat lunak murni.
- **Data:** Deep learning terkenal sangat membutuhkan data dan penggunaannya dalam robotika membutuhkan pengumpulan data. Contoh yang baik adalah banyaknya kendaraan Tesla yang dilengkapi sensor di jalan yang mengumpulkan data tentang berbagai skenario mengemudi yang sangat tidak mungkin. Skalabilitas juga menimbulkan pertanyaan terkait arsitektur dan metode pelatihan terbaik. Contoh kedua dari gangguan akibat dataset besar adalah dalam studi bahasa visual di mana dataset besar seperti ImageNet, LAION, dan Internet telah memungkinkan pengembangan jaringan saraf yang sangat mumpuni seperti ChatGPT, ResNets, dan Stable Diffusion. Hal ini dibahas lebih detail di Bab 2.
- **Pelabelan:** Pembelajaran terawasi dan pembelajaran terawasi lemah termasuk di antara jenis deep learning yang paling berkinerja tinggi. Pembelajaran terawasi, khususnya untuk data gambar, dimungkinkan oleh alur kerja pelabelan yang sangat efisien yang menghasilkan umpan balik manusia dan munculnya perusahaan pelabelan seperti Scale AI. Pengawasan lemah dari teks seperti pasangan bahasa-gambar yang diekstrak dari Internet juga mengumpulkan kumpulan data skala sangat besar yang melatih jaringan saraf.
- **Simulasi:** Ketersediaan luas mesin simulasi, seperti Gazebo, PyBullet, dan MuJoCo, meniru fisika di dunia nyata.

Mereka membuka fungsionalitas baru untuk robot. Misalnya, simulasi memungkinkan pemodelan dan penanganan adegan darurat/kritis keselamatan yang sangat tidak mungkin terjadi di dunia nyata. Pembelajaran dari data simulasi juga menghilangkan kendala yang dikenakan oleh kapasitas robot.

Menggandakan data Anda, jika Anda hanya mengandalkan robot nyata, berarti menggandakan jam kerja robot dan kapasitas robot, yang mahal dari perspektif perangkat keras dan waktu, tetapi membuat salinan robot simulasi untuk melakukan hal yang sama cukup murah. Selain itu, studi dan pengembangan (R&D) memerlukan pengembangan dan

evaluasi berulang dan akan mendapat manfaat dari siklus umpan balik yang lebih cepat yang dapat diberikan oleh simulasi.

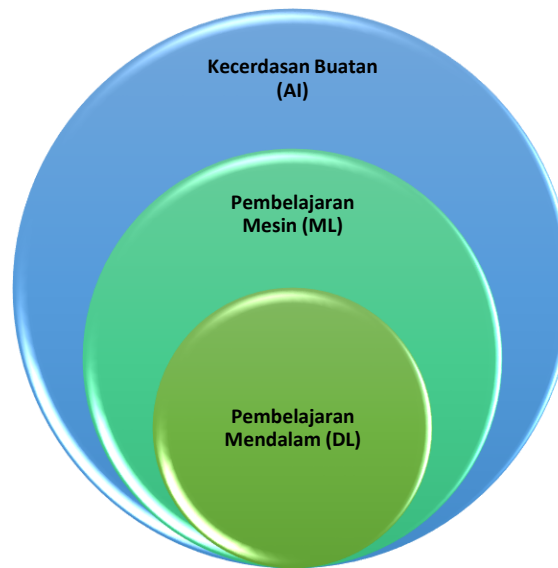
1.5 KONSEP DASAR AI, ML, DAN DL

Dalam konteks buku ini, kita cenderung menggunakan kecerdasan buatan, pembelajaran mesin, dan pembelajaran mendalam secara bergantian untuk merujuk pada metode berbasis data. Namun, secara tegas, istilah-istilah tersebut memiliki beberapa perbedaan, yang dibahas di bagian ini.

Kecerdasan Buatan (AI) adalah istilah umum untuk menggambarkan teknik komputasi yang mampu melakukan tugas dengan kecerdasan setara manusia. Tugas-tugas ini meliputi pemecahan masalah, pemahaman bahasa alami, pengenalan pola, dan pengambilan keputusan berdasarkan pembelajaran dari pelatihan model. AI adalah bidang umum yang terdiri dari ML dan DL, seperti yang ditunjukkan pada Gambar 1.7. Kecerdasan buatan saat ini dibedakan lebih lanjut menjadi Kecerdasan Buatan Umum (AGI), yang menggambarkan sistem yang memiliki berbagai kemampuan kecerdasan yang berguna dalam kehidupan sehari-hari, dan Kecerdasan Buatan Super (ASI), yang menggambarkan kecerdasan yang jauh melampaui kemampuan manusia.

Pembelajaran Mesin (ML) adalah subbidang AI yang berfokus pada pembelajaran dari data. Alih-alih diprogram secara eksplisit untuk melakukan suatu tugas, model ML meningkatkan kinerjanya melalui data yang digunakan untuk melatihnya. ML terdiri dari pembelajaran terawasi (data berlabel), pembelajaran tak terawasi (data tak berlabel), dan pembelajaran penguatan.

Pembelajaran mendalam (DL) adalah subset dari ML yang menggunakan jaringan saraf yang sering memiliki banyak lapisan (karena itu disebut mendalam) dan menggunakan data skala besar untuk melakukan tugas-tugas kompleks. Jaringan ini membantu memahami gambar dan bahasa dan dapat mencakup metode yang lebih canggih seperti transformer, GPT, dan sebagainya. Metode DL sering memiliki lebih banyak lapisan dan parameter serta membutuhkan lebih banyak data dan sumber daya komputasi daripada metode ML yang "dangkal".



Gambar 1.7. Kecerdasan buatan (AI) adalah konsep menyeluruh yang mencakup pembelajaran mesin (ML) dan subsetnya, pembelajaran mendalam (DL). Terinspirasi oleh .

Dalam pembelajaran mendalam terdapat empat jenis sistem pembelajaran utama: pembelajaran terawasi, pembelajaran tak terawasi, pembelajaran terawasi lemah, dan pembelajaran penguatan. Dalam beberapa tahun terakhir, batasan antara ketiga metode pembelajaran ini telah kabur, karena kombinasi atau ide dari ketiganya dapat dimanfaatkan dalam satu sistem.

1. Pembelajaran terawasi menggunakan dataset dengan label kebenaran dasar yang harus digunakan sistem untuk memprediksi label data yang belum pernah dilihat sebelumnya. Ini digunakan untuk klasifikasi, di mana output biasanya terdiri dari kelas diskrit, atau untuk regresi, di mana output yang diprediksi adalah bilangan riil.
2. Metode pembelajaran tak terawasi mencoba mempelajari representasi data yang berguna tanpa label. Contoh metode pembelajaran tak terawasi meliputi pengelompokan, analisis komponen utama, model campuran Gaussian, auto-encoder, dan sebagainya. Model pembelajaran tanpa pengawasan biasanya digunakan untuk pengelompokan, asosiasi, dan pengurangan dimensi.
3. Metode pembelajaran yang diawasi secara lemah menggunakan label yang bising dalam pengaturan pembelajaran yang diawasi. Metode ini digunakan dalam kasus di mana dataset mahal untuk diberi label dan penggabungan dataset besar dengan label lemah lebih layak daripada dataset yang lebih kecil dengan label yang bersih. Pembelajaran yang diawasi secara lemah digunakan untuk melatih model yang sangat kuat, termasuk CLIP, DALL-E, dan sebagainya, menggunakan data skala internet.
4. Pembelajaran penguatan memotivasi agen untuk mempelajari kebijakan yang memaksimalkan fungsi hadiah melalui pemrosesan urutan pasangan keadaan-tindakan, mengamati hadiah yang dicapai, dan menyesuaikan prediksi hingga secara akurat memprediksi jalur optimal, atau kebijakan, untuk agen tersebut. Pembelajaran

penguatan menyediakan kerangka kerja bagi robot untuk belajar secara otonom melalui interaksi coba-coba dan terus meningkatkan diri dengan umpan balik. Istilah-istilah ini bersifat menyeluruh dan mewakili seluruh bidang AI. Bagian selanjutnya membahas kerangka kerja yang secara khusus digunakan dalam pembelajaran robot.

1.6 PARADIGMA PEMBELAJARAN DAN PENENTUAN TUJUAN ROBOT

Mempelajari suatu tugas dari perspektif robotika dapat digambarkan sebagai menghasilkan distribusi tindakan berdasarkan model dunia masukan tertentu. Seperti yang disebutkan, cara utama robotika saat ini berbeda dari masa lalu adalah kita sekarang dapat merancang robot serbaguna dan generalis yang dapat melakukan berbagai tugas. Generalitas, dari perspektif pembelajaran mendalam, dapat dibingkai dalam tiga pengaturan:

1. **Pembelajaran transfer:** Diberikan jaringan yang dilatih pada tugas T_i , dapatkah kita mengadaptasinya untuk mempelajari tugas T_{i+1} ?
2. **Meta-pembelajaran:** Diberikan jaringan yang dapat melakukan tugas T_1 , T_2 , dan T_{n+1} , dapatkah kita dengan cepat mengadaptasinya untuk mempelajari tugas T_n ?
3. **Pembelajaran multi-tugas:** Dapatkah kita melatih jaringan pada semua tugas— T_1 , T_2 , hingga T_n —pada saat yang bersamaan?

Bab-bab selanjutnya membahas algoritma yang menjelaskan paradigma pembelajaran ini secara detail.

Untuk robot multi-tugas, penentuan tujuan memiliki berbagai bentuk. Otak robot dapat dikonfigurasi untuk mencapai suatu tujuan. Namun, pertanyaan tentang bagaimana menyampaikan tujuan kepada robot tetap ada. Hal ini sangat penting untuk pembelajaran mendalam, yang merupakan kerangka kerja optimasi objektif. Ide-ide yang dijelaskan dalam buku untuk menjawab pertanyaan ini meliputi hal-hal berikut:

- **Pengkondisian bahasa:** Praktik menentukan target untuk robotika menggunakan bahasa sebagai antarmuka. Bahasa telah menjadi antarmuka alami untuk interaksi antar manusia. Perluasan pembelajaran mendalam untuk pemrosesan bahasa alami telah menjadikan antarmuka bahasa untuk robot dan generatif sebagai standar.
- **Pengkondisian tujuan:** Pengkondisian tujuan sering terjadi pada kebijakan di mana Anda dapat melatih kebijakan untuk mencapai keadaan tujuan, yang diberikan sebagai masukan. Misalnya, pembelajaran penguatan terkondisi tujuan (GCRL) melatih agen untuk mencapai tujuan yang berbeda dalam skenario tertentu.
- **Tabrakan diri:** Jika robot tidak diprogram dengan benar, robot dapat bertabrakan dengan dirinya sendiri. Tujuan tabrakan diri adalah untuk membuat robot menyadari tubuhnya untuk menghindari tabrakan selama bergerak. Jenis tabrakan lainnya adalah menghindari tabrakan dengan lingkungan. Misalnya, anggaplah robot mengalami rintangan yang tidak terduga saat menavigasi lingkungannya. Dalam hal ini, robot dapat menggunakan penghindaran tabrakan untuk menentukan tindakan dan jalur terbaik di sekitar objek dan melanjutkan tugasnya.
- **Pembelajaran robot hierarkis:** Tujuan pembelajaran hierarkis adalah untuk memecah masalah yang lebih besar menjadi hierarki submasalah. Hal ini memungkinkan tugas

induk tingkat tinggi untuk memanggil tugas anak tingkat rendah untuk menyelesaikan suatu tugas.

1.7 MENUJU KECERDASAN UMUM YANG TERWUJUD

Memecahkan masalah kecerdasan umum buatan (AGI) adalah salah satu masalah yang paling hangat dan penting di generasi kita. Nick Bostrom mendefinisikan AGI sebagai “kecerdasan yang jauh lebih pintar daripada otak manusia terbaik di hampir setiap bidang”. Wikipedia mendefinisikannya sebagai “kemampuan agen cerdas untuk memahami atau mempelajari tugas intelektual apa pun yang dapat dilakukan manusia”. *Open Philanthropy* menggambarkan “AI transformatif” dengan definisi ekonomi, sebagai sesuatu yang dapat meningkatkan Produk Domestik Bruto Dunia sepuluh kali lipat.

Namun semua spesifikasi ini terutama berfokus pada AI digital, yang antarmuka imajinernya ke dunia nyata masih manusia. Transformasi industri apa pun yang ada sebelum Internet—seperti manufaktur, konstruksi, pengemudi, logistik, energi, pertambangan, dan pertanian—dan yang mencakup sebagian besar PDB global akan membutuhkan penyelesaian AI yang terwujud, yaitu kecerdasan dalam perwujudan yang memiliki penalaran fisik dan dapat memanipulasi dunia fisik. Kemajuan dalam komputasi dan data telah mempermudah siapa pun untuk membangun dan menguji model pembelajaran mendalam untuk robotika. Tetapi bagaimana seseorang mengetahui apakah AI itu terwujud dan umum? Steve Wozniak telah mengusulkan tes kopi untuk AGI yang terwujud: sebuah mesin dapat belajar cara membuat kopi di dapur manusia yang tidak terlihat. Ini bertentangan dengan paradoks Moravec, karena AI saat ini dapat menciptakan gambar dan film grafis berkualitas tinggi, tetapi membuat secangkir kopi sederhana dalam pengaturan umum masih tidak mungkin.

Robotika adalah AI yang sulit, karena perlu memecahkan masalah penglihatan komputer (untuk memahami dunia), bahasa (untuk berinteraksi dengan manusia dan berkomunikasi), manipulasi dan navigasi (untuk bertindak di dunia nyata), dan penggunaan alat (untuk pencarian, penalaran yang terwujud, dll.). Robotika adalah versi AI yang sulit dan mencakup banyak hal, yang benar-benar dikemas untuk mengubah dunia, dan yang menghuni dan hidup di antara kita, bukan hanya di balik layar dan di pusat data. Tolok ukur spesifik untuk kecerdasan yang terwujud untuk mengukur dan melacak kemajuan menuju pemecahan masalah robotika masih terus berlangsung.

Lingkungan Sangat Terkait dengan Definisi Kecerdasan

Apa bintang utara untuk kecerdasan yang terwujud? Ada beberapa bukti evolusi yang menunjukkan bagaimana solusi tersebut mungkin terlihat. Pembelajaran seumur hidup selama beberapa episode telah mengkodekan data dalam gen kita, sedemikian rupa sehingga bayi memahami gerakan terstruktur dan fisika dunia sebelum mereka memahami dan mengerti bahasa. Apa yang cerdas sangat terkait dengan apa yang memberikan keuntungan bertahan hidup di suatu lingkungan. Misalnya, hewan air memiliki sistem penglihatan yang jauh lebih baik dalam melihat di bawah air karena mereka telah berevolusi untuk mengakomodasi pembiasan oleh air dengan cara yang tidak dimiliki manusia.

Sensor kita yang mencoba meniru jangkauan penglihatan kita, dan data yang telah kita kumpulkan berdasarkan hal itu, termasuk video YouTube, mengalami overfitting pada domain kemampuan visual kita. Pada pasien yang telah menjalani operasi pengangkatan katarak, yang memungkinkan mereka untuk melihat untuk pertama kalinya, terlihat bahwa meskipun menghabiskan seluruh hidup mereka di dunia 3D, mereka kurang memahami citra spasial karena sensor mereka tidak memiliki input tersebut. Pada intinya, lingkungan dan agen tidak dapat dipisahkan dari definisi kecerdasan.

Dekade terakhir studi AI telah menyebabkan munculnya transformer besar yang sangat baik dalam multi-tugas bicara dan penglihatan. AI yang mengutamakan bahasa akan rentan terhadap mode kegagalan agen buta, di luar konteks visual yang diterimanya dari korpus pelatihan yang dikumpulkan dari manusia yang dapat melihat. Secara logis, hal ini menunjukkan bahwa model bahasa visual akan mengalami kesulitan dalam mendekati parameter aktuator yang melekat pada pelaksanaan kontrol yang tepat terhadap agen yang berwujud. Penalaran tentang dunia nyata tidak hanya membutuhkan pemikiran tentang ruang metodologis dan bahasa, tetapi juga berlandaskan pada konteks dunia nyata. Di dunia yang dibangun dan dirancang untuk manusia, kecerdasan yang tidak peka terhadap dinamika sensorimotor akan menjadi suboptimal, dan keterampilan supermanusia membutuhkan agensi fisik dan kontrol universal. Memiliki perwujudan fisik mutlak diperlukan untuk AGI. Buku ini mengeksplorasi bagaimana secara praktis mencapai tujuan tersebut dan membangun masa depan dengan robot yang cerdas secara umum.

1.8 RINGKASAN

Bab ini membahas poin-poin berikut:

- Kemajuan terbaru dalam AI memindahkan robot dari laboratorium studi yang terkontrol ke aplikasi dunia nyata, sehingga memungkinkan mereka untuk beradaptasi dan melakukan generalisasi terhadap lingkungan yang dinamis dan tidak terduga. AI memungkinkan robot untuk belajar dari data, daripada bergantung pada aturan yang telah diprogram sebelumnya, sehingga membuat mereka serbaguna di berbagai tugas dan industri.
- Sebuah robot beroperasi dengan memproses input sensor (seperti kamera, LiDAR, dan IMU) menjadi output yang dapat ditindaklanjuti melalui mikroprosesor atau mikrokontroler, yang kemudian memberi sinyal kepada aktuator (seperti motor atau material lunak) untuk mengeksekusi gerakan fisik. Siklus persepsi-aksi ini merupakan inti dari kemampuan robot untuk berinteraksi dengan dan memanipulasi lingkungannya.
- Robot dapat diklasifikasikan berdasarkan aplikasinya, termasuk robot industri (untuk perakitan, pengelasan, dll.), robot layanan (untuk tugas-tugas seperti pembersihan atau telepresence), robot medis (untuk pembedahan dan diagnostik), robot militer dan pertahanan (untuk pengintaian atau penjinakan bom), dan lain-lain. Robot juga dapat dikategorikan berdasarkan strukturnya, seperti bentuk beroda, berkaki, terbang, atau humanoid, tergantung pada fungsi dan lingkungannya.

- Prinsip desain utama meliputi penggunaan manipulator robot (lengan dengan sendi dan sambungan), derajat kebebasan (rentang gerakan independen yang dapat dilakukan robot), efektor ujung (alat di ujung lengan robot yang digunakan untuk tugas-tugas seperti menggenggam), ruang kerja (area yang dapat dijangkau robot secara fisik), dan kinematika (studi tentang gerakan sendi dan sambungan, termasuk kinematika maju dan terbalik untuk perencanaan).
- Permintaan akan robot meningkat pesat karena kemampuannya yang semakin meningkat untuk melakukan generalisasi dan berbagai tugas di berbagai industri. Pengembangan infrastruktur, kerangka kerja perangkat lunak, dan sistem pembelajaran dari AI telah mendorong kemajuan dalam kemampuan dan adopsi robot.
- Robot belajar melakukan tugas melalui empat jenis pembelajaran mesin utama: pembelajaran terawasi (pelatihan pada data berlabel), pembelajaran tak terawasi (menemukan pola dalam data tanpa label), pembelajaran terawasi lemah, dan pembelajaran penguatan.
- Berbagai kerangka kerja memandu pembelajaran robot, termasuk pembelajaran transfer, meta-pembelajaran, dan pembelajaran multi-tugas.
- Tujuan utama AI dalam robotika adalah membangun agen yang cerdas secara umum di dunia fisik.

Bab selanjutnya membahas sensor, persepsi robot, dan metode jaringan saraf dan transformator umum yang digunakan robot untuk merasakan dan memahami lingkungannya.

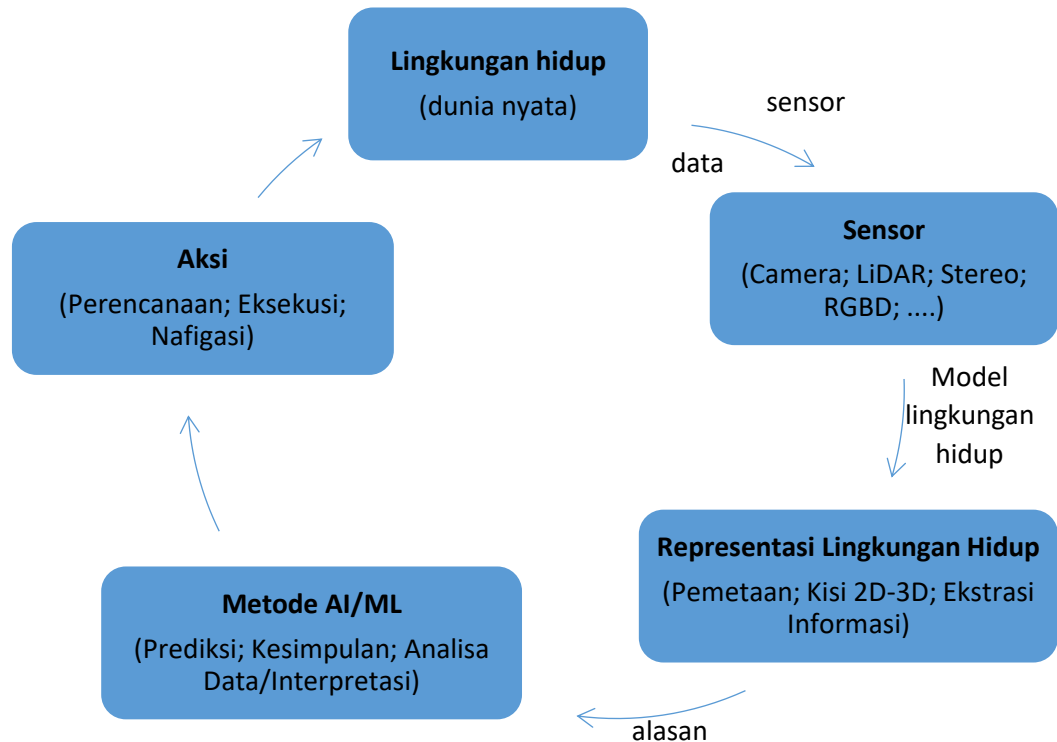
BAB 2

PERSEPSI ROBOT: SENSOR DAN PEMROSESAN CITRA

Manusia menggunakan lima indra untuk merasakan lingkungan dan berbagai jalur kognitif untuk memproses masukan ini. Gabungan indra, jalur, dan otak ini membentuk sistem persepsi kita, yang memungkinkan kita untuk mendeteksi gerakan di sekitar kita, mengenali wajah teman, dan mendeteksi aroma yang familiar. Demikian pula, robot perlu menyadari dan memahami apa yang ada di sekitarnya agar dapat berfungsi di dunia nyata. Sistem persepsi memungkinkan robot untuk mencapai dua tujuan: merasakan lingkungan sekitarnya dan memahami serta menalar tentangnya.

Tujuan pertama dari sistem persepsi dicapai oleh rangkaian sensor. Sensor bertindak sebagai mata dan telinga robot, memungkinkannya untuk mengamati dan merekam dunia fisik di sekitarnya. Selain itu, sensor memungkinkan robot untuk mengumpulkan data yang berguna yang pada gilirannya memungkinkan mereka untuk berevolusi dan beradaptasi di suatu lingkungan.

Tujuan kedua melibatkan interpretasi data dari sensor dan mengekstrak informasi yang relevan yang dapat membantu dalam menyelesaikan tujuan robot. Secara tradisional, bagian ini dilakukan dengan metode klasik yang sempit dan spesifik untuk robot dan tugas tertentu, sehingga sulit untuk digeneralisasikan ke lingkungan yang berubah. Sistem persepsi pembelajaran mesin, di sisi lain, jauh lebih tangguh. Sistem ini meningkat seiring dengan data, dan dapat digunakan untuk memetakan lingkungan sekitar robot, sehingga robot dapat menavigasi, mendeteksi, dan melacak objek. Gambaran umum dari alur kerja ini ditunjukkan pada Gambar 2.1.



Gambar 2-1. Sistem persepsi robot tipikal akan mencakup sensor yang mengumpulkan data dan algoritma ML/AI untuk interpretasi, perencanaan, dan eksekusi tindakan.

Bab ini membahas penginderaan, menjelaskan jenis-jenis sensor, dan menyoroti berbagai algoritma ML yang umum digunakan untuk persepsi robot.

2.1 SENSOR DAN PERSEPSI ROBOT

Robot menggunakan sensor untuk menerima informasi tentang lingkungan sekitarnya dan memutuskan bagaimana berinteraksi dengan dunia dan bermanuver di sekitarnya dengan aman. Kamera mengambil gambar, LiDAR menyediakan awan titik dan kedalaman yang akurat, sensor ultrasonik mengukur kedekatan (terutama objek bergerak), dan Unit Pengukuran Inersia (IMU) memberikan informasi tentang orientasi dan gerakan. Robot dapat menggunakan sensor tunggal untuk pengenalan objek, lokalisasi dan pemetaan, penghindaran tabrakan, dan kontrol umpan balik. Robot juga dapat menggunakan teknik fusi sensor untuk menggabungkan data dari berbagai sensor dan mendapatkan informasi yang lebih komprehensif.

Meskipun aplikasi robotika menggunakan berbagai macam sensor, buku ini hanya membahas sensor yang lebih populer. Sensor-sensor tersebut terbagi dalam empat kategori utama: (1) penglihatan monokular, terutama kamera, (2) sensor kedalaman, (3) sensor jarak, seperti LiDAR dan sensor ultrasonik, dan (4) unit pengukuran inersia (IMU).

Sensor Penglihatan (Kamera)

Salah satu sensor terpenting dalam sistem robot adalah kameranya. Cahaya ditangkap oleh sensor kamera, yang mengubah cahaya menjadi sinyal listrik untuk menghasilkan gambar. Kamera terdiri dari berbagai piksel atau komponen fotosensitif. Setiap piksel merekam jumlah cahaya yang mengenainya.

Dua jenis sensor gambar yang paling populer yang digunakan dalam robotika adalah CMOS (*Complementary Metal-Oxide Semiconductor*) dan CCD (*Charge-Coupled Device*). Sensor gambar CCD dan CMOS sama-sama mengubah cahaya menjadi elektron dengan menangkap foton (partikel cahaya) dengan banyak fotosit, yaitu daerah kecil yang peka terhadap cahaya pada sensor yang sesuai dengan piksel dalam gambar akhir. Saat mengambil gambar, fotosit ini mengumpulkan dan menyimpan foton sebagai sinyal listrik. Aspek kunci yang membedakan setiap sensor adalah cara muatan terakumulasi (muatan listrik yang dihasilkan oleh fotosit saat menangkap foton) dari setiap fotosit diangkut.

Sensor CCD mengangkut muatan dengan distorsi minimal, menghasilkan gambar berkualitas tinggi dan sensitif tetapi mengonsumsi daya yang jauh lebih besar. Sensor CMOS lebih fleksibel dan lebih murah tetapi cenderung lebih rentan terhadap noise dan memiliki sensitivitas cahaya yang lebih rendah. Waktu paparan, sensitivitas ISO, dan ukuran bukaan sensor kamera semuanya merupakan parameter yang dapat disesuaikan, memungkinkan mereka untuk beradaptasi dengan kondisi pencahayaan yang berbeda.

Mari kita uraikan beberapa konsep kunci kamera:

1. Resolusi piksel sensor kamera menentukan tingkat detail dalam gambar yang diambil. Sensor resolusi tinggi mencakup lebih banyak piksel, membuat gambar lebih jernih dan lebih detail, tetapi menggunakan lebih banyak memori dan daya pemrosesan.
2. Saluran mengacu pada saluran warna yang digunakan dalam kamera. Kerangka kerja umum adalah RGB, di mana setiap gambar memiliki saluran merah, hijau, dan biru. Sebagian besar kamera menggunakan berbagai filter warna—biasanya pola Bayer—di atas piksel untuk menangkap informasi warna. Karena filter ini, setiap piksel dapat menangkap cahaya merah, hijau, atau biru. Sensor kamera menghasilkan gambar berwarna penuh dengan menginterpolasi nilai warna dari piksel terdekat. Jadi, kamera dengan resolusi 480.640 dengan tiga saluran (merah, hijau, biru) menghasilkan gambar yang dapat diinterpretasikan sebagai matriks berukuran 480.640.
3. Laju bingkai mengacu pada jumlah gambar yang diambil sensor kamera setiap detik. Aplikasi robotika di lingkungan berkecepatan tinggi atau dinamis dapat memperoleh manfaat dari frame rate yang lebih tinggi karena memberikan persepsi waktu nyata, sehingga memungkinkan waktu reaksi yang lebih cepat.
4. Pemrosesan pasca. Data gambar mentah dari kamera dapat diproses lebih lanjut untuk pengurangan noise, koreksi white balance, koreksi warna, dan ekstraksi fitur. Algoritma pemrosesan gambar kemudian dapat digunakan untuk meningkatkan kualitas gambar, menemukan objek, mengenali pola, dan mengekstrak data kedalaman. Sensor kamera sering digabungkan dengan sensor lain, seperti LiDAR, radar, atau IMU, dalam strategi fusi sensor untuk memaksimalkan kegunaannya.

Kamera sebagai sensor juga memiliki keterbatasan tertentu:

- a. Kamera mungkin kesulitan dalam kondisi cahaya rendah atau kontras tinggi dan membutuhkan banyak pencahayaan untuk hasil terbaik.
- b. Kamera mungkin kesulitan untuk secara efektif merasakan kedalaman dan informasi 3D tanpa sensor tambahan.

- c. Kamera juga rentan terhadap oklusi.
- d. Pemrosesan gambar skala besar juga dapat menuntut komputasi yang tinggi.

Pertimbangan Utama untuk Kamera

Bagian ini membahas pertimbangan untuk memilih kamera untuk aplikasi robotika Anda. Seperti yang ditunjukkan pada Gambar 2.2, tiga faktor utama menentukan pemilihan kamera.



Gambar 2.2. Tiga komponen utama kamera yang perlu Anda pertimbangkan saat memilihnya untuk aplikasi robot adalah frame per detik, resolusi, dan biaya.

Mobil otonom, misalnya, membutuhkan kamera yang dapat menangkap gambar dengan frekuensi tinggi karena kecepatan mobil yang bergerak. Jika sebuah mobil bergerak dengan kecepatan 60 mil per jam dan melewati mobil lain dari arah berlawanan dengan kecepatan yang sama, setiap frame yang ditangkap mewakili jarak yang signifikan antara kedua mobil tersebut. Dalam skenario ini, idealnya kita ingin menangkap dan memproses sebanyak mungkin frame per detik (fps). Biasanya, kamera otomotif beroperasi pada 30 fps, tetapi ini dapat menimbulkan penundaan pemrosesan untuk setiap frame. Oleh karena itu, untuk mendeteksi objek dinamis seperti pejalan kaki dan memprediksi jalurnya secara akurat, diperlukan beberapa kali pengambilan gambar per frame. Jika kita melewatkan satu atau beberapa frame karena kamera tidak cukup cepat, hasilnya bisa berupa tabrakan. Misalnya, sistem bantuan pengemudi canggih (ADAS) saat ini mungkin menggunakan kamera dengan resolusi 8 megapiksel dan kecepatan frame 60 fps untuk memastikan deteksi dan pemahaman lingkungan yang andal.

Kamera yang digunakan untuk aplikasi industri memiliki spesifikasi yang berbeda. Memetik, mengemas, dan menggenggam objek tidak melibatkan gerakan kecepatan tinggi dan tidak selalu memerlukan pengambilan gambar dengan kecepatan bingkai tinggi. Biasanya, kamera 30 fps biasa sudah cukup. Kamera dengan kecepatan bingkai tinggi biasanya cenderung memiliki resolusi lebih rendah, dan resolusi yang lebih rendah dapat memengaruhi ketelitian deteksi, terutama pada objek yang jauh. Memilih kamera dengan resolusi tinggi dan frekuensi tinggi dapat meningkatkan biaya, terutama karena memerlukan bandwidth dan daya pemrosesan yang lebih tinggi, dan mungkin sama mahalnya dengan LiDAR 3D. Namun, pada akhirnya, persyaratan presisi dan batasan biaya suatu aplikasi yang menentukan pemilihan kamera.

Kamera Berbasis Peristiwa

Jenis kamera lain adalah kamera berbasis peristiwa atau neuromorfik. Kamera ini menghasilkan perubahan kecerahan tingkat piksel. Ini berbeda dengan kamera bingkai biasa, yang mengirimkan seluruh susunan informasi dari satu bingkai yang ditangkap oleh rana pada waktu tertentu. Format data dan keluaran kamera berbasis peristiwa menawarkan keuntungan

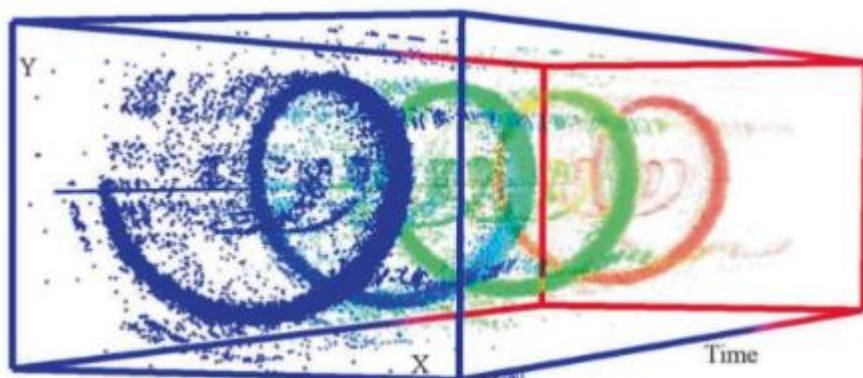
yang signifikan, karena satu-satunya data yang ditransmisikan adalah informasi piksel individual yang telah berubah dari bingkai ke bingkai. Hal ini memungkinkan kamera tersebut untuk menangkap objek yang bergerak dengan kecepatan tinggi tanpa efek buram. Kamera peristiwa ditunjukkan pada Gambar 2.3, dan keluaran data dari kamera peristiwa ditunjukkan pada Gambar 2.4.

Dibandingkan dengan kamera frame biasa, kamera berbasis peristiwa memiliki beberapa keunggulan, seperti tidak adanya blur akibat gerakan, resolusi temporal yang tinggi, dan rentang dinamis yang tinggi namun bandwidth rendah. Kamera berbasis peristiwa masih merupakan pendekatan baru untuk sebagian besar solusi, dan harganya masih belum terjangkau seperti kamera biasa. Namun, beberapa aplikasi sangat cocok untuk kamera berbasis peristiwa:

- Karena laju data yang rendah dan informasi yang jarang yang diberikan oleh sensor berbasis peristiwa, mereka dapat secara efektif melacak objek dengan daya komputasi yang rendah.
- Interpolasi frame, estimasi aliran optik, dan aplikasi perekaman kecepatan tinggi mendapat manfaat dari resolusi temporal yang tinggi dan pemodelan temporal yang lebih baik yang dimungkinkan oleh kamera berbasis peristiwa.



Gambar 2.3. Kamera berbasis peristiwa 680 x 480.



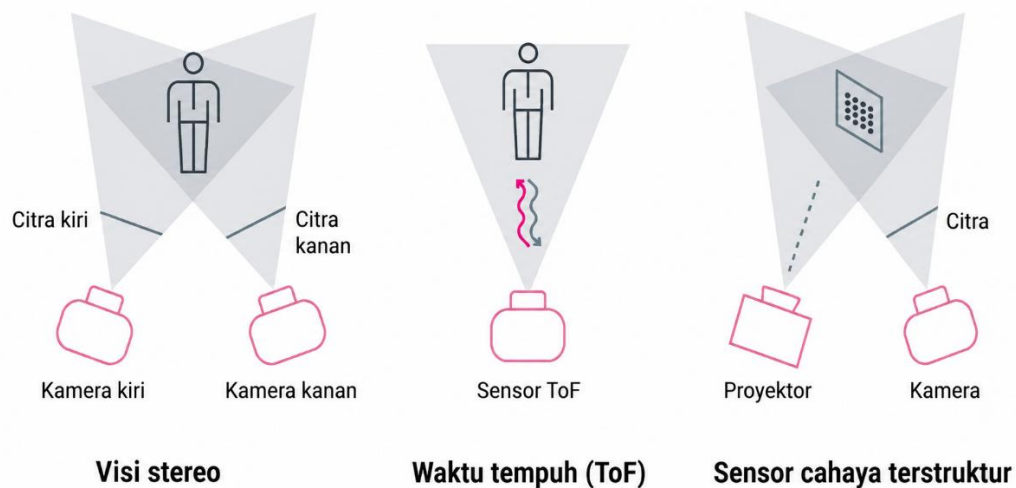
Gambar 2.4. Keluaran data dari kamera berbasis peristiwa.

Studi dari kelompok Davide Scaramuzza di Universitas Zurich menunjukkan pengembangan detektor objek hibrida berbasis peristiwa dan bingkai. Metode ini menggabungkan resolusi temporal tinggi dan efisiensi kamera berbasis peristiwa dengan pencitraan detail dari sensor tradisional, secara signifikan mengurangi latensi persepsi dan komputasi sambil mempertahankan akurasi. Dengan menggunakan kamera RGB 20 Hz dengan kamera berbasis peristiwa, sistem ini mencapai latensi yang sama dengan kamera 5.000 Hz dengan bandwidth kamera 45 Hz. Untuk melihat lebih detail studi terbaru dalam visi dan kamera berbasis peristiwa, kami merekomendasikan sumber daya ini.

Sensor Kedalaman

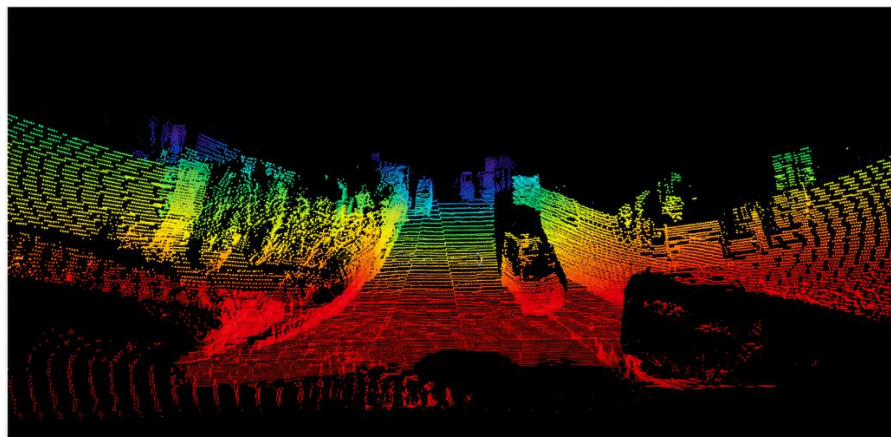
Robot juga memiliki sensor yang mengukur kedalaman atau jarak objek di sekitarnya, seperti kamera waktu tempuh dan sensor cahaya terstruktur. Sensor-sensor ini memungkinkan robot untuk memahami struktur tiga dimensi lingkungan sekitarnya dan melakukan tugas-tugas seperti rekonstruksi objek, interpretasi adegan, dan identifikasi gerakan karena informasi kedalaman yang tepat yang diberikan oleh sensor-sensor ini. Gambar 2.5 menunjukkan sensor kedalaman yang menggunakan prinsip *time-of-flight* (ToF), cahaya terstruktur, dan stereo-vision untuk mengukur informasi kedalaman.

- **Sensor waktu tempuh (ToF):** Sensor ToF mengukur waktu yang dibutuhkan sinyal cahaya untuk memantul kembali setelah dipantulkan dari objek di sekitarnya. Sinyal cahaya biasanya berupa sinyal inframerah (IR). Sensor kedalaman dapat menghitung waktu yang dibutuhkan untuk perjalanan pulang pergi untuk menentukan jarak ke objek.
- **Sensor cahaya terstruktur:** Sensor cahaya terstruktur memproyeksikan pola cahaya ke dalam adegan, seperti kisi-kisi titik inframerah atau kumpulan pola terstruktur. Kamera inframerah sensor kemudian merekam pola yang terdistorsi. Sensor kedalaman dapat menentukan informasi kedalaman berdasarkan distorsi yang dihasilkan oleh permukaan objek dengan memeriksa deformasi pola tersebut.
- **Sensor penglihatan stereo:** Mirip dengan mata manusia, sensor penglihatan stereo menggunakan sepasang kamera dengan pemisahan garis dasar yang diketahui. Sensor kedalaman dapat menentukan kedalaman dengan membandingkan perbedaan atau diskrepansi antara piksel yang sesuai dalam gambar stereo, dan setiap kamera menangkap pandangan adegan yang sedikit berbeda. Teknik triangulasi sering digunakan untuk menentukan kedalaman berdasarkan perbedaan piksel.



Gambar 2.5. Metode pengukuran kedalaman menggunakan *Time-of-flight* (ToF), cahaya terstruktur, dan penglihatan stereo

Output dari sensor kedalaman adalah awan titik, representasi 3D dari adegan yang terdiri dari kumpulan titik 3D yang mewakili permukaan adegan, atau citra kedalaman, yang mewakili jarak setiap piksel dari sensor. Banyak sensor kedalaman seperti ToF dan cahaya terstruktur akan langsung menghasilkan citra kedalaman. Dalam beberapa kasus, terutama untuk sensor yang digunakan dalam aplikasi robotika, outputnya adalah awan titik. Lokasi spesifik di ruang angkasa dan informasi warna yang sesuai diwakili oleh setiap titik dalam awan. Contoh output awan titik 3D yang dihasilkan dari sensor kedalaman ditunjukkan pada Gambar 2.6.



Simulasi Awan Titik 3D

Gambar 2.6. Contoh keluaran awan titik 3D dari sensor kedalaman

Jangkauan, presisi, sensitivitas terhadap pencahayaan, masalah oklusi, dan kesulitan memotret permukaan tembus cahaya atau reflektif adalah kekurangan dari sensor kedalaman. Namun dalam beberapa tahun terakhir, peningkatan teknologi sensor kedalaman telah meningkatkan kinerjanya. Misalnya, sensor ini dikombinasikan dengan sensor lain, seperti

kamera atau unit pengukuran inersia (IMU), untuk menciptakan sistem persepsi yang lebih komprehensif.

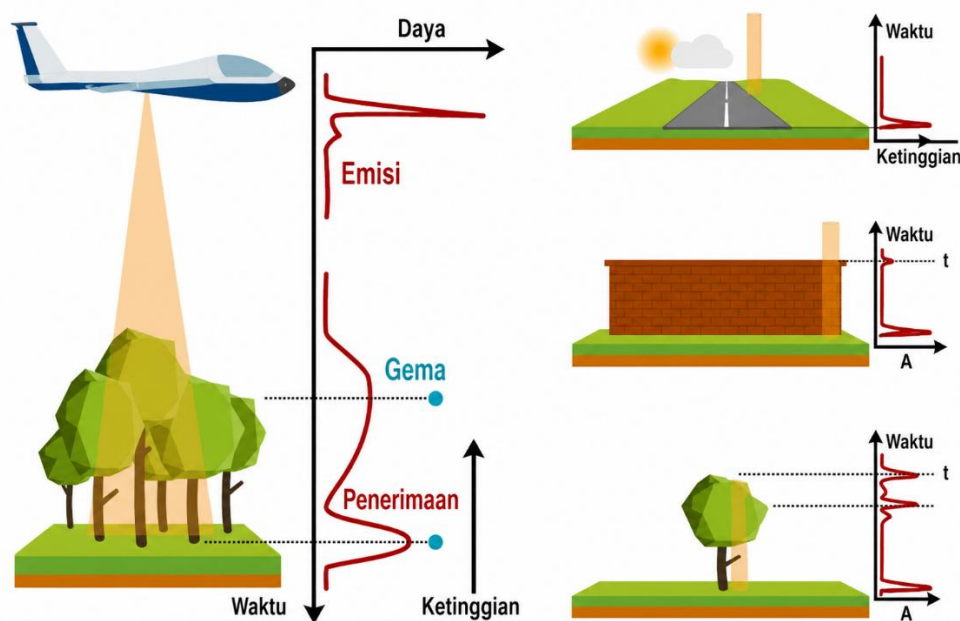
Sensor Jarak

Jenis sensor penting lainnya dalam robot adalah sensor jarak, seperti *Light Detection and Ranging* (LiDAR), dan sensor ultrasonik, yang memberikan informasi tentang jarak antara robot dan objek di sekitarnya. Jika sensor jarak memiliki fidelitas tinggi (seperti akurasi dan resolusi tinggi, yang biasanya terlihat pada LiDAR kepadatan tinggi), sensor ini berguna sebagai sensor kedalaman untuk mengukur jarak secara akurat dari robot yang bersangkutan ke objek lain. Bagian berikut membahas bagaimana LiDAR mencapai prestasi ini. Secara keseluruhan, sensor jarak membantu robot bernavigasi dan menghindari tabrakan dengan memungkinkan persepsi 3D.

LiDAR

Sensor LiDAR (*Light Detection and Ranging*) memancarkan pulsa laser dan melacak berapa lama waktu yang dibutuhkan pulsa untuk mencapai objek di dekatnya dan kembali ke sensor. Sensor LiDAR kemudian menentukan jarak ke objek dengan menghitung waktu tempuh bolak-balik, berdasarkan kecepatan cahaya.

Meskipun beberapa LiDAR juga menggunakan cahaya tampak, pulsa laser yang terutama dipancarkan oleh LiDAR adalah cahaya inframerah (IR). Tergantung pada sensor tertentu, pulsa laser dilepaskan dalam semburan singkat atau sebagai pancaran kontinu. Waktu tempuh digunakan di sini bersama dengan elektronik berkecepatan tinggi dan mekanisme pengaturan waktu yang tepat untuk mengukur waktu tempuh bolak-balik cahaya dengan lebih akurat. Gambaran umum proses refleksi LiDAR ditunjukkan pada Gambar 2.7.



Gambar 2.7. Gambaran umum sensor LiDAR. Sensor LiDAR menerima pulsa laser saat dipantulkan kembali ketika mengenai objek. Penerima pada sensor LiDAR menangkap pulsa laser yang kembali dan menghitung waktu tempuh yang sesuai

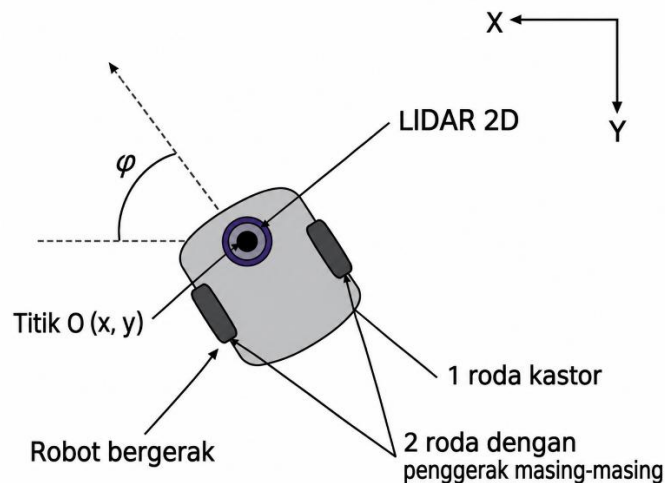
Sensor LiDAR menyediakan citra awan titik dari lingkungan sekitar dengan mengintegrasikan pembacaan jarak dari beberapa pulsa laser. Setiap titik dalam awan titik mewakili posisi yang berbeda dalam ruang 3D, bersama dengan detail jarak yang relevan. Karena sensor LiDAR menyediakan awan titik sebagai keluaran data mentahnya, pemrosesan tambahan diperlukan untuk membangun peta 3D resolusi tinggi, menghilangkan outlier, melakukan segmentasi objek, dan menyaring noise.

LiDAR dengan kepadatan tinggi adalah LiDAR yang memancarkan lebih banyak sinar secara sinkron. LiDAR lebar adalah LiDAR yang memiliki sudut bukaan sinar yang lebih besar, sehingga jika dipasang di atas mobil otonom, ia dapat memperoleh cakupan bahkan di sekitar mobil itu sendiri. Ini berguna dalam situasi seperti ketika penumpang naik dan turun, untuk menentukan apakah trotoar aman, dan sebagainya.

Sebagian besar sensor LiDAR menggunakan salah satu teknik pemindaian populer berikut:

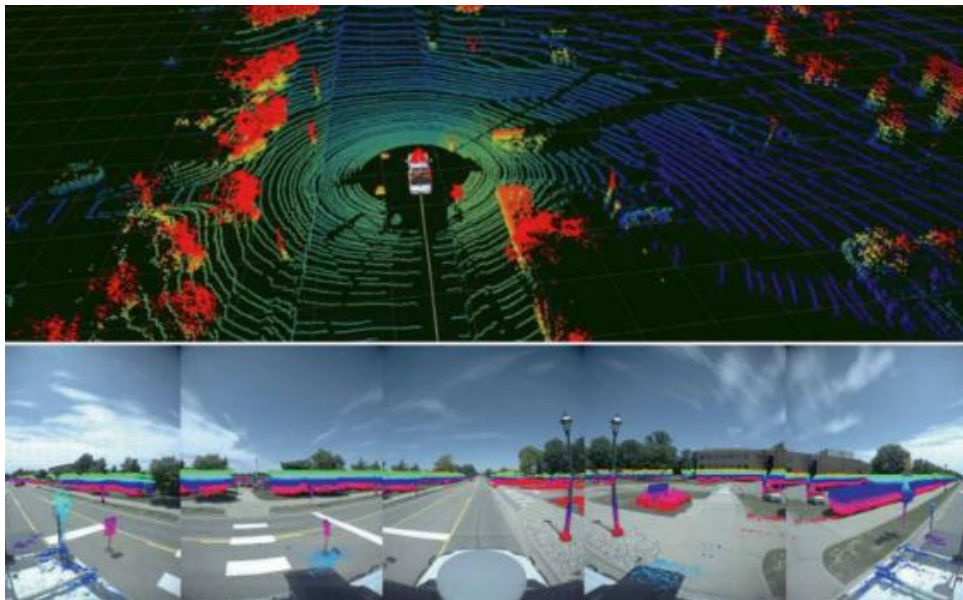
- **Pemindaian mekanis:** LiDAR pemindaian mekanis menggunakan cermin atau prisma berputar yang mengarahkan pulsa laser ke berbagai arah, memungkinkan sensor LiDAR untuk merekam gambar 360 derajat dari sekitarnya.
- **Pemindaian solid-state:** LiDAR solid-state tidak menggunakan elemen bergerak, tetapi mengarahkan pulsa laser pada berbagai sudut dengan menggunakan pemancar dan penerima solid-state yang dikontrol secara elektronik seperti susunan bertahap atau susunan bertahap optik. Karena tidak adanya bagian yang bergerak, LiDAR solid-state lebih hemat energi, lebih tahan lama, dan cenderung lebih kecil. Kekurangannya adalah jangkauan dan bidang pandangnya lebih rendah.
- **LiDAR solid-state hibrida:** Teknik ini menggabungkan elemen LiDAR mekanis dan solid-state, seringkali menggunakan cermin berputar untuk mencapai bidang pandang yang lebih luas sambil menggunakan komponen solid-state untuk pengarahannya dalam bidang pandang tersebut. Teknologi ini menawarkan kompromi antara bidang pandang yang luas dari LiDAR mekanik dan daya tahan serta ukuran yang ringkas dari LiDAR solid-state.

LiDAR 2D beroperasi pada satu bidang horizontal dan memancarkan sinar dalam pola seperti kipas. Resolusinya sangat rendah, tetapi cukup untuk banyak robot otonom sederhana untuk melakukan lokalisasi dan pemetaan simultan, seperti di lingkungan dalam ruangan seperti kantor, gudang, dan sebagainya. Contoh LiDAR 2D dapat dilihat pada Gambar 2.8. Dalam sebagian besar skenario, data ini dikombinasikan dengan sensor kedalaman seperti Intel RealSense untuk menangkap objek, rintangan, dan titik buta untuk LiDAR.



Gambar 2.8. Pengaturan robot bergerak dalam ruangan dengan LiDAR 2D untuk navigasi dan pemetaan. Robot ini memiliki dua roda bermotor individual dan satu roda kastor. Posisinya dilacak sebagai $O(x, y)$ dengan orientasi ϕ , dan sensor LiDAR 2D memindai lingkungan untuk mendeteksi rintangan dan sekitarnya.

Robot otonom canggih menerapkan LiDAR 3D untuk tumpukan persepsi karena manfaat cakupan 360 derajat dan kemampuan pemetaan yang lebih presisi dan lebih besar. Output dari LiDAR 3D ditunjukkan pada Gambar 2.9.



Gambar 2.9. Data keluaran LiDAR 360 derajat, 3D

Sensor LiDAR memberikan persepsi 3D yang presisi, penginderaan jarak jauh, bidang pandang 360 derajat, ketahanan terhadap kondisi pencahayaan, kepadatan data yang tinggi, dan kemampuan deteksi rintangan dalam aplikasi robotika.

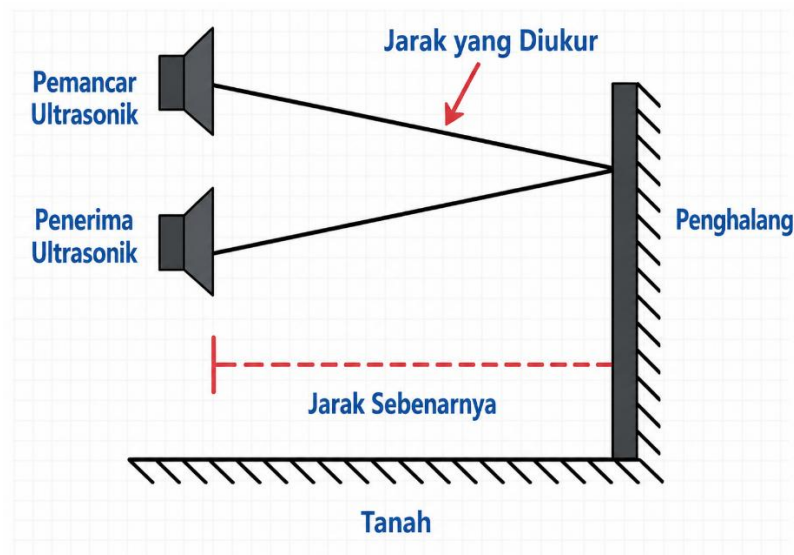
Namun, sensor ini bisa mahal, menggunakan banyak daya, memiliki resolusi rendah untuk objek kecil, memiliki masalah dengan permukaan yang mengkilap atau transparan,

bekerja lebih buruk daripada kamera dalam cuaca buruk seperti salju/hujan, dan membutuhkan sumber daya komputasi yang besar untuk pemrosesan. Sensor LiDAR harus dipertimbangkan dengan cermat terhadap kelebihan dan kekurangan ini untuk melihat apakah sensor tersebut sesuai untuk pekerjaan dan pengaturan robotika tertentu.

Sensor Ultrasonik

Untuk tugas-tugas yang melibatkan identifikasi rintangan, penghindaran tabrakan, deteksi keberadaan, pengukuran jarak, dan navigasi, sensor ultrasonik sering digunakan dalam robotika. Sensor ini menggunakan gelombang suara untuk berinteraksi dengan lingkungan. Secara khusus, sensor ultrasonik menghasilkan gelombang suara frekuensi tinggi yang seringkali tidak terdengar oleh manusia (20 kHz ke atas). Transduser piezoelektrik yang terpasang di dalam sensor mengubah energi listrik menjadi gelombang suara ultrasonik.

Setelah gelombang suara tercipta, gelombang tersebut menyebar ke luar seperti berkas berbentuk kerucut. Gelombang suara bergerak dalam garis lurus hingga bersentuhan dengan permukaan atau objek. Karakteristik permukaan objek menyebabkan gelombang suara yang dihasilkan memantul atau kembali ketika bertabrakan dengannya. Penerima sensor ultrasonik mendeteksi gelombang suara yang dipantulkan ini. Waktu yang dibutuhkan gelombang suara untuk menempuh perjalanan ke suatu objek dan kembali ke sensor diukur dan digunakan untuk menentukan jarak antara sensor dan objek menggunakan kecepatan suara dalam medium, biasanya udara. Gambar 2.10 menunjukkan gambaran umum proses ini.



Gambar 2.10. Gambaran umum sensor ultrasonik yang menunjukkan bagaimana pemancar memancarkan gelombang suara yang memantul dari suatu penghalang dan diterima oleh penerima, dengan jarak dihitung berdasarkan waktu yang dibutuhkan suara untuk kembali.

Desain sensor, frekuensi gelombang suara, dan lingkungan sekitar hanyalah beberapa variabel yang memengaruhi jangkauan dan akurasi sensor ultrasonik. Secara umum, sensor ultrasonik memiliki rentang akurasi beberapa milimeter hingga beberapa sentimeter dan dapat mendeteksi benda dalam jarak beberapa sentimeter hingga beberapa meter.

Namun, ada beberapa batasan pada sensor ultrasonik. Misalnya, sensor ini dapat mengalami kesulitan mendeteksi benda dengan permukaan yang tidak rata atau benda yang lebih kecil dari panjang gelombang gelombang suara. Sensor ini juga dapat terpengaruh oleh kebisingan latar belakang dan gema, yang mengurangi akurasi perhitungan jarak. Ini adalah faktor penting yang perlu dipertimbangkan ketika menentukan apakah sensor ultrasonik adalah yang terbaik untuk aplikasi robotika Anda.

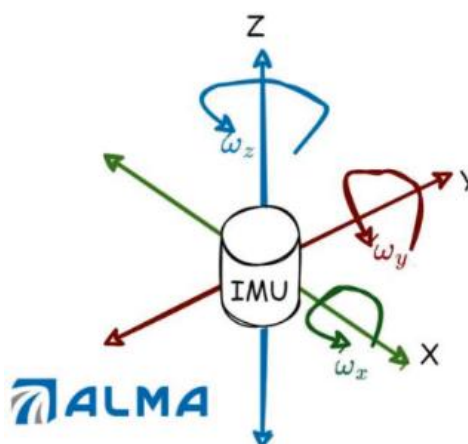
Unit Pengukuran Inersia (IMU)

Sensor lain yang perlu diperhatikan adalah IMU yang memberikan kemampuan pengukuran gerak dasar pada robot. IMU merekam data tentang percepatan, kecepatan sudut, dan medan magnet, sehingga memungkinkan robot untuk memahami gerakannya sendiri dan memberikan umpan balik untuk memperbaiki gerakannya.

Tiga sensor utama yang membentuk IMU adalah akselerometer, giroskop, dan terkadang magnetometer.

- **Akselerometer:** Akselerometer mengukur percepatan linier di sepanjang tiga sumbu ortogonal, seringkali x, y, dan z. Mereka menggunakan prinsip inersia, yang menyatakan bahwa percepatan menghasilkan sinyal listrik ketika suatu massa digerakkan. Akselerometer menghitung percepatan objek dengan menganalisis keluaran listrik.
- **Giroskop:** Giroskop menghitung laju rotasi atau kecepatan sudut di sekitar masing-masing dari tiga sumbu. Untuk mendeteksi perubahan orientasi, mereka bergantung pada prinsip momentum sudut. Giroskop melacak efek Coriolis saat objek berputar dan menghasilkan sinyal yang sesuai dengan laju rotasi.
- **Magnetometer:** Magnetometer tidak selalu menjadi bagian dari IMU, tetapi terkadang diintegrasikan untuk memberikan informasi tentang orientasi objek terhadap medan magnet Bumi. Mereka dapat digunakan untuk menentukan kekuatan dan arah medan magnet.

Gambar 2.11 menunjukkan akselerometer dan giroskop dengan pergerakan sepanjang tiga sumbu.



Gambar 2.11. Akselerometer dan giroskop dalam Unit Pengukuran Inersia (IMU) mendeteksi kecepatan sudut sepanjang sumbu X, Y, dan Z, yang diposisikan pada sudut 90° satu sama lain.

IMU menggunakan teknik fusi sensor untuk memahami gerakan objek secara menyeluruh. IMU memperkirakan posisi, orientasi, kecepatan, dan percepatan suatu objek dengan menggabungkan data dari akselerometer, giroskop, dan magnetometer dan seringkali menggunakan teknik seperti filter Kalman atau filter komplementer.

IMU sangat penting untuk melacak dan mengelola pergerakan robot. Berkat IMU, yang terus menerus mengukur dan memperbarui orientasi, kecepatan, dan percepatan, robot dapat mempertahankan stabilitas, memodifikasi pergerakannya, dan bereaksi terhadap pengaruh eksternal. Data ini digunakan untuk menyeimbangkan robot humanoid, mengoperasikan lengan robot, dan memungkinkan navigasi cepat pada mobil otonom, di antara hal-hal lainnya. Navigasi inersia menggunakan IMU untuk menentukan posisi dan lintasan robot dengan mengintegrasikan data dari akselerometer dan *giroskop* dari waktu ke waktu. Metode ini sangat membantu untuk navigasi di dalam gedung atau dalam keadaan sulit di mana sistem lokalisasi eksternal lainnya, seperti GPS, tidak tersedia atau tidak dapat diandalkan.

IMU memungkinkan pelacakan dan kontrol gerakan yang akurat dengan menyediakan data frekuensi tinggi secara real-time mengenai orientasi, percepatan, dan kecepatan robot. Namun, IMU rentan terhadap akumulasi kesalahan dari waktu ke waktu karena pergeseran sensor, yang dapat mengurangi akurasi. IMU juga tidak dapat memberikan informasi tentang posisi absolut tanpa referensi eksternal. Selain itu, interferensi medan magnet dapat menyebabkan masalah pada magnetometer di dalam IMU, mengurangi keandalannya dalam beberapa keadaan. Secara umum, IMU (*Inertial Measurement Unit*) adalah sensor yang berguna untuk deteksi dan kontrol gerakan robot, tetapi dalam aplikasi di mana akurasi tinggi dan data posisi absolut sangat penting, keterbatasan ini harus diperhitungkan.

2.2 MASALAH DALAM PERSEPSI

Sekarang Anda mengetahui berbagai jenis sensor yang digunakan dalam robotika dan cara kerjanya, Anda perlu memahami bagaimana data sensor dapat berguna untuk robotika. Jenis data input yang paling umum digunakan robot adalah gambar dari kamera. Gambar membantu robot memahami lingkungannya, seperti mengidentifikasi objek dan lokasinya, dan kemudian melakukan tugas sesuai dengan itu. Tugas persepsi umum untuk robotika meliputi klasifikasi, segmentasi semantik, segmentasi instans, dan deteksi objek. Tugas-tugas penting ini adalah dasar bagaimana banyak robot melihat, bernalar, dan memanipulasi objek.

Klasifikasi

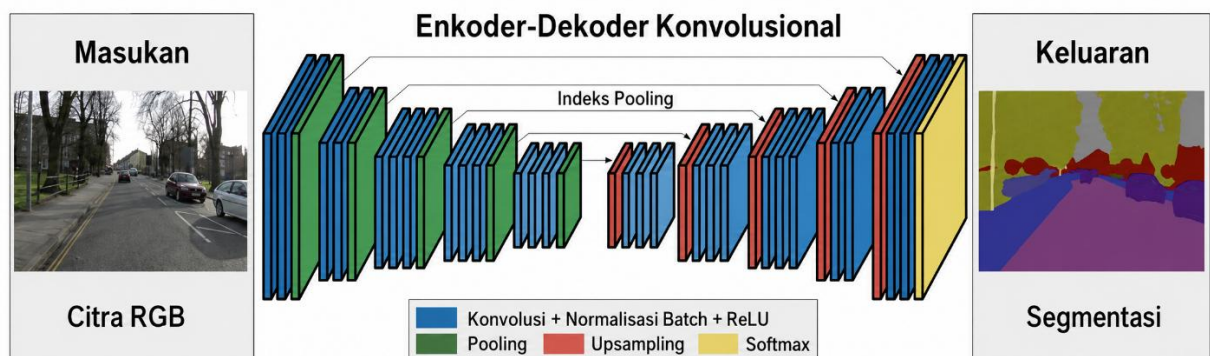
Agar robot dapat memanipulasi objek, mereka perlu memiliki pemahaman visual tentang objek dan lingkungannya. Klasifikasi gambar memungkinkan gambar diberi label berdasarkan serangkaian kategori tetap. Misalnya, jika robot dapur ditugaskan untuk mengambil cangkir, ia harus terlebih dahulu mengidentifikasi cangkir dalam gambar dan membedakannya dari kategori objek lain seperti gelas, piring, atau sendok. Robot melakukan ini dengan menerima serangkaian piksel (gambar) dan menetapkan kelas padanya. Katakanlah Anda memiliki kumpulan data N gambar, masing-masing diberi label dengan salah satu dari K kelas. Tujuannya adalah untuk menggunakan kumpulan data ini, meneruskannya melalui

model, dan mempelajari seperti apa setiap kelas tersebut. Anda kemudian dapat mengevaluasi seberapa baik model Anda mempelajari kelas objek dengan membuat prediksi pada kumpulan gambar baru.

Segmentasi

Bayangkan contoh kendaraan otonom yang mencoba menavigasi jalan dan memahami lingkungannya di jalan yang ramai dengan trotoar, mobil, dan bus di latar depan. Di latar belakang, ada bangunan, pohon, dan langit. Segmentasi gambar bertujuan untuk menetapkan setiap piksel dalam gambar yang dilihat mobil ke objek yang dimilikinya. Ini dapat memisahkan latar depan dari latar belakang, mengidentifikasi lokasi tepat dari mobil di jalan, mengidentifikasi pejalan kaki, dan menandai di mana jalan berada.

Pemahaman tingkat piksel yang diberikan oleh segmentasi gambar dapat membantu robot memahami bagaimana mereka dapat menavigasi lingkungan mereka. Sebagai contoh, segmentasi gambar dapat digunakan untuk mengekstrak objek yang mungkin Anda pegang dari sekumpulan objek atau menemukan area aman untuk berkendara bagi mobil otonom. Struktur encoder-decoder sering digunakan untuk segmentasi, seperti yang ditunjukkan pada Gambar 2.12.



Gambar 2.12. Arsitektur encoder-decoder konvolusional untuk segmentasi gambar. Gambar input RGB diproses melalui lapisan konvolusi, pooling, dan upsampling, diikuti oleh fungsi softmax untuk menghasilkan output yang tersegmentasi. Bagian CNN membahas lebih detail tentang arti masing-masing lapisan ini.

Pada dasarnya, encoder terdiri dari jaringan klasifikasi yang telah dilatih sebelumnya—yang berarti telah dilatih pada sejumlah besar data—biasanya ResNet, yang dipasangkan dengan jaringan decoder. Encoder melakukan downsampling gambar untuk mempelajari representasi yang ringkas, dan decoder melakukan upsampling representasi ini untuk merekonstruksi ruang piksel resolusi tinggi. Dengan menggunakan tensor spasial resolusi rendah yang dihasilkan oleh encoder yang merangkum informasi tingkat tinggi yang terkondensasi, decoder menghasilkan output yang tersegmentasi dengan resolusi tinggi.

Menumpuk lapisan encoder dan decoder secara langsung mengakibatkan hilangnya informasi tingkat rendah yang penting bagi model untuk dimiliki di lapisan selanjutnya. Untuk mengimbangi informasi yang hilang, koneksi lewati (*skip connection*) digunakan untuk

memungkinkan decoder mengakses fitur tingkat rendah yang dihasilkan oleh lapisan encoder. Hal ini memungkinkan decoder untuk menangkap fitur tingkat rendah dan tingkat tinggi di bagian selanjutnya dari model. Konsep utama di balik koneksi lewati adalah bahwa output perantara dari encoder digabungkan dengan input ke lapisan perantara decoder pada titik-titik tertentu.

Terdapat dua jenis segmentasi gambar utama: semantik dan instans. Dengan segmentasi semantik, semua objek yang sejenis (seseorang) ditandai menggunakan satu label kelas, sedangkan objek serupa mendapatkan label terpisah dalam segmentasi instans (yaitu, orang yang berbeda ditandai sebagai label terpisah). Perbedaan antara kedua jenis segmentasi ini ditunjukkan pada Gambar 2.13.

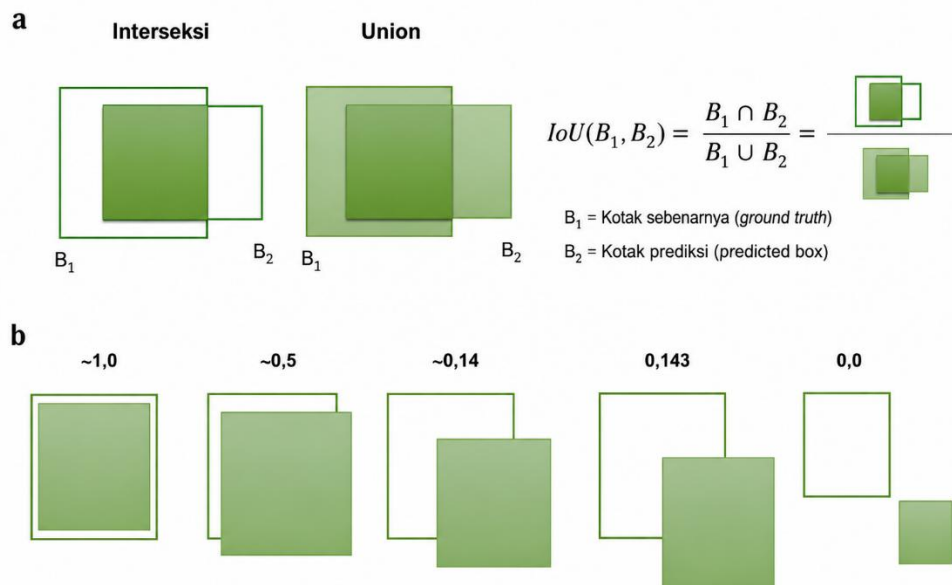


Gambar 2.13. Segmentasi semantik (kiri) dan segmentasi instans (kanan).

Segmentasi Semantik

Segmentasi semantik menerima gambar dan menghasilkan peta segmentasi di mana setiap piksel diberi label kelas, sehingga mengkategorikan objek ke dalam kategori yang telah ditentukan. Ini menggabungkan beberapa objek dari kategori yang sama (yaitu, orang yang berbeda) menjadi satu entitas (manusia). Misalnya, dalam adegan jalan mobil otonom, segmentasi dapat mengklasifikasikan kategori umum objek yang dilihat mobil, seperti pejalan kaki, sepeda, kendaraan, dan sebagainya. Peta segmentasi keluaran sesuai dengan dimensi gambar masukan (lebar dan tinggi), dengan saluran yang sesuai dengan kelas yang diprediksi. Setiap saluran memiliki masker biner yang memberi label piksel berdasarkan kelas, mengidentifikasi di mana setiap kelas berada dalam gambar masukan.

Setelah Anda mendapatkan prediksi output, bagaimana Anda tahu seberapa baik prediksi tersebut? Metrik yang umum digunakan untuk segmentasi adalah *Intersection over Union* (IoU), yang menghitung persentase tumpang tindih antara gambar yang diprediksi dan kebenaran sebenarnya dengan membagi jumlah piksel yang identik pada mask target dan prediksi dengan jumlah total piksel yang ada di masing-masing mask. Gambar 2.14 menunjukkan contoh perhitungan ini.



Gambar 2.14. Contoh perhitungan *Intersection over Union* (IoU) untuk deteksi objek. (a) IoU dihitung sebagai rasio luas irisan antara kotak pembatas ground truth dan kotak pembatas prediksi terhadap gabungannya. (b) Berbagai nilai IoU ditunjukkan untuk berbagai tingkat tumpang tindih antara kotak prediksi dan kotak ground truth.

Segmentasi Instans

Segmentasi instans mengidentifikasi objek individual dalam kategori objek, seperti orang, mobil, rumah, dan sebagainya. Misalnya, kategori seperti hewan dapat disegmentasikan lebih lanjut menjadi anjing, kucing, dan burung, dan kategori seperti anjing dapat disegmentasikan menjadi anjing 1, anjing 2, dan seterusnya. Ini dilakukan dengan mengelompokkan piksel yang termasuk dalam satu instans anjing terhadap yang lain. Segmentasi instans sangat penting untuk mobil otonom, di mana Anda ingin memiliki pemahaman detail tentang lingkungan sekitar, seperti jalanan kompleks dengan banyak pejalan kaki dan objek bergerak seperti mobil.

Bagaimana Anda mengevaluasi prediksi yang Anda dapatkan dari segmentasi instans? Salah satu caranya adalah dengan menggunakan *mean average overlap* (mAP). Sebelum melihat mAP, ada baiknya menjelaskan konsep presisi dan recall. Untuk memahami presisi dan recall, akan sangat membantu untuk menguraikan matriks kebingungan, seperti yang ditunjukkan pada Tabel 2.1:

- True positive berarti bahwa pasangan prediksi dan mask target memiliki skor IoU lebih besar dari ambang batas yang telah Anda tetapkan (biasanya 0,5 atau lebih). Ini berarti model telah berhasil memprediksi segmentasi atau mendeteksi objek berdasarkan ground truth.
- True negative terjadi ketika model tidak memprediksi label yang seharusnya tidak diprediksi (tidak ada dalam ground truth).
- False positive terjadi ketika model memprediksi objek yang tidak ada.

- False negative terjadi ketika model gagal mengidentifikasi objek yang seharusnya diidentifikasi berdasarkan kebenaran data.

Tabel 2.1. Matriks Kebingungan yang Mengilustrasikan Perbedaan Antara Presisi dan Recall.

		Aktual	
		Positif	Negatif
Prediksi	Positif	True Positif	False Positif
	Negatif	False Negatif	True Negatif

Recall mengukur jumlah objek yang diidentifikasi dengan benar dengan mengambil jumlah objek yang diidentifikasi dengan benar dan membaginya dengan jumlah objek sebenarnya. Presisi mengukur berapa banyak objek yang diidentifikasi yang benar dengan mengambil jumlah objek yang diprediksi dengan benar dan membaginya dengan jumlah total objek yang diprediksi.

$$\text{Recall} = \text{True positives} / \# \text{ ground truths}$$

$$\text{Precision} = \text{True positives} / \# \text{ predictions}$$

Presisi rata-rata dibangun di atas konsep-konsep ini dengan menghitung area di bawah kurva presisi-recall untuk mengevaluasi seberapa baik kinerja model segmentasi atau deteksi objek. Sebagai contoh, bayangkan sebuah mobil otonom yang mendeteksi mobil di jalan. Mari kita bahas ini dalam konteks interval kepercayaan, di mana interval kepercayaan yang tinggi berarti ada kepastian yang lebih besar bahwa nilai sebenarnya berada dalam rentang tersebut.

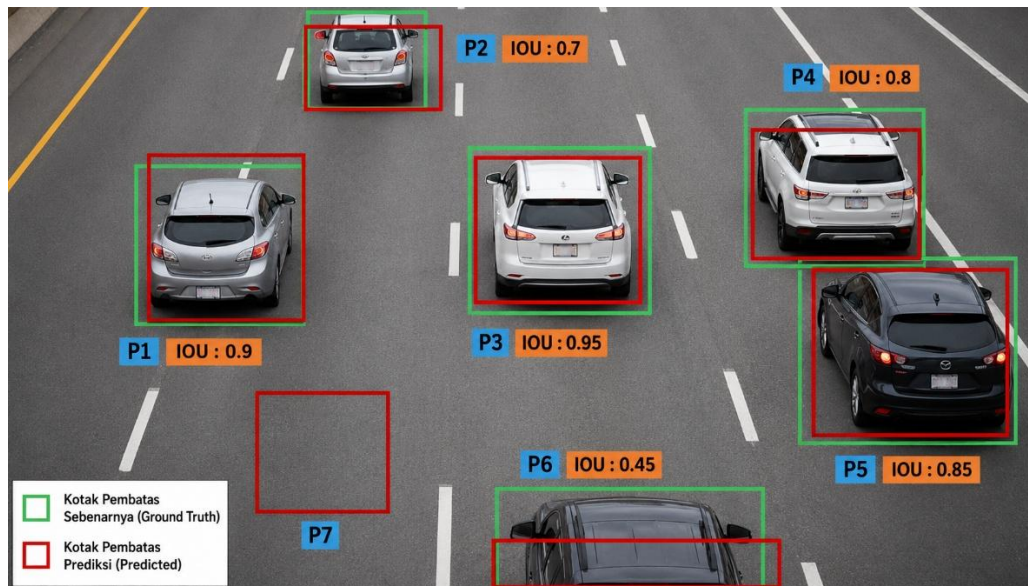
Pada interval kepercayaan yang lebih tinggi, sebuah model mungkin hanya mendeteksi beberapa mobil di jalan (presisi tinggi, recall rendah). Di sisi lain, pada interval kepercayaan yang lebih rendah, lebih banyak mobil mungkin terdeteksi tetapi banyak di antaranya bisa berupa false positive (bukan mobil sebenarnya), yang mengacu pada presisi yang lebih rendah tetapi recall yang lebih tinggi.

AP membantu Anda memahami pertukaran yang ada antara presisi dan recall dalam banyak skenario dunia nyata. mAP dibangun di atas presisi rata-rata (AP), yang mewakili area di bawah kurva presisi-recall (kurva PR). mAP hanyalah semua nilai AP yang dirata-ratakan di seluruh kelas/kategori dan berguna ketika Anda mendeteksi beberapa kelas atau objek. Harshit Kumar membuat sumber daya yang bagus jika Anda tertarik untuk mempelajari lebih dalam metrik ini dan memahami bagaimana cara perhitungannya.

Deteksi Objek

Deteksi objek mengklasifikasikan objek dalam gambar dan kemudian menentukan di mana objek tersebut berada dalam gambar menggunakan kotak pembatas. Setiap kotak

pembatas dicirikan oleh titik, lebar, tinggi, dan label kelas terkait, terkadang mengidentifikasi beberapa objek dalam gambar. Robot di lingkungan industri dan mobil otonom harus melakukan berbagai tugas seperti navigasi, pengambilan, dan penempatan, berdasarkan kemampuan mereka untuk mengenali objek. Paling umum, AP dan mAP digunakan sebagai metrik untuk mengevaluasi berbagai model deteksi objek.



Gambar 2.15. Deteksi objek pada mobil, menunjukkan kotak pembatas yang diprediksi (merah) dan kotak pembatas kebenaran (hijau) dengan skor *Intersection over Union* (IoU) untuk setiap mobil yang terdeteksi (P1 hingga P7).

Misalnya, ambil mobil otonom, di mana Anda ingin mendeteksi mobil lain di jalan raya. Output prediksi model ditunjukkan dalam kotak merah pada Gambar 2.15, sedangkan kotak hijau adalah deteksi kebenaran sebenarnya. Setiap prediksi dievaluasi berdasarkan skor IoU, yang mengukur seberapa baik kotak pembatas yang diprediksi tumpang tindih dengan kotak pembatas kebenaran sebenarnya. Skor IoU ini membantu menentukan positif sejati, positif palsu, dan negatif palsu, yang kemudian digunakan untuk menghitung presisi dan recall. Dengan menggunakan nilai presisi dan recall pada ambang batas yang berbeda, Anda dapat menghitung AP dan mAP di berbagai kelas.

2.3 GAMBARAN UMUM JARINGAN NEURAL KONVOLUSIONAL

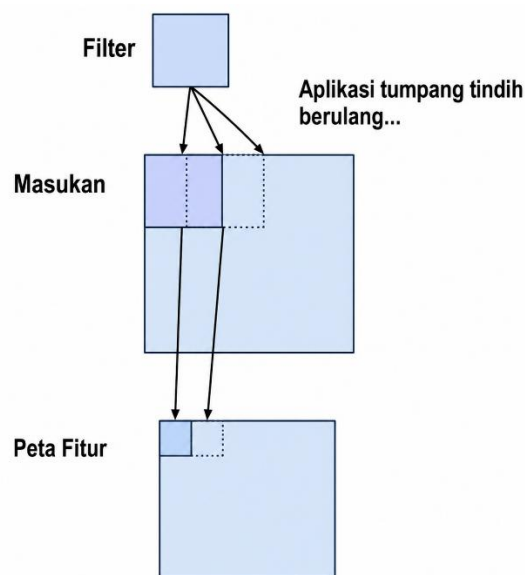
Salah satu pendekatan untuk klasifikasi gambar, segmentasi, dan deteksi objek adalah dengan menggunakan jaringan neural konvolusional (CNN). Untuk penjelasan lebih mendalam tentang cara kerja CNN, kami sarankan untuk mempelajari catatan dan tugas kuliah CS231n Stanford. Bagian ini secara singkat memberikan gambaran umum tentang CNN sebelum membahas bagaimana CNN digunakan untuk persepsi dalam robotika.

CNN adalah jaringan saraf dalam yang dapat mengklasifikasikan dan mengidentifikasi fitur spesifik dari data, seperti gambar yang dikumpulkan menggunakan sensor pada robot.

Bagian-bagian penting dari CNN adalah lapisan konvolusi, lapisan pooling, dan lapisan terhubung penuh (FC).

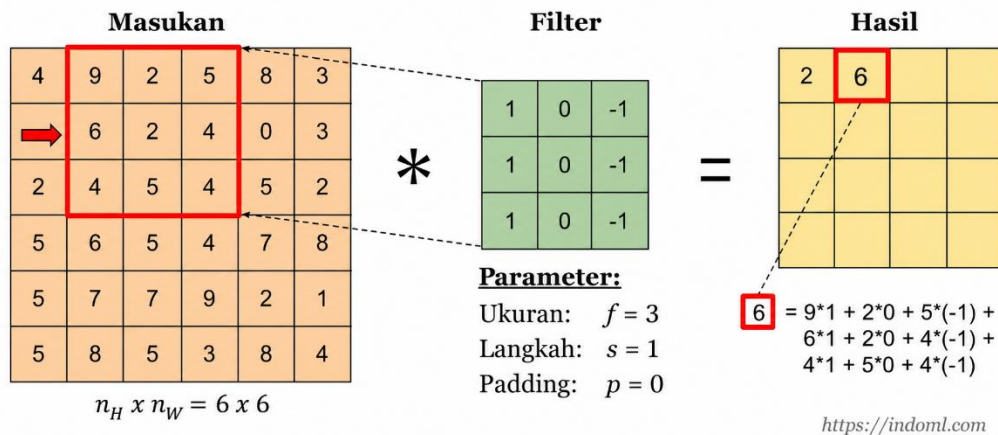
Lapisan Konvolusi

Lapisan konvolusi mengekstrak fitur gambar dengan menerapkan filter kecil (kernel) pada wilayah yang lebih kecil dari data input. Misalnya, filter 3x3 memindai gambar input 32x32x3 (tinggi, lebar, dan saluran warna) dan menangkap pola seperti tepi dan tekstur dalam bagian-bagian kecil tersebut. Saat filter bergerak melintasi gambar, ia membuat peta fitur 30x30x1 (dengan asumsi tidak ada padding dan stride 1; kita akan membahas parameter ini segera) yang mempelajari fitur seperti sudut dan tepi di setiap lokasi spasial. Gambar 2.16 menunjukkan filter yang digunakan pada input dua dimensi untuk menghasilkan peta fitur.



Gambar 2.16. Peta fitur dihasilkan dengan menerapkan filter ke gambar 2D masukan. Filter bergerak melintasi gambar, menangkap fitur-fitur penting seperti tepi dan tekstur di setiap lokasi untuk menghasilkan peta fitur akhir

Sebuah gambar dan sebuah filter, keduanya direpresentasikan sebagai matriks, pada dasarnya dikalikan pada nilai yang sesuai dan kemudian dijumlahkan untuk menghasilkan sebuah keluaran. Keluaran ini, yang dikenal sebagai peta fitur, kemudian dimasukkan ke lapisan-lapisan selanjutnya dari jaringan untuk membantu mempelajari fitur-fitur dalam gambar masukan. Gambar 2.17 memberikan gambaran visual dari proses ini.



Gambar 2.17. Contoh operasi konvolusi di mana filter 3x3 diterapkan pada gambar masukan 6x6 dengan stride 1

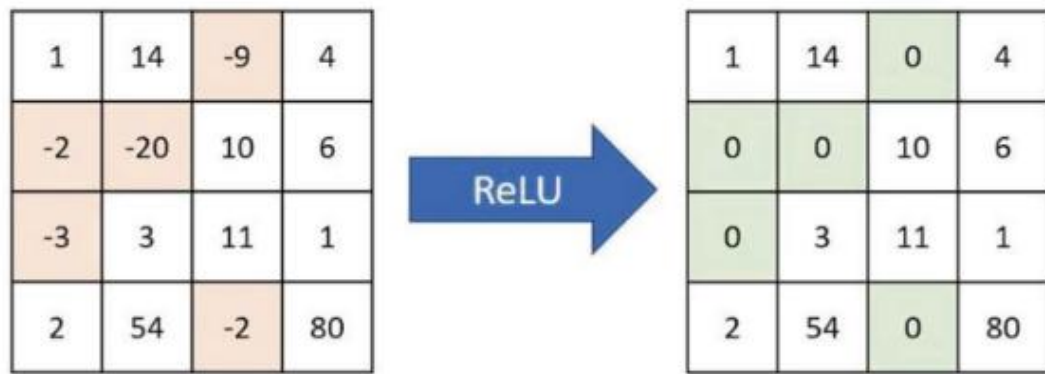
Tiga parameter menentukan ukuran peta fitur:

- **Kedalaman:** Menunjukkan berapa banyak filter yang digunakan selama proses konvolusi.
- **Stride:** Menunjukkan berapa banyak piksel yang digeser matriks filter melintasi matriks masukan di setiap langkah. Misalnya, stride 1 berarti filter bergeser satu piksel pada satu waktu. Stride yang lebih besar menghasilkan peta fitur yang lebih kecil.
- **Padding:** Menambahkan nol pada tepi matriks masukan sehingga filter dapat diterapkan pada elemen di perbatasan gambar masukan dan fitur-fitur ini dapat ditangkap.

Anda dapat mengubah nilai-nilai ini dalam matriks filter untuk menghasilkan peta fitur yang berbeda untuk gambar masukan. Dengan menumpuk lapisan konvolusi ini, mereka dapat mendeteksi berbagai pola visual secara hierarkis. Lapisan-lapisan awal membuat peta fitur yang mengidentifikasi pola sederhana, tepi, dan sudut. Lapisan-lapisan selanjutnya mulai membedakan objek yang lebih kompleks seperti hewan, furnitur, fitur wajah, dan lanskap.

Memperkenalkan Nonlinieritas (ReLU)

Setelah setiap operasi konvolusi, *Rectified Linear Unit* (ReLU) digunakan untuk meningkatkan nonlinieritas pada gambar. Fungsi ReLU terlihat seperti $f(x) = \max(x, 0)$, di mana elemen negatif diatur menjadi 0 dan elemen positif tetap sama. Secara umum, ReLU digunakan untuk mengelola informasi yang bergerak maju melalui jaringan. ReLU melakukan ini dengan mengganti semua nilai piksel negatif dalam peta fitur dengan 0. Gambaran umum proses ini ditunjukkan pada Gambar 2.18. Jenis nonlinier lain yang digunakan sebagai alternatif untuk ReLU adalah Tanh dan Sigmoid. Dalam konteks CNN, ReLU telah ditemukan berkinerja lebih baik dalam sebagian besar situasi.



Gambar 2.18. Contoh fungsi aktivasi ReLU (*Rectified Linear Unit*) yang diterapkan pada matriks. Nilai negatif dalam input diganti dengan 0, sedangkan nilai positif tetap tidak berubah. ReLU memperkenalkan nonlinier ke dalam model

Lapisan Pooling

Lapisan pooling sering ditempatkan setelah lapisan konvolusi, dengan peran utamanya adalah untuk mengurangi dimensi peta fitur yang telah dibuat sebelumnya. Mengurangi ukuran peta fitur mengurangi parameter dan kebutuhan komputasi. Jaringan juga menjadi lebih kuat terhadap perubahan kecil pada gambar masukan, seperti distorsi dan pergeseran, karena mengekstrak nilai maksimum atau rata-rata dalam area tertentu. Pooling dapat dilakukan dengan dua cara utama: pengumpulan maksimum, di mana elemen terbesar dari bagian peta fitur dipilih, dan penggabungan rata-rata, yang menghitung rata-rata setiap bagian dengan ukuran yang telah ditentukan dalam gambar.

Lapisan Terhubung Sepenuhnya (*Fully Connected Layers*)

Output dari lapisan pooling dan konvolusi terakhir diubah menjadi vektor yang diratakan dengan membuka matriks 3D menjadi satu dimensi. Vektor ini biasanya merangkum fitur tingkat tinggi dari gambar masukan, yang kemudian berfungsi sebagai masukan ke lapisan terhubung sepenuhnya (FC). Peran lapisan FC adalah untuk memanfaatkan fitur-fitur ini untuk mengklasifikasikan gambar masukan ke dalam berbagai kategori, sebagaimana ditentukan oleh data pelatihan. Lapisan FC umumnya ditempatkan tepat sebelum lapisan keluaran.

2.4 CNN UNTUK PERSEPSI

Bagian ini menjelaskan contoh sederhana robot pembersih yang menggunakan CNN untuk mendeteksi objek di atas meja dan kemudian membersihkan meja. Seperti yang ditunjukkan pada Gambar 2.19, setiap objek dalam gambar telah ditemukan dan diidentifikasi dengan tingkat akurasi tertentu.



Gambar 2.19. Contoh CNN yang mendeteksi berbagai objek di atas meja

Salah satu cara umum untuk menyelesaikan masalah ini menggunakan CNN adalah sebagai berikut:

1. Ambil gambar meja dengan barang-barang yang ingin kita bersihkan/atur menggunakan sensor.
2. Masukkan gambar ini ke dalam CNN.
3. Bagi gambar menjadi beberapa bagian yang lebih kecil, yang dikenal sebagai wilayah atau patch, dan perlakukan setiap wilayah sebagai gambar terpisah.
4. Berikan semua wilayah ke CNN dan klasifikasikan isi setiap wilayah ke dalam berbagai kelas.
5. Setelah menetapkan setiap wilayah yang dibagi ke kelasnya masing-masing, semua wilayah ini dapat digabungkan untuk membuat ulang gambar asli, yang sekarang dianotasi dengan objek yang diidentifikasi.

Meskipun metode ini pada awalnya tampak efektif, metode ini menghadapi tantangan karena objek dalam gambar memiliki rasio aspek dan lokasi yang bervariasi. Misalnya, beberapa objek mungkin memenuhi sebagian besar gambar, sementara yang lain muncul sebagai bagian kecil.

Variabilitas ini berarti sejumlah besar wilayah akan dibutuhkan, yang sangat meningkatkan kebutuhan komputasi. Untuk mengatasi hal ini, CNN berbasis wilayah (R-CNN) dapat digunakan, yang menggunakan metode proposal untuk memilih wilayah yang lebih sedikit, tetapi lebih relevan. Bagian selanjutnya menjelaskan bagaimana R-CNN bekerja untuk menyederhanakan proses ini.

R-CNN

Alih-alih memeriksa banyak wilayah, R-CNN menyarankan wilayah potensial dalam sebuah gambar untuk menentukan apakah wilayah tersebut mengandung objek. Metode ini menggunakan pencarian selektif, yang secara rekursif mengelompokkan wilayah serupa berdasarkan warna, tekstur, dan ukuran, atau kotak tepi, yang menghasilkan kotak pembatas objek berdasarkan tepi dalam gambar untuk mengidentifikasi wilayah yang diminati (RoI). Fitur

dari setiap RoI kemudian diekstrak menggunakan CNN yang telah dilatih sebelumnya. Setelah itu, pengklasifikasi digunakan untuk mengklasifikasikan objek untuk setiap RoI.

Salah satu keterbatasan signifikan dari model R-CNN adalah lambat dan membutuhkan banyak komputasi. Model ini menghasilkan dan memproses ribuan (~2.000) wilayah berbeda per gambar. Untuk aplikasi seperti robot pembersih, jika R-CNN membutuhkan waktu puluhan detik untuk menganalisis gambar baru, maka akan terlalu lambat untuk implementasi waktu nyata. Untungnya, ada model yang disebut Fast R-CNN, yang bertujuan untuk menyelesaikan beberapa masalah pada model R-CNN.

Fast R-CNN

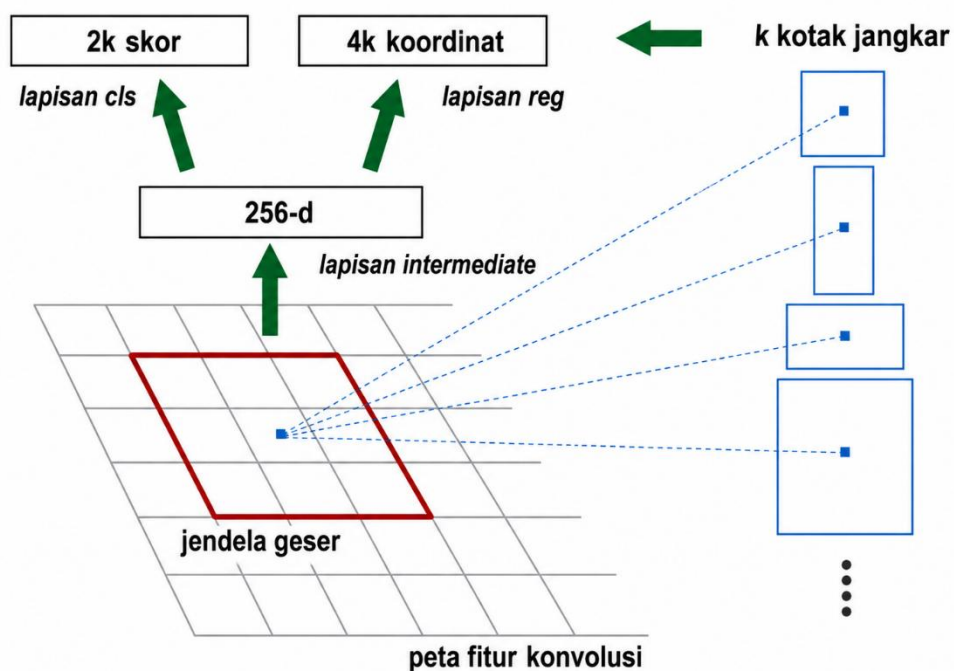
Seperti yang disebutkan, keterbatasan utama yang memengaruhi kinerja R-CNN adalah komputasi yang dilakukan untuk setiap proposal wilayah. Dalam banyak skenario dunia nyata, wilayah sering tumpang tindih dan ekstraksi fitur secara independen untuk masing-masing wilayah dapat mengakibatkan komputasi yang berlebihan. Misalnya, dalam aplikasi robotika di mana robot menavigasi lingkungan yang berantakan, beberapa wilayah yang tumpang tindih mungkin berisi bagian-bagian dari berbagai objek seperti kotak, furnitur, orang, dan sebagainya. Fast R-CNN mengatasi inefisiensi R-CNN dengan memproses seluruh gambar sekaligus untuk ekstraksi fitur, alih-alih menangani proposal wilayah individual secara terpisah. Berikut cara kerjanya:

- Seluruh gambar dimasukkan ke dalam CNN, menghasilkan peta fitur konvolusional. Anda dapat memasukkan gambar 512x512 dari kamera robot, yang kemudian menghasilkan peta fitur dengan resolusi spasial yang lebih rendah. Peta ini menangkap informasi kunci dari gambar, seperti tepi, pola, dan bentuk, yang dibutuhkan untuk deteksi atau segmentasi objek.
- Peta ini digunakan untuk mengidentifikasi ROI, yaitu area di mana objek kemungkinan besar ada. Katakanlah jaringan menemukan ROI yang berisi berbagai kotak, barang furnitur, dan sebagainya. Setiap ROI kemudian diubah ukurannya melalui lapisan pooling ROI menjadi ukuran tetap yang dapat diteruskan ke jaringan yang terhubung sepenuhnya. Peta fitur ini kemudian dimasukkan ke dalam jaringan saraf yang terhubung sepenuhnya di mana objek di setiap RoI diklasifikasikan dan regresi kotak pembatas digunakan untuk memperbaiki lokasi kotak pembatas.
- Dengan menggunakan peta fitur yang sama untuk semua proposal objek, Fast R-CNN secara signifikan meningkatkan efisiensi deteksi objek, tidak seperti R-CNN asli, yang memproses setiap wilayah secara independen.

Meskipun Fast R-CNN berkinerja lebih baik daripada R-CNN, ia masih bergantung pada pencarian selektif untuk menemukan RoI, yang merupakan proses yang lambat dan memakan waktu. Robot pembersih kemungkinan akan bekerja dengan kumpulan data yang besar dan membutuhkan model yang dapat mendeteksi setiap item di atas meja dengan sangat cepat. Untuk mengatasi masalah ini, iterasi lain dari R-CNN, yang disebut Faster R-CNN, dikembangkan.

Faster R-CNN

Faster R-CNN meningkatkan kecepatan deteksi objek dibandingkan dengan Fast R-CNN. Fitur utama yang membedakan Faster R-CNN dari model lain adalah penyertaan Jaringan Proposal Wilayah (RPN). Ditempatkan setelah lapisan konvolusi terakhir, RPN menghasilkan proposal objek secara langsung, menghilangkan kebutuhan akan pencarian selektif. Ia menggeser jendela, seperti lapisan konvolusi 3x3, di sepanjang peta fitur dari CNN, menciptakan k kotak jangkar dengan berbagai bentuk dan ukuran di setiap posisi. RPN kemudian memperkirakan kemungkinan setiap jangkar berisi objek dan memperbaiki koordinat agar lebih sesuai dengan objek yang terdeteksi. Gambaran umum cara kerja RPN ditunjukkan pada Gambar 2.20.



Gambar 2.20. Proses bagaimana RPN bekerja

Secara keseluruhan, Faster R-CNN kira-kira sepuluh kali lebih cepat daripada Fast R-CNN sambil mempertahankan akurasi yang serupa pada dataset seperti VOC-2007. Hal ini menjadikan Faster R-CNN sebagai algoritma pilihan untuk deteksi objek.

Tabel 2.2 memberikan perbandingan cepat dari versi R-CNN ini.

Tabel 2.2. Perbandingan R-CNN, Fast R-CNN, dan Faster R-CNN pada Kecepatan dan Waktu Pengujian Per Gambar

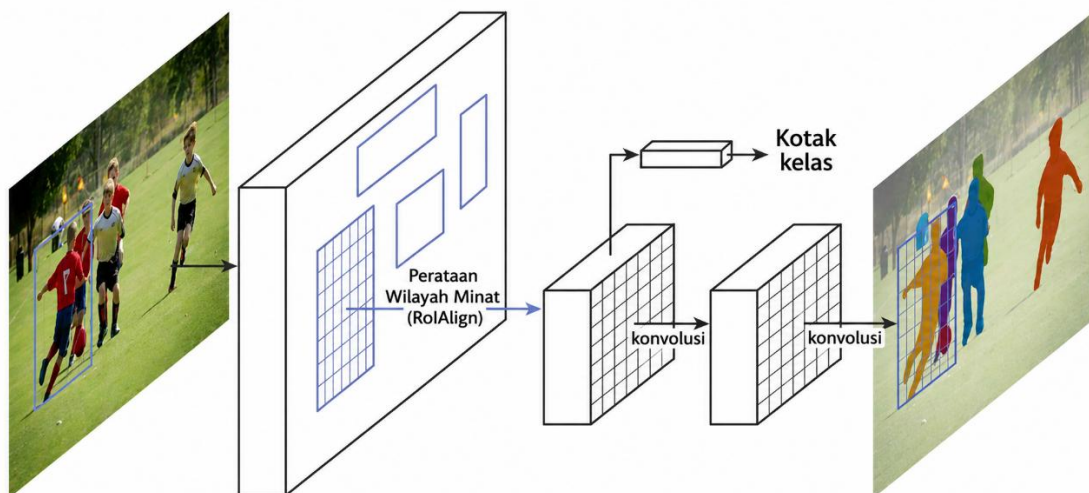
	R-CNN	Fast R-CNN	Faster R-CNN
Waktu Pengujian per Gambar	50 Detik	2 Detik	0,2 Detik
Peningkatan Kecepatan	1x	25x	250x

Mask R-CNN

Mask R-CNN adalah perluasan dari model R-CNN yang memungkinkan klasifikasi tingkat piksel, sehingga berguna untuk tugas segmentasi gambar. Sementara model R-CNN lainnya—seperti R-CNN, Fast R-CNN, dan Faster R-CNN—terutama dirancang untuk deteksi objek dan prediksi bounding box, Mask R-CNN menambahkan cabang untuk memprediksi mask segmentasi. Ini berarti ia tidak hanya mendeteksi objek tetapi juga menyediakan mask piksel demi piksel yang detail untuk setiap objek yang terdeteksi. Ini telah diterapkan di berbagai domain, seperti segmentasi objek, pengukuran jarak untuk pengambilan robot, dan deteksi kendaraan di mobil otonom. Ini juga bisa berguna untuk robot pembersih jika Anda ingin mendapatkan granularitas yang lebih besar dan mensegmentasi semua objek yang akan dibersihkan dari gambar.

Mask R-CNN memperluas Faster R-CNN dengan menambahkan cabang segmentasi, jaringan konvolusional penuh yang menghasilkan mask biner untuk setiap ROI, yang memprediksi piksel tepat yang termasuk dalam setiap objek. CNN digunakan untuk menghasilkan peta fitur dari gambar masukan. RPN kemudian mengidentifikasi ROI yang paling mungkin mengandung objek. RoI Align digunakan untuk mencocokkan ROI dari gambar dengan peta fitur. Pencocokan yang tepat dapat memastikan lokasi dan bentuk objek yang tepat dipertahankan, yang dapat berguna untuk menghasilkan masker segmentasi. Setiap ROI kemudian diproses oleh tiga cabang. Cabang klasifikasi mengidentifikasi objek di setiap ROI. Cabang regresi bounding box digunakan untuk memperbaiki koordinat bounding box. Terakhir, cabang masker menghasilkan masker biner untuk setiap ROI. Ilustrasi Mask R-CNN disajikan pada Gambar 2.21.

Fungsi kerugian yang digunakan dalam Mask R-CNN mencakup tiga elemen: kerugian klasifikasi, yang menilai akurasi prediksi relatif terhadap kelas sebenarnya; kerugian bounding box, yang mengevaluasi akurasi lokalisasi model; dan kerugian prediksi masker, yang ditentukan dengan menghitung entropi silang biner antara masker yang diprediksi dan kebenaran sebenarnya.



Gambar 2.21. Komponen utama Mask R-CNN.

2.5 RESNET DAN U-NET UNTUK PERSEPSI ROBOT

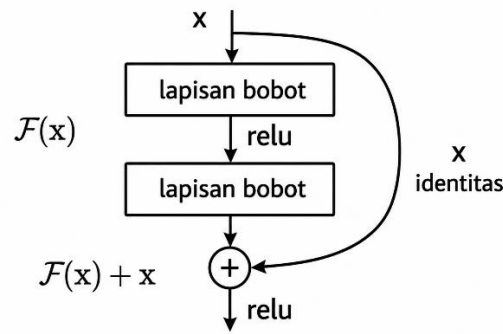
Anda telah membaca tentang betapa pentingnya mengoptimalkan arsitektur model-model ini untuk mempercepat komputasi tugas persepsi. Tetapi bagaimana dengan meningkatkan akurasi model-model ini untuk melakukan tugas persepsi dengan lebih baik? Misalnya, jika robot digunakan di lingkungan seperti operasi, robot tersebut perlu mampu mendeteksi atau mengklasifikasikan objek dengan presisi yang sangat tinggi.

Pertanyaan kunci dalam pembelajaran mendalam berfokus pada pemahaman apakah hanya dengan menumpuk lebih banyak lapisan akan meningkatkan kinerja model. Alasannya adalah bahwa lapisan tambahan ini secara progresif menangkap fitur yang lebih kompleks. Namun, telah diamati bahwa di luar kedalaman jaringan tertentu, akurasi mencapai titik jenuh dan kemudian dengan cepat menurun. Model CNN tradisional memiliki kedalaman efektif maksimum, di luar itu mereka mengalami masalah gradien yang menghilang—selama back-propagation, gradien dapat menjadi sangat kecil karena perkalian berulang, hampir menghilangkannya.

Jaringan Neural Residual (ResNet) diusulkan oleh He et al. pada tahun 2015 sebagai terobosan yang memecahkan masalah gradien yang menghilang dan memungkinkan orang untuk membangun CNN yang lebih besar yang memiliki peningkatan kinerja dalam tugas persepsi. Karena alasan ini, ResNet digunakan sebagai tulang punggung saat ini untuk banyak tugas visi komputer. Mereka tidak hanya terbatas pada klasifikasi gambar tetapi juga dapat memecahkan berbagai masalah dalam segmentasi gambar dan deteksi objek—bahkan dalam robotika.

Koneksi Lewati: Kekuatan ResNet

Aspek kunci dari ResNet terletak pada koneksi lewati, yang telah menjadi komponen fundamental dalam banyak arsitektur konvolusional. Koneksi lewati menawarkan jalur alternatif untuk aliran gradien dalam backpropagation, memungkinkan output dari suatu lapisan untuk masuk ke lapisan berikutnya. Ini dilakukan melalui penjumlahan vektor, di mana output dari suatu lapisan ditambahkan langsung ke output dari lapisan yang lebih dalam. Penjumlahan ini menciptakan jalan pintas untuk gradien selama backpropagation, memungkinkannya untuk melewati lapisan tertentu. Dengan mengalikan gradien dengan 1 (fungsi identitas), koneksi lewati memastikan bahwa gradien tidak berkurang dan mempertahankan nilai aslinya. ResNet dibangun dengan menumpuk blok residual yang menggabungkan koneksi lewati. Gambar 2.22 mengilustrasikan satu blok residual dengan koneksi lewati.

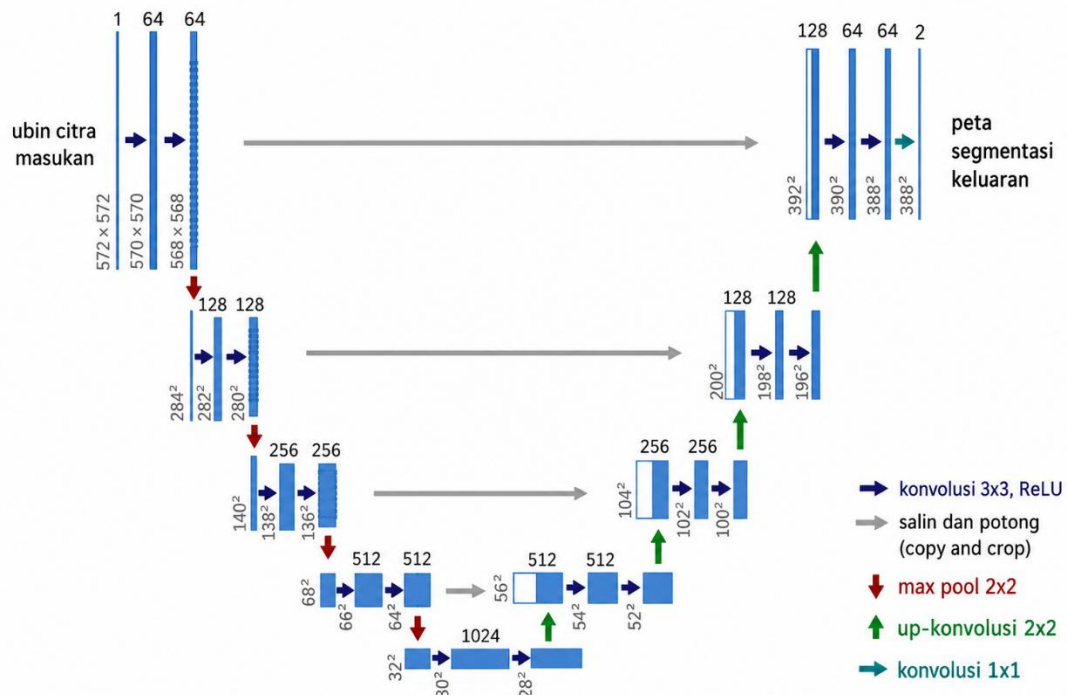


Gambar 2.22. Sebuah blok residual

U-Net

Konsep skip connection telah memengaruhi desain arsitektur seperti U-Net, yang awalnya dikembangkan oleh Ronneberger dkk. untuk segmentasi gambar biomedis. U-Net kini diterapkan pada tugas segmentasi gambar lainnya, termasuk persepsi robotika. Arsitektur ini merupakan struktur encoder-decoder dengan skip connection, yang dikenal efisien dengan jumlah gambar pelatihan yang lebih sedikit dan segmentasi yang sangat akurat. U-Net dapat melakukan segmentasi gambar hingga 512x512 piksel menggunakan GPU modern (menurut tahun 2015) dalam waktu kurang dari satu detik.

Seperti yang diilustrasikan pada Gambar 2.23, aspek khas U-Net adalah penggantian operasi pooling dengan operator upsampling, yang meningkatkan resolusi output. Encoder menggunakan lapisan konvolusi dan pooling untuk menangkap fitur dan mengurangi dimensi spasial gambar input, menciptakan representasi terkompresi dari gambar input. Dekoder kemudian menggunakan operator upsampling untuk meningkatkan resolusi output dari encoder, menyempurnakan fitur yang ditangkap oleh encoder menjadi output tersegmentasi beresolusi tinggi. U-Net juga menggabungkan skip connection, yang secara langsung mentransfer peta fitur dari encoder ke lapisan yang sesuai di dekoder. *Skip connection* ini membantu mempertahankan informasi spasial penting yang mungkin hilang selama proses *downsampling*.



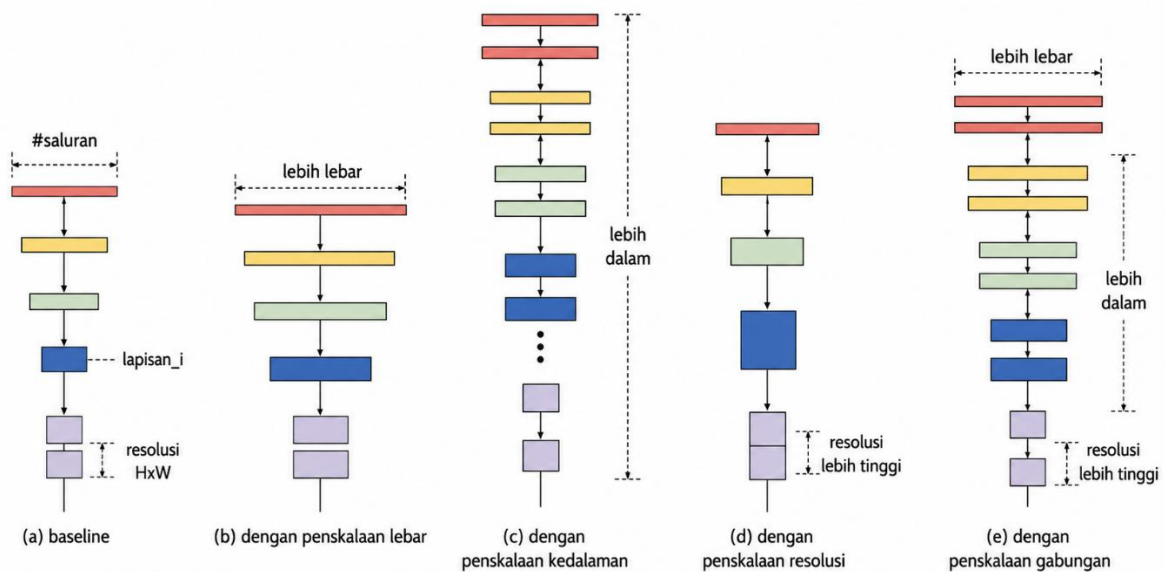
Gambar 2.23. Arsitektur U-Net

U-Net telah umum digunakan dalam tugas segmentasi untuk robotika mengingat kecepatannya dan kemampuan struktur encoder-decoder untuk mempelajari informasi spasial yang akurat. Misalnya, telah digunakan dalam studi untuk segmentasi instrumen robot untuk operasi robotik, deteksi jalur untuk mobil otonom, dan untuk pengambilan benda oleh robot.

U-Net terkenal karena membentuk tulang punggung beberapa model difusi awal dan bahkan Difusi Stabil. Sifat penting dari U-Net adalah bahwa input dan output memiliki ukuran yang sama, artinya model mempertahankan dimensi spasial input di seluruh jaringan. Desain ini memungkinkan U-Net untuk mempelajari dan mengkodekan fitur gambar terperinci (pengkodean laten) di dalam encoder, yang kemudian dapat digunakan oleh beberapa decoder atau model lain. Arsitektur terbaru menggunakan transformer sebagai pengganti CNN untuk membuat U-Net yang berjalan lebih cepat. Meskipun terjadi perubahan modul, konsep fundamental U-Net tetap berlaku.

2.6 EFFICIENTNET DAN PENSKALAAN CNN

Salah satu cara CNN dapat ditingkatkan skalanya adalah dengan menambahkan lebih banyak lapisan. Seringkali, Anda dapat menambahkan lebih banyak lapisan seperti pada ResNet dan kemudian secara manual membuat jaringan lebih dalam dan lebih lebar, atau menggunakan gambar beresolusi lebih tinggi. EfficientNet menggunakan metode penskalaan gabungan yang menyesuaikan ketiga dimensi secara bersamaan berdasarkan faktor penskalaan yang telah ditentukan sebelumnya.

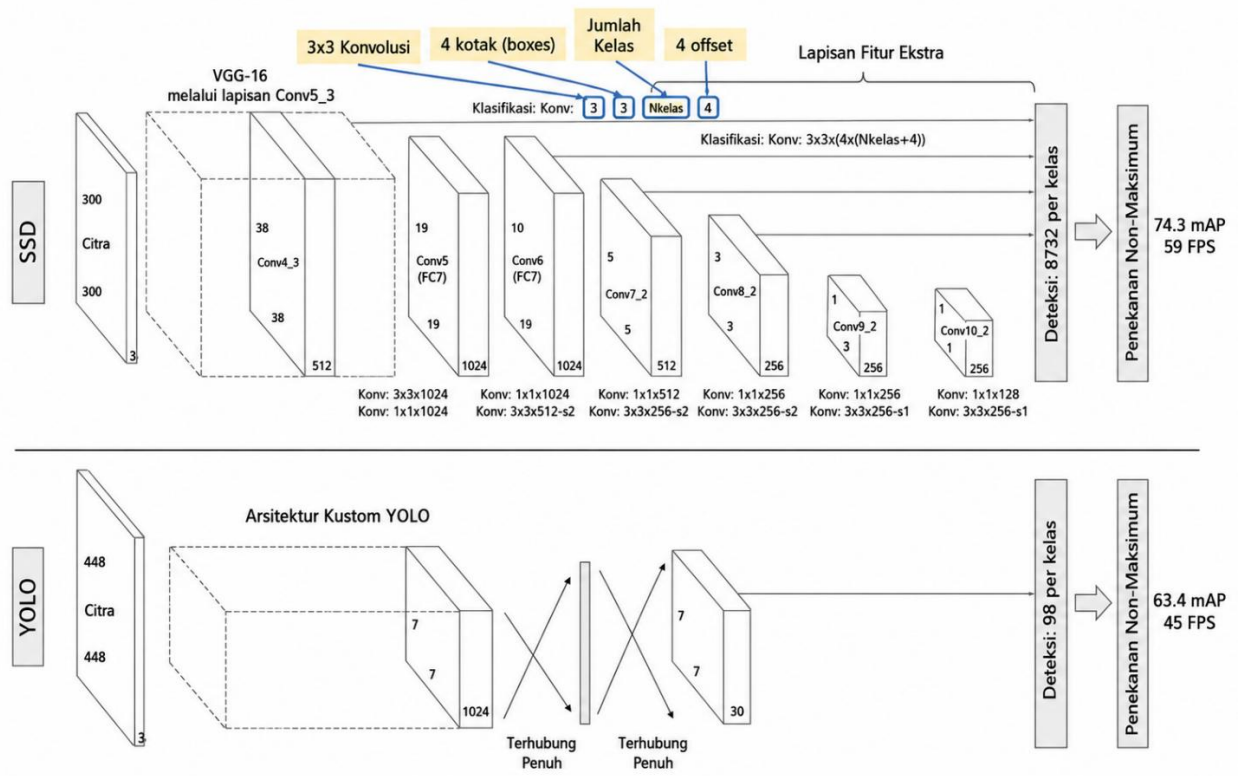


Gambar 2.24. Perbandingan metode penskalaan

Dengan menggunakan koefisien yang ditentukan pengguna dan konstanta penskalaan tetap, metode ini memastikan bahwa ada pertumbuhan proporsional di ketiga faktor tersebut. Misalnya, jika Anda ingin menggandakan sumber daya komputasi, Anda dapat meningkatkan kedalaman 1,2 kali, lebar 1,1 kali, dan resolusi 1,15 kali. Parameter yang ditentukan secara empiris ini membantu meningkatkan fitur yang ditangkap CNN dengan memungkinkan mereka mempelajari pola yang lebih kompleks (kedalaman), menangkap detail yang lebih granular (lebar), dan mendeteksi objek yang lebih kecil (resolusi). Gambar 2.24 menunjukkan perbandingan aspek penskalaan. Mereka juga mempertahankan efisiensi komputasi dengan mengoptimalkan secara holistik penggunaan semua sumber daya dan fitur yang tersedia. Pendekatan ini memungkinkan EfficientNet untuk mencapai akurasi dan efisiensi yang lebih tinggi, membuat jaringan hingga sepuluh kali lebih kecil namun lebih cepat dibandingkan dengan metode penskalaan tradisional.

2.7 YOLO DAN SSD UNTUK DETEKSI OBJEK

Banyak algoritma deteksi objek yang disebutkan sejauh ini menangani deteksi objek sebagai masalah klasifikasi dengan terlebih dahulu menghasilkan proposal objek dan kemudian mengirimkan proposal ini ke kepala klasifikasi/regresi atau segmentasi. Bagaimana jika ada cara untuk melihat seluruh gambar sekaligus daripada hanya melihat proposal wilayah yang dihasilkan? Metode ini akan mampu menghitung dengan sangat cepat dan dapat dijalankan secara real-time. Sebuah metode yang disebut *You Only Look Once* (YOLO) bertujuan untuk melakukan ini dengan membuat prediksi kotak pembatas dan probabilitas kelas secara bersamaan.



Gambar 2.26. Arsitektur SSD (atas) dan YOLO (bawah)

YOLO dan SSD sama-sama dikenal cepat. Namun, SSD dikenal memberikan akurasi yang lebih baik untuk objek dengan ukuran yang bervariasi karena peta fitur multiskala dan bidang reseptifnya. Perbandingan kinerja SSD, YOLO, dan Faster R-CNN ditunjukkan pada Gambar 2.27.

2.8 EVALUASI KINERJA MODEL DETEKSI OBJEK

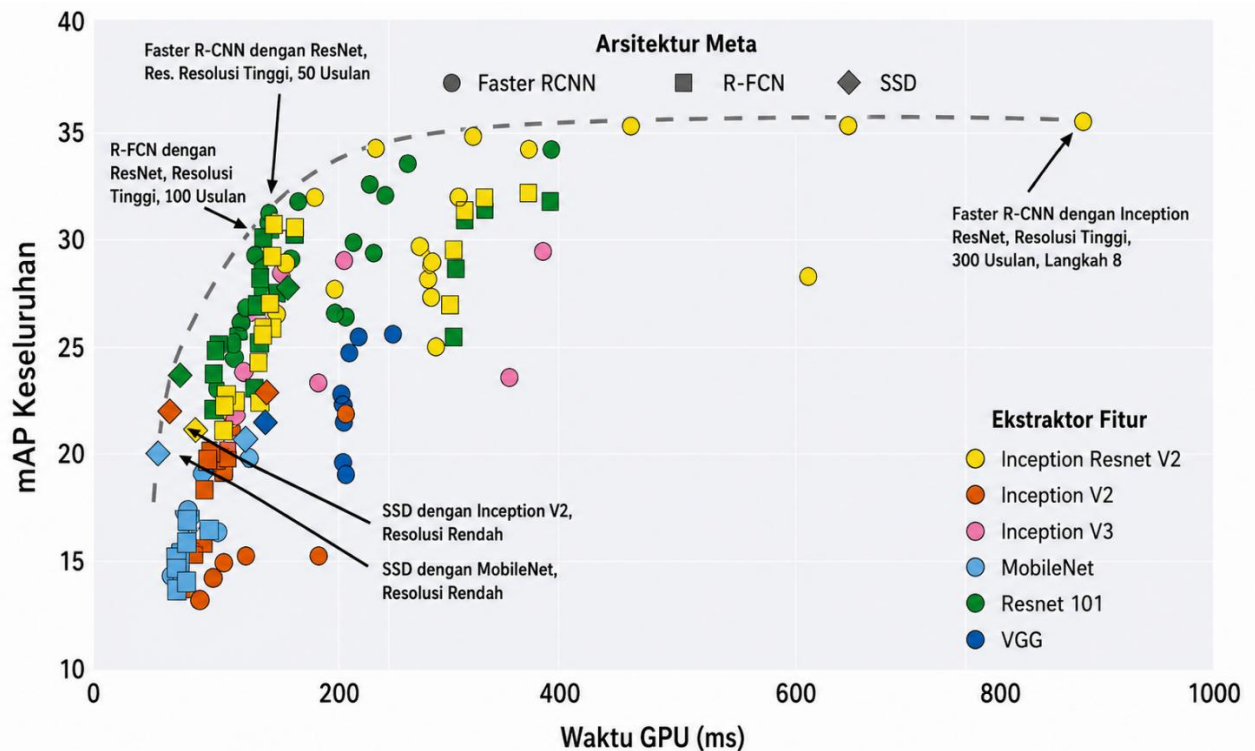
Memilih model yang tepat sangat penting untuk robotika dan bergantung pada masalah yang ingin Anda selesaikan dan bagaimana Anda mengaturnya. Hasil Faster R-CNN, YOLO, dan SSD berdasarkan mAP dan FPS pada Pascal VOC2007 ditunjukkan pada Gambar 2.27. Gambar masukan dari dataset ini diuji dengan resolusi yang berbeda untuk membandingkan hasilnya. Penting untuk dicatat bahwa versi YOLO yang lebih baru (YOLO v9, v10), yang memiliki arsitektur yang lebih baik, memiliki mAP dan FPS yang jauh lebih tinggi, tetapi kami telah memberikan hasil dari studi ini sebagai tolok ukur.

Metode	mAP	FPS	Ukuran Batch	# Kotak	Resolusi Input
Faster R-CNN (VGG16)	73,2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52,7	155	1	98	448 × 448
YOLO (VGG16)	66,4	21	1	98	448 × 448
SSD300	74,3	46	1	8732	300 × 300
SSD512	76,8	19	1	24564	512 × 512
SSD300	74,3	59	8	8732	300 × 300

SSD512	76,8	22	8	24564	512 × 512
--------	------	----	---	-------	-----------

Gambar 2.27. Perbandingan Faster R-CNN, YOLO, dan SSD berdasarkan presisi rata-rata (mAP) dan FPS, pada dataset Pascal VOC2007.

Dalam contoh lain, sebuah makalah oleh *Google Research* meneliti berbagai model deteksi objek dan trade-off antara kecepatan (waktu GPU) dan akurasi (mAP). Mereka umumnya menemukan bahwa model R-FCN dan SSD lebih cepat, karena membutuhkan lebih sedikit waktu GPU per gambar, tetapi cenderung memiliki akurasi yang lebih rendah (mAP). Di sisi lain, model Faster R-CNN cenderung lebih lambat tetapi mencapai akurasi yang lebih tinggi. Untuk model Faster R-CNN, kecepatan juga dapat ditingkatkan dengan mengurangi jumlah wilayah yang diusulkan selama deteksi.



Gambar 2.28. Kompromi berbagai model deteksi objek dalam hal waktu GPU vs mAP.

Penting untuk dicatat bahwa kombinasi berbeda dari ekstraktor fitur (misalnya, Inception Resnet V2, MobileNet) dan arsitektur (misalnya, Faster R-CNN, SSD, dan sebagainya) menghasilkan manfaat kinerja yang berbeda, seperti yang ditunjukkan pada Gambar 2.28. Kompromi ini akan berbeda tergantung pada masalah yang Anda kerjakan dan dataset Anda. Kompromi ini sangat penting untuk dipertimbangkan, terutama untuk aplikasi deteksi waktu nyata dalam robotika.

2.9 TRANSFORMER UNTUK PERSEPSI

Bagian sebelumnya membahas teknik visi berbasis CNN. Namun, akhir-akhir ini, teknik visi berbasis transformer telah memberikan alternatif yang kuat untuk memecahkan masalah

visi yang dibahas dalam bab ini. Transformer, berbeda dengan CNN, mengandalkan mekanisme self-attention untuk memodelkan koneksi global di antara elemen visual, sehingga meningkatkan pemahaman model tentang informasi kontekstual. Dalam robotika, aplikasi seperti deteksi objek, segmentasi gambar, dan penalaran visual—di mana konteks global dan interaksi yang halus sangat penting untuk pengambilan keputusan yang tepat—sangat cocok untuk transformer.

Pendahuluan Transformer

Untuk memahami bagaimana model sekuens dapat diterapkan pada robotika, Anda perlu memahami terlebih dahulu perhatian, yang merupakan elemen jaringan saraf yang secara eksplisit memotivasi jaringan untuk fokus pada bagian-bagian tertentu dari data input dan mengabaikan yang lain. Dalam sekuens yang panjang, membuka jaringan seiring waktu berarti input yang lebih lama dilupakan. Perhatian menyediakan jalur langsung ke input yang lebih lama, sehingga mengurangi masalah gradien yang menghilang/meledak.

Mekanisme perhatian pertama kali diusulkan oleh Bahdanau. sebagai bobot penyematan untuk menyelaraskan dan menerjemahkan secara bersamaan selama tugas penerjemahan mesin saraf. Sebelum karya yang diusulkan oleh Bahdanau. penerjemahan saraf melibatkan pengaturan encoder dan decoder di mana encoder mengubah input menjadi representasi penyematan dan decoder mengubah penyematan kembali menjadi token dalam bahasa target. Dalam Bahdanau jaringan feedforward sederhana digunakan untuk menghitung skor penyelarasan antara token input dan output. Skor ini digunakan untuk menimbang vektor konteks decoder RNN. Hal ini kemudian dikenal sebagai perhatian aditif karena umpan maju menerapkan fungsi linier dan aditif pada input.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Persamaan 2.1

Dalam Persamaan 2.1, α_{ij} adalah bobot yang dipelajari yang diberikan pada input ke- j untuk menghitung output ke- i . Mekanisme ini disebut perhatian lunak (*soft attention*). Sejak saat itu, beberapa mekanisme perhatian telah diusulkan:

- **Perhatian berbasis konten:** Metode ini menggunakan kesamaan kosinus antara keadaan target dan keadaan tersembunyi sumber.
- **Perhatian aditif:** Menggunakan vektor bobot yang dapat dilatih, diikuti oleh aktivasi tanh atas kombinasi atau penggabungan keadaan target dan keadaan tersembunyi sumber.
- **Perhatian umum:** Matriks bobot yang dapat dilatih diterapkan pada keadaan tersembunyi sumber dan hasil perkalian titik (*dot product*) diambil dengan keadaan target.
- **Perhatian perkalian titik (*dot product attention*):** Hasil perkalian titik antara keadaan target dan keadaan tersembunyi sumber dihitung.

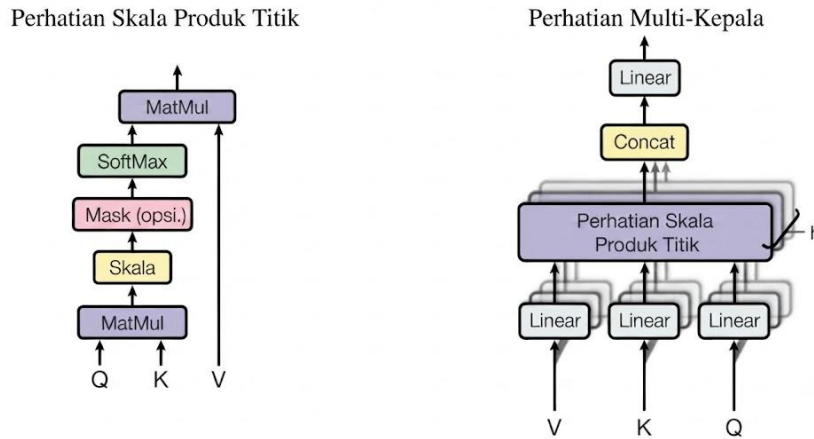
- **Perhatian produk titik berskala:** Mirip dengan perhatian produk titik, tetapi skor diskalakan dengan akar kuadrat dari dimensi n dari keadaan tersembunyi sumber.
- **Perhatian silang:** Skor perhatian dihitung sebagai produk titik antara kueri Q (dari satu urutan input) dan kunci K (dari urutan input yang berbeda), diskalakan dengan akar kuadrat dari dimensi kunci dibagi dengan jumlah kepala, h . Setelah menerapkan fungsi softmax pada skor ini, skor tersebut digunakan untuk menimbang nilai V . Teknik ini dapat digunakan untuk pengkondisian pada beberapa aliran input.
- **Perhatian kilat:** Untuk perhatian kilat, pembagian matriks perhatian yang besar menjadi blok-blok yang lebih kecil yang sesuai dengan memori cepat (SRAM). Setiap blok diproses secara independen, menjaga data perantara tetap lokal untuk menghindari pembacaan/penulisan yang sering ke memori yang lebih lambat (HBM). Hal ini mengurangi beban memori dan mempercepat komputasi, sekaligus memastikan output softmax yang benar dihitung di seluruh urutan.

Transformer

Arsitektur jaringan saraf penting yang didasarkan pada kemampuan untuk memanfaatkan perhatian pada urutan panjang adalah transformer. Diperkenalkan dalam Vaswani berjudul *"Attention Is All You Need,"* transformer telah mengubah lanskap pembelajaran mendalam dengan memberikan kinerja yang lebih baik daripada manusia dalam ucapan dan penglihatan. Model bahasa terbaik di dunia, termasuk GPT-3 dan PaLM, adalah transformer.

Selain diterapkan pada tugas-tugas seperti penerjemahan mesin, pembuatan teks, dan pemahaman bahasa, transformer dapat digunakan untuk deteksi dan pelacakan objek dalam sistem penglihatan robot. Dengan menggunakan mekanisme self-attention, transformer dapat menangkap ketergantungan global dan hubungan spasial antara berbagai wilayah, yang mengarah pada peningkatan kemampuan pengenalan dan pelacakan objek. Kasus penggunaan lain dalam robotika termasuk perencanaan jalur dan tugas navigasi dalam robotika di mana transformer dapat digunakan untuk menangkap ketergantungan jarak jauh dan menggunakan informasi kontekstual, meningkatkan kemampuan robot untuk menavigasi lingkungan yang kompleks dan dinamis.

Vaswani et al. mengkategorikan jaringan perhatian secara umum sebagai pemetaan sekumpulan kueri dan pasangan kunci-nilai ke output di mana semuanya adalah vektor. Gambar 2.29 menunjukkan bahwa output adalah jumlah tertimbang dari semua nilai, dan bobot dihitung sebagai hasil perkalian titik vektor kueri dan kunci.



Gambar 2.29. Perhatian produk titik berskala dan perhatian multi-kepala

Dalam perhatian produk titik berskala, perhatian dihitung menggunakan Persamaan 2.2:

$$Attention(Q, K, V) = softmax \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

Persamaan 2.2

di mana Q adalah kueri, K adalah kunci, V adalah vektor nilai, dan d_k adalah dimensi vektor kunci. Dibandingkan dengan perhatian aditif yang diperkenalkan dalam Bahdanau, perhatian produk titik lebih cepat dihitung dan efisien dalam penggunaan ruang. Dalam perhatian produk titik berskala, $1/\sqrt{d_k}$ menjaga nilai input dari nilai softmax tetap dinormalisasi, karena untuk nilai d_k yang besar, produk titik didorong ke area di mana fungsi softmax memiliki gradien kecil.

Makalah transformer asli menggunakan perhatian produk titik berskala dan perhatian multi-kepala. Perhatian multi-kepala digunakan untuk memproyeksikan nilai kueri dan kunci ke dalam beberapa kepala/subruang representasi untuk menerapkan perhatian produk titik berskala di atasnya. Dalam perhatian multi-kepala, Anda melakukan paralelisasi dengan terlebih dahulu memproyeksikan nilai, kunci, dan kueri ke array berdimensi d_v , d_k , dan d_q , masing-masing, kemudian menerapkan perhatian produk titik berskala, seperti yang ditunjukkan pada Persamaan 2.3. Hal ini memungkinkan model untuk memperhatikan informasi yang berbeda dari subruang penyematan yang berbeda secara bersamaan.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Persamaan 2.3

Transformator (seperti yang ditunjukkan pada Gambar 2.30) menggunakan perhatian multi-kepala dengan pengaturan encoder-decoder di tiga tempat:

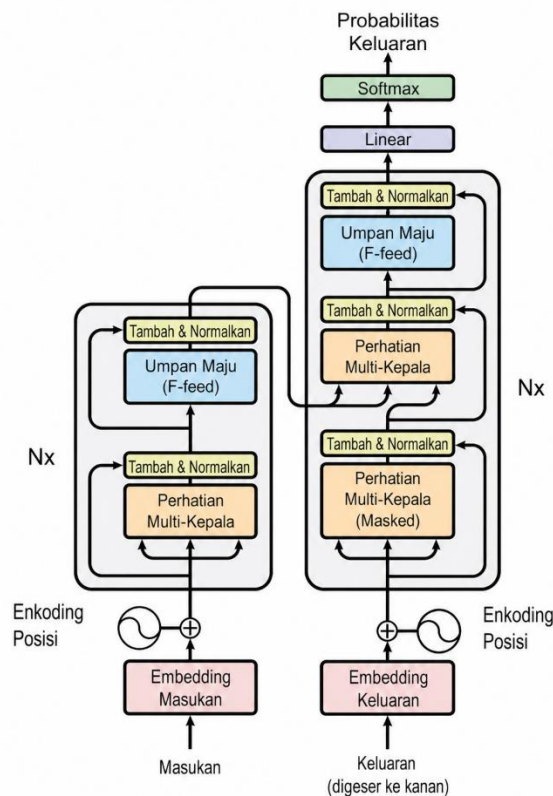
1. Di dalam encoder sebagai perhatian diri di mana kunci, nilai, dan kueri berasal dari encoder, sehingga encoder dapat memperhatikan semua posisinya.
2. Di antara encoder dan decoder, di mana kunci dan nilai berasal dari encoder dan kueri berasal dari output sebelumnya dari decoder.
3. Di dalam decoder sebagai perhatian diri, di mana kunci, nilai, dan kueri berasal dari decoder, sehingga decoder dapat memperhatikan semua posisinya.

Transformer untuk Visi

Bagian ini membahas metode yang memungkinkan Anda menerapkan transformer untuk masalah persepsi yang disebutkan dalam bab ini, yaitu klasifikasi, deteksi, segmentasi, dan pemberian keterangan.

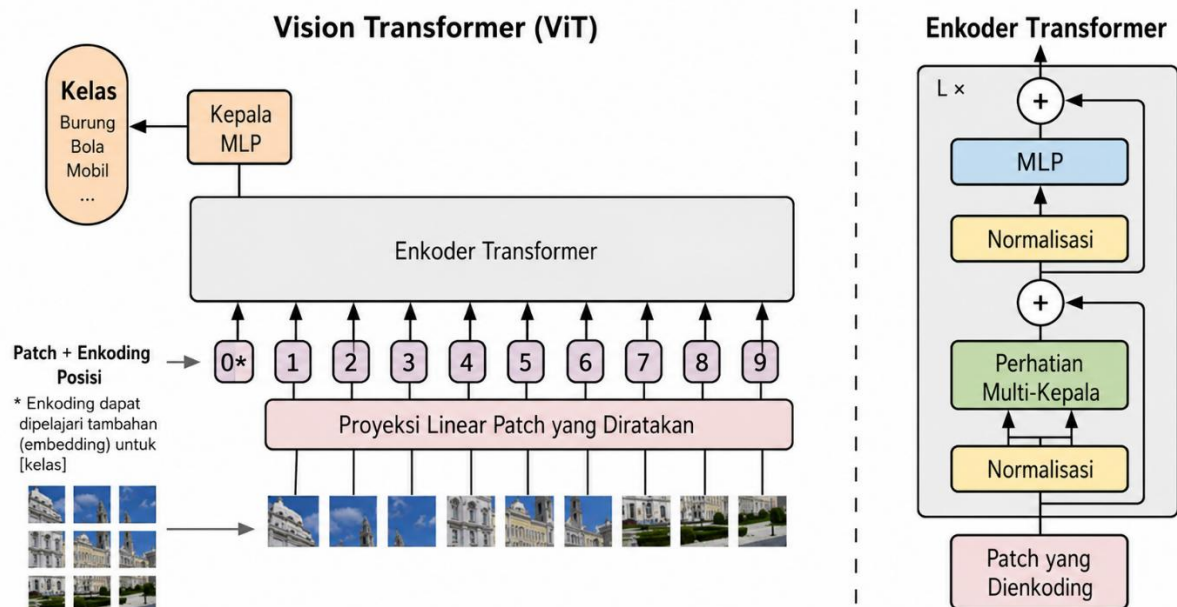
Klasifikasi Gambar dengan Vision Transformer (ViT)

Salah satu aplikasi pertama transformer untuk visi komputer yang dapat diterapkan untuk persepsi robot diterbitkan oleh Dosovitskiy. dalam bentuk *vision transformer* (ViT). Dengan memanfaatkan kemampuannya untuk menafsirkan data visual dan menangkap hubungan kontekstual, *vision transformer* (ViT) dapat digunakan untuk klasifikasi gambar. ViT berguna ketika Anda membutuhkan pemahaman menyeluruh tentang lingkungan karena mekanisme perhatiannya, yang memungkinkannya untuk memodelkan ketergantungan jarak jauh dan menangkap informasi yang sangat rinci. Kemampuan persepsi robot dapat ditingkatkan dengan memodifikasi arsitektur ViT dan melatihnya pada data spesifik domain. Hal ini memungkinkan robot untuk bergerak, berinteraksi, dan membuat penilaian yang lebih baik berdasarkan input visual.



Gambar 2.30. Transformer

ViT memecah gambar menjadi bagian-bagian kecil (patch). Kemudian, ia memasukkannya ke dalam transformer seolah-olah bagian-bagian kecil tersebut adalah urutan dengan penyematan posisi yang sesuai dengan posisi bagian kecil tersebut dalam gambar. Arsitektur ini ditunjukkan pada Gambar 2.31. Arsitektur ini, pada dasarnya, menyatukan studi bahasa alami dan studi visi komputer, karena sekarang Anda dapat memperlakukan gambar sebagai bahasa dengan memecah gambar menjadi urutan token bagian kecil, seperti halnya kalimat/dokumen adalah urutan token bahasa. Sebelum makalah ini, jaringan saraf konvolusional mendominasi pemrosesan visi dan transformer mendominasi pemrosesan bahasa, tetapi ViT meletakkan dasar bagi transformer untuk mendominasi visi dan akhirnya pemrosesan bahasa-visual atau multimodal.



Gambar 2.31. Arsitektur transformer visi.

Meskipun memecah gambar menjadi beberapa bagian tampak berlawanan dengan tujuan mempertahankan korelasi geometris di seluruh bagian, ditemukan bahwa ketika model dilatih sebelumnya pada dataset yang sangat besar (>100 juta gambar) dan kemudian disempurnakan untuk tugas klasifikasi, model tersebut mempelajari hubungan antara embedding posisi dan dapat mengekstrak fitur di seluruh bagian.

Dengan ukuran dataset yang lebih kecil, model berbasis ResNet masih dominan mengingat jaringan saraf konvolusi mempertahankan bias induktif tentang kesetaraan translasi dan lokalitas. Pemeriksaan lebih lanjut menemukan bahwa lapisan awal belajar untuk memperhatikan fitur dalam bagian-bagian tersebut, mempertahankan struktur dimensi rendahnya. Setelah embedding posisi ditambahkan, terdapat kesamaan dalam embedding antara bagian-bagian yang berdekatan dan bagian-bagian dalam baris/kolom yang sama, yang

berarti bahwa jaringan mempelajari konteks geometris gambar yang lebih besar. Dari perspektif itu, jarak perhatian ViT sebanding dengan bidang reseptif ResNet.

ViT menangani gambar dengan resolusi lebih tinggi dengan meningkatkan panjang urutan dan interpolasi 2D dari embedding posisi pra-terlatih yang sesuai dengan posisi patch baru. Makalah ini juga menemukan bahwa ViT mengungguli ResNet dalam hal komputasi vs kinerja, membutuhkan memori dua hingga empat kali lebih sedikit untuk kinerja yang sama. Menariknya, model tersebut tidak jenuh, menunjukkan kemungkinan kinerja yang lebih tinggi melalui penskalaan.

Penskalaan Vision Transformer

Meskipun ViT merupakan yang terbaik untuk banyak tugas visi komputer, penskalaan seperti transformer padat biasa terbukti tidak layak karena ketidakstabilan pelatihan. Makalah, *"Scaling Vision Transformers to 22 Billion Parameters,"* membahas mengapa metode tradisional pelatihan ViT menghasilkan ketidakstabilan selama penskalaan dan bagaimana memodifikasi arsitektur untuk mencegahnya. Sebagai hasil dari investigasi dan modifikasi ini, mereka memperkenalkan model ViT 22B, yang terdiri dari 22 miliar parameter, yang pada saat itu 5,5 kali lebih besar daripada tulang punggung visi sebelumnya ViT-e yang memiliki empat miliar parameter. ViT 22B sangat menginspirasi PaLM-e, yang menggabungkan ViT dan model bahasa besar untuk menciptakan model dasar robot multimodal awal. Bab 4 membahas hal ini lebih mendalam.

Para penulis menerapkan tiga modifikasi utama pada ViT tradisional untuk memungkinkan penskalaan:

1. **Lapisan paralel:** ViT 22B menerapkan blok perhatian dan MLP secara paralel, bukan secara berurutan seperti pada transformer tradisional. Hal ini memungkinkan pengelompokan beberapa operasi seperti paralelisasi proyeksi linier dari blok MLP dan perhatian. Teknik ini menghasilkan peningkatan kecepatan sebesar 15 persen selama pelatihan tanpa memengaruhi kinerja.

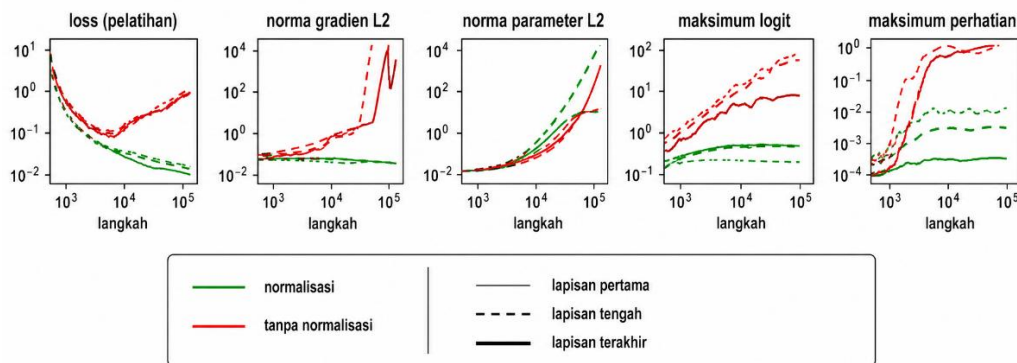
$$y' = \text{LayerNorm}(x),$$
$$y = x + \text{MLP}(y') + \text{Attention}(y').$$

2. **Normalisasi QK:** Pada sekitar parameter 8B, terlihat bahwa logit perhatian menjadi nilai yang sangat besar dan menunjukkan entropi yang sangat rendah, menyebabkan perbedaan dalam kehilangan pelatihan yang membuat pelatihan tidak stabil. Menerapkan LayerNorm ke kueri dan kunci sebelum perhatian produk titik berskala memperbaiki masalah ini, seperti yang ditunjukkan pada Gambar 2.32. Bobot perhatian kemudian dihitung sebagai

$$\text{softmax} \left[\frac{1}{\sqrt{d}} \text{LN}(XW^Q)(\text{LN}(XW^K))^T \right]$$

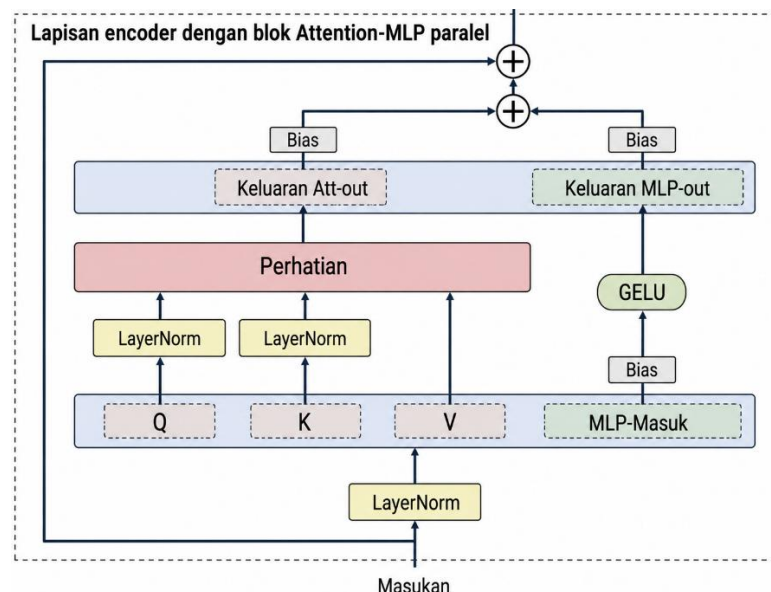
di mana X adalah input, dan W^Q serta W^K masing-masing sesuai dengan matriks bobot kueri dan kunci.

3. Menghilangkan istilah bias: Menghilangkan bias dari proyeksi QKV dan LayerNorms pertama kali ditemukan bermanfaat dalam makalah PaLM 540B dan diulangi juga dalam makalah ViT-22B, karena meningkatkan pemanfaatan akselerator tanpa mengorbankan kinerja. Berbeda dengan PaLM 540B, ViT 22B mempertahankan istilah bias dalam lapisan padat MLP karena ditemukan tidak membahayakan pemanfaatan.



Gambar 2.32. Peningkatan metrik pelatihan karena normalisasi kueri/kunci (kurva hijau).

Gambar 2.33 menunjukkan encoder ViT yang dimodifikasi dengan modifikasi yang diterapkan pada ViT-22B. Ini dilatih pada dataset JFT yang diperluas yang terdiri dari sekitar 4 miliar gambar. Implementasi praktis ViT 22B dilakukan pada JAX dengan paralelisme model dan data; ini menggabungkan langkah-langkah untuk mengoptimalkan throughput dengan meningkatkan komputasi (per perangkat) dan komunikasi (antar perangkat). ViT 22B mendekati hasil SoTA pada beberapa tugas persepsi, termasuk klasifikasi, segmentasi semantik, dan estimasi kedalaman monokular.



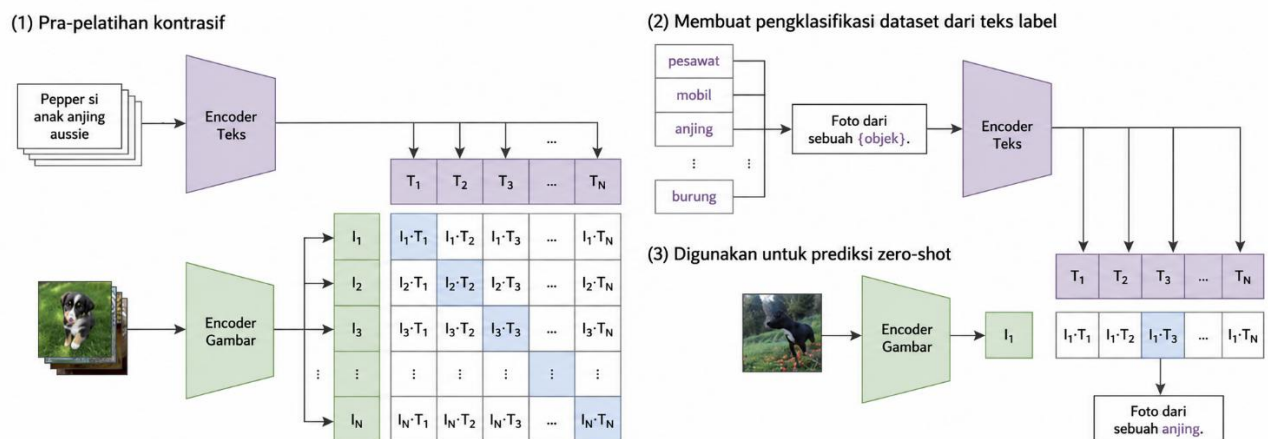
Gambar 2.33. Lapisan ViT-22B paralel dengan normalisasi QK.

Pembelajaran Fitur Gabungan Gambar-Bahasa

Menerapkan bahasa dengan visi komputer seperti menggunakan CLIP, yang diperkenalkan oleh Radford dkk. memungkinkan Anda untuk menggunakan transformer seperti ViT untuk mendapatkan fitur visual yang lebih kaya. Ide CLIP adalah untuk mempelajari embedding secara bersamaan dari korpus besar pasangan gambar-teks untuk menggunakan klasifikasi gambar zero-shot.

Alih-alih melatih encoder gambar dan pengklasifikasi, Anda melatih encoder gambar dan encoder teks secara bersamaan, yang diberi batch pasangan (gambar, teks) dengan label yang menunjukkan apakah keduanya cocok. Ada beberapa alasan kuat untuk melakukan ini. Ini menghilangkan persyaratan untuk memberi label data dalam format yang dimaksudkan untuk tugas klasifikasi. Bahkan ImageNet hanya memberi label 1.000 kelas, yang jauh lebih sedikit daripada deskriptor objek yang dibutuhkan untuk visi umum. Dibandingkan dengan itu, supervisi bahasa alami melatih teks deskriptif, tanpa pelabelan eksplisit dan terformat. Ini memungkinkan pelatihan dari sejumlah besar deskripsi gambar/teks di Internet, dibandingkan dengan tolok ukur sebelumnya, serta mempelajari atribut gambar seperti jenis objek, estetika, gaya, dan fitur lain yang cenderung ditulis orang secara online.

Tujuan kontrastif juga ditemukan lebih baik daripada tujuan prediktif dalam mempelajari representasi. Dalam pengaturan kontrastif, model dilatih untuk memprediksi teks mana yang dipasangkan dengan gambar mana, bukan memprediksi kata-kata persis dari teks tersebut. Hal ini karena encoder gambar yang dilatih bersama mempelajari representasi yang diperlukan untuk menggambarkan gambar yang sama dengan berbagai cara, alih-alih menggunakan label tunggal. Gambaran umum CLIP ditunjukkan pada Gambar 2.34.



Gambar 2.34. Ringkasan CLIP.

Rangkaian pengkode gambar dapat berupa ResNet standar atau transformer visi dan pengkode teks dapat berupa transformer teks lain seperti pada Vaswani et al. Persepsi robot dapat ditingkatkan dengan pemahaman gambar yang lebih kaya yang dimungkinkan oleh fitur CLIP.

Bahasa memungkinkan kueri objek yang lebih fleksibel daripada yang diizinkan oleh metode klasifikasi/deteksi murni.

Kemampuan CLIP dalam mengkorelasikan fitur gambar dan teks dapat membantu robot membuat kesimpulan berdasarkan instruksi bahasa dari manusia ke fitur objek/gambar yang relevan di lingkungan untuk menyelesaikan tugas bahasa visual yang menantang. Bab 4 membahas aplikasi bahasa dalam robotika secara lebih mendalam.

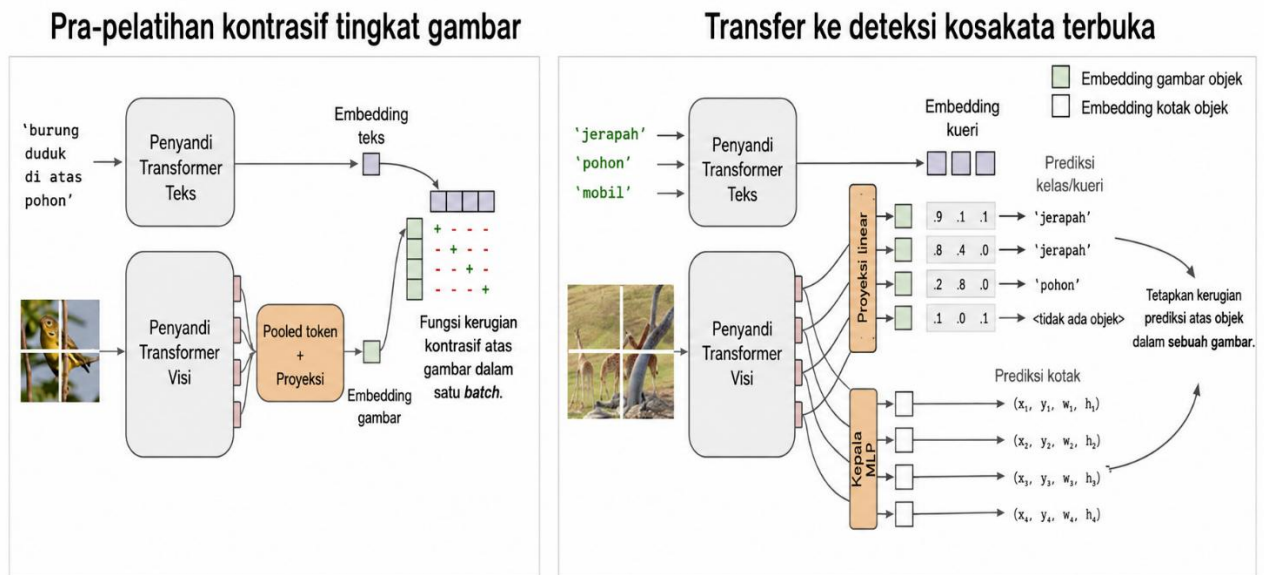
Deteksi Objek Kosakata Terbuka dengan Transformer

Meskipun model ViT asli melakukan deteksi pada benchmark ImageNet, OWL-ViT menggabungkan teknik ViT dan CLIP untuk melakukan deteksi objek kosakata terbuka. Mereka melatih jaringan CLIP besar pada miliaran data gambar-teks berpasangan skala Internet dan menyempurnakannya ke dataset deteksi yang lebih kecil dengan jutaan contoh.

Bagaimana OWL-ViT memodifikasi CLIP?

1. OWL-ViT menghilangkan lapisan token pooling dan proyeksi CLIP dan sebagai gantinya memproyeksikan embedding token output untuk mendapatkan label deteksi per objek. Untuk setiap kueri objek, probabilitas seberapa besar kueri tersebut terkait dengan gambar dan kotak pembatas diprediksi. Pada titik ini, deteksi objek kelas tertutup menjadi kasus khusus, di mana setiap label kelas diterapkan sebagai kueri ke setiap gambar.
2. Meskipun fusi awal umumnya berguna untuk mengekstrak fitur gambar yang tepat, fusi awal dapat memperlambat proses di OWL-ViT, karena Anda perlu mengurai seluruh gambar untuk setiap kueri. Sebagai gantinya, kueri dilewatkan secara independen melalui encoder teks, memungkinkan penggunaan ribuan kueri per gambar dan meningkatkan efisiensi inferensi.
3. Karena bahasa dan gambar digabungkan di tahap akhir, Anda juga dapat melakukan deteksi objek yang dikondisikan pada embedding gambar, bukan embedding teks, sehingga memungkinkan deteksi objek yang dikondisikan pada gambar.
4. Model ini dilatih menggunakan fungsi kerugian focal sigmoid cross-entropy, yang sesuai dengan dataset dengan jumlah kelas yang besar, di mana label-labelnya terpisah dan setiap gambar memiliki beberapa contoh label positif (hadir) dan negatif (tidak hadir).

OWL-ViT mendapatkan performa yang sangat baik pada deteksi objek kosakata terbuka, dengan selisih yang besar dari pekerjaan sebelumnya. Ia juga menunjukkan deteksi objek zero-shot, yang menandakan transfer pengetahuan dan representasi dari data skala internet ke tugas deteksi objek. OWL-ViT cukup cepat untuk menjalankan deteksi kosakata terbuka dalam waktu 100ms, yang berarti dapat digunakan secara luas sebagai sistem pelabelan otomatis. Gambar 2.35 menunjukkan proses pra-pelatihan encoder gambar dan teks menggunakan CLIP dan OWL-ViT yang digunakan untuk deteksi kosakata terbuka.



Gambar 2.35. Kiri: Pra-pelatihan CLIP. Kanan: OWL-ViT untuk deteksi.

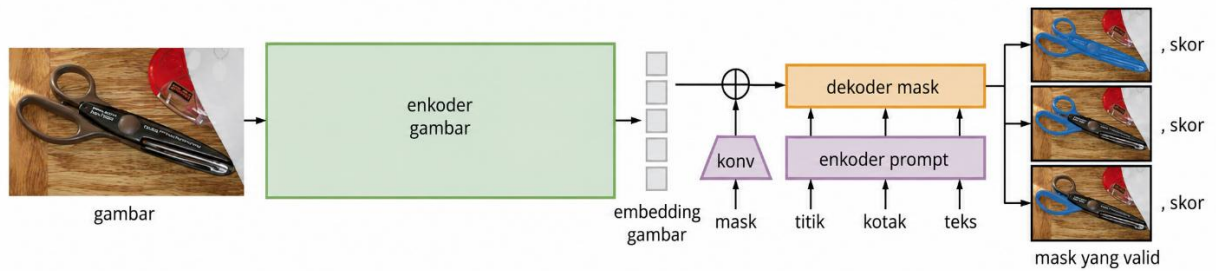
Segmentasi Kosakata Terbuka yang Dapat Diperintahkan

Transformer juga telah diterapkan untuk menyelesaikan masalah segmentasi gambar dengan metode yang disebut *Segment Anything Model* (SAM) yang dikembangkan oleh Meta Research pada tahun 2023. Mereka menindaklanjuti dengan Segment Anything 2 pada tahun 2024, yang menerapkan segmentasi di seluruh bingkai video. SAM mendefinisikan segmentasi yang dapat diperintahkan sebagai tugas menghasilkan masker segmentasi yang valid untuk setiap perintah yang menentukan informasi tekstual atau spasial yang dicari dalam gambar, dengan cara yang fleksibel, sadar akan ambiguitas, dan secara real-time. SAM dilatih pada 11 juta gambar dan 1,1 miliar masker segmentasi, menjadi model dasar yang sangat berharga dalam visi komputer.

SAM memiliki tiga komponen utama:

1. Encoder gambar yang mengkodekan gambar masukan ke dalam embedding untuk mengekstrak fitur tingkat tinggi dan representasi semantik, dalam hal ini ViT
2. Encoder prompt untuk mengkodekan berbagai jenis prompt (sparse dan dense) ke dalam vektor embedding
3. Decoder mask yang menggunakan embedding gambar dan prompt untuk memprediksi mask segmentasi. Ia memperbarui embedding ini menggunakan self-attention dan cross-attention antara embedding prompt dan gambar

Gambar 2.36 menunjukkan gambaran umum bagaimana ketiga komponen ini berinteraksi dalam kerangka kerja Segment Anything. SAM memiliki latensi end-to-end 50ms dari masukan hingga pembuatan mask dan sangat cepat.



Gambar 2.36. Pipeline pemrosesan Segment Anything.

Untuk memperluas SAM ke segmentasi video, SAM2 memperkenalkan perhatian memori, encoder memori, dan bank memori. SAM2 memperluas segmentasi dalam dimensi temporal dengan memungkinkan prompt pada satu atau lebih gambar video, dan mengharapkan output yang tersegmentasi di seluruh frame. Prompt tersebut bisa berupa klik, deskripsi teks, dan sebagainya.

Encoder gambar di SAM2 menghasilkan satu embedding per frame video. Perhatian memori mengkondisikan prediksi mask segmentasi output pada gambar masa lalu, prediksi masa lalu, dan prompt baru apa pun dengan menerapkan self-attention pada gambar saat ini dan cross-attention pada memori frame dan prediksi masa lalu. Encoder memori menghasilkan memori dengan melakukan downsampling pada output. Bank memori adalah antrian *first-in, first-out* (FIFO) yang menyimpan M memori untuk frame dan N memori untuk frame yang diminta sebagai peta fitur spasial, serta penunjuk objek untuk objek target yang akan disegmentasi sebagai embedding vektor.

SAM2 dilatih secara bersamaan pada data gambar dan video dengan pemberian petunjuk interaktif dan menunjukkan kinerja yang luar biasa untuk segmentasi kosakata terbuka zero-shot dalam video.

2.10 RINGKASAN

Bab ini membahas konsep-konsep berikut:

- Robot menggunakan berbagai sensor untuk merasakan lingkungannya, termasuk sensor penglihatan seperti kamera untuk menangkap gambar, sensor kedalaman untuk mengukur jarak, sensor jangkauan seperti LiDAR dan sensor ultrasonik untuk navigasi, dan IMU untuk melacak gerakan dan orientasi.
- Robot melakukan tugas-tugas seperti klasifikasi, segmentasi semantik, segmentasi instans, dan deteksi objek. Tugas-tugas ini membantu robot menavigasi, memanipulasi objek, dan menafsirkan lingkungan.
- CNN dapat digunakan untuk memproses gambar melalui lapisan seperti lapisan konvolusi, ReLU, dan lapisan pooling. Bab ini membahas berbagai model, termasuk R-CNN, Fast/Faster R-CNN, YOLO, dan SSD, termasuk bagaimana model-model tersebut

menyeimbangkan kecepatan dan akurasi untuk tugas-tugas seperti deteksi dan segmentasi objek, masing-masing dengan kompromi dalam kinerja.

- Transformer memperlakukan gambar sebagai urutan, memungkinkan mereka untuk menangkap konteks global dan ketergantungan dalam sebuah gambar.

Model seperti *vision transformer* (ViT) dan *detection transformer* (DETR) dapat digunakan untuk deteksi dan segmentasi objek, dan mereka mengungguli CNN tradisional dalam skenario tertentu. Bab ini juga mengeksplorasi bagaimana transformer dapat diskalakan secara efisien, dan bagaimana metode seperti ViT (untuk klasifikasi), CLIP (untuk pemahaman gambar-teks bersama), dan SAM (untuk segmentasi) dapat berguna dalam robotika. Bab ini terutama berfokus pada teknik pemrosesan gambar 2D. Bab selanjutnya membahas metode pemrosesan gambar 3D, persepsi multimodal, dan fusi sensor.

BAB 3

PERSEPSI ROBOT: DATA 3D DAN FUSI SENSOR

Data sensor 3D yang dikumpulkan dari LiDAR atau kamera kedalaman sangat penting untuk persepsi robot di dunia nyata, karena memungkinkan robot untuk merasakan struktur tiga dimensi objek di sekitarnya. Data sensor 3D sering dikumpulkan dan digunakan oleh robot industri dan konsumen, tetapi salah satu aplikasi yang paling relevan adalah dalam pengembangan kendaraan otonom. Kekuatan utama di balik kendaraan otonom adalah data sensor 3D yang dikumpulkan dari LiDAR atau kamera kedalaman. Dari data sensor 3D, kendaraan otonom dapat secara akurat merasakan struktur tiga dimensi objek, pejalan kaki, dan kendaraan lain di sekitarnya. Informasi ini memungkinkan deteksi objek yang kuat, lokalisasi yang tepat, dan perencanaan jalur yang andal, yang mengarah pada peningkatan keselamatan, kesadaran situasional, dan pengambilan keputusan yang lebih efisien dalam skenario mengemudi waktu nyata. Beberapa perusahaan kendaraan otonom juga mengandalkan pemrosesan data 3D untuk membuat peta terperinci, membantu navigasi dan memastikan pengalaman mengemudi otonom yang lancar dan andal.

Secara keseluruhan, pemrosesan data 3D sangat penting untuk persepsi, pemetaan, dan navigasi robot. Dengan memproses data sensor 3D, seperti awan titik dan peta kedalaman, robot dapat merasakan lingkungan sekitarnya dalam tiga dimensi, mendeteksi rintangan, dan merencanakan jalur bebas tabrakan untuk mencapai tujuan mereka. Bab ini membahas metode untuk memproses data dengan persepsi kedalaman seperti peta 3D serta fusi sensor. Bab ini juga mencakup teknik yang digunakan untuk mengompresi data dari sensor multimodal seperti radar, LiDAR, dan kamera.

3.1 PEMROSESAN DATA 3D

Untuk memahami pemrosesan data 3D, Anda perlu mempelajari terlebih dahulu bagaimana data 3D direpresentasikan dalam kerangka kerja pembelajaran mesin. Kemudian Anda dapat memahami metode dan belajar dari representasi data yang dihasilkan model.

Representasi Data

Data 3D, biasanya dari LiDAR, yang merupakan kamera penglihatan stereo dengan sensor kedalaman atau inframerah, umumnya direpresentasikan dalam bentuk awan titik, voxel, atau mesh.

Mari kita pertimbangkan masing-masing istilah ini.

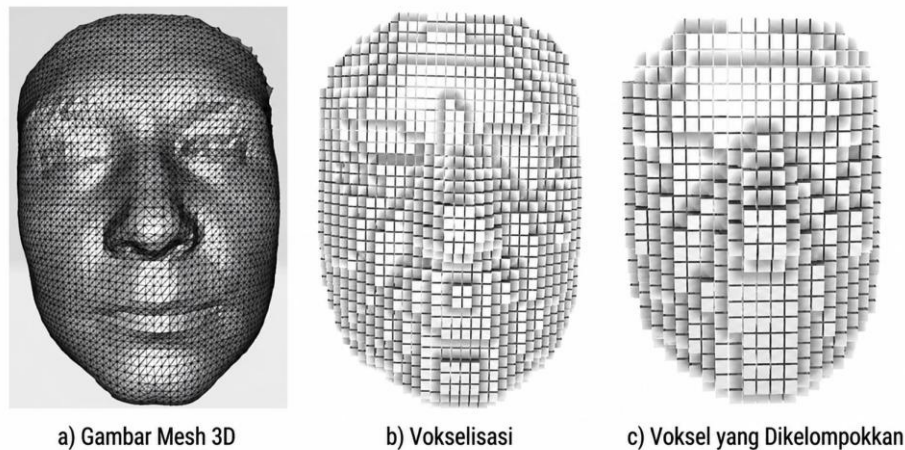
- Awan titik adalah sekumpulan koordinat titik di ruang angkasa, umumnya direpresentasikan sebagai koordinat Kartesius (x, y, z) . Sensor LiDAR dengan banyak pantulan dapat menghasilkan awan titik.
- Voxel atau piksel volumetrik adalah piksel 3D. Ini direpresentasikan sebagai titik (x, y, z) yang dipikselkan di mana koordinatnya dibatasi oleh resolusi grid voxel. Voxel jarang adalah representasi yang efisien dalam penggunaan memori di mana hanya sel yang

berisi informasi yang disimpan. Octree adalah varian dari voxel jarang di mana sel-sel yang berdekatan dan identik digabungkan, memungkinkan kompresi data.

- Mesh adalah representasi poligon dari data 3D di mana data direpresentasikan sebagai komposisi permukaan poligon, umumnya segitiga.

Peta kedalaman biasanya merupakan output dari kamera stereo vision di mana, untuk setiap piksel (x, y) , terdapat kedalaman (z) . Perhatikan bahwa ini adalah representasi 2.5D, karena Anda tidak dapat memiliki beberapa nilai kedalaman untuk piksel yang sama.

Pemodelan parametrik mengacu pada representasi objek dalam hal permukaan dan volume yang berbagi parameter, sehingga atribut yang saling terkait berubah bersamaan. Contohnya adalah model CAD suatu objek, di mana tinggi dan lebar dapat berubah dengan skala. Perbedaan antara gambar mesh 3D dan gambar voxelized ditunjukkan pada Gambar 3.1.



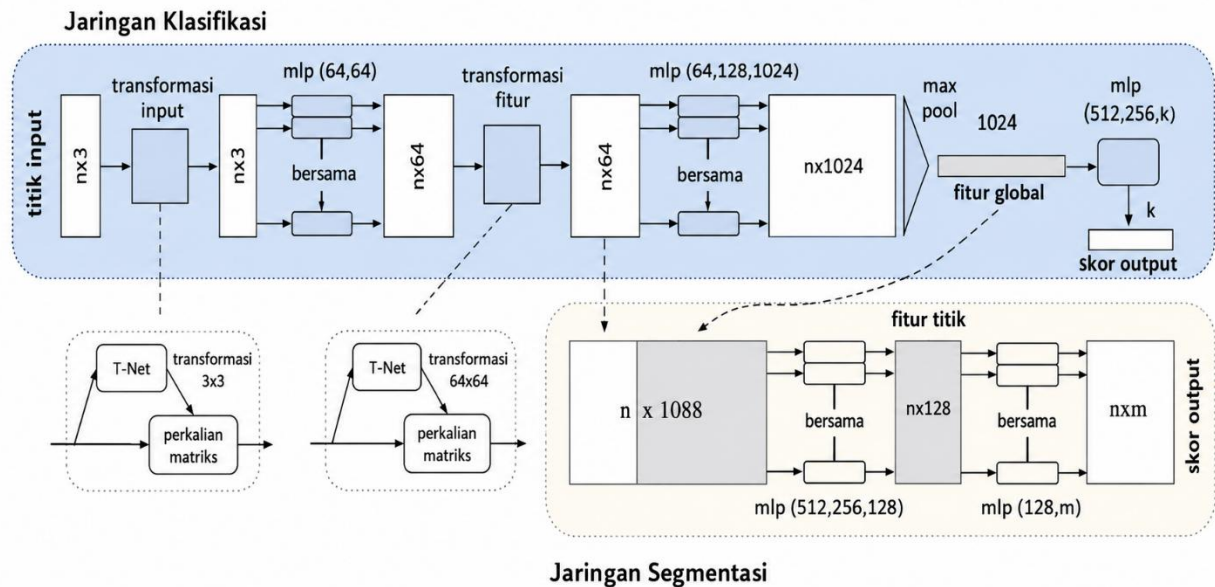
Gambar 3.1. Representasi jala 3D dan voxel dari wajah manusia

Dengan voxel, ukuran dataset meningkat secara kubik dengan ruang, sementara data tetap teratur. Dengan point cloud, Anda dapat merepresentasikan kepadatan spasial dan resolusi yang bervariasi secara efektif, tetapi himpunan titik bersifat invarian permutasi dan model yang menggunakannya perlu mempertimbangkan properti tersebut. Untuk keperluan robotika, bab ini berfokus pada pemrosesan awan titik, karena, setelah munculnya LiDAR, awan titik telah lebih banyak diadopsi dalam studi dan industri.

Pemrosesan Awan Titik

Awan titik pada dasarnya tidak teratur dalam representasinya. Awan titik juga invarian terhadap permutasi, artinya mengubah urutan titik tidak mengubah dataset. Di masa lalu, awan titik diproses setelah divokselisasi, seperti dengan VoxNet dan 3D ShapeNet. Namun, hal ini meningkatkan penggunaan memori dan memperkenalkan artefak kuantisasi, seperti menghilangkan informasi dalam kluster dengan kepadatan titik tinggi.

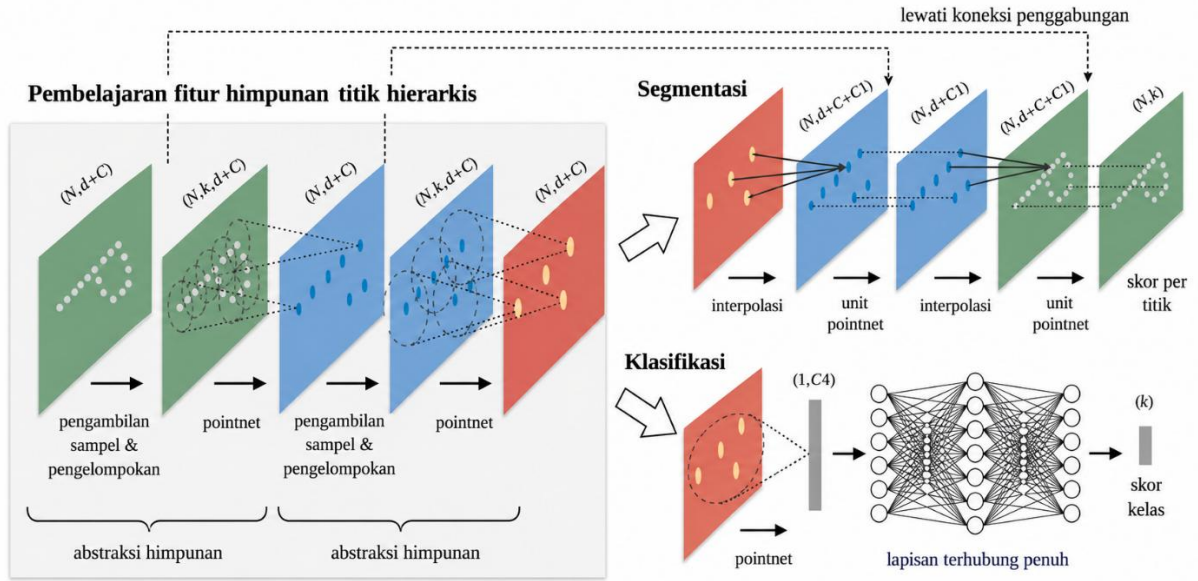
PointNet, yang ditunjukkan pada Gambar 3.2, adalah arsitektur perintis untuk klasifikasi dan segmentasi objek pada awan titik. Jaringan ini mengambil sekumpulan n titik yang tidak berurutan dan mengubahnya menjadi vektor fitur invarian transformasi menggunakan perceptron multilayer.



Gambar 3.2. Arsitektur PointNet.

Input diselaraskan ke ruang kanonik dengan menghitung transformasi affin menggunakan T-net, yang merupakan jaringan mini menyerupai jaringan yang lebih besar, ditunjukkan pada Gambar 3.2, yang berisi ekstraksi fitur independen titik, max pooling, dan lapisan terhubung penuh. Vektor fitur digabungkan ke titik asli dan fitur titik gabungan ini diagregasi menggunakan beberapa lapisan MLP dan max pooling untuk menghasilkan skor per titik. Pada saat dirilis, PointNet adalah yang terbaik dalam segmentasi semantik untuk awan titik. Namun, karena max pooling adalah agregasi dan tidak mempertahankan struktur lokal, PointNet tidak kuat dengan pola berbutir halus dan adegan yang sangat kompleks.

Kekurangan terkait struktur lokal telah diperbaiki oleh PointNet++, ditunjukkan pada Gambar 3.3. PointNet++ mencoba menyelesaikan masalah ini dengan pertama-tama mempartisi himpunan titik menjadi wilayah lokal yang tumpang tindih menggunakan metrik jarak Euclidean untuk ruang 3D. Kemudian, alih-alih menggunakan operator max pooling tunggal pada seluruh point cloud, PointNet++ membangun pengelompokan hierarkis dari titik-titik ini, mengekstrak fitur lokal menggunakan PointNet, dan memprosesnya dalam kelompok hierarkis untuk mendapatkan fitur tingkat yang lebih tinggi secara iteratif. Pada dasarnya, mereka secara rekursif menerapkan PointNet pada partisi bersarang dari himpunan input. Partisi yang tumpang tindih dihasilkan dengan mendefinisikan bola di ruang input dengan ukuran dan pusat yang bervariasi, dan centroid dari cluster/bola ini dihasilkan menggunakan algoritma pengambilan sampel titik terjauh (FPS). Dibandingkan dengan CNN volumetrik, yang selalu memiliki bidang reseptif yang sama, penerapan PointNet pada kernel Euclidean dengan ukuran yang bervariasi ini beradaptasi dengan variasi kepadatan point cloud dan variasi skala fitur. Hal ini dicapai selama pelatihan melalui dropout input acak sehingga jaringan belajar untuk secara adaptif menggabungkan fitur yang diekstrak dari lingkungan multiskala berdasarkan data input.



Gambar 3.3. Arsitektur PointNet++.

Operasi interpolasi yang ditunjukkan pada kepala segmentasi jaringan pada Gambar 3.3 dicapai dengan menginterpolasi fitur pada titik j menggunakan rata-rata tertimbang jarak terbalik atas k tetangga terdekat (lihat Persamaan 3.1).

$$f^{(j)}(x) = \frac{\sum_{i=1}^k w_i(x) f_i^{(j)}}{\sum_{i=1}^k w_i(x)} \text{ where } w_i(x) = \frac{1}{d(x, x_i)^p}, J = 1, \dots, C \quad (3.1)$$

Kemudian fitur-fitur ini digabungkan dengan fitur titik level lompatan dari lapisan abstraksi dan dilewatkan melalui PointNet unit (mendekati konvolusi 1×1). Fitur-fitur ini kemudian disebarkan kembali hingga himpunan titik asli diberi label.

Namun, PointNet++ masih memiliki ruang untuk perbaikan. Ekstraksi fitur menggunakan PointNet yang menggunakan max pooling kehilangan informasi spasial. PointCNN, yang ditunjukkan pada Gambar 3.5, bertujuan untuk menggunakan kemampuan CNN tradisional untuk menangkap fitur lokal dan hierarkis dengan bidang reseptif yang meningkat pada awan titik. Salah satu cara untuk menerapkan CNN pada awan titik adalah dengan memvoxelisasi data dan kemudian menerapkan CNN pada voxel. Namun, ini bukan representasi data yang efisien dan CNN voxel membutuhkan terlalu banyak memori karena dimensinya. PointCNN memecahkan masalah ini dengan memperkenalkan operator X -conv yang mengubah awan titik menjadi ruang laten dan kanonik. Algoritma untuk hal ini ditunjukkan pada Gambar 3.4.

ALGORITMA 1: Operator $\mathcal{X}$ -Conv

Input : $K, p, \mathbf{P}, \mathbf{F}$

Output : $\mathbf{F}_p$

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$

2: $\mathbf{F}_\delta \leftarrow \text{MLP}_\delta(\mathbf{P}')$

3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$

4: $\mathcal{X} \leftarrow \text{MLP}(\mathbf{P}')$

5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$

6: $\mathbf{F}_p \leftarrow \text{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$

▷ Fitur 'diprojektasikan', atau 'diagregasikan', ke dalam titik representatif p

▷ Pindahkan $\mathbf{P}$ ke sistem koordinat lokal p

▷ Secara individu angkat setiap titik ke ruang berdimensi C_δ

▷ Gabungkan $\mathbf{F}_\delta$ dan $\mathbf{F}$, $\mathbf{F}_*$ adalah matriks $K \times (C_\delta + C_1)$

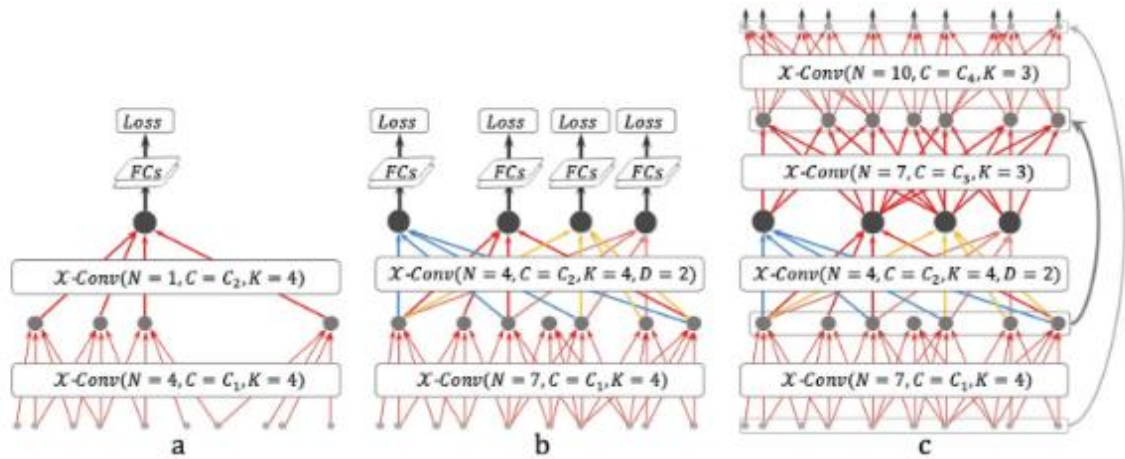
▷ Pelajari matriks transformasi $\mathcal{X}$ $K \times K$

▷ Beri bobot dan permutasikan $\mathbf{F}_*$ dengan $\mathcal{X}$ yang dipelajari

▷ Akhirnya, konvolusi tipikal antara $\mathbf{K}$ dan $\mathbf{F}_\mathcal{X}$

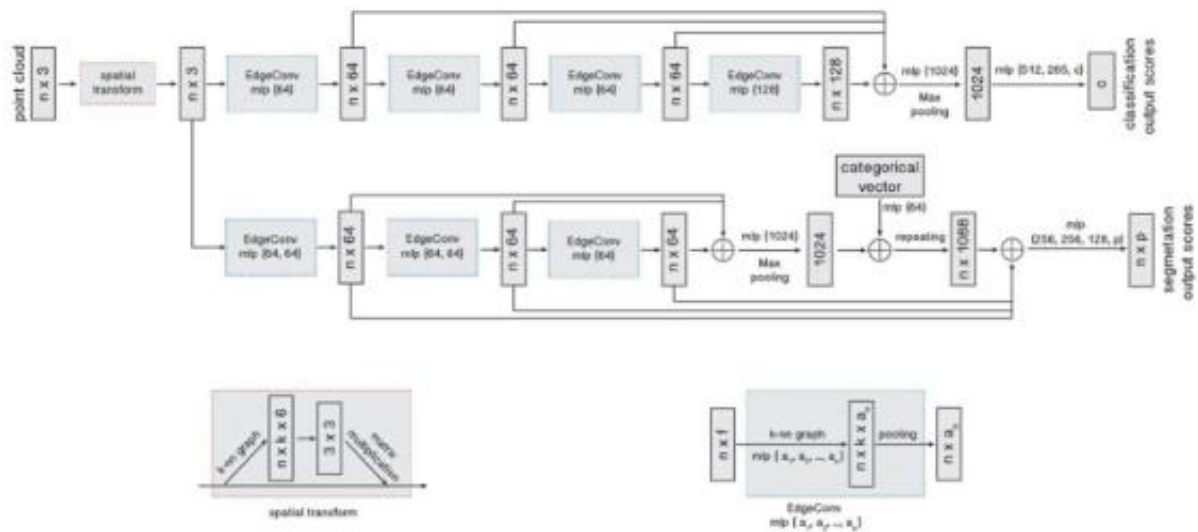
Gambar 3.4. Algoritma X -conv.

Jika terdapat N titik dalam himpunan titik, operasi X -conv mengambil sampel K titik tetangga untuk setiap titik dalam himpunan masukan, yang merupakan lingkungan lokalnya. Setiap lingkungan lokal dibawa ke sistem koordinatnya sendiri. Operasi X -conv sepenuhnya dapat didiferensiasi dan mengubah titik-titik ke ruang fitur yang lebih dalam, dengan resolusi spasial yang lebih kecil.



Gambar 3.5. Arsitektur Point CNN untuk klasifikasi (a dan b) dan untuk segmentasi (c).

Setiap lapisan X -conv mengurangi jumlah titik dengan bidang reseptif yang meningkat, dengan titik akhir melihat semua titik. Untuk deteksi, lapisan X -conv terakhir dipasang dengan lapisan yang terhubung penuh dan fungsi kerugian untuk melatih jaringan. Untuk segmentasi, X -conv diterapkan dengan cara yang mirip dengan U-Net, di mana terdapat operasi d-conv berturut-turut hingga titik-titik asli telah menyebarkan fitur. PointCNN ditemukan berkinerja lebih baik daripada PointNet++ dalam beberapa tugas klasifikasi. *Dynamic Graph CNN* (DGCNN), yang ditunjukkan pada Gambar 3.6, adalah algoritma yang bertujuan untuk meningkatkan fakta bahwa PointNet tidak dapat menangkap fitur pada skala lokal karena invariansi permutasi titik. Hal ini dilakukan dengan memperkenalkan operasi konvolusi tepi.



Gambar 3.6. DGCNN untuk klasifikasi (jalur atas) dan segmentasi (jalur bawah).

DGCNN membangun grafik lingkungan lokal dari k titik terdekat dan memperbarui grafik di setiap lapisan dengan menambahkan operasi seperti konvolusi pada tepi, sehingga menghubungkan pasangan titik dalam lingkungan. Tetangga berubah di setiap lapisan dan grafik dihitung ulang, sehingga bersifat dinamis.

Operasi edgeconv mengambil tensor $n \times f$ (di mana n adalah jumlah titik dan f adalah ukuran fitur input ke lapisan tersebut), menerapkan perceptron multilayer, dan menghitung fitur tepi untuk setiap titik. Untuk klasifikasi, lapisan edgeconv terakhir diagregasi untuk mendapatkan deskriptor 1D global, yang digunakan untuk mendapatkan label kelas untuk himpunan titik. Untuk segmentasi, deskriptor 1D digabungkan dengan fitur tepi dari setiap lapisan edgeconv untuk memberikan fitur global dan lokal untuk setiap titik. Hal ini memungkinkan untuk menghasilkan skor output segmentasi per titik.

DGCNN berkinerja mendekati PointCNN pada tugas klasifikasi dan segmentasi (+3%). Namun, ukuran tetap fitur tepi dapat membatasi kinerjanya pada skala dan resolusi yang berbeda. Mengingat bahwa fitur input berkontribusi secara berbeda pada node, mekanisme perhatian dapat lebih meningkatkan kinerja dengan melihat fitur yang relevan ketika input berukuran variabel terlibat.

Peluang Studi

1. **Memanfaatkan konteks:** Sebagian besar model berbasis point cloud memperlakukan titik pada skala lokal sebagai independen untuk mempertahankan invariansi permutasi. Tetapi ini berarti mereka tidak dapat mengekstrak fitur yang lebih halus dan memanfaatkan hubungan antara suatu titik dan tetangganya.
2. **Kinerja waktu nyata:** Terlepas dari banyaknya model untuk point cloud, sebagian besar sistem robotik memiliki komputasi onboard dan perlu merespons dengan cepat. Ini membutuhkan model yang lebih ringan dengan waktu inferensi yang lebih rendah.
3. **Jebakan pembelajaran terawasi:** Sebagian besar model point cloud yang ada menggunakan pembelajaran terawasi, yang tidak kuat terhadap skenario yang tidak terlihat dan membutuhkan sejumlah besar data untuk melatih dan menggeneralisasi.

Kerangka kerja yang diawasi secara lemah atau tidak diawasi dapat meningkatkan keterbatasan ini.

3.2 PERSEPSI MULTIMODAL DAN FUSI SENSOR

Tujuan persepsi multimodal adalah menggunakan kombinasi modalitas penginderaan yang berbeda untuk membuat robot menyelesaikan suatu tugas. Fusi sensor adalah menggabungkan input dari lebih dari satu sensor untuk mendapatkan pemahaman yang lebih dalam tentang lingkungan sekitar. Input sensor multimodal juga dapat membuat persepsi lebih kuat jika terjadi kegagalan pada salah satu sensor—ini merupakan mekanisme keselamatan yang penting.

Fusi sensor sering digunakan dalam aplikasi seperti mobil otonom untuk menggabungkan beberapa kombinasi LiDAR, radar, dan kamera untuk memanfaatkan berbagai kekuatan mereka dan meningkatkan kinerja sistem gabungan secara keseluruhan. Misalnya, LiDAR memiliki persepsi kedalaman yang baik tetapi tidak memberikan informasi warna. Sementara kamera tidak memberikan informasi tentang kedalaman suatu objek, kamera dapat memberikan informasi warna. Radar, di sisi lain, memberikan pengukuran kecepatan langsung dari rintangan dan melengkapi LiDAR dan kamera.

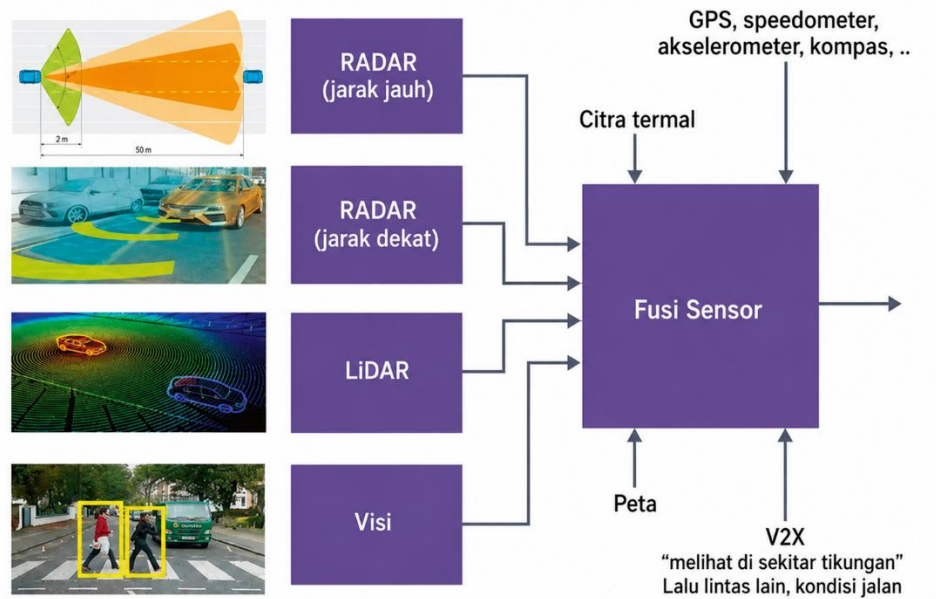
Kegunaan dan spesifikasi fusi sensor pada akhirnya dibenarkan oleh biaya dan presisi yang dibutuhkan oleh aplikasi robot. Salah satu kelemahan persepsi multisensor adalah kompleksitas sistem tambahan dalam menyinkronkan informasi dan input fusi sensor. Agar fusi sensor berfungsi, sensor perlu dikalibrasi, didaftarkan satu sama lain, dan perlu menggunakan referensi waktu yang sama. Mengingat berbagai modalitas penginderaan, Anda memerlukan cara untuk menggabungkannya guna mencapai representasi 3D dunia yang akurat agar robot dapat bertindak. Bagian berikut membahas beberapa strategi umum untuk mencapai hal ini.

Strategi Fusi

Kemajuan terbaru dalam pembelajaran mendalam telah menghasilkan dua strategi fusi sensor utama:

1. **Fusi akhir:** Memproses setiap modalitas penginderaan secara independen hingga akhir. Misalnya, akan menggabungkan kotak pembatas dari LiDAR dan dari kamera yang diproses secara terpisah.
2. **Fusi awal:** Memanfaatkan kemajuan terbaru dalam pembelajaran mendalam untuk menggabungkan pembacaan sensor mentah pada tahap awal. Misalnya, ia akan menggabungkan piksel dan awan titik secara langsung.

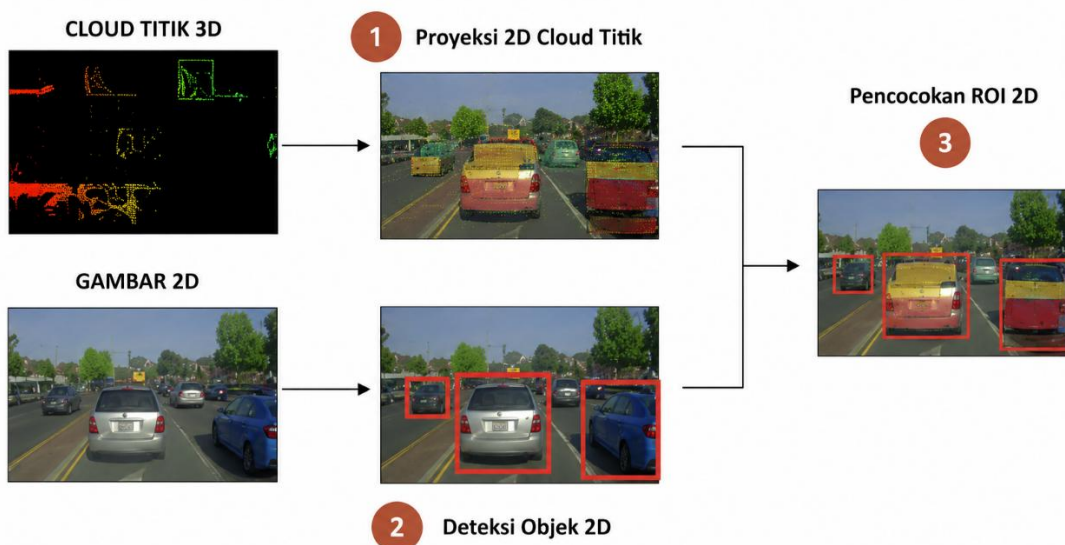
Bagian ini berfokus pada pengintegrasian data LiDAR dan kamera untuk memanfaatkan resolusi tinggi kamera dan kemampuannya untuk mengklasifikasikan objek, bersama dengan kemampuan LiDAR untuk mengukur jarak dan memahami dunia dalam 3D. Anda dapat melihat bagaimana keduanya bekerja bersama dengan melihat penggabungan awan titik dan piksel dalam mode fusi awal dan penggabungan kotak pembatas dalam mode fusi akhir. Untuk mengilustrasikan keuntungan fusi sensor, berbagai sensor dan penggunaan utamanya dalam kendaraan otonom ditunjukkan pada Gambar 3.7.



Gambar 3.7. Gambaran umum fusi sensor dalam kendaraan otonom, menggabungkan data dari RADAR jarak jauh dan jarak dekat, LiDAR, kamera penglihatan, pencitraan termal, GPS, dan sensor lainnya.

Penggabungan Data Mentah: Penggabungan Awal

Dalam penggabungan awal, data mentah seperti awan titik dari LiDAR dan gambar kamera digabungkan secara langsung. Tujuannya adalah untuk membuat proyeksi 2D dari awan titik 3D dalam bingkai kamera, di mana lokasi setiap titik sekarang dapat dibandingkan langsung dengan objek dan fitur dalam gambar kamera 2D. Proses tiga langkah ini ditunjukkan pada Gambar 3.8.



Gambar 3.8. Alur kerja untuk fusi sensor awal.

Langkah 1: Proyeksi Awan Titik

Langkah pertama menerjemahkan awan titik 3D yang ditangkap oleh LiDAR ke dalam format yang dapat digunakan mengingat tipe data 2D untuk kamera. Ini melibatkan beberapa langkah:

- Mengonversi setiap titik 3D dari data LiDAR ke koordinat homogen. Ini melibatkan konversi koordinat Euclidean standar (x, y, z) dan menambahkan dimensi tambahan, menjadikan titik tersebut (x, y, z, w) . Secara spesifik, setiap titik dalam awan titik 3D LiDAR (x, y, z) dikonversi ke koordinat homogen $(x, y, z, 1)$. Koordinat tambahan ini memungkinkan transformasi yang berbeda menggunakan perkalian matriks dan menyederhanakan penyelarasan data 3D dengan data 2D.
- Matriks transformasi (termasuk rotasi dan translasi) menyelaraskan sistem koordinat LiDAR dengan sistem koordinat kamera. Ini berguna untuk kalibrasi kamera ekstrinsik (relatif terhadap LiDAR, posisi/orientasi kamera).
- Matriks transformasi lain memproyeksikan titik-titik 3D ke bidang gambar 2D kamera. Matriks ini memperhitungkan kalibrasi intrinsik (sifat spesifik kamera seperti panjang fokus dan pusat optik). Data ini berguna untuk menerjemahkan dan memutar titik-titik LiDAR secara akurat untuk menyelaraskannya dengan sudut pandang kamera.
- Setelah proyeksi, titik-titik tersebut ditransformasikan kembali dari koordinat homogen ke koordinat Euclidean standar. Titik 2D sekarang dapat langsung dibandingkan dengan objek dan fitur dalam gambar kamera 2D.

Jika tertarik, kursus ini membahas lebih lanjut tentang apa itu koordinat homogen dan bagaimana proyeksi dan rotasi spesifik ini dilakukan. Hasil dari langkah ini ditunjukkan pada Gambar 3.9.



Gambar 3.9. Konversi dari awan titik 3D ke proyeksi awan titik 2D.

Langkah 2: Deteksi Objek

Setelah citra awan titik yang diproyeksikan sejajar dengan citra, berbagai model seperti YOLOv4 dapat digunakan untuk melakukan deteksi objek. Anda dapat mempelajari lebih lanjut tentang cara kerja model deteksi objek ini di Bab 2.

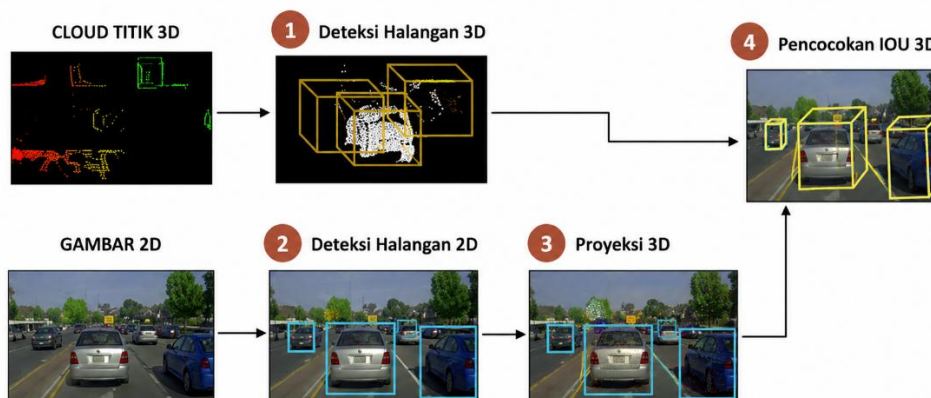
Langkah 3: Pencocokan Wilayah Minat (*Region of Interest/RoI*)

Setelah objek terdeteksi dalam citra kamera menggunakan kotak pembatas 2D, langkah selanjutnya adalah mencocokkan titik 3D dari data LiDAR dengan objek yang terdeteksi ini. Proses ini disebut pencocokan Wilayah Minat (*Region of Interest/RoI*).

- Titik-titik 3D dari LiDAR, yang diproyeksikan ke bidang gambar 2D pada langkah-langkah sebelumnya, sekarang dibandingkan dengan kotak pembatas 2D yang dihasilkan oleh model deteksi objek.
- Untuk setiap kotak pembatas 2D, setiap titik 3D dikaitkan dengan objek yang terdeteksi.
- Setelah titik-titik dikaitkan dengan objek tertentu, setiap titik 3D diberi label sesuai dengan objek yang terdeteksi. Misalnya, titik-titik dalam kotak pembatas untuk sebuah mobil diberi label sebagai bagian dari mobil tersebut.

Penggabungan Output: Penggabungan Akhir

Dalam penggabungan akhir, data dari sensor yang berbeda diproses secara terpisah sebelum menggabungkan hasilnya. Dengan pendekatan ini, deteksi objek independen dilakukan pada data setiap sensor, menghasilkan kotak pembatas 2D atau 3D, yang kemudian digabungkan. Gambaran umum tentang bagaimana proses ini bekerja dalam 3D ditunjukkan pada Gambar 3.10.



Gambar 3.10. Pipeline fusi akhir menggunakan awan titik 3D dan gambar 2D sebagai input.

Langkah 1 dan 2: Deteksi Objek 3D LiDAR dan Deteksi Objek 3D Kamera

Terdapat banyak model pembelajaran mendalam LiDAR yang dapat digunakan untuk deteksi objek 3D, seperti PointNet, PointNet++, VoxelNet, dan SECOND (jaringan konvolusional sparse spasial), beberapa di antaranya dibahas dalam bab ini. Model-model ini menghasilkan kotak pembatas 3D untuk objek yang terdeteksi dan mencakup posisi, dimensi, dan orientasinya dalam sistem koordinat LiDAR. Untuk deteksi kamera 3D, beberapa metode deteksi objek yang digunakan dalam 2D dapat diubah agar dapat bekerja dengan data 3D, seperti menambahkan jaringan estimasi kedalaman ke YOLO. Penting untuk dicatat bahwa

kotak pembatas dihasilkan sebagai (panjang x , y , z , lebar, tinggi, rotasi) dalam sistem koordinat data Anda (LiDAR atau kamera).

Dari sini, kotak pembatas 3D dari satu sistem koordinat dikonversi ke sistem koordinat lainnya. Misalnya, kotak pembatas dapat dikonversi dari sistem koordinat LiDAR ke sistem koordinat kamera. Hal ini memungkinkan kedua set kotak pembatas untuk disejajarkan dalam kerangka acuan yang sama.

Langkah 3: Pencocokan IOU

Langkah selanjutnya adalah mencocokkan kotak pembatas ini untuk mengidentifikasi objek yang sama yang dideteksi oleh kedua sensor. Proses ini melibatkan pencocokan dalam ruang atau waktu.

- **Pencocokan dalam ruang:** Untuk setiap pasangan kotak pembatas 3D dalam data LiDAR dan kamera, skor IOU dihitung. Skor IOU di atas ambang batas tertentu menunjukkan bahwa kotak pembatas kemungkinan besar sesuai dengan objek yang sama.
- **Pencocokan dalam waktu:** Ini memperluas proses pencocokan dari waktu ke waktu. Dengan menggunakan teknik seperti Filter Kalman dan Algoritma Hungaria, objek dapat dilacak di berbagai frame. Jika kotak pembatas dari satu frame tumpang tindih dengan kotak pembatas di frame berikutnya (berdasarkan IOU), objek tersebut dianggap sama.
- Filter Kalman beroperasi dengan pertama-tama memprediksi posisi masa depan kendaraan menggunakan pengetahuan sebelumnya dan persamaan kinematik, kemudian mengukur posisi aktual dengan data sensor untuk dibandingkan dengan prediksi, dan akhirnya memperbarui prediksi untuk meningkatkan akurasi berdasarkan informasi baru.

Ini membantu memberikan estimasi yang lebih akurat tentang lintasan objek. Kami tidak akan terlalu fokus pada metode klasik untuk fusi sensor, tetapi jika tertarik, Anda dapat mempelajari lebih lanjut tentang filter Kalman di sini. Beberapa informasi dari bagian ini dirujuk dari dan kami merekomendasikannya sebagai sumber yang baik.

Fusi LiDAR-Kamera

Setelah Anda memahami kerangka kerja utama untuk fusi sensor, mari kita selami model pembelajaran mendalam yang digunakan untuk fusi LiDAR-kamera.

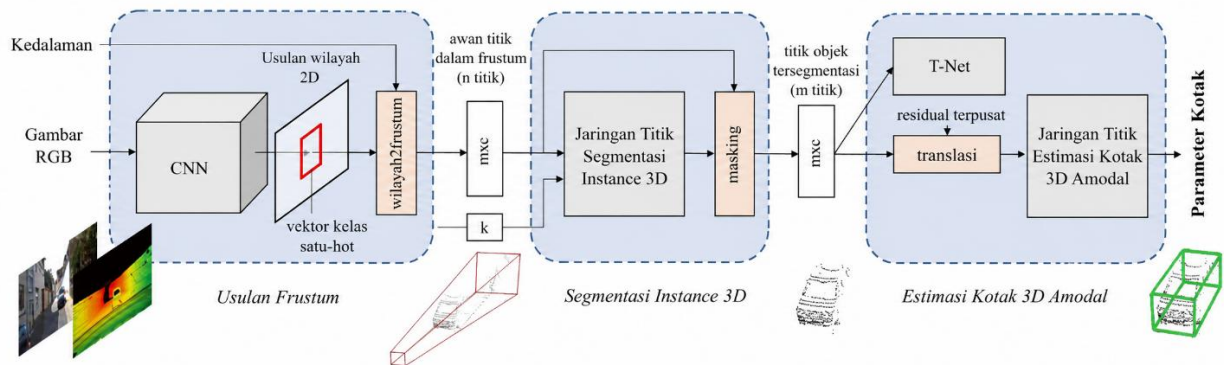
Metode Fusi Tingkat Proposal

Menggabungkan LiDAR dan kamera, fusi tingkat proposal adalah di mana "proposal" atau ROI dihasilkan dari satu jenis data sensor dan kemudian disempurnakan dan ditambah menggunakan data dari sensor lain. Alur kerja keseluruhan untuk metode fusi tingkat proposal adalah:

- Tebakan awal atau proposal diberikan untuk lokasi objek. Misalnya, CNN 2D dapat digunakan untuk menghasilkan kotak pembatas 2D di sekitar lokasi objek.
- Proposal 2D dapat diperluas menjadi proposal 3D menggunakan matriks proyeksi kamera. Ini disebut frustum 3D, yang mewakili volume yang membentang dari ruang kotak pembatas 2D ke ruang 3D yang mungkin ditempati oleh kotak pembatas 2D.

- Data dari LiDAR dapat diintegrasikan ke dalam proposal 3D ini dengan melihat awan titik mana yang berada di dalam setiap frustum, yang pada dasarnya memungkinkan Anda untuk menambahkan informasi kedalaman ke proposal 2D awal.
- Di dalam setiap frustum, data gabungan digunakan untuk lebih menyempurnakan deteksi dan klasifikasi objek. Akhirnya, posisi, dimensi, dan orientasi objek yang tepat di dalam ruang 3D ditentukan untuk memperkirakan kotak pembatas 3D yang tepat untuk objek yang terdeteksi.

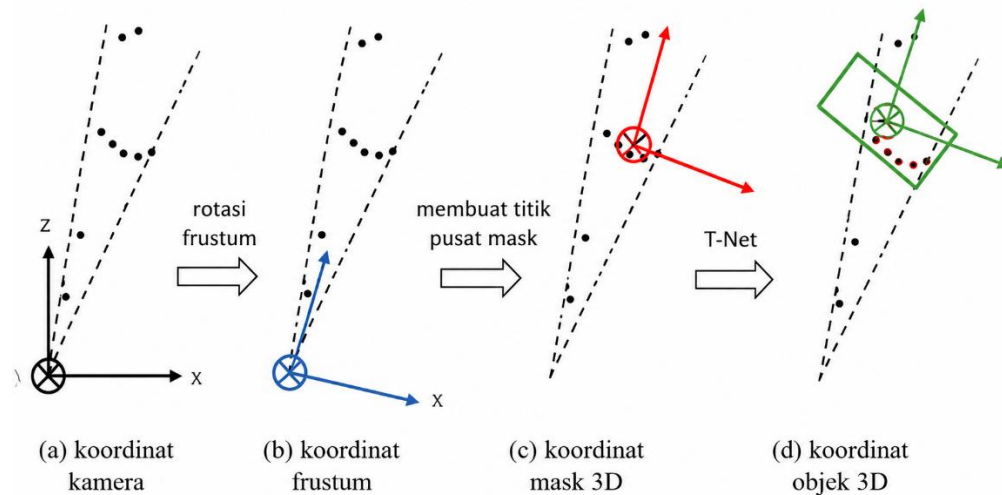
Frustum PointNets menggunakan data RGB-D untuk deteksi objek 3D. Ini dimulai dengan menggunakan CNN 2D untuk mendeteksi objek dalam gambar kamera, menghasilkan kotak pembatas 2D. Kotak pembatas 2D ini diproyeksikan ke ruang 3D, menciptakan frustum. Titik-titik di dalam frustum menciptakan awan titik frustum. Gambaran rinci tentang Frustum PointNet ditunjukkan pada Gambar 3.11.



Gambar 3.11. Komponen utama Frustum PointNet.

Titik-titik LiDAR di dalam setiap frustum diekstrak dan diproses oleh jaringan untuk mengklasifikasikan apakah setiap titik termasuk ke objek atau latar belakang. Jaringan T-Net kemudian menyempurnakan titik-titik ini dengan menggeser centroidnya agar lebih selaras dengan pusat objek yang sebenarnya. Jaringan lain memperkirakan kotak pembatas 3D yang tepat, termasuk posisi, dimensi, dan orientasi objek.

Langkah-langkah ini dan hasilnya diilustrasikan pada Gambar 3.12, yang menunjukkan bagaimana transformasi ini diterapkan untuk mendapatkan deteksi dan penentuan posisi objek akhir dalam ruang 3D.



Gambar 3.12. Transformasi dalam Frustum PointNets. Sistem koordinat kamera default (a). Sistem koordinat frustum diperoleh setelah memutar frustum untuk memusatkan pandangan (b). Koordinat masker 3D menyelaraskan titik pusat masker pada titik asal (c). Koordinat objek 3D diprediksi menggunakan T-Net (d).

Dengan menghasilkan proposal awal dengan satu sensor dan memperhalusnya dengan sensor lain, fusi tingkat proposal memiliki akurasi dan keandalan deteksi dan klasifikasi objek yang lebih baik, yang dapat sangat penting di lingkungan yang kompleks seperti yang dihadapi oleh mobil otonom.

Metode Fusi Tingkat Titik

Di sisi lain, teknik fusi tingkat titik memperkaya data awan titik LiDAR dengan fitur kamera. Langkah-langkah keseluruhan untuk metode fusi tingkat titik meliputi:

- Kalibrasi sensor, yang mencakup penentuan parameter internal (panjang fokus, pusat optik, dll.) dan parameter eksternal (posisi dan orientasi relatif antara kamera dan LiDAR). Ini dapat membantu menyelaraskan kamera dan LiDAR agar berada dalam sistem koordinat yang serupa.
- CNN dapat digunakan untuk mengekstrak fitur utama dari data kamera.
- Setiap titik LiDAR diproyeksikan ke bidang gambar kamera menggunakan parameter kalibrasi. Ini membantu memetakan setiap titik LiDAR ke setiap titik kamera.
- Titik-titik LiDAR dapat dikombinasikan dengan fitur dan metode kamera yang sesuai, seperti PointNet++ dan VoxelNet, dan dapat digunakan untuk melakukan tugas-tugas seperti deteksi objek, segmentasi, dan klasifikasi.

Dalam beberapa kasus, fusi tingkat titik menawarkan representasi yang lebih detail dan akurat karena menggabungkan fitur visual dengan setiap titik LiDAR. Hal ini dapat membantu meningkatkan kemampuan model untuk mendeteksi dan mengklasifikasikan objek.

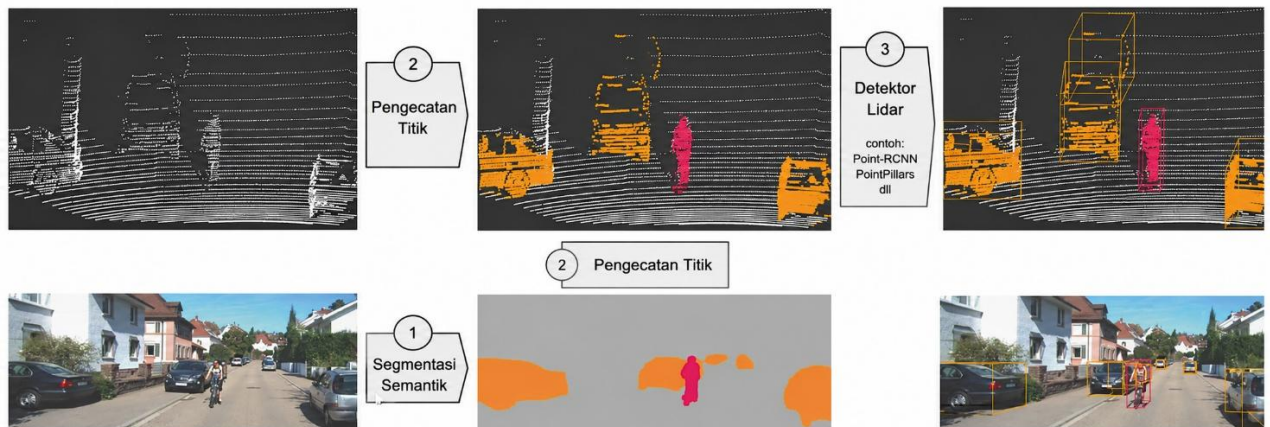
Fusi Tingkat Input: PointPainting dan PointAugmenting

PointPainting adalah contoh metode fusi tingkat titik yang terdiri dari tiga tahap utama:

- Awalnya, gambar disegmentasi, memberikan Anda masker segmentasi yang memberi skor setiap piksel berdasarkan kategori.

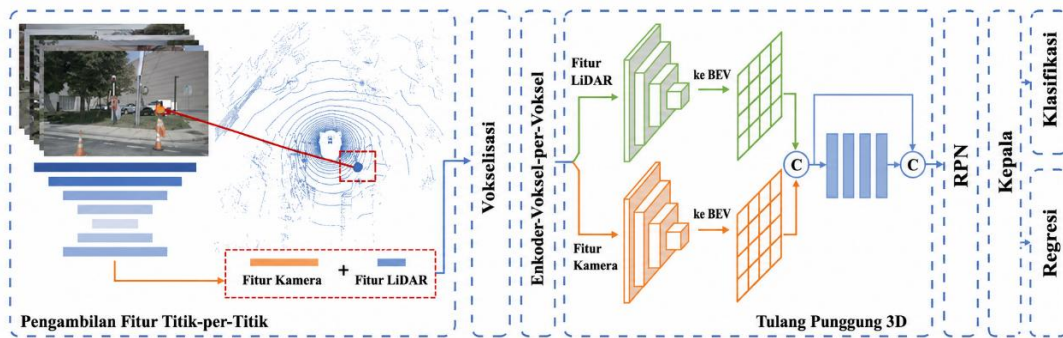
- Titik LiDAR dipetakan ke lokasi yang sesuai dalam masker segmentasi, dan setiap titik diberi skor semantik dari gambar, menambahkan informasi visual kontekstual.
- Tahap akhir menggunakan awan titik dengan informasi semantik tambahan untuk mengidentifikasi dan melokalisasi objek dalam ruang 3D. Awan titik akhir ini, yang sekarang berisi informasi geometris dan semantik, dapat digunakan untuk deteksi objek yang lebih akurat daripada hanya deteksi data LiDAR atau kamera.

Ilustrasi proses ini dapat dilihat pada Gambar 3.13, yang menunjukkan bagaimana PointPainting mengintegrasikan data gambar dan LiDAR untuk meningkatkan akurasi saat melakukan deteksi objek untuk sistem penggerak otonom.



Gambar 3.13. Gambaran umum PointPainting dan tiga tahap utamanya: (1) Jaringan segmentasi semantik berbasis gambar, (2) PointPainting (fusi), (3) Detektor objek berbasis LiDAR.

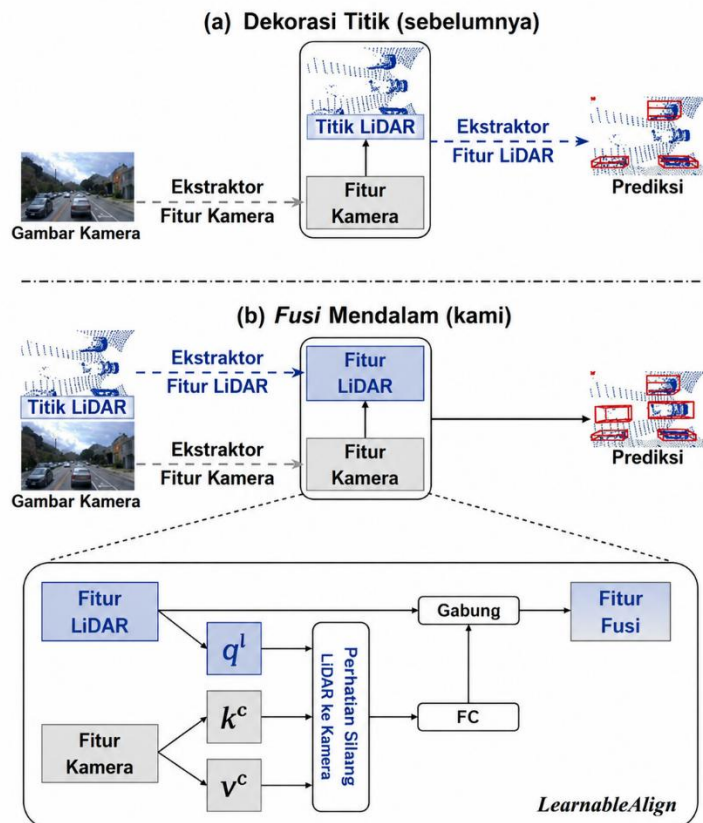
PointAugmenting dibangun berdasarkan metode PointPainting, tetapi alih-alih menggunakan skor kategorikal sederhana dari mask segmentasi, fitur yang diekstrak oleh CNN dipetakan ke titik-titik LiDAR. Ini disebut strategi fusi lintas modal. Strategi ini menggabungkan fitur visual dengan data kedalaman, yang dapat meningkatkan akurasi deteksi objek 3D. PointAugmenting ditunjukkan pada Gambar 3.14, yang mengilustrasikan bagaimana teknik ini menggabungkan dan memanfaatkan data dari sensor LiDAR dan kamera untuk meningkatkan deteksi pada kendaraan otonom.



Gambar 3.14. Gambaran Umum PointAugmenting.

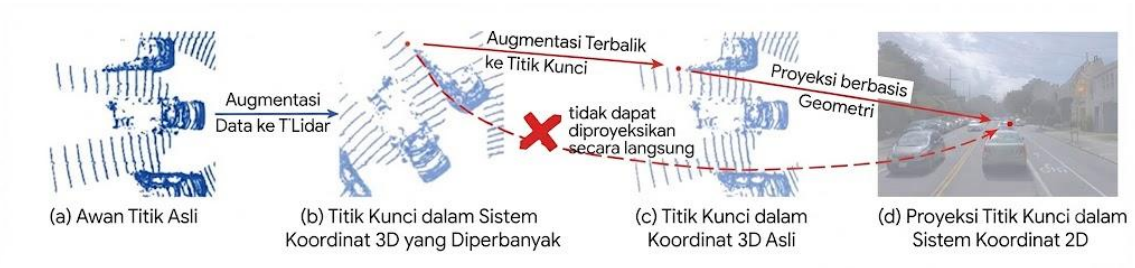
Fusi Tingkat Fitur: DeepFusion

Mekanisme yang efektif untuk menyelaraskan fitur kamera dan LiDAR merupakan komponen penting yang hilang dari pekerjaan yang ada seperti PointPainting dan PointAugmenting. Seperti yang ditunjukkan pada Gambar 3.15, DeepFusion adalah model yang melakukan fusi pada tingkat fitur, artinya menggabungkan fitur yang diekstrak dari data LiDAR dan kamera, dengan tujuan meningkatkan penyelarasan untuk meningkatkan kinerja pada tugas-tugas hilir.

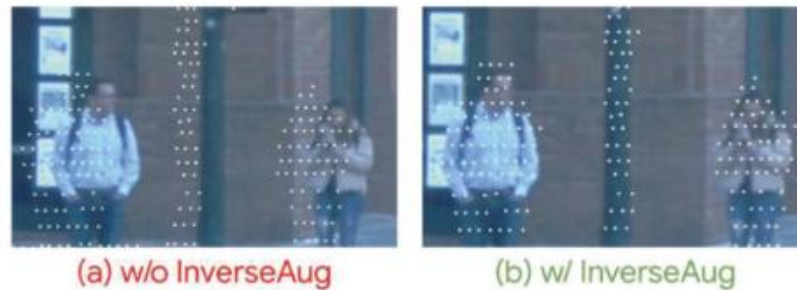


Gambar 3.15. Gambaran Umum DeepFusion (b).

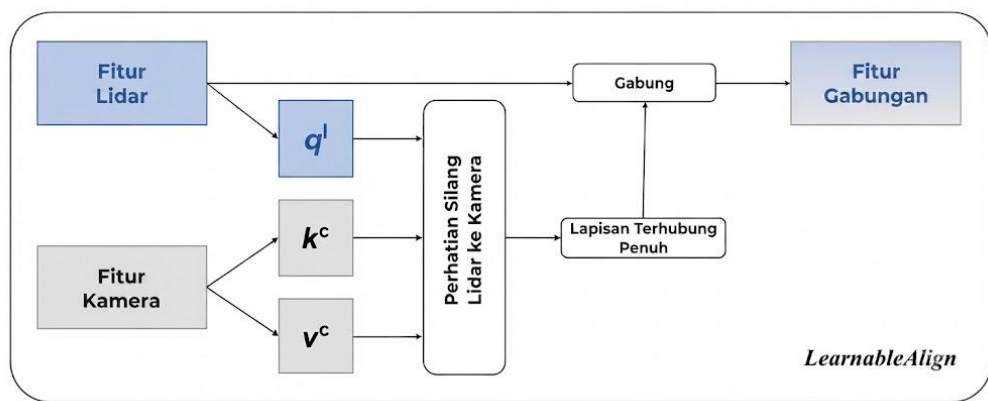
Komponen kunci DeepFusion adalah Augmentasi Terbalik (InverseAug), yang membalikkan augmentasi yang diterapkan selama pelatihan sebelum langkah fusi data. Ide utamanya adalah bahwa ketika awan titik LiDAR diputar atau ditransformasikan selama pelatihan, menjadi sulit untuk mencocokkan titik 3D dengan lokasi 2D yang benar dalam gambar kamera. InverseAug memecahkan masalah ini dengan menyimpan transformasi yang diterapkan dan membalikkannya sebelum menggabungkan data, seperti yang ditunjukkan pada Gambar 3.16. Ini memastikan bahwa titik 3D kembali ke posisi aslinya, sehingga lebih mudah untuk memetakannya ke tampilan kamera 2D. Proses ini diilustrasikan pada Gambar 3.17, menunjukkan penyelarasan baik dengan maupun tanpa penerapan InverseAug.



Gambar 3.16. Gambaran umum metode InverseAug. Awan titik LiDAR asli (a). Setelah menerapkan augmentasi pada titik LiDAR (b). Titik-titik kunci dalam sistem koordinat 3D asli (c). Titik-titik yang diproyeksikan dalam sistem koordinat 2D (d).



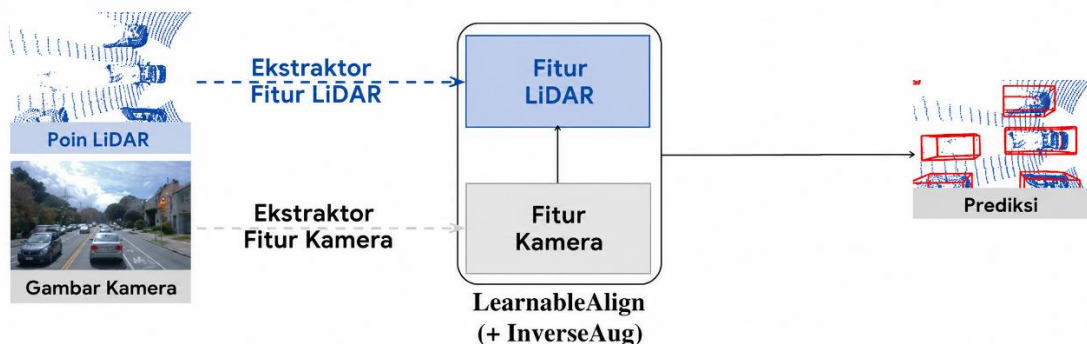
Gambar 3.17. Penyelarasan tanpa InverseAug (a). Penyelarasan dengan InverseAug (b).



Gambar 3.18. Proses Penyelarasan yang Dapat Dipelajari dalam DeepFusion di mana fitur LiDAR dan kamera diselaraskan

Untuk lebih meningkatkan presisi penyelarasan antara berbagai jenis data, DeepFusion memperkenalkan metode yang disebut *Learnable Alignment* (LearnableAlign), seperti yang diilustrasikan pada Gambar 3.18. Ini terutama merupakan mekanisme perhatian lintas modalitas. Mekanisme ini memproses fitur dari data LiDAR (setiap voxel) dan data kamera yang sesuai. Kemudian, algoritma ini menghitung output yang merupakan penjumlahan berbobot dari fitur-fitur kamera, yang berarti memberikan bobot lebih besar pada fitur-fitur kamera yang paling relevan. Dengan berfokus pada fitur-fitur yang paling relevan dari data LiDAR dan kamera melalui perhatian berbobot, LearnableAlign memastikan bahwa penyelarasan fitur lebih tepat, yang membantu meningkatkan akurasi tugas deteksi dan klasifikasi objek.

Dengan menggabungkan semuanya, DeepFusion meningkatkan deteksi objek dengan menggabungkan data LiDAR dan kamera. Pertama, ia mengubah titik LiDAR menjadi fitur yang berguna dan mengekstrak fitur dari gambar kamera menggunakan ResNet. Fitur-fitur ini kemudian diselaraskan dan digabungkan menggunakan InverseAug, yang membalikkan transformasi untuk penyelarasan yang lebih baik, dan LearnableAlign, yang menggunakan mekanisme perhatian untuk fokus pada fitur yang paling relevan. Data yang digabungkan diproses oleh model deteksi 3D, termasuk tulang punggung dan kepala deteksi dari sistem PointPillars, untuk menghasilkan hasil deteksi akhir. Seluruh alur kerja proses ini, dari input data awal hingga produksi output deteksi, ditunjukkan pada Gambar 3.19.



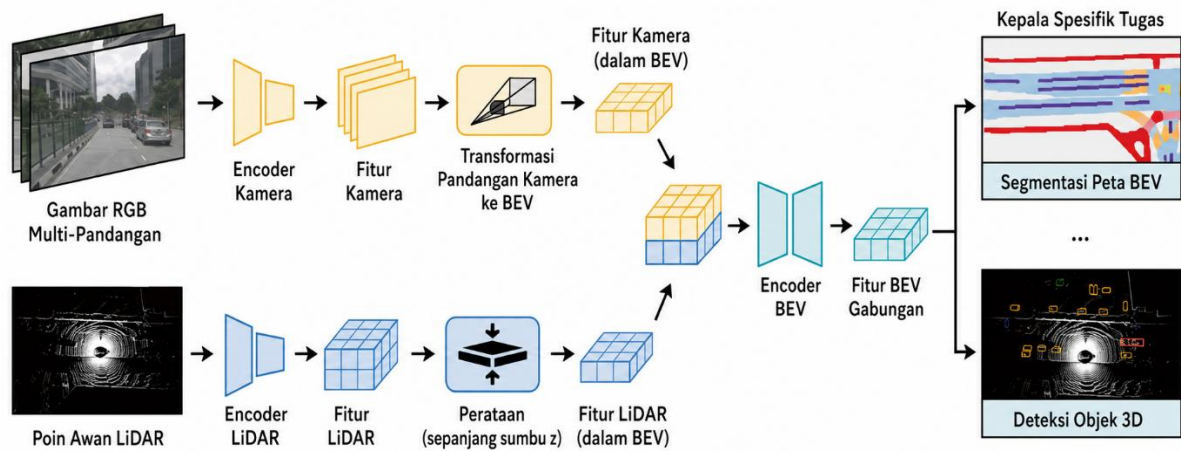
Gambar 3.19. Gambaran umum alur kerja DeepFusion.

BEVFusion

Berdasarkan ide fusi multisensor, BEVFusion mengambil pendekatan berbeda untuk tugas persepsi 3D. Ia mengubah input dari kedua sensor menjadi fitur dan mentransformasikannya ke ruang pandangan mata burung (BEV), yang merupakan perspektif dari atas ke bawah lingkungan. Pandangan terpadu ini memudahkan integrasi data. Encoder BEV kemudian memproses fitur gabungan ini, yang digunakan oleh komponen khusus untuk tugas tertentu untuk melakukan berbagai tugas persepsi 3D, seperti deteksi dan pelacakan objek, seperti yang diilustrasikan pada Gambar 3.20.

Jika Anda ingin mempelajari lebih lanjut tentang pendekatan ini, kami sarankan untuk membaca makalah aslinya. Namun, manfaat utama BEV adalah memberikan perspektif top-down yang konsisten dan terpadu yang menyederhanakan fusi data LiDAR dan kamera. Karena

data berada dalam ruang bersama, terdapat peningkatan keselarasan spasial yang dapat meningkatkan akurasi tugas persepsi 3D.



Gambar 3.20. Proses BEVFusion mengekstrak fitur dari berbagai input dan mengubahnya menjadi ruang BEV bersama.

Secara keseluruhan, fusi sensor dapat menggabungkan data dari banyak sensor, seperti kamera dan LiDAR, untuk memanfaatkan kekuatan komplementernya. Integrasi ini dapat meningkatkan kinerja tugas persepsi robot umum seperti deteksi, segmentasi, dan pelacakan.

3.3 RINGKASAN

Dalam bab ini Anda telah mempelajari:

- Bagaimana data 3D dapat direpresentasikan sebagai voxel (piksel 3D dalam grid), point cloud (kumpulan titik 3D), dan mesh (permukaan poligonal). Point cloud dapat diproses menggunakan metode seperti PointNet, PointNet++, PointCNN, dan Dynamic Graph CNN (DGCNN).
- Terdapat dua strategi fusi utama: fusi awal (menggabungkan data sensor mentah seperti point cloud dan gambar secara langsung) dan fusi akhir (menggabungkan data yang telah diproses seperti bounding box dari sensor yang berbeda).
- Metode fusi LiDAR-kamera menggabungkan data dari LiDAR dan kamera. Teknik seperti Frustum PointNets, bersama dengan metode fusi tingkat titik seperti PointPainting dan PointAugmenting, meningkatkan deteksi dengan memproyeksikan titik LiDAR ke gambar kamera.
- Pendekatan fusi tingkat fitur seperti DeepFusion menyelaraskan fitur kamera dan LiDAR menggunakan teknik seperti InverseAug dan LearnableAlign. Selain itu, BEVFusion mengubah data sensor menjadi tampilan dari atas (*bird's-eye view*/BEV) yang terpadu, sehingga meningkatkan deteksi dan pelacakan objek 3D.

Bab selanjutnya membahas bagaimana model bahasa besar (*large language models/LLM*) diterapkan pada perencanaan, kontrol, dan pemetaan robot, termasuk penggunaan model dasar, pendekatan multimodal, dan metode kontrol robot ujung-ke-ujung serta model difusi.

BAB 4

MODEL DASAR DALAM ROBOTIKA

Model dasar yang dikembangkan untuk studi bahasa kini dengan berani mengambil alih banyak bidang, termasuk robotika. Banyak kemajuan ini didorong oleh upaya membuat informasi tampak seperti bahasa, yaitu membuat informasi menyerupai urutan token. Anda telah mempelajari di Bab 2 bagaimana studi visi kini sepenuhnya didominasi oleh transformer dengan tokenisasi input (gambar dan output) agar tampak seperti token. Model dasar dan perkembangannya dalam robotika secara alami tumbuh sebagai hasil dari penggunaan bahasa dalam robotika. Bab ini pertama-tama menjelaskan, secara singkat, bagaimana model dasar yang besar dilatih. Kemudian, bab ini membahas bagaimana bahasa semakin banyak digunakan sebagai penghubung dalam robotika dan bagaimana studi tersebut berkembang menjadi kontrol robot ujung-ke-ujung dengan transformer besar. Terakhir, bab ini membahas perkembangan model difusi dan aplikasinya dalam kontrol robot.

4.1 MODEL BAHASA BESAR: PELATIHAN, PENSKALAAN, DAN EVALUASI

Studi bahasa telah memimpin dalam pembelajaran mesin selama dekade terakhir. Beberapa model bahasa terbaru berdasarkan transformer skala besar telah menjadi berita, termasuk GPT-4, Gemini, dan Llama. Mereka adalah produk dari serangkaian studi panjang yang dimulai dari Bert, GPT sebelumnya, PaLM, dan banyak lagi. Bagian ini membahas bagaimana model multimodal besar dilatih dengan Llama sebagai contoh.

Pengembangan model multimodal besar melibatkan dua langkah utama:

1. **Pra-pelatihan:** Pada tahap ini model dilatih pada sejumlah besar data multibahasa dari Internet. Dalam kasus Llama 3, model 405 B dilatih pada 15,6 T token, dengan kosakata token 128K, di mana 28K token kosakata mendukung bahasa non-Inggris. Transformer Llama menggunakan perhatian kueri yang dikelompokkan dan cache nilai kunci untuk meningkatkan kecepatan inferensi (lihat sumber daya untuk mempelajari lebih lanjut tentang topik ini).
2. **Pasca-pelatihan:** Pada tahap ini, model ditingkatkan dalam mengikuti instruksi dengan menggunakan kumpulan data berkualitas tinggi yang lebih kecil yang berasal dari anotasi manusia dan data sintetis.

Llama 3 menggunakan pendekatan komposisional untuk multimodalitas dan memiliki encoder visi dan ucapan yang terpisah.

1. **Encoder gambar dan adaptor:** Encoder gambar dilatih pada korpus besar pasangan teks gambar, mirip dengan CLIP, yang dibahas dalam Bab 2, untuk menghasilkan representasi yang menyadari konten visual dan representasi bahasanya. Dua adaptor terpisah dilatih untuk gambar dan video pada pasangan teks gambar dan teks video masing-masing, selama pelatihan model bahasa yang telah dilatih sebelumnya tetap dibekukan, sementara bobot encoder gambar yang telah dilatih sebelumnya diedit. Adaptor ini menggunakan perhatian silang untuk memasukkan representasi encoder

gambar yang telah dilatih sebelumnya ke dalam model bahasa yang telah dilatih sebelumnya.

2. **Encoder ucapan dan adaptor gambar:** Mirip dengan pengaturan gambar, encoder ucapan terpisah dilatih dan adaptor ucapan dilatih dengan perhatian silang untuk menyelaraskan embedding dari encoder ucapan yang telah dilatih sebelumnya dan model bahasa besar yang telah dilatih sebelumnya.

Pelatihan awal memiliki tiga komponen utama:

1. **Kurasi data web:** Melibatkan pembuatan dataset besar berkualitas tinggi sambil menerapkan teknik untuk deduplikasi, penghapusan informasi identitas pribadi dan konten dewasa melalui penyaringan heuristik dan pembelajaran.
2. **Penentuan campuran data:** Eksperimen hukum penskalaan dilakukan untuk memprediksi kinerja model besar dengan menginterpolasi kinerja beberapa model yang lebih kecil untuk mencapai campuran data yang optimal, dan untuk memahami bagaimana dataset yang berbeda berkontribusi/memengaruhi campuran model besar. Pada akhirnya, mereka sampai pada 50 persen token pengetahuan umum, 25 persen token matematika dan penalaran, 17 persen token kode, dan 8 persen data multibahasa.
3. **Pra-pelatihan awal:** Pra-pelatihan Llama menggunakan jadwal laju pembelajaran kosinus dengan peluruhan dan pemanasan, dan dimulai dengan ukuran batch dan panjang urutan yang lebih kecil, yang secara bertahap ditingkatkan seiring berjalannya pra-pelatihan.
4. **Pra-pelatihan konteks panjang:** Pra-pelatihan pada konteks panjang diperkenalkan selama tahap akhir pra-pelatihan, karena komputasi di lapisan self-attention tumbuh secara kuadratik dengan panjang konteks. Panjang konteks yang didukung ditingkatkan secara bertahap, dengan adaptasi terhadap peningkatan panjang konteks diukur dengan pemulihan kinerja pada panjang konteks yang lebih pendek dan tes jarum di tumpukan jerami untuk peningkatan panjang konteks.
5. **Annealing:** Pelatihan pada sejumlah kecil data matematika dan kode berkualitas sangat tinggi ditemukan dapat meningkatkan penalaran secara keseluruhan untuk model yang lebih kecil.

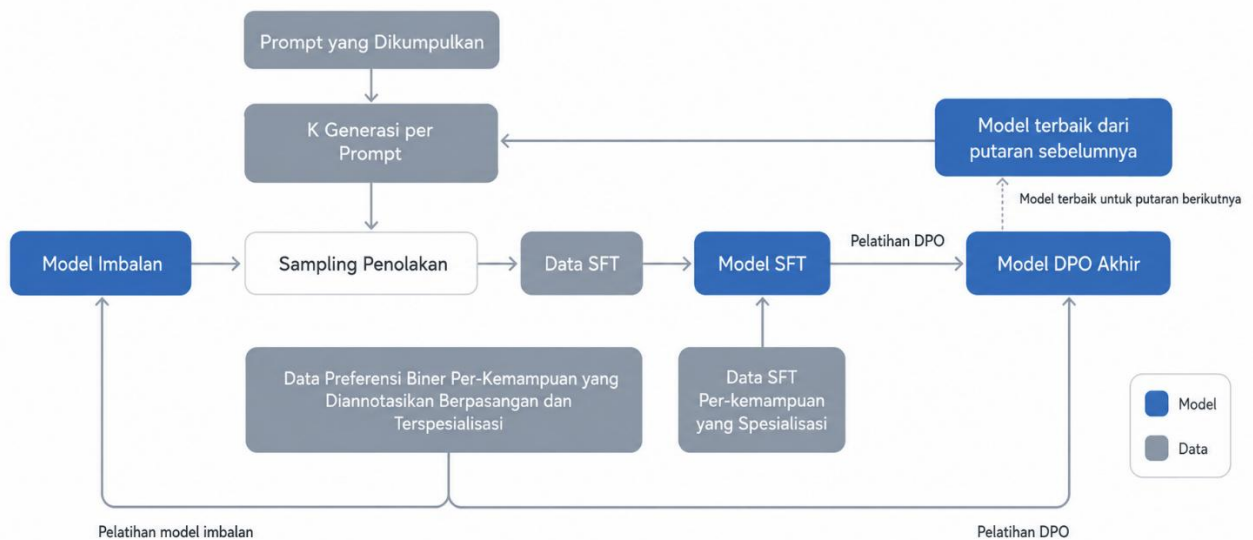
Pasca-pelatihan memiliki dua komponen utama: model penghargaan dan model bahasa. Ini menggunakan fine-tuning terawasi dan optimasi preferensi langsung (DPO). Contoh strategi pasca-pelatihan ditunjukkan pada Gambar 4.1. Pertama, dataset preferensi dibangun dengan memberi perintah pada model dan menghasilkan output yang kemudian dianotasi oleh manusia sesuai urutan preferensi mereka. Annotator manusia juga dapat mengedit output yang dipilih, dan mekanisme pemeringkatan (diedit ➤ dipilih ➤ ditolak) digunakan untuk memberi peringkat preferensi.

1. **Penyesuaian halus terawasi (SFT):** Selama tahap ini, model penghargaan dilatih pada preferensi. Model penghargaan ini kemudian digunakan dalam skala besar dengan memberi perintah pada model bahasa yang telah dilatih sebelumnya dan menolak output berdasarkan penghargaan untuk membuat dataset sampel penolakan dari

output LLM. Data sintetis tambahan dihasilkan untuk kode dan matematika dengan menerapkan berbagai strategi:

- Melatih pakar kode dengan mencabangkan proses pra-pelatihan dan melatihnya pada token kode berkualitas tinggi. Intuisinya di sini adalah bahwa pra-pelatihan berkelanjutan pada data spesifik domain meningkatkan kinerja di domain tersebut. Pakar ini kemudian digunakan untuk membuat generasi berkualitas tinggi yang membentuk dataset.
- Penerjemah kode dan linting digunakan untuk meningkatkan kualitas generasi sintetis. Koreksi diri melalui penalaran rantai pemikiran digunakan untuk lebih meningkatkan kualitas dataset.

Model bahasa yang telah dilatih sebelumnya kemudian disempurnakan pada dataset berkualitas tinggi yang diperoleh melalui generasi sintetis dan pengambilan sampel penolakan dengan model penghargaan yang dipelajari.



Gambar 4.1. Strategi pasca-pelatihan LLM.

- Optimasi preferensi langsung (DPO):** Sementara metode sebelumnya dalam penyediaan instruksi model bahasa menggunakan pembelajaran penguatan pada umpan balik manusia (RLHF), DPO semakin populer sekarang karena lebih sederhana. Dengan tidak mempelajari model penghargaan perantara, dan melewati beberapa ketidakstabilan dalam pelatihan model RL aktor-kritik. Bab 7 membahas RLHF secara detail. DPO malah secara langsung mengoptimalkan penghargaan yang dibatasi oleh divergensi KL dengan secara langsung meningkatkan log likelihood relatif dari respons yang disukai dibandingkan dengan yang tidak disukai. Ukuran divergensi KL dalam DPO menyeimbangkan pertukaran antara memaksimalkan penghargaan dengan penyesuaian preferensi dengan meminimalkan divergensi dari model dasar yang telah dilatih sebelumnya.

Llama3 diadaptasi untuk penggunaan alat dengan memiliki format obrolan di mana setiap pesan memiliki header yang menentukan sumber dan tujuan pesan. Setiap pesan juga memiliki token penghentian yang menentukan kapan harus berganti antara AI dan manusia.

Hukum Skala untuk Model Bahasa

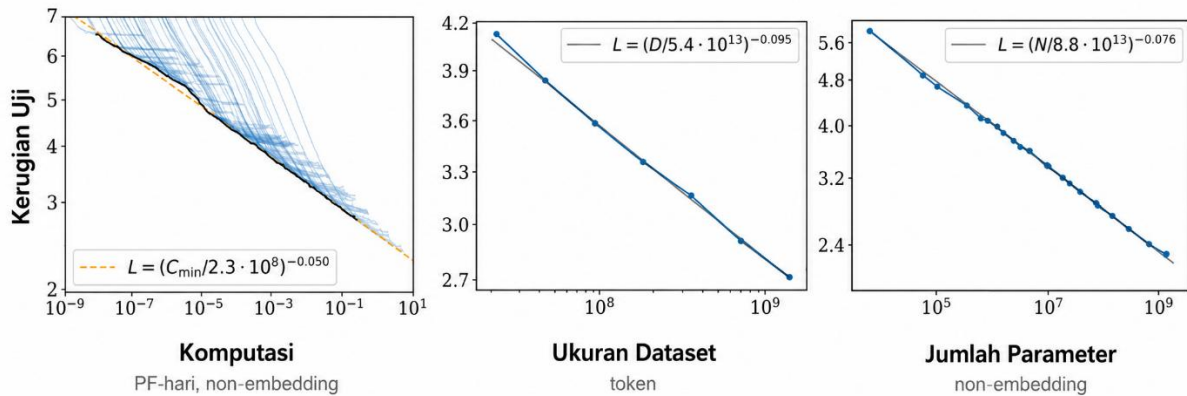
Hukum skala untuk model bahasa memprediksi kinerja atau kerugian pengujian apa yang dapat dicapai saat menskalakan data, komputasi, atau ukuran parameter model. Karena menjalankan pelatihan model besar sangat mahal, memahami perilaku penskalaan membantu memprediksi kemampuan model bahasa dan membuat pertimbangan yang tepat saat berinvestasi dalam menjalankan pelatihan model besar. Perhatikan bahwa hampir semua hukum penskalaan neural ditentukan secara empiris dan didekati melalui interpolasi.

Kaplan menemukan hukum penskalaan neural pertama untuk model bahasa besar dengan menjalankan banyak model yang lebih kecil dan menginterpolasi perilakunya. Mereka mempertimbangkan tiga aspek skala: jumlah parameter model N , ukuran dataset D , dan jumlah komputasi yang digunakan C . Mereka menemukan bahwa kerugian model memiliki hubungan hukum pangkat dengan masing-masing dari tiga faktor tersebut ketika tidak dibatasi oleh faktor lainnya.

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha N} ; \alpha N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13} \text{ (parameter non – embedding)}$$

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha D} ; \alpha D \sim 0.095, \quad D_c \sim 5.4 \times 10^{13} \text{ (token)}$$

$$L(C_{min}) = \left(\frac{C_c^{min}}{C_{min}}\right)^{\alpha_c^{min}} ; \alpha_c^{min} \sim 0.050, \quad C_c^{min} \sim 3.1 \times 10^8 \text{ (PF – hari)}$$



Gambar 4.2. Kerugian pengujian sebagai fungsi komputasi, ukuran dataset, dan ukuran model.

Ini menunjukkan bahwa ada batas efisien yang menyeimbangkan ukuran model, ukuran dataset, dan komputasi untuk mencapai kinerja terbaik dengan jumlah sumber daya yang diberikan. Mereka juga menemukan bahwa model yang lebih besar lebih efisien dalam penggunaan sampel daripada model yang lebih kecil dan bahwa pelatihan hingga konvergensi

bersifat suboptimal. Hasilnya ditunjukkan pada Gambar 4.2. Hoffman dua tahun kemudian, memperkenalkan hukum penskalaan optimal Chinchilla. Mereka mencoba menjawab pertanyaan, dengan anggaran FLOPs yang tetap, apa pertukaran antara ukuran model dan ukuran dataset?

Mereka mencoba dua pendekatan:

1. Tetapkan ukuran model dan latih dengan ukuran dataset yang berbeda untuk memplot kerugian.
2. Tetapkan ukuran model dan latih untuk jumlah FLOP yang berbeda untuk memplot kerugian.

Berdasarkan hasil dari kedua pendekatan tersebut, mereka menyesuaikan persamaan parametrik (Persamaan 4.1) untuk data. Penyesuaian yang diusulkan adalah:

$$\hat{L}(N, D) \cong E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}.$$

Persamaan 4.1

Kemudian mereka memperkirakan parameter ini dengan meminimalkan kerugian Huber antara kerugian log yang diprediksi dan kerugian log yang diamati. Sebuah studi selanjutnya oleh Epoch AI yang mencoba mereplikasi hasil dari Hoffman et al. menghasilkan versi koreksi dari estimasi untuk parameter ini, seperti yang ditunjukkan pada Tabel 4.1.

Tabel 4.1. Estimasi Fungsi yang Hilang dari Upaya Replikasi Chinchilla Epoch AI dan Estimasi Asli Chinchilla. Angka dalam Kurung Adalah Deviasi Standar.

Parameter	Estimasi kami	Estimasi Hoffman et al.
A	482,01 (124,58)	406,4
B	2085,43 (1293,23)	410,7
E	1,82 (0,03)	1,69
α	0,35 (0,02)	0,34
β	0,37 (0,02)	0,28
$a = \beta/(\alpha + \beta)$	0,512 (0,018)	0,454
Titik data	240	> 400

Berdasarkan hal ini, alokasi komputasi yang hampir optimal dapat dipetakan ke N dan D secara terpisah, sebagai berikut:

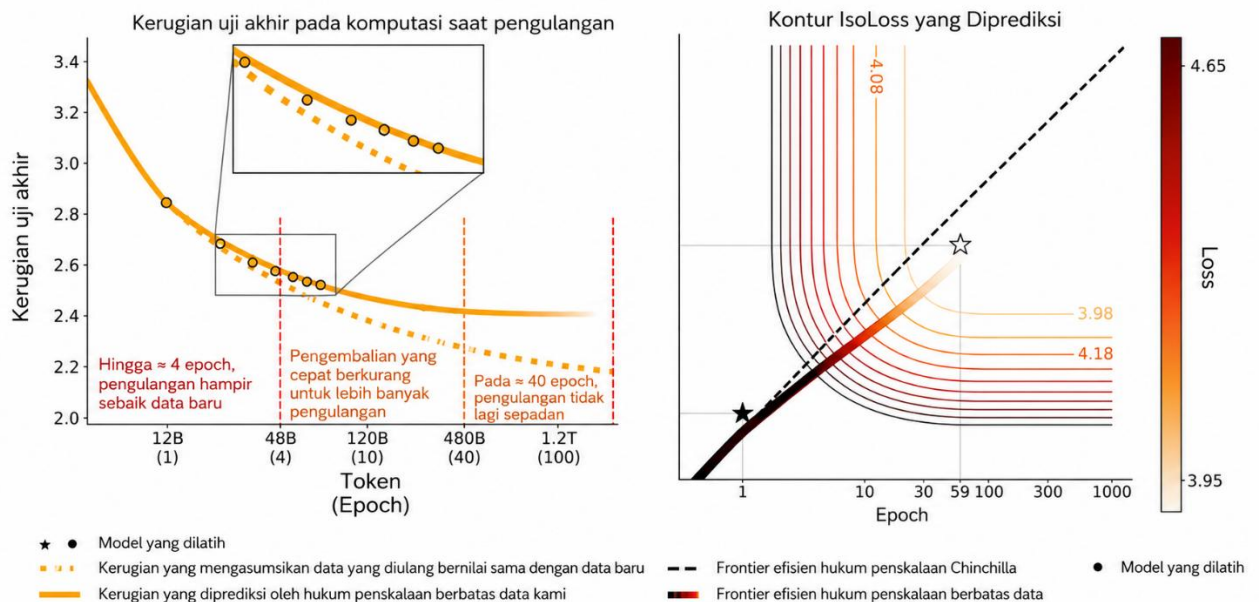
$$N_{opt}(C) = G \left(\frac{C}{6} \right)^a \quad D_{opt}(C) = G^{-1} \left(\frac{C}{6} \right)^b$$

$$\text{where } G = \left(\frac{\alpha A}{\beta B} \right)^{\frac{1}{\alpha + \beta}} \quad a = \frac{\beta}{\alpha + \beta} \quad b = \frac{\alpha}{\alpha + \beta}$$

Persamaan 4.2

Hasil dari Chinchilla menunjukkan bahwa banyak model terdepan yang dilatih pada saat itu, seperti GPT3, kurang terlatih pada data, sehingga model optimal akan jauh lebih kecil (15B dari 175B) dalam jumlah parameter. Dari hasil tersebut juga dapat disimpulkan bahwa, untuk anggaran komputasi tetap, penskalaan data jauh lebih efektif daripada penskalaan parameter model.

Muennighoff mencoba memprediksi hukum penskalaan dalam rezim yang dibatasi data dan menyimpulkan bahwa pengulangan data hingga empat epoch tidak menghasilkan perbedaan dalam kerugian, dan untuk mengoptimalkan kinerja saat mengulang data, pelatihan model yang lebih kecil untuk waktu yang lebih lama lebih disukai. Hal ini ditunjukkan pada Gambar 4.3.



Gambar 4.3. Atas: Kerugian LLM yang dilatih pada data berulang berkurang secara dapat diprediksi. Bawah: Bagaimana hukum penskalaan yang dibatasi data menyimpang dari penskalaan optimal yang tidak dibatasi: peningkatan jarak kontur menunjukkan penurunan keuntungan dari data berulang.

Studi terbaru dari Sorscher et al. menunjukkan bahwa hubungan hukum pangkat antara kerugian pelatihan dan ukuran token dataset dapat dipatahkan dengan memberi peringkat contoh dari yang sulit ke yang mudah pada metrik berkualitas tinggi dan kemudian memangkas contoh yang mudah dari dataset. Mereka berpendapat bahwa pemangkasan tersebut bahkan dapat menyebabkan hubungan eksponensial.

Mengevaluasi Model Bahasa

Sekarang Anda memahami bagaimana model bahasa dilatih dan diskalakan, Anda akan mempelajari bagaimana model tersebut dievaluasi. Studi model bahasa sebagian besar menggunakan benchmark untuk mengevaluasi kinerja. Berikut adalah beberapa tolok ukur yang banyak digunakan, diurutkan berdasarkan kategori dan apa yang diujinya:

1. Tolok Ukur Umum

- MMLU (*Massive Multitask Language Understanding*): Mengevaluasi pemahaman dan penalaran di 57 disiplin ilmu akademik dengan 14.000+ pertanyaan pilihan ganda.
- MMLU-Pro: Versi MMLU yang lebih menantang dengan tugas penalaran yang lebih sulit.
- AGIEval: Menilai kecerdasan umum model AI di berbagai tugas.

2. Matematika dan Penalaran

- GSM8K: Menguji pemecahan masalah matematika tingkat sekolah dasar dengan 8.000 tugas penalaran langkah demi langkah.
- MATH: Mengevaluasi kemampuan pemecahan masalah matematika tingkat sekolah menengah dan kompetisi.
- ARC-C (AI2 Reasoning Challenge - Challenge Set): Menilai penalaran kompleks dan keterampilan pemecahan masalah.

3. Pemahaman Akal Sehat

- CommonSenseQA, OpenBookQA, PiQA, SiQA: Mengevaluasi penalaran akal sehat dan pemahaman logis dalam skenario sehari-hari.
- CommonSenseQA: Tolok ukur untuk mengevaluasi penalaran akal sehat, menguji model pada pertanyaan pilihan ganda.
- OpenBookQA: Menguji jawaban pertanyaan buku terbuka dengan fokus pada pertanyaan sains yang membutuhkan pengetahuan tambahan di luar teks.
- PiQA: Mengevaluasi penalaran akal sehat fisik dengan mengharuskan pilihan penyelesaian yang paling masuk akal untuk skenario tertentu.
- SocialIQA (SiQA): Berfokus pada penalaran akal sehat tentang situasi sosial dan interaksi manusia.

4. Kode

- HumanEval, MBPP (Benchmark Multi-Tugas untuk Masalah Pemrograman): Menilai kemampuan pengkodean dan pemecahan masalah dalam konteks pemrograman.

5. Pemahaman Bacaan

- SQuAD (Dataset Tanya Jawab Stanford): Menguji pemahaman bacaan dan ekstraksi informasi dari artikel Wikipedia.
- QuAC (Tanya Jawab dalam Konteks): Mengevaluasi pemahaman kontekstual dan koherensi dalam menjawab pertanyaan percakapan.
- RACE (*Reading Comprehension from Examinations*): Menilai pemahaman bacaan menggunakan soal ujian Bahasa Inggris tingkat SMP dan SMA.

6. Tolok Ukur Konteks Panjang

- Needle-in-a-Haystack: Menguji pemahaman dokumen dengan fokus pada pencarian informasi yang relevan dalam teks panjang.
- ZeroSCROLLS: Menilai kemampuan model untuk melakukan berbagai tugas NLP menggunakan anotasi data minimal.

- InfiniteBench: Berfokus pada pemahaman dokumen panjang dan tugas penalaran, termasuk peringkasan dan menjawab pertanyaan.

7. Tolok Ukur Penggunaan Alat

- API-Bank, API-Bench: Mengevaluasi kemampuan untuk berinteraksi dengan dan memanfaatkan berbagai API eksternal secara efektif.

Selain itu, mereka juga dievaluasi pada tes AP, GMAT, GRE, ujian pengacara, dan ujian medis serta berbagai tes khusus lainnya.

Bagian ini membahas bagaimana model bahasa dilatih, bagaimana skalabilitasnya, dan berdasarkan apa model tersebut dievaluasi. Bagian selanjutnya menjelaskan bagaimana model bahasa digunakan dalam robotika. Anda akan mempelajari bahwa hampir semua data robotika dapat dimodelkan sebagai token dan urutan serta diproses seperti bahasa untuk berbagai kasus penggunaan hilir.

4.2 BAHASA SEBAGAI JARINGAN PENGHUBUNG DALAM ROBOTIKA

Bahasa telah lama digunakan sebagai media komunikasi antar manusia, sedemikian rupa sehingga sebagian besar kekayaan pengetahuan dan pengalaman manusia dikodekan dalam bahasa. Bahasa juga cukup fleksibel dan cair, memungkinkan komunikasi berbagai topik dengan tingkat presisi tertentu. Munculnya transformer menyebabkan terobosan dalam pemahaman bahasa, pembangkitan bahasa, dan penalaran dengan bahasa. Dalam beberapa tahun terakhir, bahasa dalam robotika telah menjadi bidang studi baru yang menarik dan menjanjikan, memungkinkan robot untuk melakukan generalisasi yang lebih besar dengan pemahaman konteks dunia yang didasarkan pada bahasa. Model bahasa telah menyediakan metode untuk fusi multimodal modalitas yang sederhana dan dapat diperluas, termasuk penglihatan dan tindakan.

Robotika yang dikondisikan bahasa adalah gagasan untuk menentukan tujuan robot dalam bahasa alami dan mengukur keberhasilan berdasarkan pencapaian tujuan tersebut dengan benar. Pengkondisian bahasa dapat digunakan untuk perencanaan (“bagaimana Anda akan mengambilkan saya minuman ringan dari kulkas”), kontrol (“ambil kaleng minuman ringan”), pemetaan (“di mana kaleng minuman ringan itu”), dan navigasi (“pergi ke kaleng minuman ringan”). Bagian ini menjelaskan bahwa hampir semua data robotika dapat dimodelkan sebagai token dan urutan dan diproses seperti bahasa dengan cara yang serupa untuk berbagai kasus penggunaan hilir.

Bahasa untuk Perencanaan

Ketika manusia berinteraksi dengan robot dalam bahasa alami, instruksi tipikal dapat berupa “ambilkan saya minuman dari kulkas,” yang terdiri dari beberapa langkah seperti “pergi ke kulkas,” “buka pintu kulkas,” “ambil minuman,” “tutup kulkas,” “pergi ke manusia,” dan “letakkan minuman.” Ini mengharuskan robot untuk melakukan penalaran yang terwujud dalam lingkungan yang tidak terlihat. Jika kita bertanya kepada model bahasa, misalnya, “bagaimana Anda membersihkan tumpahan,” model tersebut mungkin memberikan langkah-langkah yang tidak dapat diakses oleh robot.

Sebagai contoh, sistem dapat mengatakan "temukan penyedot debu" sebagai salah satu langkah, tetapi jika penyedot debu tidak ada di tempat kejadian, ini akan menjadi rencana yang salah. Mode kegagalan kedua dapat terjadi ketika LLM menyarankan rencana yang layak tetapi bukan bagian dari keahlian robot, misalnya, "gunakan penyedot debu," padahal penyedot debu ada tetapi robot belum dilatih untuk menggunakannya. Salah satu tantangan interaksi robot-manusia adalah bahwa robot memiliki serangkaian keterampilan terbatas yang telah dilatihkan kepada mereka dan kosakata instruksi tugas yang disampaikan kepada mereka perlu dibatasi oleh keahlian mereka.

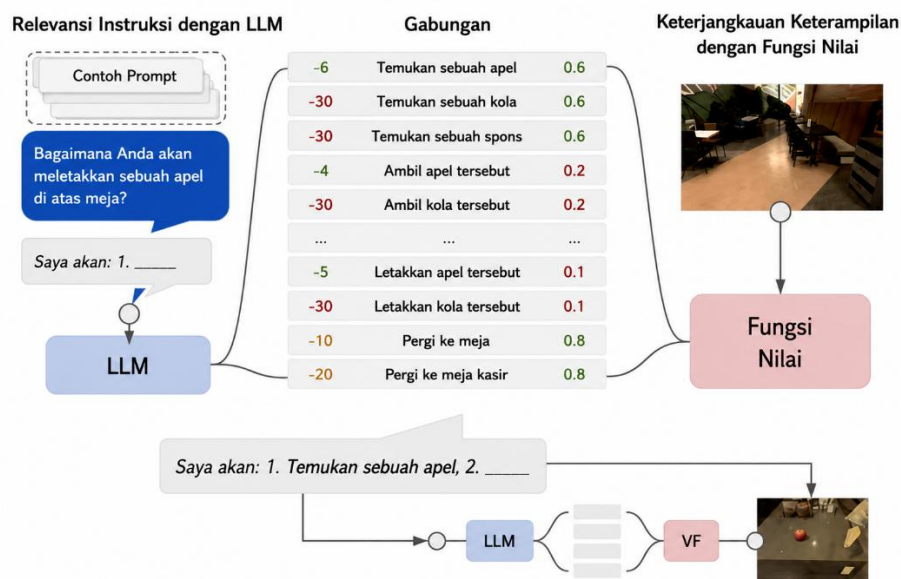
Properti yang mengacu pada seberapa layak dan sukses suatu tugas dalam skenario tertentu secara formal disebut sebagai *affordance*, biasanya dinyatakan sebagai nilai float antara 0,0 dan 1,0, di mana 0,0 berarti sangat tidak mungkin dan 1,0 berarti mungkin dan sangat mungkin berhasil.

Terdapat dua kelas pendekatan perencanaan:

1. **Pembelajaran dalam konteks:** Di sini, model bahasa diperlihatkan contoh rencana robot dalam konteks melalui pemberian petunjuk dan diminta untuk menghasilkan rencana baru. Model bahasa tidak secara khusus disesuaikan untuk pembelajaran robot. Contoh algoritma pembelajaran dalam konteks adalah SayCan/Inner Monologue, yang dijelaskan di bagian selanjutnya.
2. **Model perencanaan yang disesuaikan/dipelajari:** Di sini, model bahasa tujuan umum/multimodal dilatih/disesuaikan secara khusus untuk tujuan perencanaan robot. Roboplan dan PaLM-E adalah contohnya.

SayCan Loop Terbuka

SayCan adalah algoritma yang mengintegrasikan bahasa untuk perencanaan robot. Gambaran umum SayCan ditunjukkan pada Gambar 4.4.



Gambar 4.4. SayCan menggabungkan probabilitas dari LLM untuk menentukan keterampilan mana yang berguna untuk instruksi tertentu dengan probabilitas dari fungsi nilai (VF) yang menilai kelayakan pelaksanaan keterampilan tersebut.

Ide utamanya adalah sebagai berikut:

- Model bahasa digunakan untuk memecah instruksi tugas tingkat tinggi menjadi primitif keterampilan yang kemudian dapat dieksekusi oleh kebijakan manipulasi terlatih. LLM memodelkan probabilitas bahwa instruksi tugas tingkat tinggi dicapai melalui komposisi primitif keterampilan.
- Untuk mendasarkan model bahasa pada ruang keterampilan robot, model affordance kedua dipelajari secara independen. Model affordance mempelajari probabilitas keberhasilan primitif keterampilan berdasarkan keadaan robot dan lingkungan saat ini. Fungsi nilai merupakan perkiraan yang mendekati model affordance tersebut dan dapat dilatih menggunakan pembelajaran penguatan.
- Kedua probabilitas tersebut dikalikan untuk setiap primitif keterampilan dan yang memiliki nilai maksimum dipilih untuk dieksekusi, seperti yang ditunjukkan di sini.

$$p_{\pi}^{LLM} = p(l_{\pi}|i, l_{\pi_{n-1}}, \dots, l_{\pi_o})$$

Mengevaluasi penilaian LLM

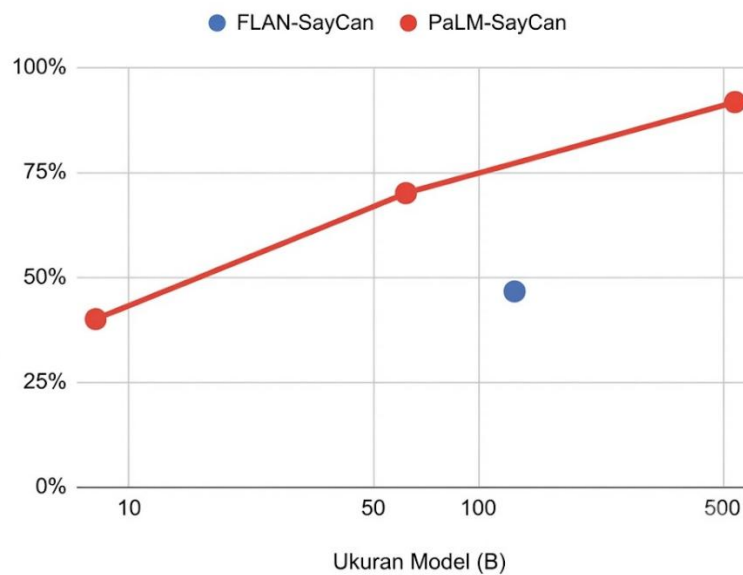
$$p_{\pi}^{affordance} = p(c_{\pi}|s_n, l_{\pi})$$

Mengevaluasi fungsi affordance

$$p_{\pi}^{combined} = p_{\pi}^{affordance} p_{\pi}^{LLM}$$

Pengamatan menarik lainnya dari studi ini adalah bahwa hanya dengan memperbarui model bahasa yang mendasarinya, robot menjadi lebih baik dalam perencanaan. Seperti yang ditunjukkan pada Gambar 4.5, PaLM-SayCan telah terbukti memiliki peningkatan akurasi perencanaan ketika model bahasa ditingkatkan skalanya.

Kinerja Perencanaan



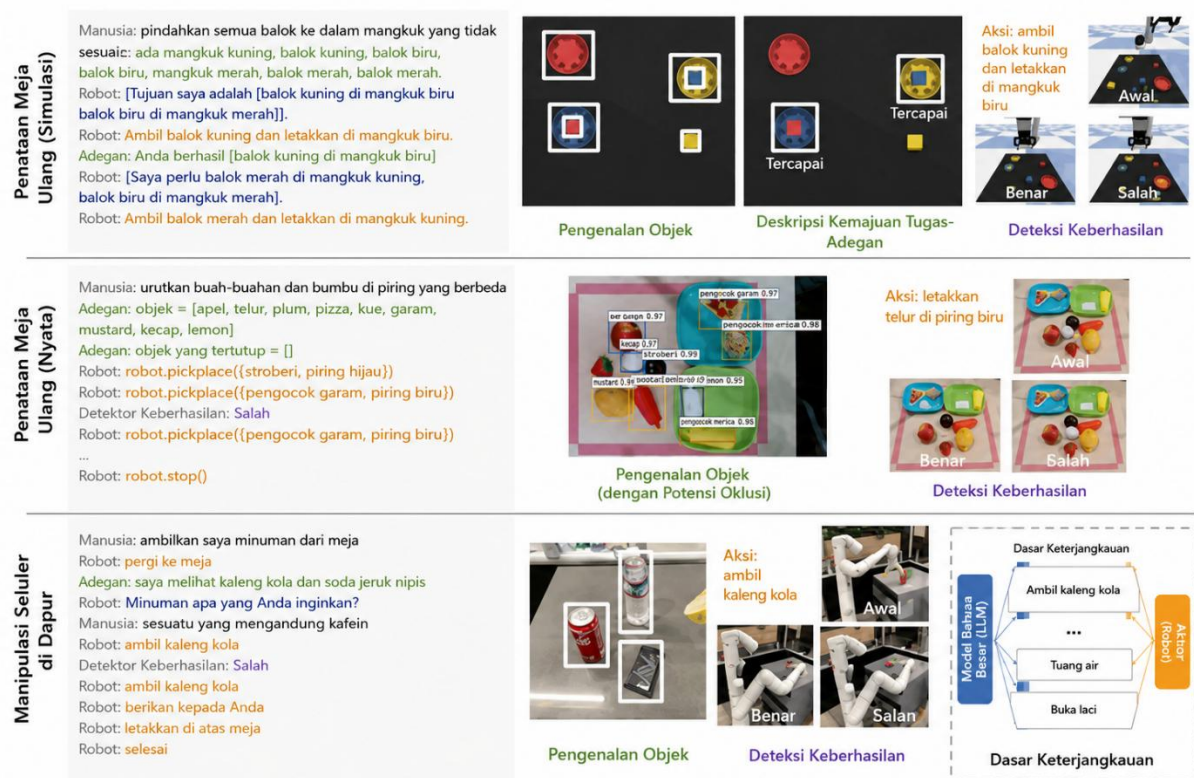
Gambar 4.5. Kinerja perencanaan SayCan dengan ukuran model.

Perencanaan Lingkaran Tertutup

Salah satu masalah dengan perencanaan berbasis bahasa gaya SayCan adalah jika salah satu langkah perantara gagal, seluruh eksekusi proses gagal. Algoritma ini tidak melakukan perencanaan ulang secara interaktif atau mengambil umpan balik dari lingkungan. Monolog Dalam[menawarkan peningkatan dibandingkan SayCan dengan menutup lingkaran umpan balik dari lingkungan ke model bahasa. Ia mempertimbangkan tiga jenis umpan balik:

1. **Deteksi keberhasilan:** Klasifikasi biner, dalam bentuk bahasa, apakah tugas berhasil atau tidak pada langkah waktu tertentu.
2. **Deskripsi adegan pasif:** Umpan balik dari adegan yang ditanyakan melalui detektor objek/deskriptor adegan eksternal. Mereka terutama terdiri dari dua jenis:
 - a. *Umpan balik objek:* Keluaran tekstual lokasi objek dari detektor objek
 - b. *Umpan balik adegan:* Deskripsi adegan kemajuan tugas dalam bentuk teks
3. **Deskripsi adegan aktif:** Model bahasa secara interaktif mengajukan pertanyaan tentang adegan/tugas, yang dijawab oleh model tanya jawab visual.

Gambar 4.6 menunjukkan Monolog Batin dalam berbagai skenario.



Gambar 4.6. Contoh perencanaan Monolog Batin dalam tugas manipulasi robot yang menunjukkan skenario simulasi dan dunia nyata untuk penataan ulang meja dan manipulasi peralatan dapur bergerak.

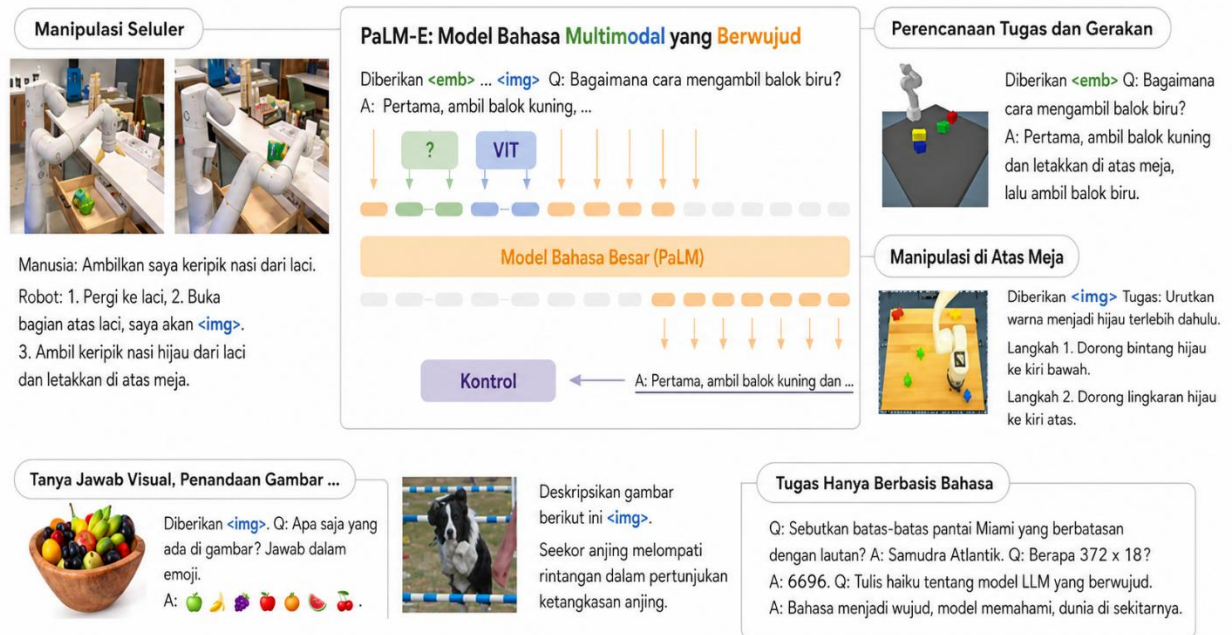
Monolog Batin melaporkan tingkat kinerja yang lebih tinggi daripada SayCan dan beberapa kemampuan yang muncul, seperti robot bereaksi terhadap umpan balik perantara dari

manusia, perencanaan ulang dalam kondisi tidak layak, dan pemahaman adegan interaktif yang lebih baik yang ditunjukkan melalui tanya jawab.

Perencanaan Multimodal

Selain umpan balik bahasa, model multimodal yang secara khusus disempurnakan untuk perencanaan robot memiliki dua keunggulan:

1. Lebih presisi daripada perencanaan loop terbuka di lingkungan yang nyata dan tidak terlihat
2. Kemampuan untuk tetap mengingat konteks yang lebih besar dari pengetahuan skala Internet
3. Umpan balik visual dan keadaan dalam loop menghasilkan model yang dapat melihat dan berpikir secara bersamaan
4. Salah satu karya terkemuka dalam perencanaan multimodal adalah PaLM-e (lihat Gambar 4.7), juga disebut Embodied PaLM.



Gambar 4.7. Gambaran umum PaLM-E yang digunakan untuk berbagai tugas, seperti manipulasi seluler, tanya jawab visual, dan perencanaan tugas dan gerakan.

PaLM-e mengkodekan input kontinu ke dalam LLM yang hanya berfungsi sebagai dekoder (misalnya, PaLM). Untuk mengkodekan gambar, ia menggunakan ViT 22B dan memasukkan token gambar yang digabungkan dengan bahasa. Input ke PaLM adalah kalimat multimodal yang menyisipkan input kontinu dengan teks. Contoh kalimat multimodal dari makalah tersebut adalah “T: Apa yang terjadi antara dan ?” (mewakili penyematan gambar).

PaLM-e kemudian mengeluarkan teks secara autoregresif. Outputnya bisa berupa jawaban atas pertanyaan, atau urutan keputusan/rencana yang dieksekusi oleh kebijakan tingkat bawah. Salah satu hasil yang mengesankan dari makalah PaLM-e adalah bahwa

penyempurnaan pada teks/rencana robot tidak secara signifikan menurunkan kinerja dalam keterampilan bahasa umum, membuka jalan bagi satu model untuk semua tugas tingkat tinggi/rendah di masa depan.

Perencanaan melalui Dialog Multimodal

Pendekatan kedua untuk perencanaan berbasis bahasa adalah melalui tanya jawab visual menggunakan model multimodal. RoboVQA adalah contoh kerangka kerja tersebut, menggunakan model yang dipelajari/disempurnakan. Ini melibatkan menampilkan model beberapa detik terakhir dari umpan sensor dan mengajukan pertanyaan seperti:

1. “Tujuan tingkat tinggi adalah menumpuk cangkir. Apa yang harus dilakukan robot selanjutnya? Sejauh ini kita telah mengeksekusi 1. Tumpuk cangkir biru di atas cangkir merah 2.”
2. “Apa yang baru saja terjadi?”
3. Dalam konteks ini, *affordance* generatif adalah jenis pertanyaan yang menghasilkan tugas-tugas yang mungkin, seperti “Tugas apa yang dapat dilakukan robot sekarang?” dan *affordance* diskriminatif adalah jenis pertanyaan yang mengklasifikasikan tugas yang diusulkan menjadi layak atau tidak. Misalnya, “Apakah mungkin untuk mengambil cangkir sekarang?”

Output dari model multimodal digunakan untuk eksekusi robot.

Tantangan

Meskipun munculnya model bahasa besar dan komputasi serta perencanaan melalui bahasa, beberapa masalah masih perlu dipecahkan terkait perencanaan robot LLM:

1. Terdapat mode kegagalan mendasar ketika model bahasa tidak dapat melihat dan tidak memiliki pemahaman tentang fisika. Salah satu cara untuk menjembatani kesenjangan ini adalah dengan menggunakan VLM dalam loop atau, seperti pada PaLM-E, menggunakan model visi yang terhubung. Kedua pendekatan tersebut rentan terhadap kerapuhan. Pada pendekatan pertama, pemahaman tentang hubungan objek dibatasi oleh kemampuan VLM untuk mendeskripsikan, yang dapat melewati beberapa hal. Misalnya, jika VLM menemukan telepon dan seseorang, robot mungkin berencana untuk mengambil telepon meskipun telepon tersebut sedang dipegang oleh orang tersebut, sehingga mengabaikan hubungan objek. Metode deteksi objek memiliki kegagalan tersendiri yang secara keseluruhan menurunkan akurasi perencanaan.
2. Mode kegagalan kedua adalah kegagalan dalam memahami fisika. Misalnya, LLM mungkin tidak mengetahui berapa berat suatu objek, apakah aman untuk ditangani, bagaimana mendekati objek target berdasarkan rintangan di sekitarnya, dan sebagainya. Memberikan pemahaman fisika pada model bahasa[50] adalah bidang pekerjaan yang sedang berlangsung.
3. Mode kegagalan ketiga adalah perencanaan spesifik perwujudan. Robot tertentu mungkin memiliki kekuatan dan kelemahan tertentu yang perlu diperhitungkan dalam perencanaannya. Ini dapat mencakup muatan, jumlah lengan, ketinggian yang dapat dijangkau, titik buta rangkaian sensor, dan sebagainya. Meskipun fungsi *affordance* yang dipelajari melalui RL dapat memberikan gambaran tentang imbalan yang

diharapkan, cakupan atas ruang aksi keadaan mungkin tidak cukup untuk pertimbangan menyeluruh terhadap atribut perwujudan.

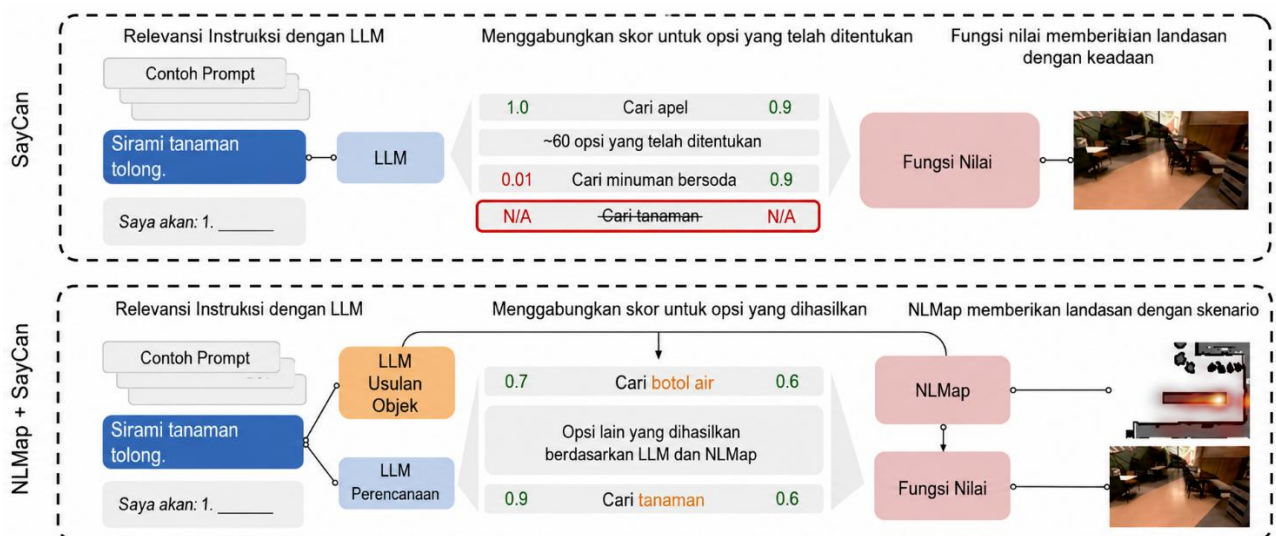
4. Perencanaan di lingkungan yang tidak terlihat sangat sulit bagi LLM yang menderita halusinasi, penalaran yang salah, dan sebagainya.

Bahasa untuk Pemetaan

Selain perencanaan tingkat tinggi, bahasa dapat digunakan untuk pemetaan dan lokalisasi. Sebelumnya dalam bab ini, Anda telah mempelajari bahwa melalui fungsi affordance, yang memetakan lingkungan dan potensi interaksi yang dapat dilakukan robot dengan objek, model bahasa besar dapat digunakan untuk perencanaan robot, seperti pada SayCan. Salah satu kekurangan SayCan adalah ketidakmampuannya untuk menalar tentang apa yang dapat dilakukan robot dalam suatu adegan di luar apa yang dapat dilihat robot. Hal ini penting karena banyak tugas manipulasi mobile dalam ruangan melibatkan mengetahui di mana objek berada meskipun objek tersebut tidak terlihat dalam gambar.

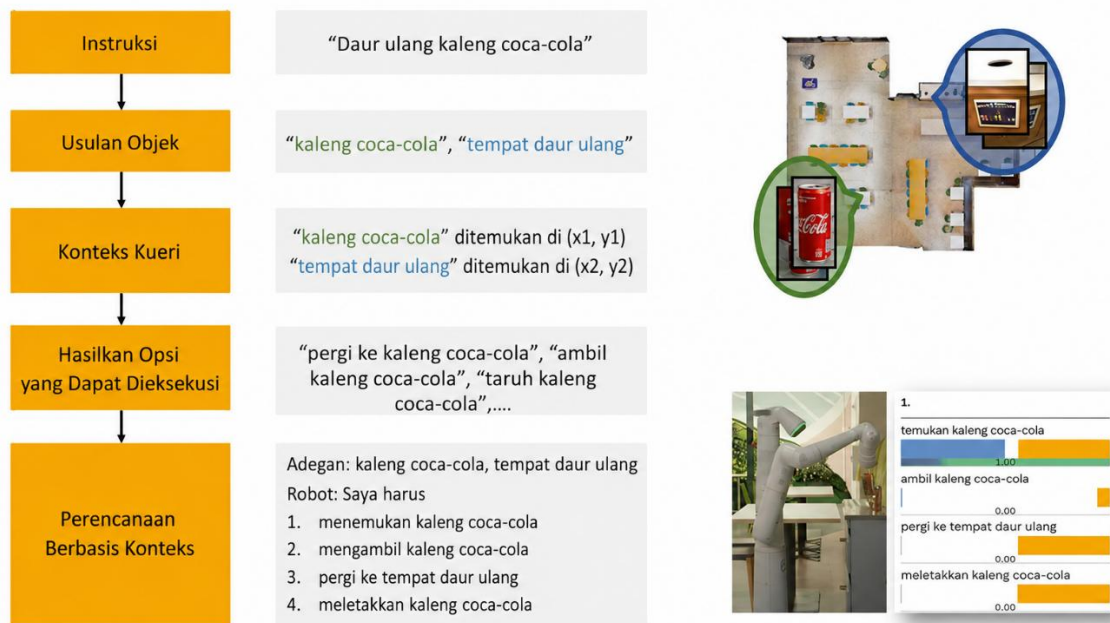
NLMap-SayCan bertujuan untuk mengatasi masalah ini dengan mengusulkan objek potensial dan menghasilkan rencana untuk mengeksekusi tindakan bahkan ketika objek tersebut tidak terlihat secara langsung. Sebagai ringkasan, SayCan menerima instruksi melalui LLM untuk menghasilkan dan menilai kemungkinan tindakan dari daftar yang telah ditentukan sebelumnya, kemudian menggunakan fungsi nilai untuk memilih tindakan yang paling layak berdasarkan keadaan lingkungan saat ini. Sebaliknya, NLMap membuat peta lingkungan dan menggunakan fungsi nilai untuk mengevaluasi kelayakan tindakan. Hal ini memastikan bahwa tindakan yang dipilih relevan dengan instruksi tetapi juga layak dalam konteks lingkungan.

Gambar 4.8 menunjukkan perbedaan utama antara SayCan dan NLMap-SayCan. Secara keseluruhan, NLMap mengatasi keterbatasan SayCan dalam perencanaan dengan pemahaman kontekstual yang lebih kuat melalui peta inklusi.



Gambar 4.8. Perbandingan SayCan dan NLMap-SayCan. NLMap-SayCan menghasilkan objek yang relevan secara dinamis menggunakan LLM dan menanyakan NLMap untuk lokasi objek. Hal ini memungkinkan perencanaan tugas yang lebih fleksibel dan spesifik konteks.

NLMap dimulai dengan instruksi, seperti “ambil botol Coca-Cola,” dan dengan membiarkan robot menjelajah, mengumpulkan gambar RGB-D. Kemudian ROI dihasilkan dari gambar-gambar ini untuk menentukan di mana objek berada. Informasi 3D tentang adegan dipelajari melalui VLM, yang membuat peta adegan terperinci. LLM kemudian mengusulkan objek yang relevan berdasarkan instruksi, seperti “Coca-Cola” dan “botol,” dan VLM memeriksa peta adegan untuk menentukan apakah dan di mana objek-objek ini berada, memberi skor seberapa besar kemungkinan objek tersebut ditemukan di berbagai tempat dalam adegan. Objek adegan kemudian diperbarui agar sesuai dengan objek yang paling relevan dan LLM merencanakan cara mengeksekusi tugas. Proses ini ditunjukkan pada Gambar 4.9.



Gambar 4.9. NLMap-SayCan menghasilkan objek yang relevan berdasarkan instruksi, kemudian melakukan kueri ke NLMap untuk memfilter daftar objek dan menemukan lokasinya. Sistem ini membuat opsi yang dapat dieksekusi dan menggunakan perencanaan berbasis LLM untuk memandu robot melalui tugas langkah demi langkah.

Pendekatan ini memang memiliki beberapa keterbatasan yang perlu disebutkan yang disorot dalam makalah: ia kesulitan memahami bagaimana objek saling berhubungan dalam lingkungan dan menangani perubahan adegan. Penting juga untuk dicatat bahwa akurasi proposal objek bergantung pada kinerja LLM. Meskipun keterbatasan ini ada, NLMap-SayCan tetap merupakan lompatan maju dalam perencanaan robot yang lebih tangguh.

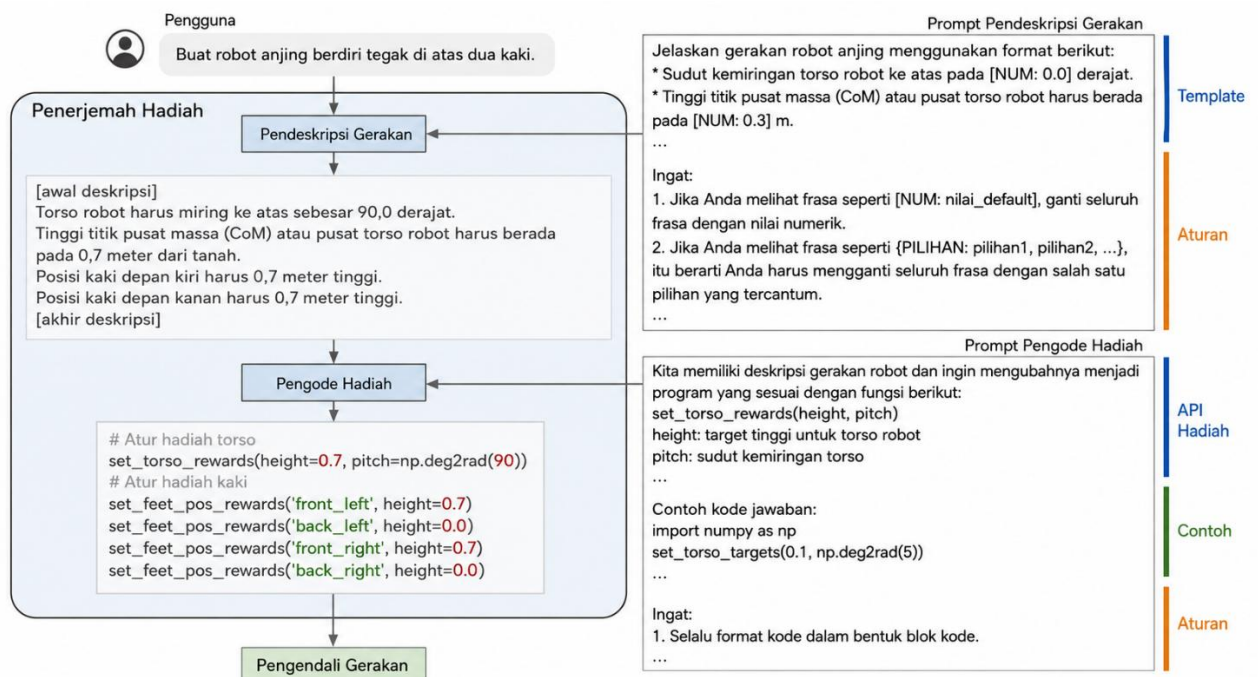
Bahasa untuk Imbalan

Setelah perencanaan selesai, robot harus mengeksekusi tindakan yang direncanakan, yang melibatkan kontrol dan gerakan robot secara akurat. Data pelatihan LLM seringkali kurang informasi tentang tindakan fisik dan perwujudan, sehingga lebih sulit untuk menggunakan LLM untuk kontrol dan gerakan robot secara langsung, karena skenario ini berada di luar distribusi pelatihan tipikalnya.

Bagaimana jika robot dapat belajar untuk mengeksekusi tugas dengan lebih akurat dengan memiliki kerangka internal tentang apa yang "salah" dan apa yang "baik"? Inilah tepatnya yang dapat dilakukan fungsi penghargaan untuk LLM dan kebijakan kontrol, dengan memberikan umpan balik kepada robot untuk memandu tindakan robot dan meningkatkan akurasi. "*Language to Rewards for Robotic Skill Synthesis (L2R)*" adalah makalah yang menggunakan fungsi penghargaan untuk menentukan kontrol bagi robot untuk mengeksekusi berbagai tugas di lingkungannya.

L2R bekerja sebagai berikut:

- Ia menerima instruksi pengguna, yang dalam kasus Gambar 4.10 adalah: "Buat anjing robot berdiri di atas dua kaki."
- Sebagian besar pekerjaan dilakukan oleh Penerjemah Penghargaan, yang memecah perintah tingkat tinggi menjadi tindakan yang lebih detail dan spesifik yang dapat dieksekusi oleh robot. Di dalam komponen ini, Deskriptor Gerak menerjemahkan instruksi pengguna menjadi deskripsi gerak yang detail. Misalnya, ia dapat mengatakan sesuatu seperti, "Kedua kaki depan harus diangkat setinggi 0,7 meter."
- Deskripsi gerak diteruskan ke Pengkode Imbalan, yang mengubah deskripsi tersebut menjadi fungsi imbalan yang dapat ditindaklanjuti untuk sistem kontrol robot menggunakan API yang telah ditentukan sebelumnya yang telah diatur agar dipahami oleh robot.
- Fungsi imbalan ini dikirim ke Pengontrol Gerak, yang bertanggung jawab untuk mengeksekusi tindakan menggunakan fungsi imbalan untuk mengontrol pergerakan robot dan memperbaikinya.



Gambar 4.10. Gambaran L2R, dengan deskriptor gerak dan pengkode hadiah.

Keterbatasan utama pendekatan ini adalah bahwa pendekatan ini membutuhkan templat yang telah ditentukan sebelumnya yang dapat digunakan agar robot memahami deskripsi gerak. Hal ini dapat mempersulit generalisasi di berbagai robot dan modalitas.

Bahasa untuk Kode Robot

Tema utama dalam metode seperti L2R adalah mengembangkan cara "standar" bagi robot untuk memahami deskripsi gerak yang diperlukan untuk menyelesaikan tugas. Mengingat bahwa robot dapat diprogram dan dikendalikan, dapatkah kita memanfaatkan LLM untuk meningkatkan kemampuan mereka dalam mengeksekusi tugas-tugas ini secara lebih efektif dengan menulis kode?

Kode sebagai Kebijakan adalah metode yang bertujuan untuk melakukan hal ini dengan menggunakan LLM untuk menerjemahkan perintah bahasa alami ke dalam kode kebijakan robot yang dapat dieksekusi. Saat pengguna memberikan instruksi, seperti "susun buah-buahan ini dalam satu baris," LLM memproses masukan tersebut dan menghasilkan kode yang mencakup API persepsi untuk mengidentifikasi objek terkait di lingkungan serta API kendali untuk memanipulasi objek-objek tersebut. Kode ini memuat fungsi-fungsi spesifik yang mengidentifikasi objek-objek tersebut dan menjalankan tindakan yang diperlukan dengan cara memindahkan, menyusun, atau memanipulasinya. Contoh kode yang dapat dihasilkan oleh *Code as Policies* ditampilkan pada Gambar 4.11. Aspek menarik dari **Code as Policies** adalah ketergantungannya semata-mata pada **prompt engineering** menggunakan LLM yang telah dilatih sebelumnya (**pretrained LLM**) tanpa pelatihan khusus untuk tugas tertentu; hal ini berarti, secara teoretis, metode tersebut dapat melakukan generalisasi terhadap tugas-tugas yang melibatkan objek dan tindakan yang tercakup dalam distribusi data pelatihan LLM tersebut.. Dengan menghasilkan dan menyesuaikan kode dari contoh, Robot dapat mempelajari berbagai tugas dan mengikuti perintah bahasa setiap hari.

```
# susun blok-blok di dalam mangkuk kosong.  
empty_bowl_nama = parse_obj('mangkuk kosong')  
nama_blok = parse_obj('blok')  
nama_objek = [empty_bowl_nama] + nama_blok  
susun_objek_berurutan(nama_objek=nama_objek)
```

Gambar 4.11. Contoh instruksi masukan yang diformat sebagai komentar (hijau). Instruksi tersebut diinterpretasikan oleh LLM, menghasilkan kode Python yang valid (disorot) untuk menyelesaikan tugas-tugas seperti menumpuk balok di dalam mangkuk kosong.

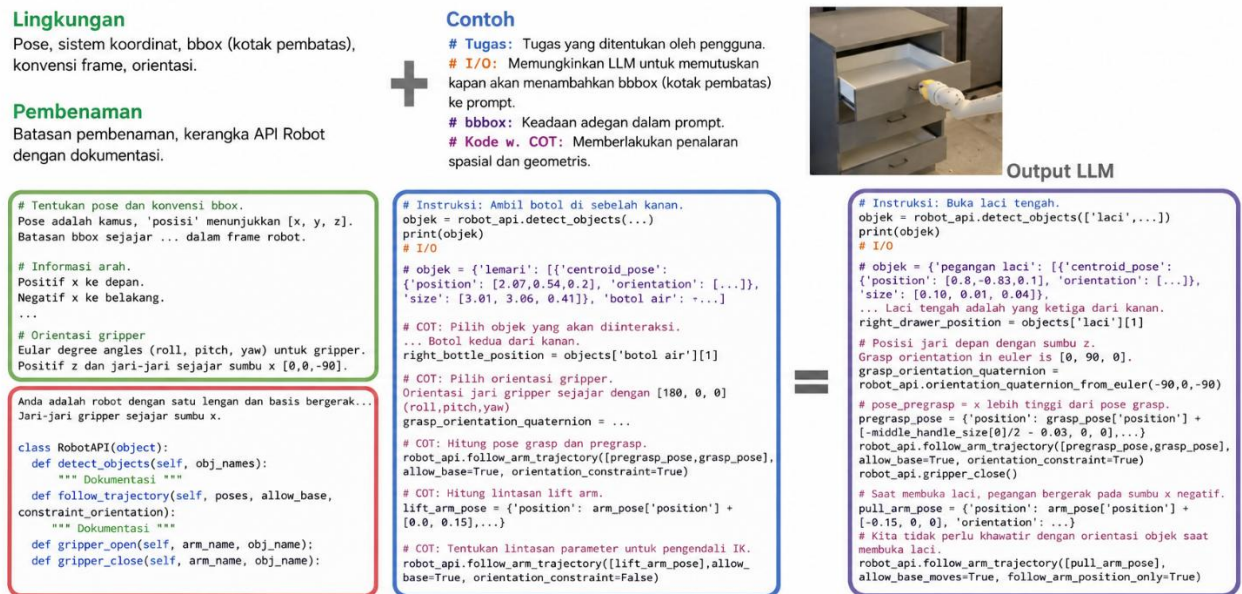
Untuk menghasilkan kebijakan ini, model bahasa diberi beberapa contoh. LLM kemudian secara autoregresif dan hierarkis membuatnya dengan menggabungkan fungsi-fungsi yang sudah dikenal atau memanggil program lain untuk mendefinisikan fungsi-fungsi yang belum didefinisikan dengan jelas. Sebagai kerangka kerja, ini sangat menarik bagi robot karena mengumpulkan sejumlah besar data spesifik tugas merupakan tantangan, dan menggunakan

model besar yang sudah terlatih memungkinkan solusi yang lebih efisien. Kunjungi untuk mempelajari lebih lanjut.

Studi terbaru menunjukkan bahwa pembangkitan kode robot dapat ditingkatkan dalam lebih banyak cara seiring dengan kematangan studi LLM:

1. Pembangkitan kode robot dapat dikondisikan secara multimodal dengan menggunakan VLM, yang mahir dalam pembangkitan kode, dan memasukkan gambar dari kamera robot secara langsung, daripada memasukkan deskripsi bahasa alami dari gambar ke dalam LLM. Hal ini memungkinkan penalaran menjadi lebih multimodal secara alami daripada dalam Kode sebagai Kebijakan.
2. Memberi VLM definisi API robot yang sangat deskriptif dan menggunakan model konteks panjang yang mahir dalam penalaran rantai pemikiran meningkatkan pembangkitan kode secara signifikan dan dapat memunculkan perilaku yang belum pernah dilihat sebelumnya tanpa data pelatihan apa pun.
3. Menunjukkan contoh pembangkitan kode yang benar juga meningkatkan kemungkinan kebenarannya.

Ini dan beberapa trik lain tentang cara memberi petunjuk kepada robot dapat dipelajari lebih lanjut di sini. Contohnya ditunjukkan pada Gambar 4.12.



Gambar 4.12. Contoh input dan output pembangkitan kode dari “Cara Memberi Perintah pada Robot Anda.”

Keberhasilan pembangkitan kode untuk sinyal kontrol zero-shot menunjukkan bahwa ada lebih banyak peluang dalam mengekstrak gerakan dari pengetahuan dunia yang terkandung dalam LLM melalui kode, dan menjadikannya resep yang dapat dipelajari untuk meningkatkan skala pembelajaran robot. Salah satu potensi penentangan terhadap gagasan ini adalah anggapan bahwa, meskipun beberapa metode memiliki kinerja zero-shot yang sangat baik, metode tersebut mungkin tidak cukup untuk meningkatkan skala pembelajaran yang menyerupai

metode yang berpotensi mencapai keberhasilan 100 persen. Meningkatkan metode zero-shot melalui umpan balik adalah strategi pasca-pelatihan yang umum dalam studi LLM, sehingga sangat mungkin bahwa Kode sebagai Kebijakan dapat dibingkai menjadi resep peningkatan berkelanjutan, jika juga diadaptasi untuk mengumpulkan keterampilan baru, yang tidak diketahui oleh LLM melalui demonstrasi.

4.3 KONTROL ROBOT UJUNG-KE-UJUNG

Konsep penting dalam robotika adalah generalisasi: dapatkah sebuah model melakukan tugas baru yang tidak dilatih sebelumnya? Pentingnya konsep ini muncul dari fakta bahwa robot yang cerdas secara umum harus berhasil di berbagai lingkungan dan tugas penerapan, tetapi akan sulit untuk merepresentasikan keragaman ini dalam set pelatihan. Jadi, metode yang dapat melakukan generalisasi ke situasi baru, melalui pembelajaran representasi yang lebih baik atau penalaran tentang semantik tugas, akan mendapatkan keuntungan.

Pada saat ini dalam studi robotika, terdapat beberapa sumbu generalisasi:

1. **Generalisasi objek:** Dapatkah sebuah model menangani objek yang tidak terlihat?
2. **Generalisasi lingkungan:** Dapatkah sebuah model bertindak di lingkungan yang tidak terlihat?
3. **Generalisasi gerakan:** Dapatkah sebuah model menciptakan gerakan baru?
4. **Generalisasi perspektif:** Dapatkah model beralih dari perspektif orang ketiga ke perspektif egosentris dan sebaliknya? Apakah posisi kamera penting?
5. **Generalisasi perwujudan:** Dapatkah data pada satu jenis tubuh robot digunakan untuk meningkatkan keterampilan pada jenis tubuh robot yang berbeda?

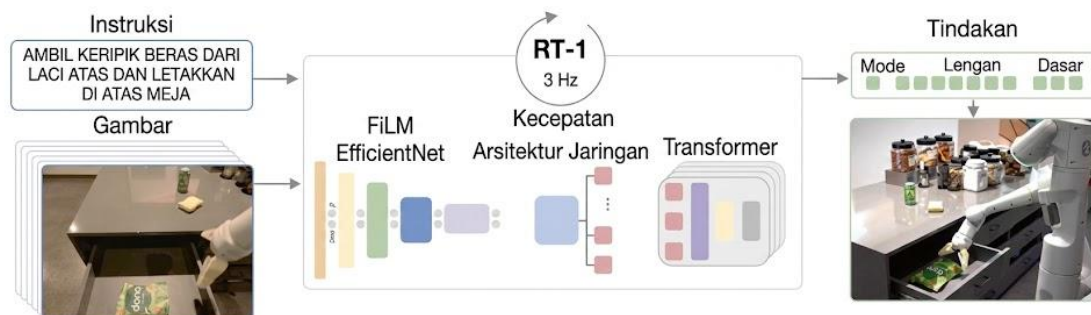
Selain itu, model kontrol robot dapat diukur berdasarkan pemahaman simbolik, penalaran, perencanaan jangka panjang, pengenalan manusia, keselamatan fisik, dan sebagainya.

Kontrol robot ujung-ke-ujung mencakup masalah pembelajaran kontrol tingkat rendah sedemikian rupa sehingga gradien mengalir dari output kontrol ke input robot. Dalam penggunaan konvensional, ini berarti mempelajari tindakan secara langsung dengan menggunakan gambar kamera dan pengamatan lain sebagai input. Bagaimana kita mengembangkan metode yang dapat mengontrol robot ujung-ke-ujung? Dapatkah kita menghubungkan model-model ini dengan tren besar penskalaan dan model besar dalam AI? Bagaimana kita mengukur dan meningkatkannya pada sumbu generalisasi?

Kontrol Robot Ujung-ke-Ujung dengan Transformer Autoregresif

Hingga tahun 2022, penskalaan model pembelajaran penguatan multi-tugas besar dianggap sebagai kunci untuk memecahkan masalah robotika, tetapi sejak saat itu gelombang telah bergeser ke arah model pembelajaran imitasi besar, terutama karena pra-pelatihan imitasi mulai berkinerja lebih baik daripada metode RL pada tolok ukur multi-tugas. *Robotics Transformer 1* (RT-1), yang ditunjukkan pada Gambar 4.13, dibuat pada era ini, dan merupakan model multi-tugas pembelajaran imitasi skala besar awal. Model ini dilatih pada korpus besar data robot dunia nyata yang mencakup beragam tugas dan merupakan model dasar awal. RT-1 bekerja sebagai berikut:

- Instruksi pengguna dalam teks dan serangkaian gambar (riwayat) yang ditangkap oleh robot adalah input ke model RT-1. Teks dikodekan oleh tokenizer teks beku dan kemudian gambar dan teks digabungkan bersama oleh jaringan FiLM-efisien. Penggabungan awal token visual dan bahasa terbukti sangat penting dalam mengekstrak konteks yang tepat dari gambar.
- Token bahasa visual ini kemudian dikompresi oleh TokenLearner, yang menggunakan perhatian untuk fokus pada bagian gambar dan teks yang paling relevan.
- TokenLearner mengompresi jumlah total token, dan membuat inferensi tiga kali lebih cepat.
- Token ini kemudian diproses oleh transformer, yang menggunakan self-attention untuk memahami hubungan antara objek dalam gambar dan perintah dalam teks. Kemudian menghasilkan aksi: parameter kontrol tunggal untuk setiap derajat kebebasan. Dalam kasus kontrol efektor akhir Cartesian, itu menjadi posisi 3D gripper, rotasi 3D, gerakan dasar, dan sebagainya. Aksi yang berupa nilai kontinu di-tokenisasi dengan memetakannya ke distribusi seragam diskrit, dan kemudian memprediksi nomor bucket yang sesuai dengan aksi float.
- Model ini dilatih dengan fungsi kerugian cross-entropy, dan data pelatihan (output aksi dan input gambar/teks) dikumpulkan oleh manusia yang mengoperasikan robot dari jarak jauh.



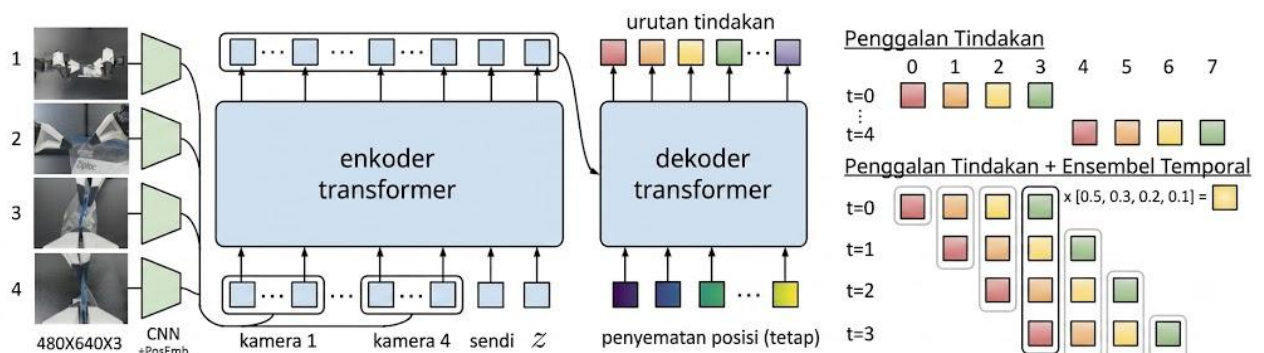
Gambar 4.13. RT-1 memproses gambar dan instruksi bahasa alami untuk mengontrol tindakan robot.

RT-1 melakukan kontrol loop tertutup dan mengeksekusi tindakan hingga menghasilkan tindakan penghentian atau mencapai jumlah langkah waktu yang telah ditentukan. Karya ini menunjukkan bahwa dengan data domain yang cukup, model berbasis transformer dapat disesuaikan untuk menciptakan transformer multi-tugas besar untuk kontrol robot. Karya ini juga menunjukkan bahwa data simulasi, ketika ditambahkan dalam campuran yang tepat, dapat meningkatkan kinerja, seperti yang Anda lihat dengan data sintesis di domain LLM. RT-1 juga menunjukkan tanda-tanda transfer antar robot yang kemudian muncul sebagai area studi yang lebih luas, seperti yang dibahas kemudian dalam bab ini.

Salah satu kekurangan RT1 adalah bahwa ia diuji terutama pada manipulasi kasar—mengambil dan menempatkan—dan ini membuatnya lebih sulit untuk mengetahui apakah ia dapat bekerja untuk tugas-tugas yang sangat cekatan. Tugas-tugas cekatan adalah tugas manipulasi yang membutuhkan kontrol motorik halus dan koordinasi yang mirip dengan tangan manusia yang memanipulasi objek di lingkungannya. Dalam bidang robotika, ini berarti berurusan dengan objek dengan tekstur, ukuran, dan bentuk yang berbeda. Kelemahan kedua adalah bahwa meskipun menunjukkan generalisasi yang mengesankan dalam domain, kemampuannya untuk melakukan generalisasi di luar domain—dengan objek yang sepenuhnya baru dan dengan penalaran—terbatas.

Model lain, yang sangat mirip dengan RT-1, yang banyak digunakan saat ini adalah ACT (*Action Chunking Transformer*). ACT adalah model encoder-decoder yang memperkenalkan konsep yang disebut "pengelompokan aksi." Pengelompokan aksi mengacu pada metode pemodelan k langkah ke depan. Pada titik waktu tertentu, Anda mendapatkan pengamatan tetapi memprediksi k tindakan, secara efektif mengurangi cakupan tugas sebanyak k kali lipat. Ini mengungguli kebijakan satu langkah dalam situasi di mana terdapat faktor pengganggu yang berkorelasi temporal, seperti ketika ada jeda dalam demonstrasi dan tindakan selanjutnya bukan hanya fungsi dari keadaan tetapi juga dari langkah waktu. Selama faktor pengganggu ini berada dalam panjang pengelompokan, kebijakan pengelompokan aksi dapat pulih.

Dalam praktiknya, panjang pengelompokan sesuai dengan 1 detik cakupan ke depan. Artinya, jika kontrol Anda berjalan pada 30hz, satu observasi, alih-alih memprediksi satu tindakan seperti pada RT1, akan memprediksi 32 tindakan, atau horizon 1 detik. Dalam praktiknya, pelatihan pada chunk juga menggunakan penggabungan temporal, bersama dengan chunking. Seperti yang ditunjukkan pada Gambar 4.14 di sebelah kanan, ini berarti bahwa alih-alih pelatihan pada chunk yang terpisah, chunk untuk observasi saling tumpang tindih, memungkinkan pemodelan yang sangat padat.

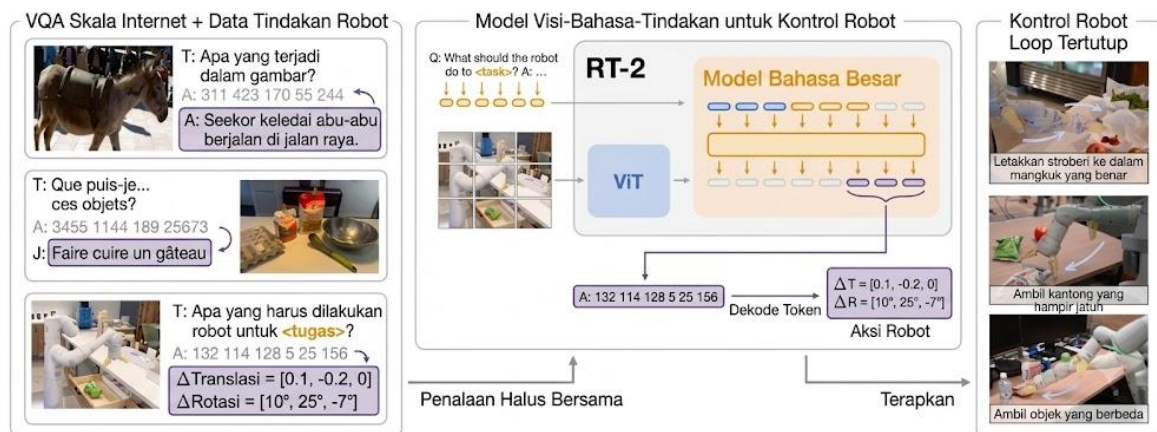


Gambar 4.14. Arsitektur transformer encoder-decoder chunking aksi di atas; chunking dengan penggabungan temporal di bawah.

RT-2 adalah metode yang dibangun di atas RT-1 yang menggunakan informasi semantik yang dimiliki model bahasa-visi yang telah dilatih sebelumnya untuk menghasilkan aksi robot tingkat rendah. RT-2 menggunakan data tanya jawab visual (VQA), di mana terdapat gambar yang

dipasangkan dengan pertanyaan dan jawaban yang menggambarkan konteks dalam gambar, dan data aksi robot, yang memasangkan gambar dan instruksi teks dengan aksi robot yang perlu dieksekusi untuk menyelesaikan tugas tersebut dengan sukses.

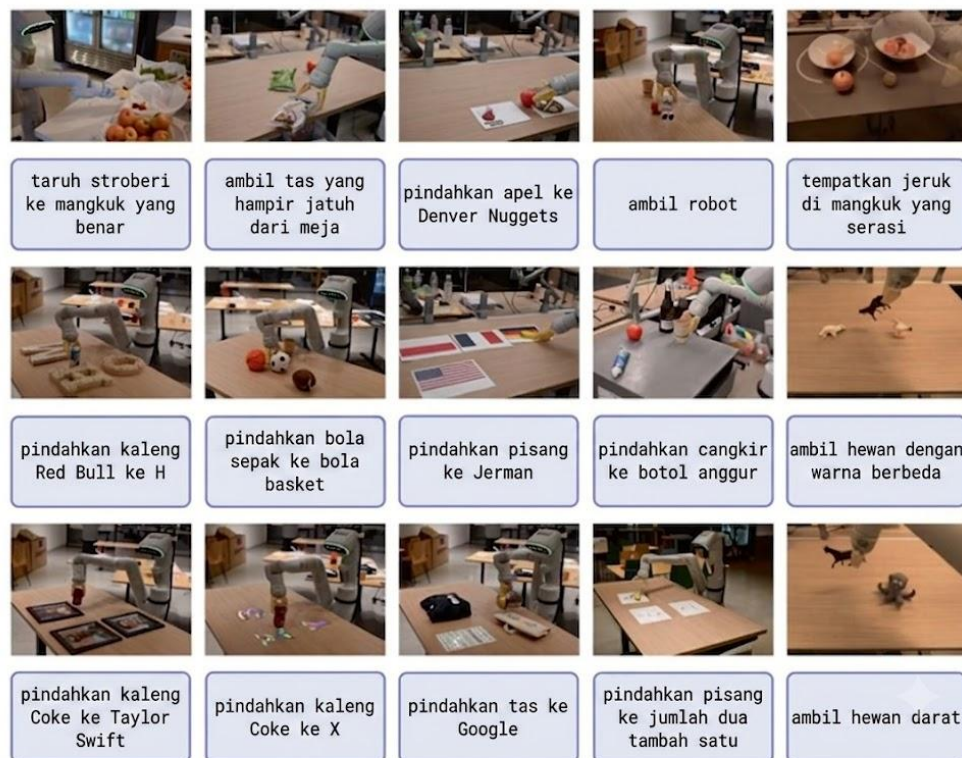
Model ini pertama-tama menggunakan VLM yang telah dilatih sebelumnya yang dilatih pada data skala Internet, kemudian melatihnya bersama dengan aksi robot dan data skala Internet untuk mendapatkan model bahasa-visi-aksi (VLA) untuk kontrol robot. Resep pelatihan ini digambarkan pada Gambar 4.15.



Gambar 4.15. RT-2 merepresentasikan tindakan robot sebagai token teks, dilatih bersama dengan dataset bahasa-visi skala besar.

Inferensi dimulai dengan pertanyaan pengguna, seperti "Apa yang harus dilakukan robot untuk mengambil apel?" (What should the robot do to pick up the apple?). Pertanyaan ini dan gambar adegan tempat pengguna ingin tugas tersebut dieksekusi diproses oleh ViT dan LLM. ViT mengekstrak fitur visual dari gambar dan LLM memahami bagian bahasa dari pertanyaan tersebut. Fitur gabungan ini menghasilkan urutan tindakan yang direpresentasikan sebagai translasi dan rotasi spesifik untuk dilakukan robot.

Hasil makalah menunjukkan bahwa pelatihan bersama pada dataset Internet dan robot secara bersamaan dengan cara ini memungkinkan robot untuk memahami konsep yang hanya terlihat di Internet dan secara praktis menggunakannya dalam situasi dunia nyata. Sebagai contoh, robot mampu mengidentifikasi selebriti ("pindahkan kaleng ke Taylor Swift"), melakukan perhitungan matematika sederhana ("pindahkan kaleng Coca-Cola ke jumlah 1 + 2") dan memahami simbol ("pindahkan kaleng Coca-Cola ke Google"). Robot ini bahkan memahami konsep relatif lainnya, seperti "ambil objek dengan warna berbeda," "letakkan stroberi ke dalam mangkuk yang tepat," dan sebagainya, yang membutuhkan penalaran di antara pilihan-pilihan dalam adegan, ketika pertanyaan diajukan dalam bahasa alami. Studi ini mencatat bahwa pelatihan awal skala internet menghasilkan representasi yang jauh lebih baik daripada yang mungkin dilakukan sebelumnya. Beberapa hasil ini ditunjukkan pada Gambar 4.16.

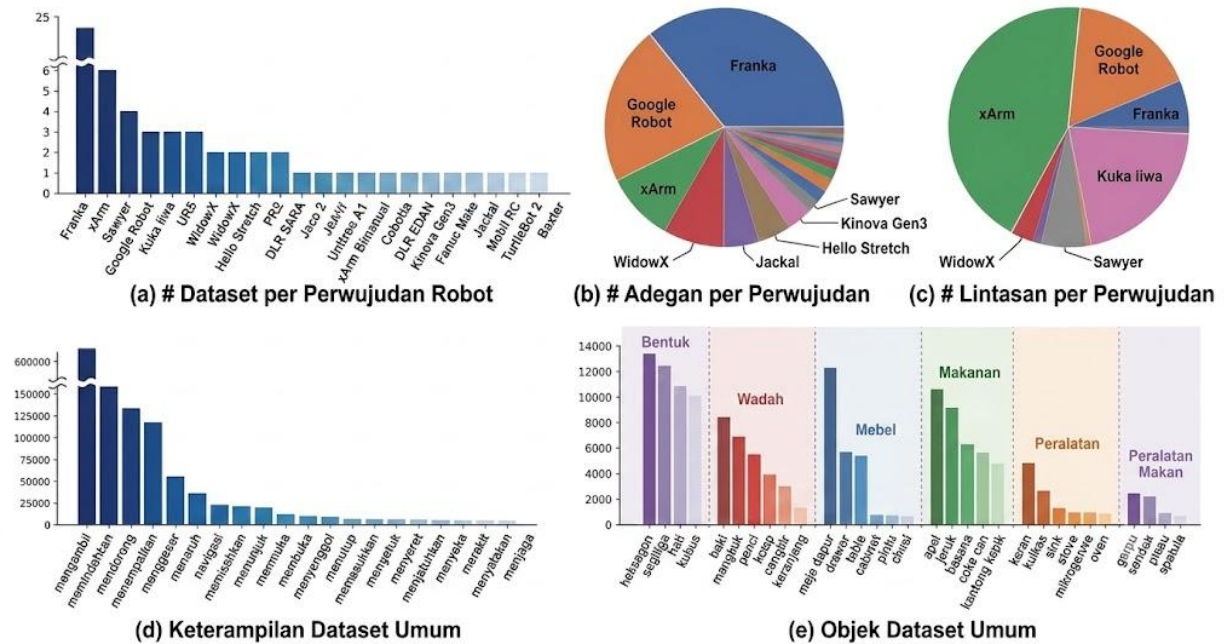


Gambar 4.16. Hasil evaluasi dari RT2. Robot memahami konsep dari Internet seperti Google, huruf, warna, Taylor Swift, dan sebagainya.

Keuntungan dari pelatihan pada data multimodal adalah bahwa model, dengan sedikit penyesuaian pada rencana robot dengan bahasa dan gambar, juga dapat melakukan perencanaan tingkat tinggi, seperti yang ditunjukkan oleh model SayCan. Ini menunjukkan peluang di masa depan untuk menyatukan perencanaan tingkat tinggi dan kontrol tingkat rendah ke dalam satu otak robot yang mampu melakukan keduanya.

Meskipun RT1 dan RT2 menunjukkan cara untuk memodelkan data robot dalam domain, dan memodelkan data Internet dan robot skala besar secara bersamaan, masing-masing robot masih memiliki ruang aksi yang berbeda, dan tidak jelas apakah representasi yang dipelajari ditransfer antar robot. Upaya perwujudan Open X yang diprakarsai oleh Google DeepMind mempelajari masalah pembelajaran robot lintas perwujudan—dapatkah kita membangun otak tunggal untuk mengontrol banyak perwujudan robot yang berbeda?

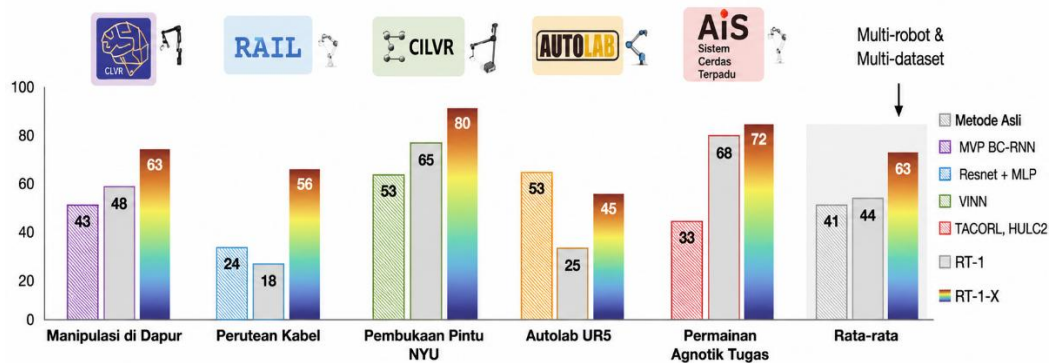
Untuk mempelajari masalah ini, sebuah dataset besar yang mewakili data dari berbagai perwujudan robot yang berbeda dikumpulkan, dengan laboratorium di seluruh dunia berpartisipasi dalam studi ini. Dataset tersebut berisi lebih dari satu juta episode robot dari lebih dari 34 laboratorium studi, yang mewakili 22+ perwujudan robot yang berbeda. Gambar 4.17 menunjukkan detail dataset tersebut. Dataset ini bersifat open source dan dapat diakses di sini.



Gambar 4.17. Distribusi data dalam dataset perwujudan Open X.

Untuk melatih model yang dapat mengontrol beberapa perwujudan, dua kandidat dipertimbangkan: RT1 dan RT2. Versi model yang dilatih pada data lintas perwujudan masing-masing dikenal sebagai RT1-X dan RT2-X. Hanya robot lengan tunggal yang dipertimbangkan untuk studi ini dan semua dataset diubah menjadi parameter kontrol posisi kartesian sebagai target pembelajaran, seperti pada RT1 dan RT2.

- Evaluasi menemukan bahwa model yang dilatih pada semua data robot mampu mengalahkan model individual yang dilatih oleh setiap lab pada data mereka sendiri. Dengan kata lain, model generalis berkinerja lebih baik daripada spesialis, seperti yang ditunjukkan pada Gambar 4.18.
- Studi ini juga menemukan bahwa data pelatihan bersama Internet meningkatkan kinerja dibandingkan pelatihan pada data robot dari awal dan bahwa pelatihan bersama dengan data Internet dan data robot yang dicampur dalam kelompok lebih baik daripada penyempurnaan pada data robot diikuti dengan pra-pelatihan pada data Internet. Yang pertama membantu mempelajari representasi gabungan yang lebih baik antara data skala robot dan Internet, dan untuk mempertahankan konsep Internet setelah beberapa masa pelatihan.



Gambar 4.18. Evaluasi model RTX.

RTX memberikan bukti untuk mendukung gagasan bahwa otak tunggal untuk banyak robot mungkin dimungkinkan di masa depan, dan bahwa robot tidak terlalu berbeda satu sama lain. Mereka mungkin hanya berbeda seperti misalnya bahasa Inggris dan Cina, dan banyak sinergi umum dapat dieksploitasi. Ini juga menyajikan cara untuk mengumpulkan data secara berbeda: jika Anda dapat mengumpulkan data Anda pada perwujudan yang lebih murah dan terukur, lalu mentransfernya ke perwujudan yang lebih mahal dan lebih sulit untuk diterapkan, Anda meningkatkan efisiensi operasi dan kemampuan untuk meningkatkan skala secara signifikan.

Gagasan ini dieksplorasi lebih lanjut dalam karya seperti CrossFormer, yang memperluas perwujudan lintas platform ke robot navigasi, robot bimanual, dan bahkan drone. Pembelajaran lintas platform kini menjadi area studi utama dalam robotika, dengan banyak perkembangan yang akan datang.

Kontrol Robot Ujung-ke-Ujung dengan Model Difusi

Baru-baru ini, model difusi menjadi sangat populer di bidang pembelajaran robot karena kemampuannya untuk memodelkan multimodalitas dengan baik. Mereka juga mudah diskalakan, dan lebih cocok untuk menangani dataset yang bising, terutama saat menghasilkan data berdimensi tinggi yang sangat multimodal. Misalnya, katakanlah dataset Anda memiliki dua lintasan untuk keadaan yang sama: satu dengan robot bergerak dalam lingkaran searah jarum jam dan yang lain dengan robot bergerak dalam lingkaran berlawanan arah jarum jam. Transformer autoregresif umumnya dilatih untuk memprediksi rata-rata dari kedua lintasan tersebut, karena menggunakan tujuan entropi silang pada rata-rata kerugian.

Pendekatan ini dapat menghasilkan hasil yang tidak masuk akal dalam situasi ini. Difusi, di sisi lain, cenderung kembali ke satu atau lebih mode data dan mengikutinya secara konsisten. Sifat ini dapat sangat berguna ketika ada strategi yang berbeda, seperti dalam manipulasi. Aspek lain di mana hal ini menjadi berguna adalah ketika Anda membutuhkan lintasan yang dihasilkan agar sangat presisi dan memiliki fidelitas tinggi. Langkah-langkah penghilangan noise dalam difusi memungkinkan koreksi untuk mendapatkan lintasan yang sangat halus yang dibutuhkan untuk gerakan yang sangat cekatan, ketika model autoregresif

menghasilkan suatu tindakan, tetapi tidak dapat memperbaikinya untuk menghilangkan suboptimalitas. Bagian ini menjelaskan bagaimana model difusi bekerja secara fundamental dan bagaimana difusi dapat diterapkan dalam robotika.

Proses Difusi Maju

Model difusi belajar menghasilkan data dengan membalikkan proses penambahan noise iteratif. Dalam proses difusi maju, noise ditambahkan ke data dalam langkah-langkah kecil. Dalam proses sebaliknya, model belajar untuk menghilangkan noise ini langkah demi langkah. Difusi maju dimulai dengan gambar awal yang diambil sampelnya dari distribusi data. Pada setiap langkah waktu t , noise secara bertahap ditambahkan ke gambar hingga menyerupai noise Gaussian murni.

Proses ini dijelaskan oleh rantai Markov, di mana setiap $x(t)$ adalah versi noise dari $x(t - 1)$. Persamaan 4.3 menunjukkan cara menghasilkan gambar noise pada langkah waktu t . Gambar noise pada beberapa langkah dibuat dengan mengambil gambar asli, memperkecil ukurannya (menggunakan akar kuadrat dari α), dan kemudian menambahkan noise acak. Noise acak ditambahkan sedemikian rupa sehingga variansnya dikendalikan oleh $(1 - \alpha)$, memastikan noise yang ditambahkan sesuai dengan distribusi normal.

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_0, (1 - \alpha_t)I)$$

Persamaan 4.3

Proses Difusi Terbalik

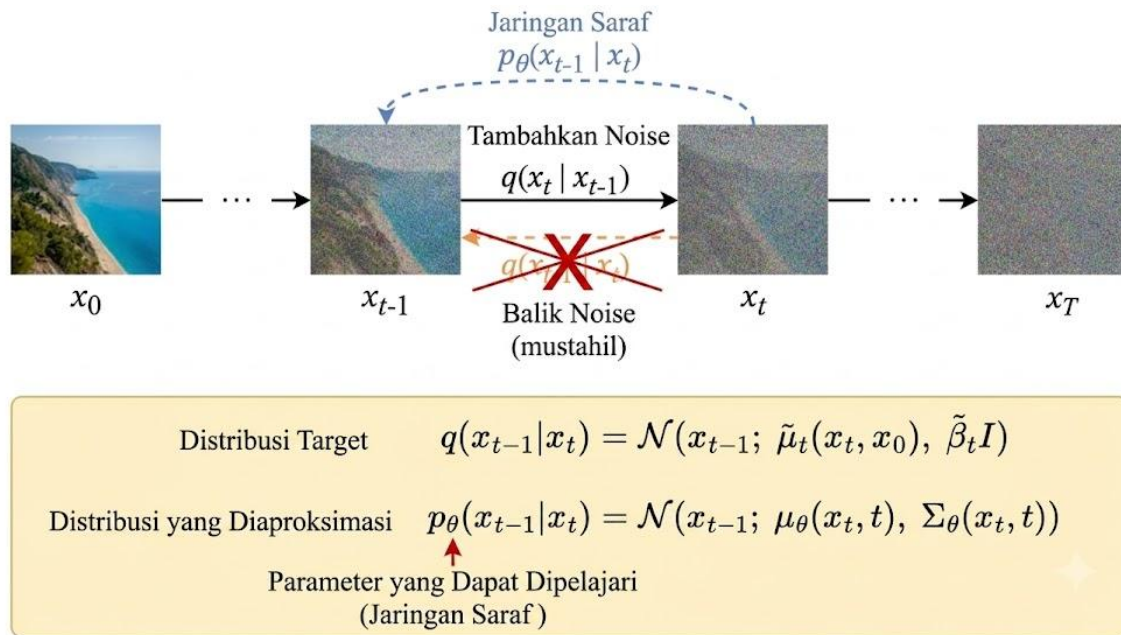
Dalam proses terbalik, Anda mulai dengan sampel noise murni. Pada setiap langkah, jaringan saraf terlatih digunakan untuk memprediksi gambar sebelumnya yang kurang bising dari sampel noise saat ini. Prediksi ini memiliki dua komponen: nilai harapan dari gambar sebelumnya dan ketidakpastian atau varians dari prediksi ini. Jaringan saraf mengambil gambar bising saat ini, langkah waktu sebagai input, dan menghasilkan nilai harapan (rata-rata) dan varians untuk langkah sebelumnya. Noise ini kemudian dikurangi dari gambar bising untuk mendapatkan gambar yang lebih jelas. Proses ini diulang secara iteratif, memungkinkan jaringan untuk secara bertahap menghilangkan noise dari gambar langkah demi langkah, seperti yang ditunjukkan pada Persamaan 4.4.

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Persamaan 4.4

Tujuan dari proses pelatihan adalah untuk membuat proses balik seakurat mungkin dengan meminimalkan perbedaan antara distribusi yang diprediksi (balik) dan distribusi posterior sebenarnya (dihitung dari proses maju), sehingga keluaran dari proses balik terlihat seperti distribusi keluaran target yang terlihat selama pelatihan. Salah satu cara untuk melakukan ini adalah dengan menggunakan divergensi KL, yang mengukur seberapa besar perbedaan satu distribusi probabilitas dengan distribusi probabilitas lainnya.

Dalam konteks difusi, divergensi mengukur perbedaan antara prediksi jaringan tentang noise yang akan dikurangi dan noise sebenarnya yang ditambahkan selama proses maju. Dengan mengurangi divergensi KL antara keduanya, jaringan saraf dapat ditingkatkan untuk membuat prediksi lebih akurat dan lebih dekat dengan noise sebenarnya yang ditambahkan selama proses maju, dan dengan demikian menghasilkan citra yang menyerupai distribusi pelatihan dari noise murni. Gambaran umum dari proses ini ditunjukkan pada Gambar 4.19.



Gambar 4.19. Ilustrasi proses maju dan mundur model difusi. Dalam proses maju, noise ditambahkan langkah demi langkah ke gambar hingga menjadi noise acak. Proses mundur, yang didekati oleh jaringan saraf, memprediksi gambar yang telah dihilangkan noise-nya pada setiap langkah.

Dengan menggabungkan ini, langkah-langkah inti untuk pelatihan difusi adalah:

- Memuat beberapa dataset, melakukan praproses gambar (augmentasi, normalisasi), dan menggunakan pemuat data untuk memuat data dalam batch.
- Seperti yang dijelaskan dalam Bab 2, model U-Net dapat digunakan untuk pembangkitan gambar. Model U-Net kustom dapat dibuat dengan menentukan saluran input/output dan menyusun blok downsampling dan upsampling.
- Penjadwal noise dapat digunakan untuk menentukan bagaimana noise (intensitas dan frekuensi) ditambahkan ke gambar secara bertahap selama beberapa langkah waktu. Ini penting untuk memastikan bahwa noise ditambahkan secara terkontrol dan bertahap, yang dapat membantu menstabilkan pelatihan.
- Untuk setiap masa, tambahkan noise ke langkah waktu saat ini. Model memprediksi noise yang ditambahkan dan kerugian dihitung sebagai Mean Squared Error antara noise yang diprediksi dan noise aktual. Parameter model diperbarui menggunakan backprop dan metrik ini (loss, learning rate, dll.) dicatat.

Di sisi inferensi:

- Anda memuat model U-Net yang telah dilatih dan penjadwal noise.
- Dimulai dengan sekumpulan gambar yang bising (noise acak), model digunakan untuk memprediksi noise pada gambar saat ini.
- Noise yang diprediksi dihilangkan dari gambar yang bising untuk mendapatkan gambar yang kurang bising.
- Proses ini diulangi untuk semua langkah waktu untuk secara bertahap menghilangkan noise dari gambar.

Kami menyarankan untuk memeriksa dokumentasi HuggingFace sebagai cara untuk memulai implementasi langkah-langkah ini.

DDPM (*Denoising Diffusion Probabilistic Models*)

DDPM adalah salah satu makalah pertama yang menunjukkan bagaimana model difusi dapat digunakan untuk menghasilkan gambar. Dua algoritma utama dalam makalah tersebut adalah untuk pelatihan dan pengambilan sampel, seperti yang ditunjukkan pada Gambar 4.20. Tahap pelatihan bertujuan untuk mengajarkan model memprediksi noise yang ditambahkan ke gambar pada berbagai langkah waktu. Langkah-langkah ini dilakukan per masa:

- Mengambil sampel sejumlah gambar dari beberapa distribusi data (dataset gambar). Mengambil sampel langkah waktu t secara seragam untuk setiap gambar dalam kelompok. Ini akan menentukan kapan dalam proses difusi kita menambahkan noise ke gambar. Setiap langkah waktu dimodelkan secara individual, dibandingkan dengan memodelkan seluruh proses difusi dengan satu fungsi. Ini meningkatkan stabilitas dan kecepatan pelatihan, karena model hanya perlu memprediksi noise untuk langkah waktu t , karena t diambil sampelnya secara seragam.
- Menghasilkan noise acak dari distribusi normal standar. Setiap gambar hingga langkah waktu t dalam batch diberi noise menggunakan persamaan yang menggabungkan gambar asli dengan noise yang diambil sampelnya.
- Model memprediksi noise berdasarkan gambar yang bising dan langkah waktu t . Kerugian *Mean Squared Error* (MSE) dihitung antara noise yang diprediksi dan noise aktual yang ditambahkan pada langkah waktu t .

Tujuan dari tahap pengambilan sampel adalah untuk menghasilkan gambar baru dengan menghilangkan noise secara iteratif dari gambar yang bising. Langkah-langkah utama untuk pengambilan sampel adalah:

- Mulailah dengan mengambil sampel gambar bising dari distribusi normal standar. Gambar saat ini berada pada langkah waktu akhir T .
- Secara iteratif dari $t = T$ hingga 1, jika $t > 1$, maka ambil sampel noise dari distribusi normal standar yang akan digunakan pada langkah waktu berikutnya. Jika $t = 1$, maka tidak diperlukan noise karena Anda berada pada langkah penghilangan noise terakhir. Noise ini akan digunakan pada bagian stokastik terakhir untuk menghitung gambar yang telah dihilangkan noise-nya pada setiap langkah waktu.
- Tujuannya adalah untuk memperkirakan gambar yang telah dihilangkan noise-nya pada langkah waktu sebelumnya ($t - 1$) berdasarkan gambar bising saat ini dan prediksi

noise dari model. Model yang terlatih memprediksi noise pada gambar saat ini pada t dan kemudian prediksi tersebut digunakan untuk menghilangkan noise dan memindahkannya ke $t - 1$.

- Perhitungan gambar yang telah dihilangkan noise-nya memiliki dua bagian utama. Bagian pertama adalah bagian deterministik di mana noise yang diprediksi dikurangi dari noise saat ini.

Bagian stokastik menambahkan sejumlah noise yang terkontrol kembali ke gambar menggunakan faktor penskalaan. Ini membantu memastikan adanya variasi dan model tidak terlalu deterministik.

Algoritma 1 Pelatihan	Algoritma 2 Pencuplikan
1: ulangi 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 3: $t \sim \text{Uniform}(\{1, \dots, T\})$ 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 5: Lakukan langkah penurunan gradien pada $\nabla_{\theta} \ \epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\ ^2$ 6: sampai konvergen	1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 2: untuk $t = T, \dots, 1$ lakukan 3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ jika $t > 1$, sebaliknya $\mathbf{z} = \mathbf{0}$ 4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 5: akhir untuk 6: kembalikan $\mathbf{x}_0$

Gambar 4.20. Algoritma pelatihan dan pengambilan sampel DDPM.

Untuk contoh bagaimana DDPM diimplementasikan, kami merekomendasikan tutorial ini.

Masalah utama dengan DDPM asli adalah skor log-likelihood-nya yang buruk. Ini berarti bahwa meskipun dapat menghasilkan gambar berkualitas tinggi, ia tidak secara akurat sesuai dengan distribusi data gambar nyata. DDPM yang ditingkatkan memperkenalkan beberapa variasi untuk mengatasi hal ini:

- **Prediksi varian:** Alih-alih menggunakan varians tetap, DDPM yang ditingkatkan mempelajari varians distribusi noise, yang membantu meningkatkan log-likelihood dan stabilitas model. Ini dilakukan dengan menginterpolasi varians antara batas atas dan batas bawah.
- **Penjadwalan laju pembelajaran:** Penjadwalan linear noise asli digantikan dengan interpolasi kosinus, yang menghasilkan kontrol yang lebih baik atas penambahan dan penghapusan noise.
- **Peningkatan langkah waktu:** Jumlah langkah waktu dalam proses difusi ditingkatkan dari 1.000 menjadi 4.000, yang meningkatkan kualitas sampel. Namun, ini meningkatkan waktu dan komputasi yang dibutuhkan untuk pelatihan dan pengambilan sampel.

DDIM (*Denoising Diffusion Implicit Models*)

Dalam pendekatan DDPM, model memprediksi noise yang ditambahkan ke gambar bersih untuk menghasilkan gambar yang bising. Ini dilakukan secara bertahap dan dapat dilakukan melalui banyak langkah. Alih-alih memprediksi noise yang ditambahkan ke gambar pada setiap langkah seperti yang dilakukan DDPM, DDIM memprediksi noise sehingga ketika

noise dan gambar bersih dicampur, mereka menghasilkan x_t . Hal ini memungkinkan model untuk mendekati gambar bersih akhir dalam langkah yang lebih sedikit.

Proses ini dikendalikan oleh σ , yang menentukan berapa banyak noise yang ditambahkan pada setiap langkah. Jika nilainya 0, maka prosesnya deterministik. Model kemudian memprediksi seperti apa gambar bersih dari gambar yang bising. Prediksi ini diberikan oleh Persamaan 4.5, yang mengurangi noise dari gambar yang bising dan kemudian menskalakannya:

$$f_{\theta}^t(x_t) := (x_t - \sqrt{1 - \alpha_t} \epsilon_{\theta}(x_t, t)) / \sqrt{\alpha_t}$$

Persamaan 4.5

Untuk menghasilkan gambar pada langkah $t - 1$ dari gambar pada langkah t , digunakan persamaan (Persamaan 4.6), yang menggabungkan gambar bersih yang diprediksi dan beberapa noise untuk melangkah mundur dari x_t ke x_{t-1} . Gambar bersih yang diprediksi diskalakan agar sesuai dengan langkah waktu sebelumnya. Beberapa noise ditambahkan kembali dan diskalakan kembali agar sesuai dengan langkah waktu sebelumnya. Sedikit noise, yang dikendalikan oleh σt , juga ditambahkan untuk memastikan adanya variabilitas dan prosesnya tidak sepenuhnya deterministik.

$$x_{t-1} = \sqrt{\alpha_{t-1}} f_{\theta}^t(x_t) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \epsilon_{\theta}(x_t, t) + \sigma_t \epsilon$$

Persamaan 4.6

Terdapat juga kasus Persamaan 4.6 di mana σt diatur ke 0 sehingga tidak ada noise yang ditambahkan dan prosesnya lebih deterministik. Model dan tujuan pelatihan yang digunakan sama dengan DDPM, tetapi idenya adalah bahwa teknik ini memungkinkan langkah yang lebih sedikit dan lebih besar untuk diambil.

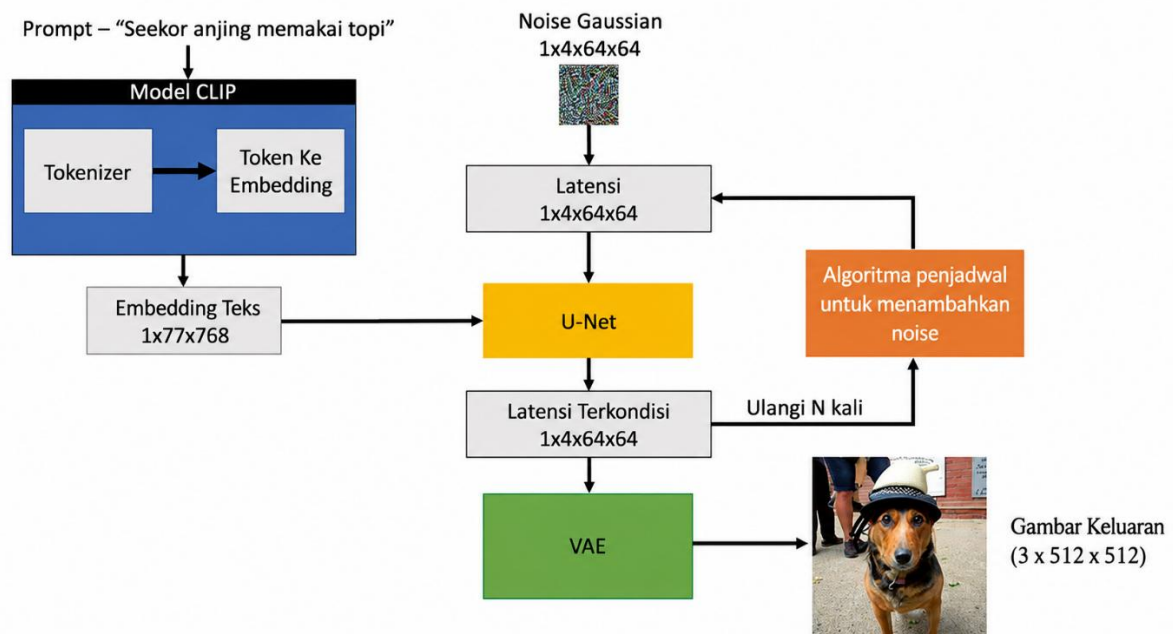
Difusi Stabil

Proses difusi yang dijelaskan sejauh ini bekerja langsung di ruang piksel gambar mentah. Ini bisa memakan waktu lama untuk gambar yang sangat besar (banyak piksel untuk dihilangkan noise-nya) dan dapat mengalami masalah pemodelan detail halus dalam gambar secara tidak tepat. Difusi stabil adalah metode populer yang mengatasi kedua masalah ini dengan mengompresi gambar ke dalam ruang laten sebelum menerapkan model difusi. Difusi stabil menggemparkan dunia dengan output yang dihasilkan AI yang sangat luar biasa dan realistis.

Selama fase pelatihan difusi stabil, Anda mengambil gambar dan melewatinya melalui *Variational Autoencoder* (VAE), yang mengompresinya menjadi vektor laten. Encoder membantu mengurangi citra menjadi representasi berdimensi lebih rendah sambil mempertahankan fitur-fitur penting—anggap ini sebagai kompresi semantik. Vektor ini kemudian digunakan sebagai titik awal proses difusi dan noise ditambahkan ke vektor laten pada setiap langkah waktu untuk membuatnya lebih bising. Vektor bising yang dihasilkan

menjadi titik awal proses difusi terbalik, di mana U-Net digunakan untuk memprediksi dan menghilangkan noise pada setiap langkah. U-Net ini memiliki pengkondisian teks, yang berarti bahwa ia juga menggunakan embedding teks dari kueri input pengguna (selama inferensi) atau keterangan (dalam dataset pelatihan) untuk memandu pembuatan citra.

Vektor laten yang telah dihilangkan noise-nya dilewatkan melalui decoder, yang merekonstruksi citra asli—anggap ini sebagai dekompresi. Secara keseluruhan, tujuan dari pendekatan ini adalah untuk memastikan citra berkualitas tinggi yang sesuai dengan input pengguna pada saat inferensi dengan merekonstruksinya dari vektor laten yang telah dikondisikan. Karena proses difusi beroperasi pada pengkodean laten dan bukan pada gambar asli, proses ini cukup cepat untuk digunakan dalam produk. Anda dapat melihat langkah-langkah ini pada Gambar 4.21.



Gambar 4.21. Difusi stabil menghasilkan gambar dengan menghilangkan derau secara iteratif pada representasi gambar laten, yang dikondisikan pada embedding teks yang dihasilkan oleh model CLIP. U-Net memperbaiki laten yang bising selama beberapa iterasi, dan representasi laten akhir didekode oleh VAE menjadi gambar keluaran.

Generasi Terkondisi

Anda telah melihat dalam difusi stabil bagaimana kueri atau keterangan masukan pengguna digunakan untuk memandu gambar yang dihasilkan melalui pengkondisian. Bagian ini menggali lebih dalam tentang pengkondisian untuk difusi. Ada dua varian pengkondisian yang populer digunakan: panduan pengklasifikasi dan tanpa pengklasifikasi. Perbedaan utama antara keduanya adalah bahwa dalam kasus panduan pengklasifikasi, pengklasifikasi terpisah perlu dilatih, sedangkan panduan tanpa pengklasifikasi tidak memerlukan jaringan pengklasifikasi eksternal.

Difusi yang Dipandu Pengklasifikasi

Tujuan dalam difusi yang dipandu pengklasifikasi adalah untuk menghasilkan gambar yang berada dalam kelas tertentu atau memiliki atribut tertentu. Hal ini dicapai dengan melatih jaringan pengklasifikasi terpisah pada label dari dataset yang sama dengan yang digunakan untuk melatih jaringan difusi. Kemudian, jaringan pengklasifikasi digunakan untuk memandu proses pembangkitan gambar dengan menggunakan aturan Bayes untuk menggabungkan probabilitas gambar dan kelas (lihat Persamaan 4.7):

$$p(x_t|c) \propto p(c|x_t)p(x_t)$$

Persamaan 4.7

Pembuatan gambar diperbarui berdasarkan kombinasi gradien probabilitas log gambar dan kelas (lihat Persamaan 4.8).

$$\nabla_{x_t} \log p(x_t|c) = \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(c|x_t)$$

Persamaan 4.8

Menggabungkan sub-waktu sinyal membutuhkan parameter linier yang disebut skala panduan, γ . Panduan yang lebih tinggi berarti gambar lebih memperhatikan kelas dan kurang beragam, dan panduan yang lebih rendah berarti gambar kurang spesifik kelas tetapi lebih beragam.

Panduan Tanpa Klasifikasi

Ide utama dalam panduan tanpa klasifikasi adalah melatih model difusi untuk memahami label kelas secara langsung tanpa memerlukan pengklasifikasi terpisah, yaitu, pengklasifikasi Anda adalah model itu sendiri. Model dilatih untuk menghasilkan gambar dengan dan tanpa informasi kelas tertentu, dengan memasukkan label 20-30 persen dari waktu selama pelatihan. Selama penghilangan noise, dua prediksi dibuat—satu dengan label sebagai pengkondisian dan satu tanpa. Probabilitas generasi bersyarat dan generasi tanpa syarat digabungkan sambil menerapkan bobot untuk menyeimbangkan pentingnya panduan kelas, γ . Setelah persamaan gradien gabungan ini diperluas dan disederhanakan, Anda mendapatkan Persamaan 4.9:

$$\nabla_{x_t} [(1 - w) \log p(x_t) + w \log p(x_t|c)]$$

Persamaan 4.9

Di mana $\gamma = w/(1 - w)$. Seperti pada generasi yang dipandu pengklasifikasi, γ yang lebih tinggi menyiratkan lebih banyak perhatian pada label kelas dengan mengorbankan keragaman. Panduan tanpa pengklasifikasi lebih sederhana sekaligus memberikan lebih banyak kendali dan fidelitas atas generasi. Ini juga menghindari generasi yang merugikan yang dipicu oleh jaringan pengklasifikasi terpisah.

Panduan Berbasis Teks

Panduan berbasis teks digunakan dalam banyak produk AI generatif yang menghasilkan gambar/video yang secara khusus diminta oleh pengguna. Untuk tujuan ini, perintah teks yang menggambarkan gambar keluaran diubah menjadi embedding dengan encoder teks. Embedding teks diintegrasikan ke dalam model U-Net menggunakan cross-attention (lihat Persamaan 4.10):

$$\epsilon_{\theta}(x_t, t, c)$$

Persamaan 4.10

Persamaan 4.11 merepresentasikan prediksi model pada langkah t , menggunakan embedding gambar dan teks. Selama pembuatan gambar, model menggunakan embedding teks untuk memandu setiap langkah:

$$\nabla_{x_t} \log p(x_t | c)$$

Persamaan 4.11

Gradien akan menyesuaikan gambar berdasarkan deskripsi teks, seperti panduan yang dibahas sebelumnya. Secara keseluruhan, deskripsi teks digunakan untuk memandu model dalam menghasilkan gambar yang sesuai dengan deskripsi.

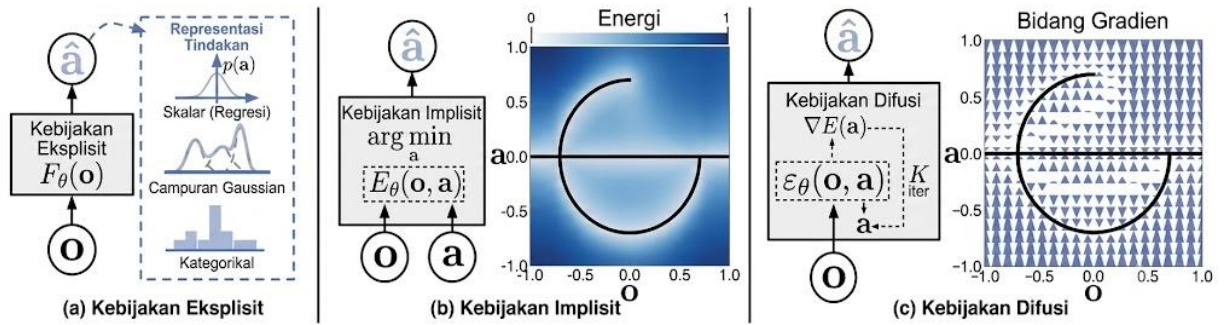
Difusi Aksi untuk Kontrol Robot

Kebijakan Difusi adalah metode yang, alih-alih langsung menentukan tindakan robot, menggunakan difusi peredam bising bersyarat untuk menghasilkan tindakan terbaik yang harus diambil. Kebijakan difusi dapat mengambil sampel dari ruang keluaran berdimensi sangat tinggi. Penyempurnaan iteratif dalam difusi juga memungkinkan tindakan yang lebih fleksibel dan mudah beradaptasi. Beberapa kebijakan umum lainnya yang digunakan untuk tindakan meliputi:

- **Kebijakan eksplisit:** Ini secara langsung memetakan input visual ke tindakan robot dan dapat memiliki representasi tindakan yang berbeda, seperti regresi, campuran Gaussian, dan sebagainya. Salah satu keterbatasan kebijakan ini adalah bahwa tindakan yang langsung dikeluarkan dapat kurang fleksibel dan lebih sulit untuk dilatih untuk perilaku yang lebih kompleks karena ada pemetaan langsung yang dipelajari dari input ke output.
- **Kebijakan implisit:** Metode ini menggunakan fungsi energi untuk mendefinisikan kebijakan secara implisit, di mana kebijakan mempelajari fungsi energi yang dikondisikan pada tindakan dan pengamatan. Salah satu keterbatasan metode ini adalah bahwa optimasi selama inferensi dapat memakan banyak sumber daya komputasi ketika harus menyelesaikan konfigurasi energi minimum.

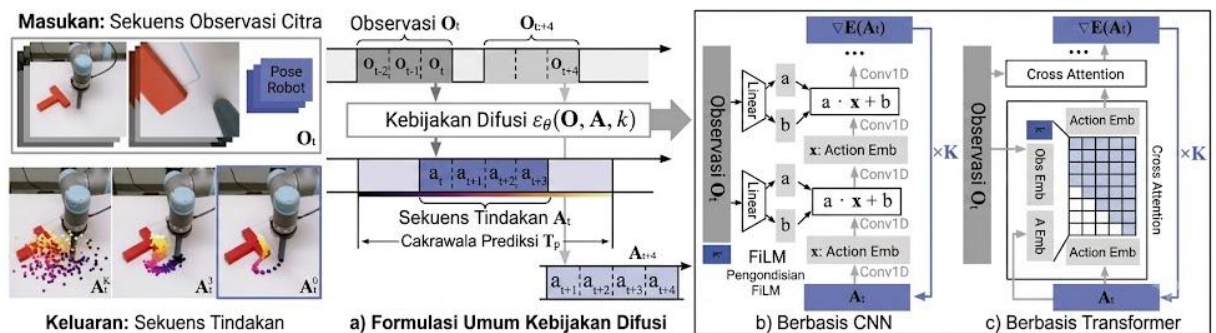
Alih-alih langsung menghasilkan tindakan, dalam kebijakan difusi, kebijakan tersebut menyimpulkan gradien skor tindakan berdasarkan pengamatan visual. Proses dimulai dengan serangkaian gambar sebagai masukan, yang dikodekan untuk mengekstrak fitur yang relevan.

Kebijakan difusi mengambil pengamatan ini dan menghasilkan serangkaian tindakan selama rentang waktu tertentu. Ini dapat dilihat sebagai tindakan awal yang bising yang diberikan sebagai tebakan kasar, yang disempurnakan selama K iterasi. Seperti yang ditunjukkan pada Gambar 4.22, model menyempurnakan kebisingan menjadi tindakan menggunakan medan gradien dalam kebijakan difusi.



Gambar 4.22. Perbedaan antara kebijakan eksplisit, implisit, dan difusi.

Seperti yang ditunjukkan pada Gambar 4.23, langkah-langkah umumnya adalah robot mendapatkan serangkaian gambar dari lingkungan, yang dikodekan. Kebijakan tersebut menggunakan pengamatan yang dikodekan untuk menghasilkan serangkaian tindakan, yang disempurnakan secara iteratif. Berdasarkan pengamatan baru, tindakan-tindakan ini disempurnakan untuk memandu pergerakan robot. Implementasi kebijakan spesifik menggunakan dua pendekatan: a) berbasis CNN, yang menggunakan lapisan konvolusional dan FiLM untuk mengkondisikan setiap lapisan dengan beberapa fitur pengamatan dan menyempurnakan tindakan dan b) berbasis transformer, yang menggunakan cross-attention untuk mengintegrasikan pengamatan dan tindakan sambil memanfaatkan kekuatan transformer untuk menangani data sekuensial. Untuk detail lebih lanjut, kami sarankan untuk melihat makalah.



Gambar 4.23. Gambaran umum dari pipeline kebijakan difusi yang menghasilkan urutan aksi untuk dilakukan robot.

Sejak karya ini, telah ada varian lain dari kebijakan difusi untuk kontrol robot, seperti Aloha Unleashed, di mana dekode transformer secara iteratif memperbaiki aksi dari noise, sambil menerima embedding observasi dan timestep difusi sebagai input, melalui cross-attention. Model difusi memberikan alternatif untuk model autoregresif untuk kontrol robot, dan mereka dapat memodelkan aksi fidelitas tinggi, karena menghasilkan aksi float secara langsung alih-alih menghasilkan token. Dalam praktiknya, mereka juga dapat memodelkan multimodalitas dan dataset yang bising dengan jauh lebih baik, seperti yang dibahas sebelumnya dalam bab ini.

Menggabungkan VLM dan Model Difusi

Baru-baru ini, kita telah melihat arsitektur yang menggabungkan VLA dan model difusi sebagai model sistem 1 / sistem 2 dan dilatih bersama. Pi0, Groot N1 dan Gemini Robotics adalah beberapa varian ini. Menggabungkan VLM besar dengan difusi memungkinkan untuk memanfaatkan generalitas dan kapasitas VLA sambil mempertahankan perilaku halus dari difusi. Dalam Groot N1, input ke model difusi adalah output dari salah satu lapisan akhir VLM. Selama waktu inferensi, model dapat dipisahkan, dengan model besar berada di cloud dan model yang lebih kecil berada di robot seperti pada.

4.4 PEMBELAJARAN DARI DEMONSTRASI VIDEO

Paradigma kunci dalam pembelajaran robot adalah pembelajaran dari demonstrasi video (LfV). Tantangan terbesar dalam pembelajaran dari demonstrasi video adalah tidak adanya label aksi, yang membuat konversi data video menjadi data robotik yang dapat ditindaklanjuti menjadi lebih sulit. Tantangan kedua adalah bahwa sebagian besar data video skala internet memiliki pergeseran distribusi relatif terhadap penyebaran robot, tetapi dalam beberapa kasus ini mungkin menjadi keuntungan karena keragaman data video memungkinkan generalisasi ke berbagai macam adegan yang belum pernah dilihat robot sebelumnya. Tantangan ketiga, atau peluang, adalah bahwa kumpulan data video sumber terbuka yang dikurasi dengan baik dengan anotasi bahasa yang dapat digunakan untuk robotika masih relatif terbatas atau berukuran kecil.

Tantangan pertama dapat diatasi dengan cara berikut:

1. **Menyimpulkan gerakan dari video:** Menambahkan label aksi proksi ke data video, melalui penggunaan aliran optimal. Misalnya dengan memberi anotasi pada posisi tangan, atau menyimpulkan posisi tangan melalui pelacakan monokular/stereo.
2. **Bahasa tingkat tinggi sebagai tindakan:** Anotasi menggunakan model keterangan, deskripsi bahasa tentang gerakan, dapat diturunkan. Meskipun jenis informasi ini berguna untuk mempelajari semantik, bahasa tersebut kurang memiliki ketelitian yang diperlukan untuk kontrol yang tepat.
3. **Mempelajari tindakan laten dari video:** Biasanya, auto-encoder mengompresi dan mendekomposisi video, dan kemudian menggunakan kompresi yang dipelajari sebagai variabel laten.

Pemodelan Dunia Menggunakan Data Video

Model dunia adalah representasi internal dunia: fisika dunia, bagaimana objek berinteraksi dan dinamikanya, dan apa yang mungkin dimiliki robot untuk mengantisipasi hasil dari tindakannya sendiri atau tindakan agen dinamis lainnya. Salah satu cara untuk melihatnya adalah bahwa itu adalah simulasi internal.

Cara umum untuk membuat model dunia adalah dengan mempelajari representasi laten yang baik dari pengamatan. Video memberikan peluang yang sangat kaya untuk membuat model dunia yang ekstensif. Biasanya, jaringan encoder mengkodekan video ke dalam representasi laten, dan kemudian model dinamika memprediksi keadaan laten berikutnya dan imbalannya, menggunakan keadaan saat ini dan tindakan yang tertanam sebagai masukan. Kerugian rekonstruksi biasanya digunakan untuk melatihnya. Dalam sebuah video, dataset menyajikan keadaan masa depan, sehingga memberikan sinyal pelatihan yang berguna. Model dunia yang dipelajari kemudian dapat disempurnakan untuk mempelajari dinamika aksi umum berdasarkan perwujudan robot tertentu. Setelah Anda membuat model dunia ini, Anda dapat menggunakannya untuk berbagai aplikasi robotik hilir.

Penalaran dari Video

Perencanaan bahasa video adalah contoh dari metode ini. Metode ini menalar tentang dunia dan membuat rencana dalam video dan teks multimodal, mirip dengan PaLM-e, yang dibahas sebelumnya, tetapi dengan komponen temporal. Data video skala internet dengan anotasi bahasa menyimpan banyak pengetahuan kontekstual, menghadirkan peluang untuk memperoleh generalisasi. Robot dapat mempelajari tindakan tingkat rendah dari "dataset robot" mereka saat ini sambil memperoleh pemahaman tugas tingkat tinggi dari dataset video melalui isyarat visual dan langkah-langkah yang diperlukan untuk menyelesaikan tugas.

Aksi dari Video dengan Penyempurnaan Model Dunia

Model Dunia Terstruktur untuk Intentionalitas, atau SWIM, adalah sebuah karya yang membantu robot mempelajari tugas manipulasi dengan pertama-tama menciptakan model dunia dari video manusia dan kemudian menyempurnakannya pada data robot. SWIM melatih model dunia (yang memprediksi bagaimana lingkungan berubah ketika tindakan dilakukan) dan model affordance (yang mempelajari di mana dan bagaimana berinteraksi dengan objek, yaitu menggenggam) menggunakan data video manusia. Model affordance ini digunakan untuk memandu perilaku robot yang mengumpulkan data di dunia nyata, dan data dari robot digunakan untuk menyempurnakan model dunia, sehingga mewujudkannya secara tidak terawasi. Dapat dikatakan bahwa model dasar saat ini sudah memiliki model dunia implisit karena dilatih di Internet dan berbagai macam data. Oleh karena itu, menyempurnakannya pada tindakan merupakan ekspresi dari mentransfer pemahaman model dunia mereka ke konsep tindakan dan gerakan. Hasil dari RT2 yang dibahas sebelumnya dalam bab ini memberikan bukti untuk beberapa klaim ini.

Model Aksi dengan Pengkondisian pada Representasi Video

Pembelajaran kebijakan yang dikondisikan video menggunakan gagasan bahwa robot dapat mengamati manusia (atau robot) yang melakukan suatu tugas dan menyimpulkan tindakan yang diperlukan untuk menyelesaikan tugas tersebut di lingkungannya sendiri. Hal ini

bermanfaat, karena memungkinkan non-ahli untuk menentukan tugas bagi robot secara intuitif tanpa memerlukan desain fungsi penghargaan yang kompleks atau definisi tujuan yang eksplisit. Misalnya, Anda cukup menunjukkan kepada robot cara melakukan suatu tugas, kemudian video tersebut menjadi masukan untuk kebijakan. Vid2Robot adalah contoh dari metode jenis ini.

Selain memberikan petunjuk langsung dengan video, pengguna juga dapat mengekstrak representasi tugas tingkat tinggi dari video melalui pelatihan awal skala besar, dan kemudian menggunakannya untuk pembelajaran kebijakan hilir, sehingga mengkondisikan kebijakan pada embedding yang diekstrak dari video. Misalnya, *Time-Contrastive Networks* (TCN) mengekstrak fitur visual seperti interaksi tangan-objek dan relasi spasial dari video multi-view dan *Contrastive Video Representation Learning* (CVRL) mempelajari representasi spasial-temporal dari data video.

Model Aksi dari Pemikiran dalam Video

Pendekatan menarik terhadap prediksi aksi yang ingin kami soroti adalah Policy-as-Video. Pendekatan ini memodelkan dunia dalam gambar. Pendekatan ini memperkenalkan *Unified Predictive Decision Process* (UPDP) sebagai alternatif untuk MDP yang menggunakan gambar untuk merepresentasikan lingkungan alih-alih keadaan numerik. Generator video memprediksi lintasan gambar yang menunjukkan bagaimana robot harus bergerak dari keadaan awalnya untuk mencapai tujuan yang dijelaskan oleh teks.

Setelah video dihasilkan, model dinamika invers (IDM) mengekstrak aksi yang sesuai (misalnya, perintah motor) dari frame yang diprediksi. Model difusi digunakan untuk menghasilkan urutan video dalam UPDP dan secara bertahap menghilangkan noise pada gambar menuju tujuan akhir. Salah satu fitur utama UPDP adalah pemisahan proses perencanaan (pembuatan video) dari eksekusi (ekstraksi aksi), yang dapat membuat metode ini lebih fleksibel untuk berbagai robot dan lingkungan.

4.5 KEAMANAN AI UNTUK ROBOTIKA

Karena kita mempercayai model AI untuk melakukan tindakan di dunia fisik secara otonom, kita perlu menyadari risiko keamanan yang berasal dari aplikasi ini. Sebuah studi praktis tentang risiko keamanan menunjukkan bahwa serangan fisik, jaringan, dan perangkat lunak adalah skenario yang mungkin terjadi. Serangan fisik pada robot, seperti merusak perangkat keras atau memberi mereka data sensor palsu (pemalsuan sensor) dapat secara langsung mengubah perilaku robot, menyebabkan mereka mengalami kerusakan atau kegagalan.

Serangan pada lapisan persepsi (sensor seperti IMU, GPS, LiDAR, kamera, dll.) dapat mengganggu kemampuan robot untuk memahami lingkungannya dan ini dapat lebih lanjut memengaruhi kemampuan navigasi dan kontrolnya. Di sisi perangkat keras, peretasan aktuator dan sistem kontrol dapat secara langsung menyebabkan perilaku berbahaya. Serangan jaringan, seperti *Denial-of-Service* (DoS), dapat membebani jaringan robot dan menyebabkan perlambatan atau kegagalan sistem. Serangan tingkat perangkat lunak, seperti menyuntikkan

kode berbahaya, dapat mengubah tindakan robot dan menyebabkan tindakan yang merugikan.

Untuk melindungi robot dari berbagai ancaman keamanan, beberapa strategi pertahanan meliputi:

1. Sistem deteksi anomali yang memantau data sensor untuk ketidaknormalan dan kemudian memicu mode pengaman ketika masalah terdeteksi.
2. Penyaringan dan fusi sensor dapat mengurangi pemalsuan dan gangguan dengan merujuk silang beberapa sumber data (misalnya, GPS, LiDAR) untuk memverifikasi apakah data konsisten.
3. Sanitasi dan validasi data dapat membantu memastikan bahwa input diperiksa untuk konten berbahaya sebelum diproses.
4. Pemeriksaan integritas kriptografi dan pembaruan firmware yang aman melindungi dari kerusakan data dan eksekusi kode yang tidak sah.
5. Mekanisme pengaman dan protokol pembaruan yang aman dapat berguna bagi robot untuk kembali ke keadaan aman dan hanya menerima pembaruan yang terverifikasi.

Untuk detail lebih lanjut tentang serangan dan mekanisme pertahanan potensial, kami sarankan untuk membaca.

Meskipun ancaman ini lebih praktis, ada juga jangkauan yang lebih luas dan resistensi filosofis terhadap otonomi. Seiring dengan semakin cerdasnya AI dan semakin terintegrasinya ke dalam kehidupan sehari-hari, kesadaran situasionalnya dapat meningkat, sehingga memungkinkan mereka untuk menipu manusia, membujuk kita untuk melakukan perilaku berbahaya, menyebarkan informasi yang salah, dan sebagainya. Ada beberapa organisasi yang bekerja untuk mengatasi ancaman eksistensial jangka panjang terhadap umat manusia dari AI. Hal ini sangat kontras dengan orang-orang yang bekerja pada etika AI, yang berfokus pada risiko jangka pendek dari AI seperti deep fake, bias dalam keluaran AI, masalah privasi, dan sebagainya.

Prinsip-prinsip inti keselamatan AI Anthropic adalah standar yang menentukan industri untuk merancang sistem AI yang lebih aman. Berikut adalah beberapa cara kita dapat menggunakan prinsip-prinsip ini untuk meningkatkan sistem robotika yang digerakkan oleh AI:

- Interpretasi mekanistik berfokus pada pemahaman proses pengambilan keputusan internal model AI (interpretasinya). Dalam robotika, ini berarti mampu mendeteksi apakah robot mungkin salah menafsirkan suatu situasi sebelum berpotensi melakukan tindakan berbahaya. Salah satu bidang studi keselamatan AI adalah rekayasa balik model dan membangun alat untuk interpretasi model sehingga para insinyur dapat mendeteksi apakah robot mungkin memiliki perilaku yang tidak diinginkan, seperti bergerak ke area berbahaya atau salah mengidentifikasi suatu objek. Kemajuan terbaru dalam interpretasi adalah studi dari OpenAI yang menunjukkan bahwa model yang lebih kecil dapat dijelaskan dengan model yang lebih besar. Dalam studi tersebut, GPT-4 menghasilkan penjelasan untuk peran neuron di GPT-2, dan kemudian neuron disimulasikan untuk menilai penjelasan tersebut, memberikan cara untuk interpretasi yang terukur. Namun, masih ada pertanyaan apakah semua neuron dapat

diinterpretasikan, jika mereka tidak memiliki penjelasan semantik yang jelas dan mudah dipahami manusia untuk peran mereka.

- Red teaming adalah teknik di mana model AI diuji coba dengan kondisi sulit atau antagonis sebelum digunakan untuk mengidentifikasi dan memperbaiki kerentanan sebelum menerapkannya di dunia nyata. Pengujian model AI dalam lingkungan yang terkontrol dapat membantu mengidentifikasi dan memperbaiki titik kegagalan. Dalam konteks robotika, misalnya, robot dapat menghadapi skenario di mana sensor mereka diberi informasi yang menyesatkan, yang dapat menyebabkan tindakan berbahaya, seperti menabrak atau salah navigasi. Dengan sengaja menempatkan robot dalam kondisi antagonis yang terkontrol ini, para insinyur dapat mendeteksi dan mengurangi titik kegagalan ketika diserang oleh ancaman fisik atau siber, seperti pemalsuan sensor.
- Penyelarasan mengacu pada masalah memungkinkan sistem AI untuk mengikuti nilai-nilai manusia dan mematuhi preferensi manusia. Seperti yang dibahas sebelumnya dalam bab ini, fase pasca-pelatihan terutama digunakan untuk meningkatkan penyelarasan model, bersama dengan kemampuan mengikuti instruksi mereka. Dalam banyak situasi, terutama pada robot yang dikerahkan secara real-time, tidak praktis bagi manusia untuk memberikan pengawasan real-time yang konstan. Mengembangkan teknik yang memungkinkan sistem robot untuk memantau dan memperbaiki perilaku mereka sendiri tanpa pengawasan manusia akan menjadi penting. Constitutional AI dan RLAIIF adalah dua karya yang memajukan hal ini. RLAIIF menyediakan cara untuk meningkatkan umpan balik manusia dengan melatih model dari preferensi manusia yang dapat memberikan umpan balik tanpa hambatan keterlibatan manusia dan bertindak sebagai kritikus selama pelatihan jaringan besar. Constitutional AI memperkenalkan konstitusi atau sistem nilai, yang digunakan oleh model bahasa untuk menalar tentang respons mereka dan mengeditnya agar selaras dengan konstitusi selama pasca-pelatihan.
- Robot harus mampu beroperasi di berbagai lingkungan tanpa pemrograman ulang. Membangun generalisasi ke dalam pelatihan model, dengan melatih pada berbagai kondisi dan lingkungan dan mengujinya, dapat menjadi sangat penting. Jika robot dilatih di pabrik tertentu, ia mungkin memiliki perilaku yang tidak diinginkan tanpa pemrograman ulang di lingkungan baru. Memastikan dan menguji generalisasi dapat bermanfaat tidak hanya untuk kinerja tetapi juga untuk keselamatan dalam robotika.

Untuk aspek keselamatan praktis lebih lanjut selama penerapan, lihat Bab 11.

4.6 RINGKASAN

Singkatnya, bab ini membahas poin-poin berikut:

- 🚦 Bagaimana perencanaan, kontrol, dan pemetaan robot dapat dibingkai sebagai masalah bahasa dan berbagai cara untuk menyelesaikannya dengan LLM menggunakan teknik loop terbuka dan loop tertutup.
- 🚦 Model dasar besar dilatih terlebih dahulu pada korpus data yang besar dan kemudian disempurnakan menggunakan penyempurnaan terawasi (SFT) atau optimasi preferensi

langsung (DPO). Hukum penskalaan membantu memandu pengembangannya dengan memprediksi peningkatan kinerja seiring bertambahnya ukuran, data, atau komputasi model. Tolok ukur seperti MMLU dan HumanEval mengevaluasi kinerja model bahasa di berbagai tugas.

- ✚ Model seperti SayCan memecah instruksi tingkat tinggi untuk dieksekusi oleh robot dan PaLM-e menggabungkan input multimodal (teks, visi) untuk perencanaan. Model-model ini membantu robot memahami tugas dalam konteks, menghasilkan tindakan, dan menyesuaikan diri.
- ✚ Metode seperti NLMap-SayCan memungkinkan robot untuk membangun peta lingkungan sekitarnya dan merencanakan berdasarkan objek yang tidak terlihat. Code as Policies memungkinkan LLM untuk menerjemahkan instruksi bahasa alami ke dalam kode yang dapat dieksekusi untuk robot menggunakan model dan contoh yang telah dilatih sebelumnya untuk memandu eksekusi tugas.
- ✚ Model seperti RT-1, RT-2, dan RT-X menggunakan transformer untuk mengontrol tindakan robot secara langsung dari input visual dan tekstual, memungkinkan kontrol multi-tugas dan dunia nyata di berbagai robot dan tugas.
- ✚ Kebijakan berbasis difusi menawarkan fleksibilitas dalam menghasilkan urutan tindakan berdimensi tinggi melalui penyempurnaan iteratif.
- ✚ Robot dapat mempelajari tugas dengan mengamati demonstrasi video, mengekstrak representasi tugas tingkat tinggi, dan menerapkan tugas-tugas tersebut pada tindakan mereka sendiri, menjembatani kesenjangan antara demonstrasi manusia dan tugas robot.
- ✚ Robot yang digerakkan oleh AI menghadapi risiko keamanan praktis, seperti perusakan fisik, serangan jaringan, dan kerentanan perangkat lunak, sekaligus menimbulkan risiko jangka panjang terkait otonomi dan generalisasi. Mekanisme keselamatan yang kuat, deteksi anomali, dan interpretasi sangat penting untuk operasi robot yang aman.

Bab selanjutnya membahas bagaimana simulasi dapat digunakan dalam robotika, simulator umum, kelebihan dan kekurangannya, serta metode seperti pengacakan domain dan adaptasi domain, yang digunakan untuk mengatasi kesenjangan Sim2Real. Bab selanjutnya juga membahas metode pembelajaran umum menggunakan simulator untuk RL dan IL.

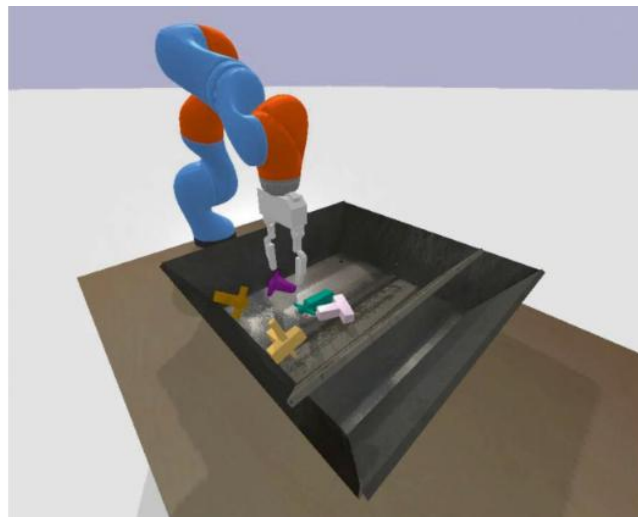
BAB 5

SIMULASI ROBOT

5.1 SIMULASI DALAM PELATIHAN ROBOTIKA

Memperoleh data dunia nyata untuk melatih model robot bisa mahal dan memakan waktu. Banyak model membutuhkan triliunan lintasan untuk mempelajari semua skenario di mana robot dapat beroperasi dan setiap kombinasi tindakan potensial yang dapat dilakukannya. Ini bisa memakan waktu puluhan tahun untuk dikumpulkan, banyak modal untuk dikerahkan, dan lebih sulit untuk diskalakan karena adanya perangkat keras dalam prosesnya. Untuk mengatasi masalah ini, beberapa peneliti dan perusahaan telah menerapkan teknologi simulasi yang bertujuan untuk menciptakan data sintesis yang sangat mirip dengan dunia nyata. Salah satu contoh mesin tersebut adalah Isaac Sim dari NVIDIA, yang menghasilkan data sintesis untuk grafis yang realistis.

Memasukkan data simulasi dalam alur pelatihan robotika memiliki berbagai manfaat. Simulator menyediakan lingkungan terkontrol di mana robot dapat belajar dan mengeksekusi tindakannya tanpa bahaya yang terkait dengan pengujian dunia nyata. Data simulasi yang mencakup berbagai adegan memungkinkan model untuk melakukan generalisasi lebih efektif ke skenario baru. Lihat Gambar 5.1. Namun, sangat penting bahwa data simulasi secara memadai mewakili dinamika dunia nyata dan menjembatani kesenjangan "simulasi-ke-nyata" (Sim2Real), yaitu kesenjangan yang menjelaskan mengapa model yang dilatih dalam simulasi mungkin tidak berkinerja sebaik di dunia nyata.



Gambar 5.1. Lingkungan genggam robot yang disimulasikan di PyBullet.

Pertimbangan untuk Simulasi dalam Robotika

Sebelum Anda mempelajari cara kerja mesin simulasi ini dan bagaimana penggunaannya untuk robotika, penting untuk mempertimbangkan manfaat dan keterbatasan penggunaan simulasi dalam aplikasi Anda.

Manfaat simulasi meliputi:

- **Skalabilitas lebih cepat dengan keragaman:** Keuntungan utama menggunakan simulasi adalah Anda dapat menggunakan AI generatif untuk menghasilkan adegan yang sangat beragam dalam simulasi, yang kemudian dapat digunakan untuk melatih model. Ini hemat biaya dan dapat diskalakan lebih baik daripada menyiapkan adegan tersebut di dunia nyata dan mengumpulkannya dengan robot.
- **Siklus iterasi lebih cepat:** Lingkungan simulasi memungkinkan pengembangan dan pengujian algoritma baru yang lebih mudah, yang mengarah pada iterasi rekayasa yang lebih cepat. Sebagai perbandingan, siklus iterasi ini pada robot fisik di dunia nyata mungkin membutuhkan waktu yang sangat lama atau mungkin sangat mahal.
- **Pengujian fitur kritis keselamatan:** Banyak fitur keselamatan yang penting dalam situasi berbahaya lebih sulit diuji di dunia nyata. Fitur-fitur ini lebih baik diuji secara menyeluruh dalam simulasi, di mana Anda dapat dengan mudah dan aman mensimulasikan perilaku agen yang berbahaya, tabrakan, dan sebagainya, tanpa menyebabkan bahaya nyata. Ini sangat berguna untuk aplikasi kritis keselamatan seperti robot otonom atau robot bedah.
- **Reproduksibilitas dan perbandingan untuk eksperimen:** Seringkali sulit untuk mereproduksi hasil tertentu di dunia nyata karena hasil dan data dunia nyata dapat bising atau terus berubah. Lingkungan simulasi dirancang agar dapat dikontrol dan direproduksi sehingga dapat diandalkan, yang memungkinkan lingkungan tersebut juga digunakan untuk membandingkan algoritma dalam eksperimen yang identik. Lingkungan simulasi juga dapat digunakan untuk membuat data bersih untuk diablasikan untuk eksperimen yang sangat spesifik.

Beberapa keterbatasan penggunaan simulator meliputi:

- ❖ **Pemodelan dunia:** Pembelajaran hanya di lingkungan mainan dapat menyebabkan kecerdasan mainan. Pada akhirnya, model-model tersebut mungkin gagal mempelajari dinamika dunia nyata, dan pengetahuan ini mungkin gagal ditransfer ke robot fisik nyata.
- ❖ **Kesenjangan Sim2Real:** Masih ada kesenjangan di mana kinerja robot yang dilatih dalam simulasi mungkin tidak sepenuhnya sesuai dengan dunia nyata. Hal ini disebabkan oleh fisika, dinamika, dan faktor eksternal yang tidak sepenuhnya ditangkap dalam lingkungan virtual tetapi ada di lingkungan fisik.

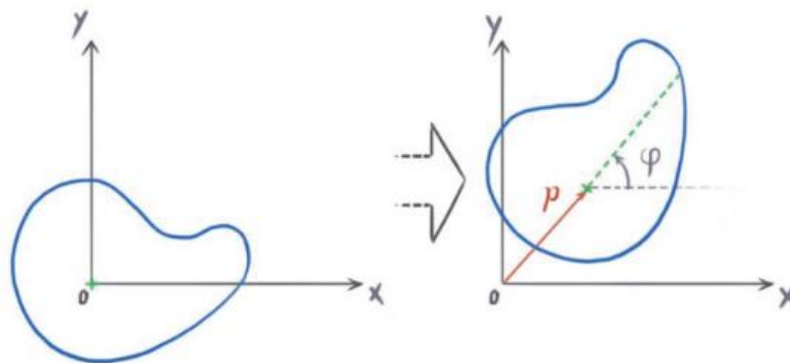
Meskipun ada banyak manfaat dari data simulasi, keterbatasan utamanya adalah adanya perbedaan antara dunia nyata dan dunia simulasi. Kombinasi simulasi dan data dunia nyata untuk pelatihan dan strategi adaptasi/randomisasi domain yang cermat seringkali diperlukan untuk menutup kesenjangan ini. Nantinya dalam bab ini, kita akan membahas strategi-strategi ini lebih lanjut.

5.2 KOMPONEN SIMULATOR ROBOT

Bagian ini membahas simulator robot yang umum digunakan—cara kerjanya serta manfaat dan kekurangannya. Mesin fisika benda kaku mensimulasikan berbagai interaksi,

seperti tabrakan dan gaya kontak. Benda kaku didefinisikan sebagai benda padat yang tidak mengubah bentuknya saat berinteraksi dengan lingkungan. Benda ini memiliki massa, lokasi, kecepatan, volume, dan bentuk. Hal ini memungkinkan benda tersebut berputar di sekitar pusat massanya, seperti yang ditunjukkan pada Gambar 5.2, yang mewakili posisi rata-rata massa benda. Ketika benda kaku berputar, ia memiliki tiga sifat sudut: kecepatan sudut (seberapa cepat ia berputar), torsi (gaya yang menyebabkannya berputar), dan momen inersia (resistensi terhadap perubahan rotasi).

Ketika Anda menerapkan gaya pada suatu titik pada benda kaku, torsi tercipta, menyebabkan benda tersebut berputar. Perhitungan torsi sederhana dalam dua dimensi dan membutuhkan kuaternion dan matriks 3x3 dalam tiga dimensi. Integral dapat membantu menemukan pusat massa dan momen inersia karena objek tersebut kontinu (tidak terbuat dari partikel terpisah).



Gambar 5.2. Posisi dan rotasi benda kaku pada waktu tertentu didefinisikan sebagai offset dari keadaan awal.

Simulasi benda lunak, di sisi lain, berkaitan dengan objek yang dapat berubah bentuk ketika gaya diterapkan, yang berarti bahwa benda lunak tersebut berubah bentuk. Seringkali, pendekatan simulasi yang lebih canggih diperlukan untuk memperhitungkan gaya dan deformasi internal. Pembengkokan, peregangan, dan kompresi dimodelkan menggunakan teknik seperti model pegas-massa atau pendekatan elemen hingga, sehingga simulasi benda lunak membutuhkan biaya komputasi yang tinggi tetapi diperlukan untuk simulator robot.

Sebagian besar simulator robot seringkali mencakup fitur-fitur berikut:

1. **Mesin fisika:** Mesin fisika digunakan untuk mensimulasikan dinamika fisik robot, termasuk dinamika benda kaku, pemodelan benda lunak, dan dinamika fluida. Dinamika benda kaku seringkali berfokus pada deteksi dan pemodelan tabrakan, gesekan, dan dinamika interaktif lainnya. Pemodelan benda lunak berfokus pada perilaku objek yang dapat berubah bentuk seperti jaringan, kain, atau material fleksibel. Dinamika fluida digunakan untuk mensimulasikan perilaku cairan dan gas, yang sangat penting dalam situasi yang melibatkan aerodinamika (drone, pesawat terbang, bagian yang bergerak) dan robot bawah air.

- a. Mensimulasikan fisika mencakup penyelesaian persamaan yang memodelkan gerakan benda kaku dan gaya lain yang mungkin bekerja pada badan robot.
 - b. Untuk simulasi benda lunak, metode elemen hingga atau metode pegas-massa sering digunakan untuk memahami gaya internal dan untuk memodelkan bagaimana deformasi terjadi pada material.
 - c. Dinamika fluida menggunakan berbagai metode dinamika fluida komputasional (CFD), yang menyelesaikan persamaan Navier-Stokes di balik layar untuk mensimulasikan aliran cairan dan gas.
2. **Pemodelan robot:** Simulator menggunakan format standar seperti URDF (*Unified Robot Description Format*) untuk mendeskripsikan dan memuat kinematika robot. Hal ini biasanya dilakukan dengan mengurai file berbasis XML yang mendeskripsikan sendi (seperti berat, dimensi tautan, dan pusat gravitasinya) dan transformasi antar sendi. Biasanya juga mencakup dukungan untuk sensor seperti kamera dan LiDAR, dan memetakan intrinsik dan ekstrinsik kamera (transformasi kamera terhadap robot). Setelah Anda memiliki spesifikasi lengkap robot dan sensornya, Anda dapat membuat gambar dari sudut pandang sensor dan mensimulasikan gerakan dengan menerapkan gaya atau menentukan posisi untuk efektor ujung dan kemudian menghitung posisi semua sendi menggunakan kinematika maju atau invers.
 3. **Pemodelan lingkungan:** Simulator biasanya memiliki pustaka yang memungkinkan Anda untuk mengkonfigurasi adegan dan lingkungan. Menambahkan objek, mengubah lokasinya, dan menambahkan medan dan permukaan, misalnya. Selain objek dan permukaan, simulator juga memungkinkan Anda untuk mensimulasikan cuaca (misalnya, sinar matahari), pencahayaan (dalam/luar ruangan) dan bayangan, serta kondisi lain, seperti kelembaban/suhu, dan sebagainya. Selain objek statis, simulator juga dapat menghasilkan agen dinamis seperti manusia dengan menggunakan model humanoid untuk meniru gerakan manusia yang realistis. Setelah pengguna mengkonfigurasi lingkungan dan pengaturan, mesin fisika kemudian memodelkannya, dan pustaka grafis merenderkannya di layar.
 4. **Berinteraksi dengan pengguna:** Untuk berinteraksi dengan pengguna, mesin simulasi menyediakan API yang memungkinkan pengguna untuk menulis kode untuk mengontrol lingkungan dan robot serta untuk menjalankan model/kebijakan tertentu atau tujuan tingkat tinggi tertentu. Selain API, simulator juga memiliki GUI agar pengguna dapat berinteraksi dengan simulator. Sebagian besar simulator memiliki algoritma perencanaan lintasan bawaan, seperti A*, Dijkstra, atau kontrol prediktif model (MPC) untuk perencanaan jalur. Kami membahas kontrol prediktif model secara detail di bab ini.
 5. **Visualisasi dan rendering:** Adegan itu sendiri dirender menggunakan pustaka grafis seperti OpenGL sehingga pergerakan dan interaksi robot dapat dilihat di layar secara real-time. Sebagian besar simulator juga terintegrasi dengan kerangka kerja middleware seperti *Robot Operating System* (ROS). ROS adalah kerangka kerja terdistribusi untuk mempublikasikan dan berlangganan informasi, memungkinkan

sensor/sambungan/program dalam robot untuk beroperasi pada frekuensi yang berbeda (seperti ketika kamera memperbarui pada 30fps, tetapi IMU memperbarui pada 100 Hz) dan tetap berkomunikasi satu sama lain. Banyak simulator memungkinkan Anda untuk mempublikasikan informasi keluar dan masuk ke simulator sebagai pesan ROS atau protokol komunikasi umum lainnya.

Setelah Anda memahami komponen-komponen kunci dari simulator robot, mari kita lihat bagaimana komponen-komponen tersebut berinteraksi untuk meniru perilaku objek di dunia nyata:

1. Dimulai dengan model matematika yang merepresentasikan keadaan sistem saat ini, seperti posisi dan kecepatan objek.
2. Mesin menggunakan persamaan untuk memprediksi bagaimana keadaan ini akan berubah seiring waktu tergantung pada gaya, bentuk objek, dan pergerakan.
3. Mesin menggunakan metode numerik untuk menyelesaikan persamaan-persamaan ini. Runge-Kutta adalah teknik yang banyak digunakan untuk memperkirakan posisi dan kecepatan yang bervariasi langkah demi langkah.
4. Ketika objek bertabrakan, mesin mengidentifikasi tabrakan dan menyesuaikan kecepatan dan arah objek untuk mencerminkan dampak dengan benar.
5. Ada banyak tugas optimasi kinerja dan keandalan yang terlibat, seperti menampilkan objek di layar, mengelola input pengguna, dan memastikan bahwa semuanya berjalan secara real-time.

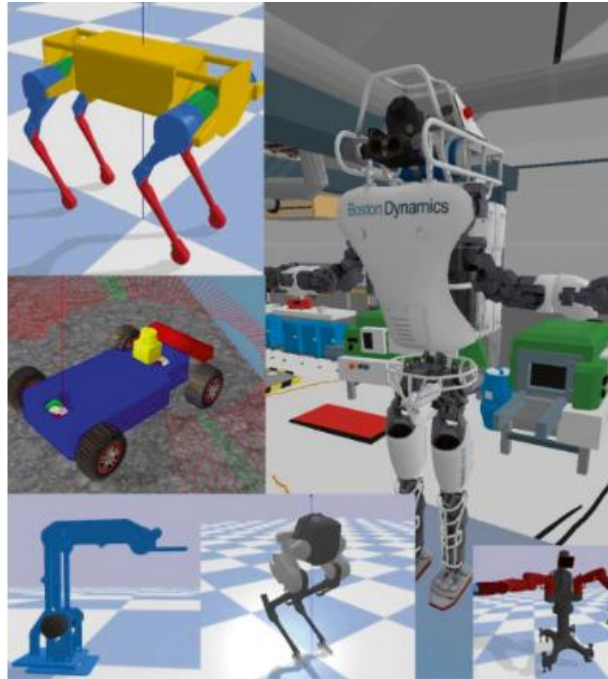
Sekarang mari kita lihat beberapa simulator populer yang digunakan oleh peneliti dan pengembang robotika serta properti spesifiknya.

Modul PyBullet

Modul Python PyBullet menggunakan mesin fisika Bullet untuk mensimulasikan interaksi fisik dan gerakan robot. Aplikasi penting PyBullet adalah sistem QT-Opt, sebuah algoritma yang akan kita bahas secara detail di bagian pembelajaran bab ini, di mana Bullet membantu meniru lingkungan menggenggam untuk melatih lengan robot pada sejumlah besar tugas menggenggam. PyBullet memungkinkan Anda untuk mengimpor model robot dalam berbagai format:

1. SDF format (*Simulation Description Format*)
2. *Unified Robotics Description Format* (URDF)
3. MJCF

PyBullet mendukung dinamika kontak, tabrakan, gesekan, dan benda kaku untuk model robot yang sudah ada atau robot yang diimpor oleh pengguna. PyBullet umumnya digunakan untuk mensimulasikan berbagai tugas robot, seperti tugas manipulasi robot. Gambar beberapa robot yang didukung ditunjukkan pada Gambar 5.3. Untuk perencanaan gerak tingkat lanjut, Anda dapat mengintegrasikan PyBullet dengan pustaka eksternal, seperti *Open Motion Planning Library* (OMPL), yang banyak digunakan dalam robotika. Terdapat tutorial tentang memulai dengan PyBullet di akhir bab ini.



Gambar 5.3. Berbagai macam robot yang didukung oleh simulasi PyBullet. Ini termasuk robot berkaki seperti Atlas dari Boston Dynamics, robot berkaki empat, robot beroda, lengan robot, dan manipulator.

MuJoCo

MuJoCo (*Multi-Joint Dynamics with Contact*) adalah platform yang dimiliki dan bersifat open-source oleh DeepMind yang dikembangkan untuk robotika. MuJoCo berfokus pada aplikasi robotika tradisional, termasuk lengan multi-sambungan, menggenggam, dan berjalan bipedal. Platform ini telah mendapatkan banyak popularitas karena kecepatannya, presisi dalam fisika yang sangat akurat, dan desain robotika yang ramah pengguna. Platform ini mendukung dinamika gesekan, kontak, dan benda kaku, serta dapat mengakomodasi benda yang lebih fleksibel. Sebagai contoh, tangan bayangan dari Open AI, yang merupakan salah satu contoh pertama tangan robot yang dapat memanipulasi lingkungannya dengan ketangkasan tinggi, dibangun menggunakan mesin fisika MuJoCo. Kami merekomendasikan untuk mempelajari dokumentasi MuJoCo untuk memulai penggunaannya.

Gazebo

Platform simulasi robotika populer lainnya adalah Gazebo. Platform ini sering digunakan untuk manipulasi bergerak, penggengaman robot, dan mobilitas off-road, serta untuk aplikasi robotika yang lebih konvensional. Kami merekomendasikan untuk mempelajari dokumentasi Gazebo untuk memulai penggunaannya. Secara khusus, berikut adalah tutorial yang kami rekomendasikan untuk menggunakan lingkungan Gazebo sederhana.

Antarmuka middleware Gazebo dengan *Robot Operating System* (ROS) memungkinkan penggunaannya dalam kombinasi dengan tumpukan robotika yang ada yang digunakan dalam robot industri. Sebaliknya, PyBullet dan MuJoCo menyediakan integrasi yang lebih baik dengan kerangka kerja DL dan RL, serta lingkungan gym, sehingga lebih cocok untuk kasus penggunaan ini. Simulator yang Anda pilih dapat ditentukan oleh berbagai kriteria, termasuk fitur yang

Anda butuhkan, kecepatan, dan aplikasi. Perbandingan yang lebih komprehensif dari berbagai simulator ditunjukkan pada Gambar 5.4.

Simulator	RGBD + LiDAR	Sensor Gaya	Aktuator Linear + Kabel	Imulti-Body Import	Kontak Bodi Lunak	Simulasi DEM	Mekanika Mechanics	Mode Tanpa Tampilan	Dukungan ROS	HITL	Teleoperasi	Perenderaan Realistik	Kinematika Terbalik
AirSim	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓, unreal	✗
CARLA	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓, unreal	✗
CoppeliaSim	✓	✓	Hanya linear	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓
Gazebo	✓	✓	Hanya linear	✓	✗	Melalui Fluidix	Melalui Fluidix	✓	✓	✓	✓	✗	✓
MuJoCo	✓	✓	✓	✓	✓	✓	Terbatas	✓	✗	HAPTIX only	Hanya HAPTIX	✗	✗
PyBullet	✓	✓	Hanya linear	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓
SOFA	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓, Unity	✗
UWSim	Hanya RGBD	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓, kustom	✗
Chrono	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓, huring	✓
Webots	✓	✓	linear	✓	✗	✗	Terbatas	✓	✓	✗	✓	✗	✗

Gambar 5.4. Perbandingan platform simulasi robotika berdasarkan berbagai fitur, seperti dukungan sensor (RGBD, LiDAR, sensor gaya), jenis aktuator, impor multi-body, dan kontak soft-body. Tabel ini menunjukkan kekuatan dan keterbatasan setiap simulator, termasuk PyBullet, MuJoCo, Gazebo, dan lainnya.

Secara keseluruhan, Gazebo menawarkan integrasi ROS yang luas dan kemampuan untuk mensimulasikan lingkungan yang berbeda. PyBullet terintegrasi dengan baik dengan kerangka kerja pembelajaran mesin dan MuJoCo mengungguli kecepatan dan akurasi simulasi.

5.3 KONSEP DALAM SIM2REAL

Sim2Real mengacu pada strategi yang menggunakan simulasi untuk mempelajari kebijakan untuk bertindak di dunia nyata. Mereka mungkin juga menggunakan data dunia nyata atau tidak. Pada bagian ini, kita akan melihat beberapa strategi yang paling sering digunakan dalam Sim2Real. Banyak prinsip ini dapat dibahas dalam konteks pembelajaran penguatan (*reinforcement learning*/RL), tetapi prinsip-prinsip ini juga dapat diterapkan pada tugas-tugas yang lebih umum, seperti identifikasi objek, kontrol dunia nyata, dan sebagainya. Secara khusus dalam konteks RL, tujuannya adalah untuk melatih agen RL dalam lingkungan simulasi dan menerapkannya/mengujinya di dunia nyata menggunakan teknik seperti adaptasi domain atau pengacakan domain.

Adaptasi Domain

Prinsip utama di balik adaptasi domain (DA) adalah menggunakan data dari satu area (sumber) untuk meningkatkan kinerja model di area lain (target) ketika data yang tersedia lebih sedikit. Untuk mencapai hal ini, Anda ingin membuat data dari kedua sumber tampak lebih mirip. Untuk memahami apa arti domain sumber dan domain target, pertimbangkan proses keputusan Markov (*Markov decision process*/MDP). MDP adalah model pengambilan

keputusan di mana agen membuat keputusan dalam serangkaian langkah. Pada setiap langkah, agen melakukan tindakan yang mengubah skenario saat ini (keadaan) ke skenario baru (keadaan berikutnya) dengan probabilitas yang bervariasi. Agen diberi imbalan atas tindakan yang dilakukannya. RL berupaya mengidentifikasi teknik (kebijakan) paling efektif yang memaksimalkan total imbalan dari waktu ke waktu. MDP direpresentasikan dengan keadaan (S), tindakan (A), transisi (P) yang terjadi dengan probabilitas tertentu, dan imbalan (R):

$$D \equiv (S, \mathcal{A}, P, \mathcal{R})$$

Persamaan 5.1

Proses Keputusan Markov bersifat Markovian, yang berarti bahwa keadaan sepenuhnya dapat diamati dan informasi tentang bagaimana keadaan tersebut tercapai tidak relevan dalam memutuskan aspek apa pun di masa depan. Domain sumber adalah lingkungan yang dapat Anda akses sepenuhnya (dalam hal ini, simulator), dan domain target adalah dunia fisik yang sebenarnya. Domain sumber dalam tugas RL dan domain target dirancang sebagai MDP khusus. Keadaan mereka dapat sangat berbeda, sementara tindakan, transisi, dan imbalan memiliki kesamaan, karena Anda ingin tindakan dari simulasi diterjemahkan ke dunia nyata.

Terdapat beberapa teknik adaptasi domain umum, yang pada akhirnya bertujuan untuk membuat data sumber menyerupai data target:

- Salah satu metode menggunakan teknik statistik untuk menilai dan mengoreksi perbedaan antara fitur dari beberapa sumber. Hal ini dapat dilakukan dengan menyelaraskan rata-rata dan varians fitur atau meminimalkan jarak antara distribusi domain dalam ruang berdimensi tinggi.
- Metode lain melatih model untuk mengenali dari sumber mana fitur tersebut berasal sebelum menyesuaikannya agar lebih mirip. Di sini, Anda dapat menggunakan pengklasifikasi domain untuk membedakan antara fitur dari setiap domain, yang dapat dilatih secara adversarial untuk mempersulit klasifikasi.
- Terakhir, sebuah teknik akan mengidentifikasi karakteristik bersama dengan mempelajari cara mereproduksi data asli menggunakan fitur bersama ini, memastikan bahwa informasi penting ditangkap dari kedua sumber. Struktur encoder-decoder dapat digunakan, di mana encoder memetakan data dari kedua domain ke ruang laten dan decoder digunakan untuk merekonstruksi data input berdasarkan representasi bersama ini.

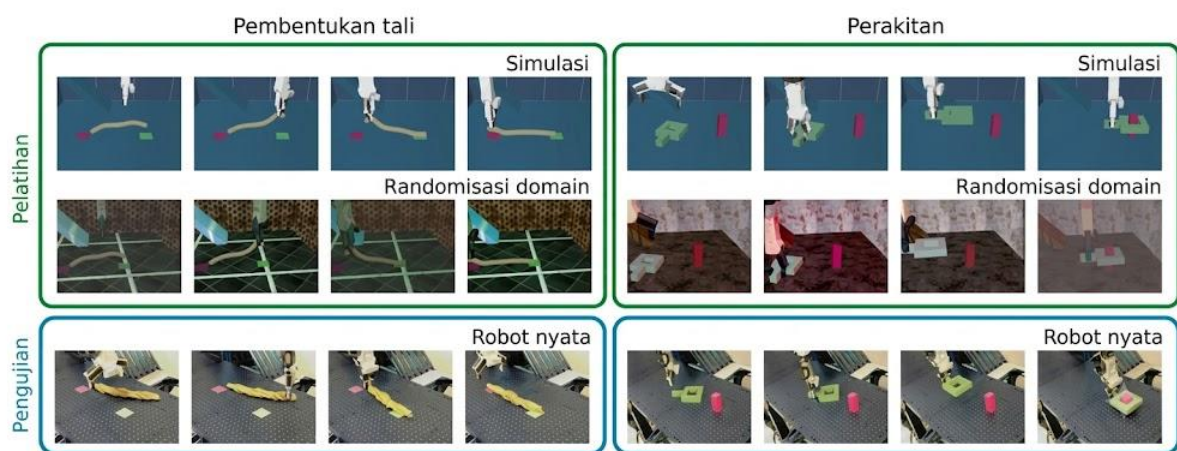
Kami merekomendasikan sebagai sumber untuk mempelajari lebih lanjut.

Pengacakan Domain

Pengacakan domain adalah strategi untuk meningkatkan ketahanan model dengan mengekspos model ke berbagai situasi simulasi. Tujuan pengacakan simulasi adalah untuk membuat model responsif terhadap fluktuasi dan situasi dunia nyata yang mungkin muncul.

Dengan cara ini, model belajar untuk menangani berbagai keadaan, sehingga lebih berhasil ketika diterapkan pada data dunia nyata.

Pada Gambar 5.5, diagram menggambarkan pengacakan domain dengan menunjukkan bagaimana data simulasi diacak untuk mereplikasi distribusi data dunia nyata. Ini menunjukkan dua tugas—Pembentukan Tali dan Perakitan—yang dilatih dalam lingkungan simulasi standar dan acak. Pengacakan domain menambahkan variabilitas visual dan fisik untuk membantu model melakukan generalisasi yang lebih baik pada pengujian robot dunia nyata. Sebaliknya, adaptasi domain menyelaraskan fitur dari domain sumber (simulasi) dan target (nyata) untuk menciptakan ruang fitur bersama untuk pembelajaran penguatan.



Gambar 5.5. Contoh pengacakan domain: Kebijakan manipulasi visuomotor dilatih dalam simulasi (baris atas) dengan pengacakan domain menggunakan tekstur, pencahayaan, warna objek, dan pengaturan kamera yang bervariasi (baris tengah). Kebijakan ini ditransfer langsung ke tugas dunia nyata (baris bawah).

Pengacakan domain memiliki tiga pendekatan pelatihan utama:

- Pengacakan statis, yang memperkenalkan perubahan acak pada lingkungan pelatihan di awal dan mempertahankan variasi ini sepanjang proses pelatihan. Hal ini sering dilakukan melalui variasi kondisi cahaya, tekstur, dan penempatan. Perubahan ini kemudian ditetapkan sepanjang proses pelatihan model. Strategi ini mudah dan cepat, tetapi mungkin tidak begitu berguna dalam situasi dunia nyata karena mungkin tidak menangkap seluruh rentang variasi yang dihadapi oleh robot.
- Pengacakan adaptif memodifikasi modifikasi acak ini selama pelatihan berdasarkan kemajuan pembelajaran model. Misalnya, lingkungan mungkin memiliki variasi yang lebih kecil di awal pelatihan tetapi seiring peningkatan model, perubahan yang lebih kompleks dapat diperkenalkan untuk memastikan bahwa model tersebut belajar. Pendekatan ini dapat menghasilkan peningkatan kinerja tetapi membutuhkan data dunia nyata yang substansial untuk memandu penyesuaian.
- Pengacakan adversarial menambahkan gangguan adversarial selama pelatihan dengan menggunakan model yang berbeda untuk menghasilkan skenario yang menantang,

sehingga membuat lingkungan pelatihan lebih sulit bagi model "utama". Hal ini membantu model menjadi lebih kuat, tetapi perlu dikalibrasi dengan hati-hati untuk mencegah tugas menjadi terlalu sulit dan memperburuk pembelajaran.

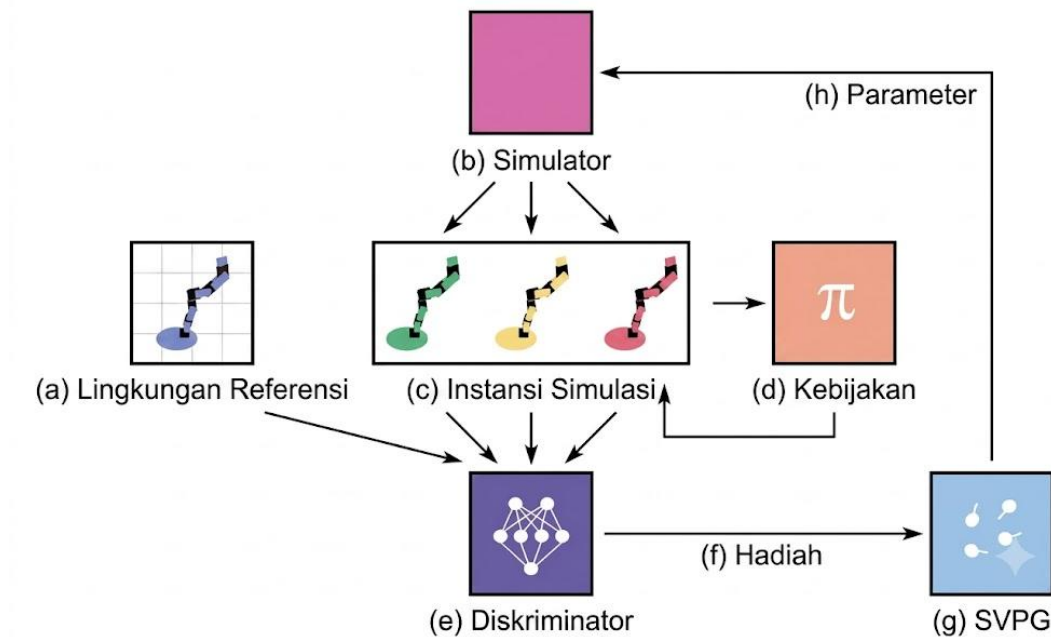
Kami merekomendasikan sebagai sumber tempat kami mengumpulkan ide untuk bagian ini.

Cara lain untuk mengekspos model ke berbagai skenario selama pelatihan adalah melalui Randomisasi Domain Seragam (*Uniform Domain Randomization/UDR*), yaitu proses memodifikasi properti lingkungan pelatihan secara acak di setiap langkah. Variasi ini dapat mencakup aspek pencahayaan, tekstur, dan penempatan, sehingga model tidak mengalami overfitting pada lingkungan tertentu. Ini menyiratkan bahwa model terus-menerus terpapar skenario baru, membuatnya lebih adaptif dan kuat dalam menghadapi variasi antara konteks pelatihan dan dunia nyata. Namun, pengambilan sampel seragam mengasumsikan bahwa semua perubahan dalam lingkungan pelatihan sama pentingnya, yang tidak benar karena skenario tertentu lebih sulit dipelajari daripada yang lain dan membutuhkan lebih banyak fokus selama proses pelatihan. Randomisasi Domain Terarah (*Guided Domain Randomization/GDR*) diusulkan untuk menyelesaikan beberapa masalah ini dengan teknik pengambilan sampel yang lebih terarah. Bagian ini membahas dua metode GDR: Randomisasi Domain Aktif dan Randomisasi Domain Otomatis.

Randomisasi Domain Aktif

Alih-alih sekadar mengacak pengaturan lingkungan secara seragam, Randomisasi Domain Aktif (ADR), yang dipresentasikan dalam Mehta, berfokus pada identifikasi dan pelatihan model dalam pengaturan paling sulit yang dihadapinya. Teknik ini melibatkan pembuatan situasi simulasi yang beragam dan kompleks, mengevaluasi mana yang paling sulit bagi model, dan kemudian memfokuskan pelatihan pada skenario-skenario sulit ini.

Simulator secara khusus digunakan untuk menghasilkan berbagai lingkungan simulasi dengan parameter yang bervariasi. Kebijakan model kemudian diuji dalam lingkungan simulasi ini. Diskriminator menilai tingkat kesulitan lingkungan ini dengan membandingkannya dengan lingkungan referensi; ia memberikan imbalan yang sesuai. Lingkungan referensi dapat dipilih berdasarkan kondisi tipikal yang diharapkan akan dihadapi oleh model. *Stein Variational Policy Gradient* (SVPG) kemudian menggunakan imbalan ini untuk memperbarui parameter model dengan mengarahkan model untuk menghabiskan lebih banyak waktu pelatihan dalam skenario yang sulit. Hal ini dicapai dengan menyesuaikan parameter kebijakan secara iteratif untuk memaksimalkan imbalan yang diharapkan. Langkah-langkah utama bagaimana randomisasi domain aktif bekerja ditunjukkan pada Gambar 5.6.



Gambar 5.6. Kerangka kerja ADR menggunakan simulator untuk menghasilkan lingkungan acak untuk melatih kebijakan agen. Diskriminator membedakan antara lingkungan referensi dan lingkungan acak, menghasilkan sinyal hadiah yang digunakan untuk melatih partikel SVPG. Partikel-partikel ini mengeksplorasi parameter lingkungan yang sulit untuk meningkatkan ketahanan kebijakan.

Fokus dari randomisasi domain aktif adalah untuk secara dinamis menyesuaikan proses pelatihan berdasarkan kesulitan. Di sisi lain, Randomisasi Domain Otomatis memberikan pendekatan yang lebih sistematis untuk randomisasi domain.

Randomisasi Domain Otomatis

Teknik GDR yang relatif sederhana, Randomisasi Domain Otomatis, diperkenalkan oleh Open AI pada tahun 2018, dan berhasil membantu tangan robot sungguhan memecahkan Kubus Rubik. Randomisasi Domain Otomatis dimulai dengan melatih kebijakan dalam lingkungan dengan parameter default. Ia melalui beberapa episode pelatihan di mana ia secara acak memutuskan apakah akan memberikan batasan yang lebih rendah atau lebih tinggi pada parameter untuk menciptakan variabilitas. Kebijakan tersebut berinteraksi dengan lingkungan, dan kinerjanya dievaluasi. Jika kebijakan tersebut berkinerja baik secara konsisten, batasan parameter secara bertahap ditingkatkan, membuat lingkungan menjadi lebih sulit. Jika kinerja menurun, batasan diperketat untuk membuat lingkungan lebih mudah. Proses iteratif ini mengajarkan kebijakan tersebut cara menangani berbagai skenario berdasarkan umpan balik langsung. Untuk detail lebih lanjut tentang desain algoritma, kami sarankan untuk membaca.

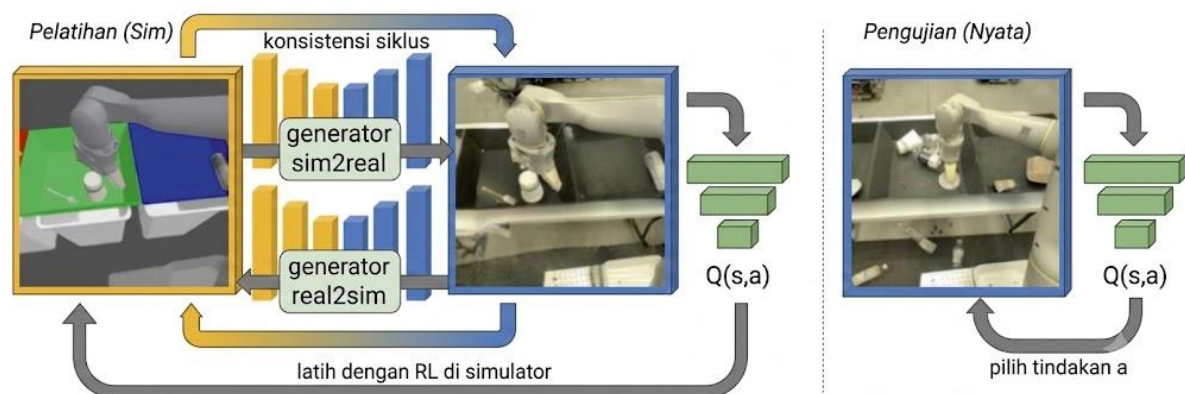
5.4 MENUTUP KESENJANGAN SIM2REAL UNTUK RL

RL-CycleGAN adalah metode penting yang bertujuan untuk menutup kesenjangan Sim2Real untuk RL dan telah diuji untuk tugas-tugas pengambilan benda berbasis visi robot. Secara garis besar, RL-CycleGAN menggabungkan RL dengan CycleGAN (yang akan kami

jelaskan lebih detail nanti) untuk mentransfer pengetahuan dari lingkungan simulasi ke dunia nyata. Komponen utama RL-CycleGAN adalah:

- **Generator Sim2Real:** Komponen ini mengubah gambar dari simulator agar terlihat lebih mirip dengan gambar dunia nyata. Generator mengambil gambar dari simulator dan menerjemahkannya menjadi gambar yang memiliki karakteristik tekstur, pencahayaan, dan noise dari gambar dunia nyata.
- **Generator Real2Sim:** Untuk memastikan bahwa transformasi antara dua domain (gambar simulasi dan nyata) konsisten, gambar nyata dikonversi menjadi gambar simulasi dan sebaliknya. Ini berarti bahwa ketika sebuah gambar dikonversi dari simulasi ke nyata dan kembali ke simulasi, gambar tersebut harus sangat mirip dengan gambar simulasi aslinya.

RL belajar dengan berinteraksi dengan lingkungan dan mendapatkan semacam imbalan sebagai umpan balik. Misalnya, robot mungkin mendapatkan imbalan positif karena berhasil mengambil suatu objek dan imbalan negatif karena menjatuhkannya. Secara keseluruhan, tujuan agen adalah untuk memaksimalkan imbalan kumulatifnya dengan mempelajari tindakan terbaik yang harus diambil dalam berbagai keadaan. Dalam kasus CycleGAN, agen RL dilatih menggunakan gambar yang dikonversi oleh generator Sim2Real (lihat Gambar 5.7). Idealnya, gambar-gambar ini terlihat senyata mungkin, sehingga agen belajar untuk secara efektif melakukan tugas di luar simulator. Setelah agen RL terlatih dengan baik di lingkungan simulasi dengan gambar Sim2Real, agen tersebut dikerahkan dan diuji di dunia nyata.



Gambar 5.7. RL-CycleGAN melatih CycleGAN untuk memetakan gambar simulator ke gambar realistis dan sebaliknya. Model RL ditransfer ke robot nyata untuk pengujian dan eksekusi tugas.

CycleGAN

Untuk sepenuhnya memahami bagaimana CycleGAN dapat digunakan untuk tugas pengambilan gambar robot berbasis RL, akan sangat membantu untuk memahami arsitektur dan kerugian utama. CycleGAN sendiri merupakan jenis *Generative Adversarial Network* (GAN) yang digunakan untuk mempelajari pemetaan antara dua domain gambar (gambar simulasi dan gambar nyata) tanpa contoh berpasangan. Terdiri dari dua generator—generator Sim2Real

dan generator Real2Sim. Generator Sim2Real memetakan gambar dari domain simulasi ke domain nyata. Generator Real2Sim memetakan gambar dari domain nyata kembali ke domain simulasi. Terdapat diskriminator untuk setiap jenis domain gambar: satu diskriminator akan membedakan antara gambar asli dari domain dunia nyata dan gambar palsu yang dihasilkan oleh generator Sim2Real dari gambar simulasi, sementara diskriminator lainnya membedakan gambar simulasi dari domain simulasi dan gambar palsu yang dihasilkan oleh generator Real2Sim dari gambar asli.

Tiga kerugian utama digunakan dalam CycleGAN—kerugian adversarial, kerugian konsistensi siklus, dan kerugian total akhir. Kerugian adversarial digunakan untuk membuat gambar yang dihasilkan tampak serealistik mungkin. Terdapat dua diskriminator: satu untuk gambar nyata dan satu untuk gambar simulasi. Setiap generator (Sim2real dan Real2Sim) mencoba untuk mengelabui diskriminatornya masing-masing dengan menghasilkan gambar yang tampak nyata. Diskriminator, pada gilirannya, mencoba untuk mengidentifikasi gambar nyata dan palsu dengan benar. Generator bertujuan untuk meminimalkan kerugian adversarial dengan menghasilkan gambar yang sangat realistis, sementara diskriminator bertujuan untuk memaksimalkannya dengan mengidentifikasi gambar palsu secara akurat. Istilah pertama dalam setiap fungsi kerugian mengukur seberapa baik diskriminator mengidentifikasi gambar nyata dan istilah kedua mengukur seberapa baik ia mengidentifikasi gambar palsu yang dihasilkan oleh generator.

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim \mathcal{P}_{data}(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim \mathcal{P}_{data}(x)} [\log (1 - D_Y(G(x)))]$$

$$L_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim \mathcal{P}_{data}(x)} [\log D_X(x)] + \mathbb{E}_{y \sim \mathcal{P}_{data}(y)} [\log (1 - D_X(F(y)))]$$

Persamaan 5.2

Kerugian konsistensi siklus memastikan bahwa transformasi antara gambar simulasi dan gambar nyata konsisten dan dapat dibalik. Fungsi kerugian konsistensi siklus memastikan bahwa ketika sebuah gambar ditransformasikan dari satu domain ke domain lain dan kembali lagi, gambar tersebut tetap mirip dengan aslinya. Hal ini dilakukan dengan memberikan penalti pada model jika gambar yang telah ditransformasikan dua kali berbeda secara signifikan dari aslinya. Secara keseluruhan, ini mendorong model untuk mempertahankan fitur-fitur penting selama transformasi.

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim \mathcal{P}_{data}(x)} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim \mathcal{P}_{data}(y)} [\|G(F(y)) - y\|_1]$$

Persamaan 5.3

Fungsi kerugian total akhir menggabungkan kerugian adversarial untuk generator dan diskriminator dengan kerugian konsistensi siklus. Ini memastikan bahwa gambar yang dihasilkan realistis dan transformasi antar domain konsisten.

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F)$$

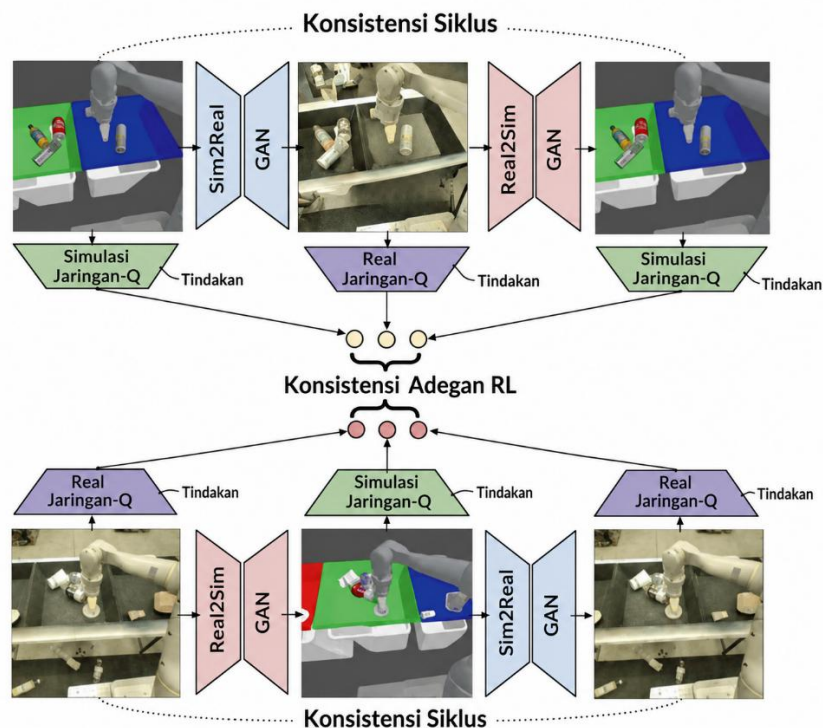
Persamaan 5.4

RL-CycleGAN

Komponen utama RL-CycleGAN meliputi:

- GAN, yang mentransformasikan gambar antara domain simulasi dan nyata. Ini termasuk Sim2Real GAN, yang mengubah gambar simulasi agar terlihat realistis, dan Real2Sim GAN, yang mengubah gambar nyata kembali menjadi simulasi.
- Konsistensi siklus memastikan bahwa ketika sebuah gambar ditransformasikan dari simulasi ke nyata dan kembali ke simulasi (atau sebaliknya), gambar tersebut mempertahankan fitur-fitur pentingnya.
- Dua jaringan Q dilatih: satu pada gambar simulasi untuk membantu robot berinteraksi dengan objek di lingkungan simulasi dan satu pada gambar nyata untuk membantu robot berinteraksi dengan objek di dunia nyata.
- Terdapat komponen konsistensi adegan RL yang memastikan bahwa tindakan robot konsisten di seluruh lingkungan simulasi dan nyata. Misalnya, jika robot mempelajari cara menggenggam objek dalam simulasi, robot tersebut seharusnya dapat menerapkan teknik menggenggam yang sama di dunia nyata. Selain itu, hal ini memastikan bahwa nilai q untuk adegan serupa (baik di dunia nyata maupun simulasi) adalah serupa. Hal ini dilakukan karena kerugian konsistensi adegan RL memberikan penalti pada perbedaan nilai q untuk adegan yang sesuai di berbagai transformasi (misalnya, simulasi ke nyata ke simulasi, nyata ke simulasi ke nyata).

Gambar 5.8 menunjukkan bagaimana komponen-komponen ini saling terhubung.



Gambar 5.8. RL-CycleGAN menggabungkan CycleGAN dengan konsistensi adegan RL.

Pelatihan RL-CycleGAN menggabungkan beberapa tujuan:

- **Generasi Gambar Realistis (L_GAN):** Memastikan bahwa gambar yang dihasilkan oleh GAN terlihat realistis.
- **Konsistensi Siklus (L_cycle):** Memastikan bahwa gambar yang ditransformasikan dari simulasi ke nyata dan kembali ke simulasi (atau nyata ke simulasi dan kembali ke nyata) tetap konsisten dengan gambar aslinya.
- **Konsistensi Adegan RL (L_RL-scene):** Memastikan bahwa adegan serupa memiliki nilai q yang serupa.
- **Kerugian RL Tradisional (L_RL):** Melatih jaringan q menggunakan kerugian RL standar; model mempelajari nilai q berdasarkan tindakan dan imbalan.

Perbedaan antara gaya dan semantik penting dalam konteks ini. Gaya mengacu pada aspek visual seperti pencahayaan dan tekstur, yang idealnya tidak memengaruhi kinerja robot. Misalnya, perubahan pencahayaan lingkungan seharusnya tidak mengubah cara robot menggenggam suatu objek. Semantik, di sisi lain, mengacu pada fitur seperti posisi dan jenis objek. Hal ini memengaruhi cara robot melakukan tugasnya. Misalnya, mengetahui posisi dan identitas objek secara tepat sangat penting bagi robot untuk menggenggam objek tersebut. RL-CycleGAN berfokus pada pelestarian semantik sambil mentransformasikan gambar antara lingkungan simulasi dan nyata. Ini penting, karena semantik untuk tugas tersebut harus dipertahankan sementara aspek visual (gaya) dapat berubah selama transformasi.

5.5 PEMBELAJARAN DARI SIMULASI

Sekarang Anda memahami bagaimana pengacakan domain dapat meningkatkan pelatihan model dari simulasi dan menerjemahkannya secara lebih efektif ke dunia nyata, bagian ini mengeksplorasi beberapa contoh kunci tentang bagaimana simulasi telah digunakan dalam robotika. Secara khusus, robot dapat dilatih untuk menangani lingkungan yang kompleks dan dinamis menggunakan pembelajaran penguatan, tetapi biaya akuisisi data yang tinggi merupakan keterbatasan yang signifikan. Banyak metode telah dikembangkan untuk mengatasi hal ini dengan menggabungkan data dunia nyata dan data simulasi untuk pengembangan dan pelatihan model.

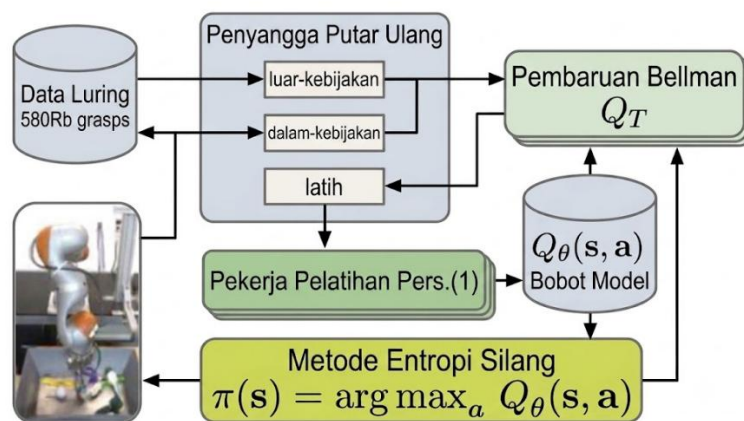
Simulasi untuk Bootstrapping RL

QT-Opt melatih fungsi Q dengan puluhan ribu genggam dunia nyata dan simulasi. Dalam hal ini, fungsi Q adalah imbalan masa depan yang diharapkan untuk mengambil tindakan tertentu dalam keadaan tertentu. Pendekatan ini dapat menggeneralisasi genggam hingga 96 persen keberhasilan genggam pada objek yang tidak terlihat, yang merupakan peningkatan substansial dibandingkan pendekatan yang ada.

QT-Opt dimulai dengan koleksi besar data offline yang terdiri dari 580.000 genggam yang direkam, yang mencakup pasangan keadaan-tindakan-imbalan. Data ini disimpan dalam buffer pemutaran ulang, di mana data tersebut dilengkapi dengan data dari interaksi robot nyata secara online. Pembaruan Bellman mengambil sampel transisi dari buffer pemutaran

ulang ini untuk menghasilkan contoh pelatihan dan menyempurnakan estimasi nilai model menggunakan persamaan Bellman.

Persamaan ini memperbarui nilai Q , yang mewakili imbalan yang diharapkan untuk setiap pasangan keadaan-tindakan, berdasarkan imbalan langsung dan masa depan. Pekerja pelatihan kemudian menggunakan contoh-contoh ini untuk memperbarui parameter fungsi Q . Untuk memilih aksi genggam terbaik, QT-Opt menggunakan Metode Entropi Silang, yang mengidentifikasi aksi yang memaksimalkan nilai estimasi model (Q). Akhirnya, robot menggunakan model terlatih untuk melakukan tugas genggam di dunia nyata. Proses ini ditunjukkan pada Gambar 5.9.



Gambar 5.9. Alur kerja untuk QT-Opt.

Selain menggunakan data nyata, komponen penting dari model ini adalah melatihnya pada beragam objek menggunakan data simulasi dari simulator Bullet Physics. Model, metode pelatihan, dan kontrol yang sama digunakan untuk pelatihan di lingkungan nyata dan simulasi. Hal ini memastikan bahwa kebijakan yang dipelajari dapat digeneralisasi dan robot dapat melakukan pengambilan objek di berbagai lingkungan. Pengaturan simulasi ditunjukkan pada Gambar 5.10.



Gambar 5.10. Pengaturan manipulasi robot QT-Opt dalam simulasi.

Melalui simulasi, mereka mampu dengan cepat menjalankan eksperimen skala besar dengan hingga 1.000 robot virtual yang beroperasi secara bersamaan. Eksperimen ini menggunakan kebijakan dunia nyata yang dipelajari dari data dunia nyata, dan menemukan bahwa pembelajaran dunia nyata lebih menantang dengan lebih banyak data yang dibutuhkan dan waktu pelatihan yang lebih lama agar kinerjanya sama efektifnya dengan simulasi. Secara keseluruhan, eksperimen ini menyoroti pentingnya menggunakan simulasi dan data dunia nyata untuk pengujian dan pelatihan model robot.

Agen Dasar dalam Simulasi

Selama beberapa tahun terakhir, arah utama dalam pembelajaran mesin adalah mengembangkan model dasar yang dilatih pada korpus data besar yang dapat digeneralisasikan ke banyak tugas. Model dasar sering digunakan dalam simulasi untuk memungkinkan robot melakukan berbagai aktivitas yang lebih luas dan lebih memahami lingkungannya. Model bahasa besar dapat menghasilkan rencana aksi, tetapi banyak robot masih membutuhkan bantuan untuk memperoleh dan memperbarui pengetahuan dalam jangka waktu yang lama.

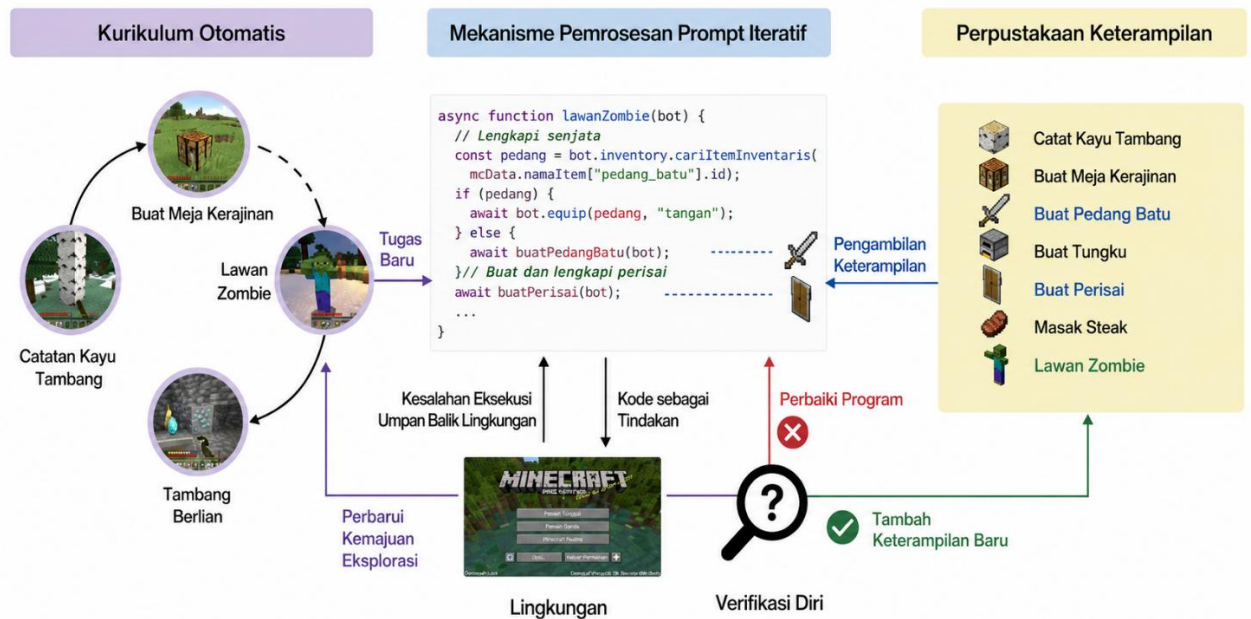
Untuk mendorong eksplorasi dan generalisasi ke berbagai kemampuan yang lebih luas dalam lingkungan simulasi Minecraft, Voyager mencoba mengatasi masalah ini dengan menggunakan LLM untuk agen seperti robot. Ini adalah contoh bagaimana pembelajaran dalam simulasi dapat membantu untuk mengeksplorasi berbagai tugas dan konteks serta mengevaluasi tugas dalam skala yang lebih besar. Voyager memiliki loop di mana, berdasarkan inventaris pemain, apa yang ada di dekatnya, kesehatan, kelaparan, dan lingkungan (bioma dan waktu siang), agen dalam simulasi menyarankan tindakan terbaik berikutnya. Namun eksplorasi, pembelajaran, dan memorinya dikelola melalui tiga komponen berikut:

- **Kurikulum otomatis:** Tugas baru untuk agen diselesaikan dihasilkan oleh GPT-4, dengan menganalisis informasi keadaan. Penalaran model menjelaskan mengapa tugas yang disarankan bermanfaat, dan dari daftar ini, tugas spesifik diantrekan untuk dieksekusi oleh pemain. Karena kurikulum atau tugas yang ditingkatkan oleh agen ditentukan oleh LLM, maka kurikulum atau tugas tersebut dihasilkan secara terprogram. Contoh tugas yang dihasilkan oleh agen ditunjukkan pada Gambar 5.11.



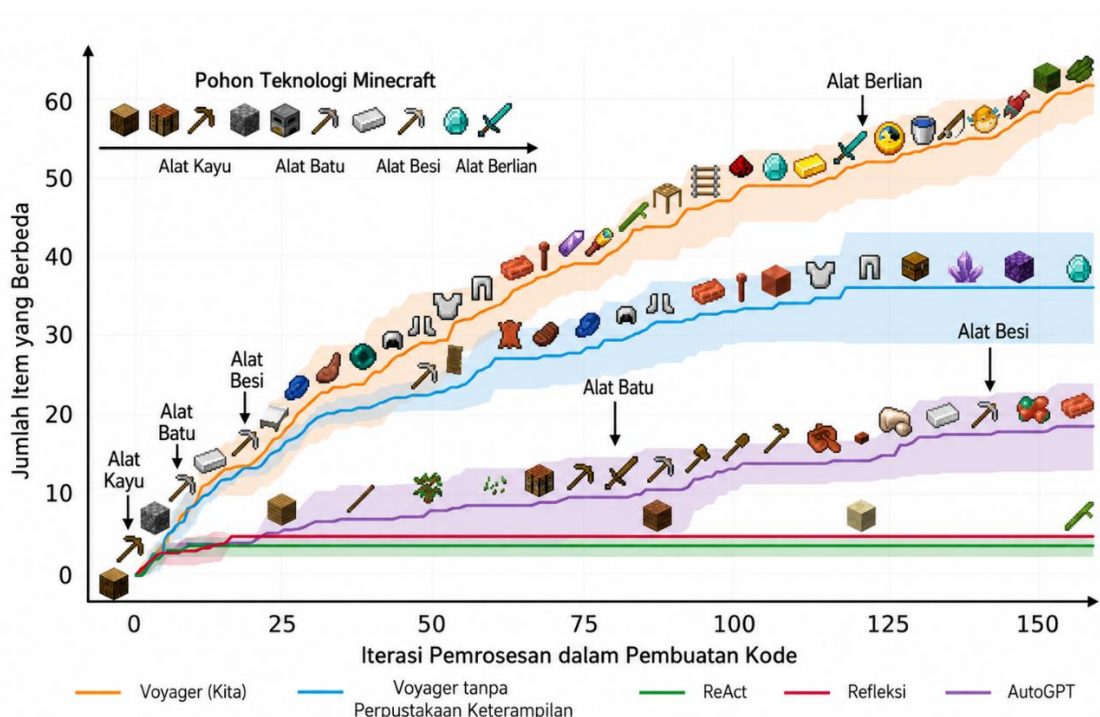
Gambar 5.11. Contoh tugas yang diusulkan oleh kurikulum otomatis.

- **Perpustakaan keterampilan:** Untuk mengumpulkan dan mengembangkan keterampilan yang dikumpulkan dalam simulasi, agen memiliki perpustakaan keterampilan. Ketika agen melakukan suatu tugas, dengan menghasilkan kode, tugas ini dan program yang dihasilkan disimpan dalam kamus yang disebut perpustakaan keterampilan. Kunci kamus adalah penyematan deskripsi keterampilan dan nilai yang sesuai adalah program itu sendiri. Keterampilan baru ditambahkan ke kamus ketika tugas yang sesuai ditemui, setelah eksekusinya. Keterampilan lama diambil melalui kueri dan diperbarui menggunakan perbaikan atau umpan balik dari putaran terakhir ketika keterampilan tersebut dipanggil.
- **Mekanisme pemberian petunjuk iteratif:** Terdapat proses iteratif untuk menyempurnakan tindakan dengan mengeksekusinya, melaporkan kesalahan, dan menyempurnakan tindakan menggunakan umpan balik dari model bahasa dan kesalahan dari simulator. Gambaran umum tentang bagaimana semua ide ini saling terkait ditunjukkan pada Gambar 5.12.



Gambar 5.12. Kurikulum otomatis, mekanisme dorongan berulang, dan perpustakaan keterampilan di Voyager.

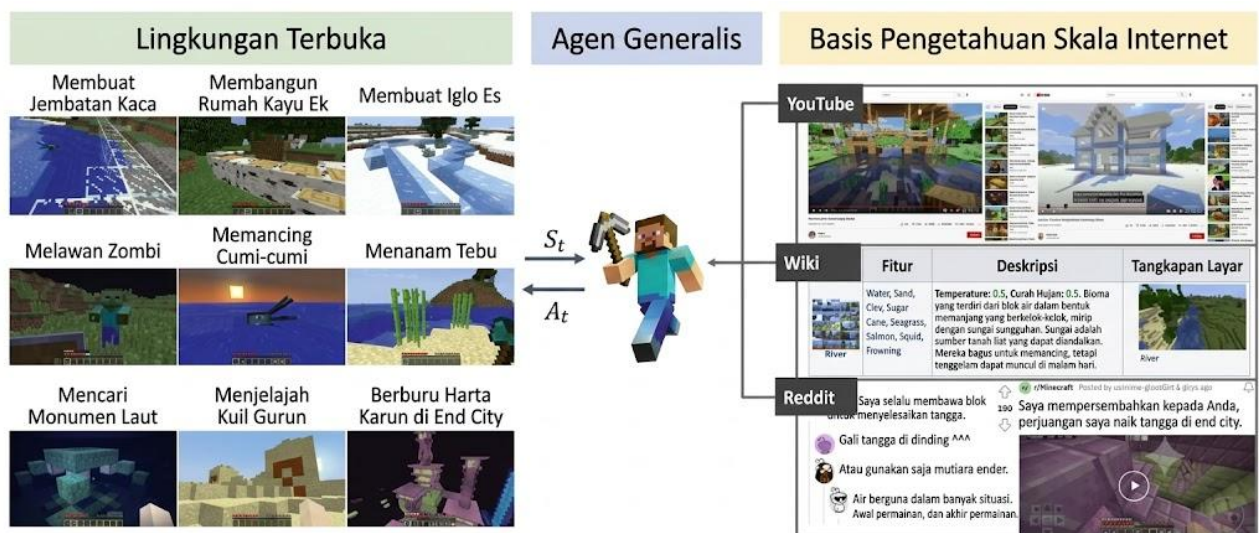
Voyager belajar dan meningkatkan permainan Minecraft lebih baik dari metode sebelumnya. Ia juga beradaptasi dengan tugas-tugas baru di dunia Minecraft baru menggunakan perpustakaan keterampilan yang dipelajari. Gambar 5.13 menunjukkan perbandingan metode lain dengan Voyager.



Gambar 5.13. Jika dibandingkan dengan baseline, Voyager terus menemukan item dan keterampilan Minecraft baru melalui eksplorasi mandiri.

Bersamaan dengan Voyager, tim di NVIDIA juga mengembangkan MineDojo, yang merupakan platform yang dibangun di atas Minecraft yang dapat digunakan untuk melatih agen AI menggunakan berbagai tugas dalam lingkungan simulasi (lihat Gambar 5.14). Platform MineDojo:

- Menyediakan cara standar untuk mendefinisikan tugas, pengaturan dunia, dan perilaku agen, semuanya di dalam Minecraft, sehingga memudahkan pengembangan dan pengujian berbagai model.
- Mencakup benchmark besar dengan ribuan tugas Minecraft yang berbeda.
- Mengkurasi basis pengetahuan multimodal skala besar dari video tutorial, siaran langsung, dan sebagainya, yang dapat digunakan untuk mengajarkan agen tugas dan strategi untuk mencapai suatu tujuan.



Gambar 5.14. MineDojo adalah kerangka kerja untuk mengembangkan agen generalis yang belajar dari tugas terbuka dan data skala internet.

MineCLIP adalah metode pembelajaran baru yang menggunakan model video pra-terlatih untuk mempelajari fungsi penghargaan yang digunakan untuk memandu agen di Minecraft. MineCLIP menghubungkan model bahasa besar dengan pemahaman visual dunia Minecraft, memungkinkan agen untuk:

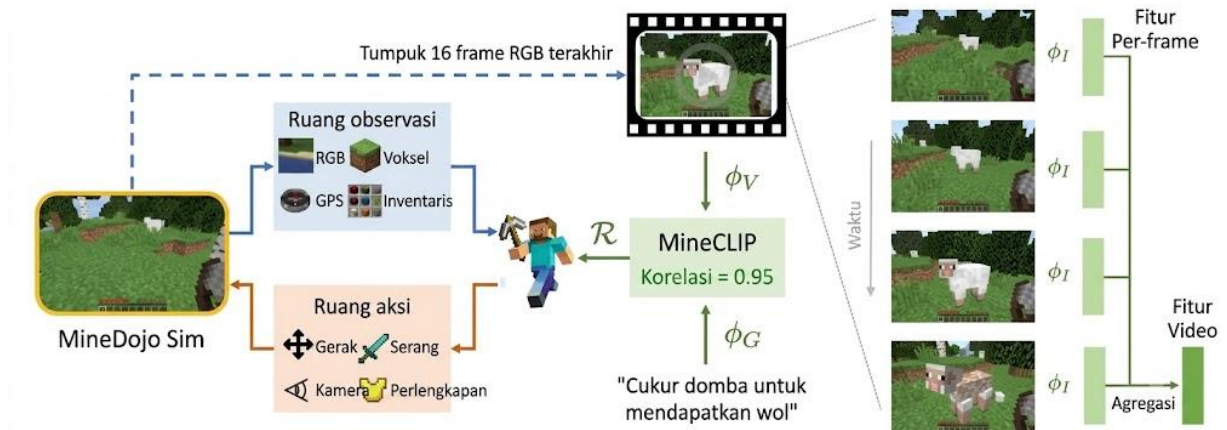
- **Memahami instruksi bahasa alami:** Menafsirkan perintah dan tujuan seperti manusia (misalnya, "bangun rumah").
- **Belajar dari video:** Memperoleh keterampilan dan strategi dengan menonton video permainan Minecraft.
- **Bertindak secara otonom:** Melakukan tugas-tugas kompleks dalam permainan berdasarkan instruksi bahasa dan pengetahuan yang dipelajari.

Pada intinya, MineCLIP bertindak sebagai jembatan antara bahasa dan tindakan di Minecraft. Hal ini dilakukan dengan membangun model bahasa-video kontrastif yang telah dilatih sebelumnya pada video YouTube MineDojo. Model ini memproses frame video dan

deskripsinya untuk mempelajari korelasi. Model ini mengambil urutan 16 frame RGB terakhir, memprosesnya untuk mengekstrak fitur, dan menggabungkannya menjadi fitur video.

Tujuan tekstual dikodekan ke dalam fitur teks dan kemudian fitur video dan teks dibandingkan untuk menghitung skor korelasi, yang menunjukkan seberapa baik frame video yang diamati agen sesuai dengan tujuan, sehingga mempelajari apakah agen benar-benar berhasil mencapai tujuan atau tidak. Misalnya, jika tugasnya adalah "lakukan tugas x untuk mendapatkan makanan y", skor korelasi akan mencerminkan seberapa baik frame video selaras dengan deskripsi tugas ini.

Skor ini digunakan sebagai hadiah untuk melatih agen RL, membantunya memprioritaskan tindakan yang selaras dengan tujuan. MineCLIP dirancang untuk RL multi-tugas, menggunakan skor korelasi untuk memandu agen pada berbagai tugas tanpa merekayasa fungsi hadiah secara manual (lihat Gambar 5.15).



Gambar 5.15. Pipeline untuk MineCLIP menggunakan pembelajaran kontrastif pada data video-bahasa.

Meskipun fokus proyek khusus ini adalah Minecraft, lingkungan simulasi serupa dapat digunakan untuk menunjukkan bagaimana robotika dapat memperoleh manfaat dari simulasi. Lingkungan ini menyediakan ruang aman bagi robot untuk berlatih, menyempurnakan, dan menyimpan keterampilan mereka, yang memungkinkan pembelajaran dan adaptasi yang lebih baik terhadap tugas dan lingkungan baru.

Simulasi untuk Desain Imbalan

Mengajarkan robot untuk melakukan tugas-tugas kompleks seringkali membutuhkan rekayasa fungsi imbalan untuk memandu proses pembelajaran robot. Namun, mendesain fungsi imbalan ini dapat memakan waktu dan mungkin tidak secara efektif menangkap kompleksitas tugas seperti manipulasi yang cekatan. Menggunakan data simulasi dapat bermanfaat sebagai cara untuk menghasilkan, mengevaluasi, dan menyempurnakan fungsi imbalan dalam lingkungan yang lebih terkontrol dan aman. Misalnya, Eureka, yang dikembangkan oleh NVIDIA, menggunakan IsaacGym untuk menciptakan tugas manipulasi robot yang beragam dan kompleks. Tugas-tugas ini berkisar dari penanganan objek sederhana

hingga manipulasi bimanual seperti memutar cangkir 180 derajat atau melakukan trik memutar pena dengan Tangan Bayangan. Ide inti di balik Eureka adalah menggunakan LLM seperti GPT-4 untuk menghasilkan dan meningkatkan fungsi penghargaan menggunakan proses iteratif dalam simulasi. Lihat Gambar 5.16.

Dalam kerangka kerja Eureka, robot diberi deskripsi rinci tentang tugas yang akan dilakukan: detail tentang lingkungan, tujuan, dan batasan apa pun yang harus dipatuhi robot. Sebuah perintah awal diteruskan ke LLM, yang mencakup deskripsi tugas dan detail tentang lingkungan yang memandu pembuatan fungsi penghargaan awal. Dengan jumlah iterasi tertentu, hal-hal berikut dilakukan:

- **Contoh kode penghargaan:** LLM menghasilkan beberapa fungsi penghargaan berdasarkan perintah awal. Fungsi penghargaan ini adalah "strategi" yang berbeda yang memberikan umpan balik kepada robot tentang seberapa baik ia melakukan tugas tersebut.
- **Evaluasi fungsi hadiah dalam simulasi:** Setiap fungsi hadiah diuji dalam lingkungan simulasi di mana robot mencoba menyelesaikan tugas menggunakan fungsi hadiah tersebut, dan kemudian kinerja diukur dengan menggunakan fungsi kebugaran yang mengukur efektivitas setiap fungsi hadiah dalam melatih robot.
- **Refleksi hadiah:** Fungsi hadiah dengan kinerja terbaik dipilih berdasarkan analisis bagaimana penalti rotasi, jarak, dan kecepatan sudut berkontribusi pada kinerja robot. Nilai komponen-komponen ini dan fungsi kebugaran tugas dipantau selama proses pelatihan. Perintah awal kemudian disempurnakan untuk membantu LLM menghasilkan fungsi hadiah yang lebih optimal menggunakan umpan balik dari komponen dan penyesuaian.
- **Perbarui hadiah:** Jika fungsi hadiah terbaik dari iterasi saat ini berkinerja lebih baik daripada yang terbaik sebelumnya, maka Eureka memperbarui hadiah referensinya.

Algorithm 1 EUREKA

```
1: Wajib: Deskripsi tugas  $l$ , kode lingkungan  $M$ ,  
   LLM koding LLM, fungsi kecocokan  $F$ , prompt awal prompt  
2: Hiperparameter: Iterasi pencarian  $N$ , ukuran batch iterasi  $K$   
3: untuk  $N$  iterasi lakukan  
4:   // Ambil Sampel  $K$  kode imbalan dari LLM  
5:    $R_1, \dots, R_K \sim \text{LLM}(l, M, \text{prompt})$   
6:   // Evaluasi kandidat imbalan  
7:    $s_1 = F(R_1), \dots, s_K = F(R_K)$   
8:   // Refleksi Imbalan  
9:    $\text{prompt} := \text{prompt} : \text{Reflection}(R_{\text{best}}^n, s_{\text{best}}^n),$   
     di mana  $\text{best} = \arg \max_k s_1, \dots, s_K$   
10:  // Perbarui imbalan Eureka  
11:   $R_{\text{Eureka}}, s_{\text{Eureka}} = (R_{\text{best}}^n, s_{\text{best}}^n), \text{ jika } s_{\text{best}}^n > s_{\text{Eureka}}$   
12: Luaran:  $R_{\text{Eureka}}$ 
```

Gambar 5.16. Algoritma Eureka.

Teknik pencarian evolusioner digunakan untuk mengambil sampel beberapa fungsi hadiah dari LLM di setiap iterasi, meningkatkan peluang menemukan setidaknya satu fungsi hadiah yang baik dengan meningkatkan jumlah sampel.

Fungsi-fungsi ini disempurnakan dengan memutasi fungsi-fungsi berkinerja terbaik dan menggunakan restart acak untuk menghindari optimasi lokal dan menemukan solusi yang lebih baik. Fungsi hadiah itu sendiri terdiri dari beberapa komponen utama:

- Hadiah rotasi mengukur seberapa dekat orientasi objek dengan orientasi tujuan.
- Hadiah jarak berfokus pada kedekatan antara ujung jari robot dan objek.
- Penalti kecepatan sudut diterapkan, yang mencegah gerakan tiba-tiba atau cepat dan membantu menciptakan tindakan yang lebih terkontrol.
- Total hadiah adalah jumlah tertimbang dari komponen-komponen ini berdasarkan seberapa besar pengaruhnya terhadap robot dalam menyelesaikan tugasnya dengan sukses.

Peningkatan berkelanjutan ini tidak akan mungkin terjadi tanpa lingkungan simulasi di mana mudah untuk mengontrol robot dan lingkungannya serta menguji berbagai fungsi hadiah. Dengan menggunakan teknik ini, Eureka menemukan bahwa fungsi hadiah yang dihasilkan lebih baik daripada fungsi hadiah yang dirancang manusia dan dalam beberapa kasus bahkan menghasilkan formulasi hadiah baru.

Simulasi untuk Pemodelan Dunia

Metode tradisional untuk menghasilkan kumpulan data skala besar untuk pembelajaran robot seringkali bergantung pada:

- **Demonstrasi:** Membutuhkan keterlibatan manusia yang signifikan untuk mengajarkan tugas kepada robot, sehingga dapat sangat memakan waktu dan tenaga.
- **Pengumpulan data otonom:** Dapat membutuhkan banyak rekayasa untuk meningkatkan keterampilan dan sulit untuk diskalakan meskipun otonom.

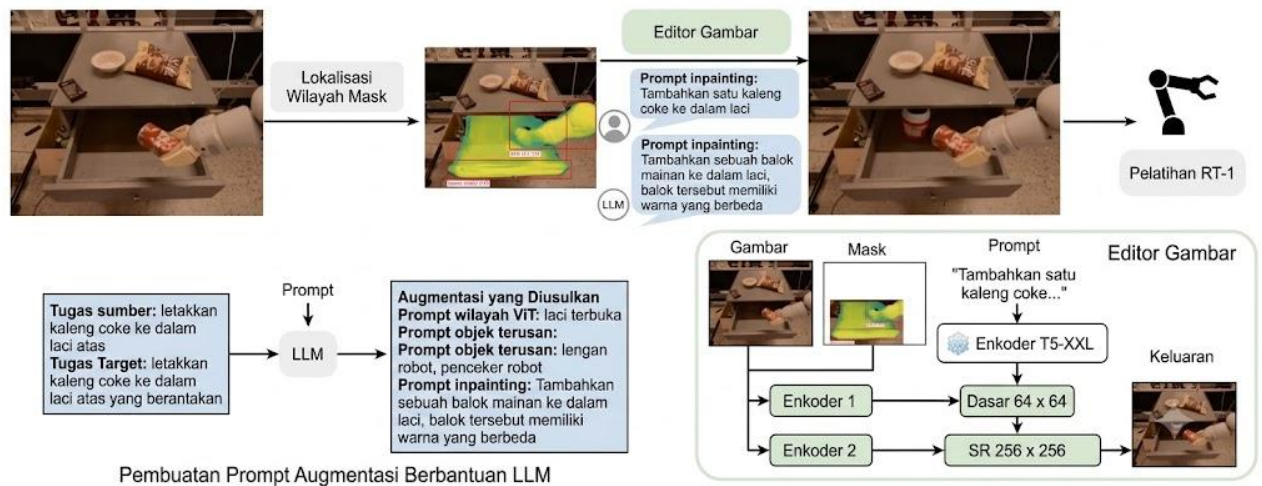
Karena kedua metode tersebut sulit diimplementasikan dalam skala besar, hal ini dapat membatasi keragaman dan volume data yang tersedia untuk melatih model robot. Simulasi, seperti yang telah dibahas, adalah salah satu cara untuk mengatasi masalah ini karena memungkinkan robot untuk belajar di lingkungan virtual. Solusi lain adalah melalui pemodelan dunia tingkat lanjut, yang dapat membantu menghasilkan data yang realistis dan beragam.

Pemodelan dunia penting bagi robot untuk memahami lingkungannya, tetapi juga merupakan cara yang baik untuk mengumpulkan data yang dapat digunakan untuk pembelajaran robot. Pertimbangkan contoh ROSIE (*Scaling Robot Learning with Semantically Imagined Experience*), karya dari Google DeepMind, di mana model difusi yang dipandu teks digunakan untuk pemodelan dunia. Lihat Gambar 5.17.

ROSIE memperkaya data demonstrasi untuk meningkatkan variasi dalam dataset pembelajaran robot dan meningkatkan kemampuan adaptasi kebijakan pembelajaran imitasi.

- Dengan menggunakan model segmentasi kosakata terbuka, ROSIE mengidentifikasi dan melokalisasi berbagai wilayah yang diminati dalam gambar untuk mengidentifikasi di mana augmentasi harus diterapkan.

- Wilayah yang diidentifikasi diedit menggunakan pengeditan gambar berbasis teks. Misalnya, ia dapat menambahkan objek baru atau mengubah objek yang ada berdasarkan petunjuk teks yang merupakan variasi dari tugas awal. Dengan menggunakan petunjuk teks, editor gambar membuat versi baru yang bervariasi dari tugas asli. Data yang diperkaya ini mencakup berbagai objek, latar belakang, dan sebagainya. Setiap gambar dalam episode diproses untuk menyertakan elemen-elemen baru ini, yang pada akhirnya meningkatkan keragaman dataset.
- Data yang diperkaya kemudian digunakan untuk melatih kebijakan manipulasi RT-1, yang merupakan kebijakan imitasi multi-tugas yang dibahas dalam Bab 4. Idennya adalah bahwa, dengan menggabungkan berbagai skenario, kebijakan tersebut menjadi lebih kuat terhadap tugas-tugas baru dan yang belum pernah dilihat sebelumnya.



Gambar 5.17. Arsitektur untuk ROSIE.

Metode lain, seperti InstructPix2Pix, telah mencoba menggunakan pendekatan serupa. Namun, ROSIE menemukan bahwa augmentasi yang dihasilkannya lebih realistis secara fisik dan konsisten dalam konteks tugas aslinya. Hal ini dilakukan dengan memastikan tidak ada perubahan global yang lebih luas pada gambar yang dapat mengubah konteks gambar asli. Lihat Gambar 5.18.



Gambar 5.18. Proses augmentasi yang digunakan dalam ROSE.

Dari perspektif infrastruktur, NVIDIA telah merilis banyak alat baru untuk simulasi dan pemodelan dunia yang mungkin menarik bagi pengembang robotika yang tertarik untuk membangun kebijakan pembelajaran robot yang realistis dan terukur. Pada GTC 2024, mereka mengumumkan Proyek GROOT, model dasar untuk robot humanoid dan Jetson Thor, komputer baru yang dirancang khusus untuk robot humanoid.

Isaac Lab dari NVIDIA mendukung menjalankan ribuan simulasi paralel untuk mendukung RL dan pembangkitan data skala besar, pelatihan model, dan alur kerja terdistribusi. Alat-alat ini secara kolektif memajukan kemampuan robot untuk dapat melakukan tugas yang lebih kompleks dan meningkatkan beberapa masalah kelangkaan data yang ada dalam robotika melalui dukungan simulasi skala besar.

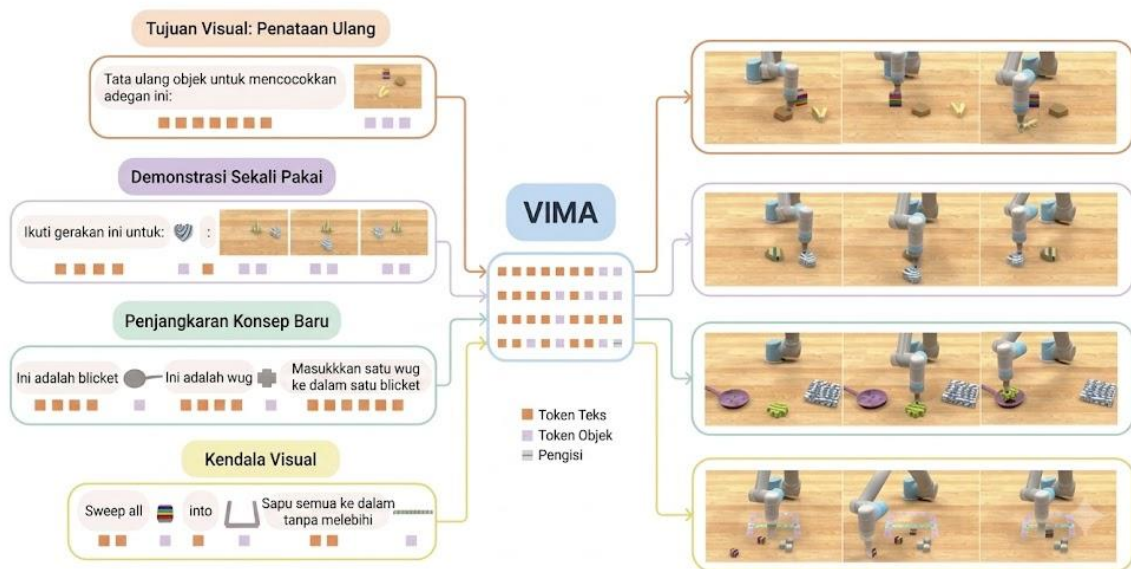
Simulasi untuk Pembelajaran Imitasi

Lingkungan simulasi dapat menyediakan sarana untuk meningkatkan skala pendekatan pembelajaran imitasi dan mengulang ide-ide studi yang terukur. *VIMA: General Robot Manipulation with Multimodal Prompts* adalah karya dari NVIDIA yang mendorong ke arah ini, dengan fokus khusus pada pengkondisian tugas multimodal. Meskipun VIMA adalah metode generik yang dapat dieksekusi dalam pengaturan dunia nyata, ini adalah contoh yang mengesankan dari penggunaan simulasi untuk pembelajaran ujung-ke-ujung, konsep dasar, dan iterasi cepat dalam skala besar. VIMA membangun tolok ukur simulasi dengan ribuan tugas menggenggam meja yang dihasilkan dengan campuran petunjuk teks dan gambar dan lebih dari 600.000 episode ahli untuk membantu robot belajar melalui imitasi. VIMA menggunakan petunjuk multimodal—yaitu petunjuk dengan kombinasi gambar dan teks—dengan arsitektur berbasis transformer untuk mengeksekusi berbagai tugas manipulasi robot. VIMA menunjukkan beberapa cara menarik untuk menentukan tugas untuk pembelajaran robot, melalui petunjuk multimodal:

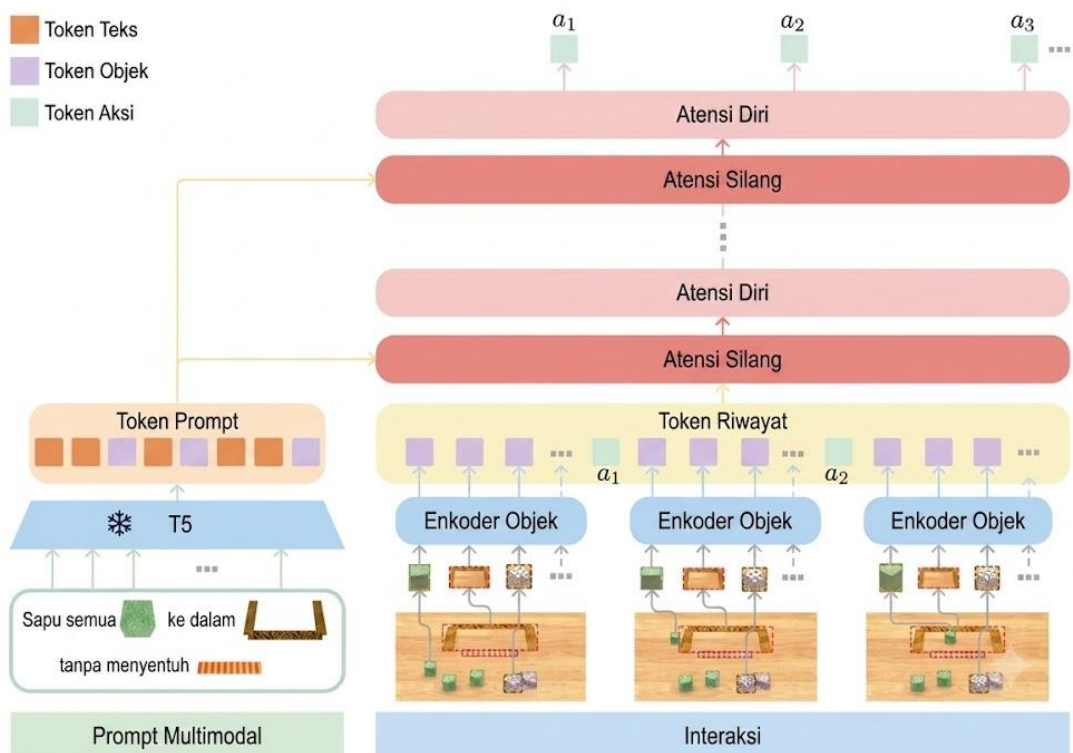
- **Pengkondisian tujuan untuk orientasi:** Dalam tugas seperti menata ulang objek, lebih sulit untuk menentukan tujuan tersebut hanya melalui teks, dan membutuhkan spesifikasi multimodal. Salah satu cara untuk melakukannya adalah dengan memberikan gambar dan meminta robot untuk mengatur ulang adegan agar sesuai dengan konfigurasi akhir yang diharapkan. Katakanlah Anda memiliki meja dengan berbagai buah (apel, pisang, jeruk) yang tersebar di sekitarnya. Perintah tersebut mencakup gambar piring buah yang tersusun rapi, di mana setiap buah disusun dalam urutan tertentu. Teks yang sesuai, "Susun ulang buah-buahan agar sesuai dengan tata letak ini," juga diberikan.
- **Pembelajaran satu kali:** Tujuannya di sini adalah agar robot belajar melakukan suatu tugas hanya dengan satu demonstrasi dari ahli. Misalnya, jika Anda ingin robot belajar menumpuk buku dalam urutan tertentu, Anda dapat memberikan serangkaian gambar yang menunjukkan seseorang menumpuk buku yang berbeda di setiap bingkai bersamaan dengan teks "Tumpuk buku dalam urutan ini."
- **Pemahaman konsep:** Memeriksa apakah robot dapat memahami konsep simbolik dan melakukan tugas yang memanipulasinya. Misalnya, Anda memperkenalkan objek baru yang disebut "x" dan "y" kepada robot. Perintah tersebut mencakup gambar yang menunjukkan "Ini adalah x" di samping gambar objek silinder dan "Ini adalah y" di samping objek bola. Tugasnya mungkin "Letakkan x di samping y."
- **Batasan visual:** Di sini Anda menentukan batasan melalui kombinasi gambar dan teks. Misalnya, jika Anda ingin robot membersihkan area bermain, Anda dapat memberikan perintah "Pindahkan semua mainan ke dalam kotak tanpa menjatuhkan menara balok," tetapi sebagian dari perintah tersebut adalah token visual.

Gambar 5.19 menunjukkan contoh kasus penggunaan ini dan bagaimana VIMA diberi perintah multimodal untuk masing-masing kasus.

Arsitektur VIMA ditunjukkan pada Gambar 5.20. Mereka menggunakan tokenizer T5 terlatih untuk tokenisasi perintah multimodal dan menggunakan perhatian diri dan silang untuk menyisipkan token perintah dan riwayat interaksi yang telah di tokenisasi. Konteks ini kemudian digunakan oleh dekoder kausal untuk menghasilkan tindakan, dan seluruh sistem dilatih melalui kloning perilaku pada kumpulan data demonstrasi ahli.



Gambar 5.19. VIMA memproses perintah multimodal yang menggabungkan input bahasa dan visual untuk menentukan tugas manipulasi robot.



Gambar 5.20. Arsitektur VIMA.

Varian VIMA yang telah dilatih diuji untuk berbagai tingkat generalisasi:

1. Generalisasi adegan: Dengan menempatkan objek dalam urutan/orientasi yang berbeda dalam adegan.

2. Objek yang telah dilihat, tetapi kombinasi baru: Memanipulasi dua objek berbeda dalam kombinasi yang belum pernah dilihat dalam set pelatihan, tetapi di mana setiap objek telah dilihat dalam konteks yang berbeda selama pelatihan. Ini memeriksa transfer pemahaman konsep dari tahap pelatihan ke tahap inferensi.
3. Objek baru: Objek baru diperkenalkan ke tugas yang belum pernah dilihat selama pelatihan untuk memeriksa generalisasi.
4. Tugas baru: Tugas yang sama sekali belum pernah dilihat dengan objek yang belum pernah dilihat.

Gambar 5.21 menunjukkan kerangka evaluasi generalisasi yang digunakan dalam VIMA. Setiap tingkat ini berbeda secara progresif dari data pelatihan, membuat tugas lebih menantang dan bertindak sebagai kerangka evaluasi yang kuat untuk generalisasi. Meskipun studi ini dilakukan sepenuhnya dalam simulasi, semua konsep ini dapat ditransfer ke domain dunia nyata, memberikan contoh nyata bagaimana simulasi digunakan untuk memajukan studi pembelajaran robot.



Gambar 5.21. Empat tingkat evaluasi untuk VIMA untuk mengukur generalisasi zero-shot.

5.6 RINGKASAN

Selama bab ini, Anda telah mempelajari bahwa:

- Simulasi menawarkan solusi yang hemat biaya dan terukur untuk menghasilkan beragam data sintesis, yang dapat membantu robot belajar dengan aman dan efisien. Namun, keterbatasan seperti kesenjangan Sim2Real mengharuskan penggabungan simulasi dengan data dunia nyata.
- Simulator robot, seperti PyBullet, MuJoCo, dan Gazebo, mencakup fitur inti seperti mesin fisika untuk dinamika benda kaku dan lunak, pemodelan robot melalui URDF, simulasi lingkungan, dan interaksi pengguna melalui API dan GUI untuk kontrol dan visualisasi.
- Teknik seperti adaptasi domain dan pengacakan domain digunakan untuk menjembatani kesenjangan antara simulasi dan dunia nyata. Pendekatan seperti pengacakan domain terpandu berfokus pada pelatihan model dengan skenario yang semakin menantang.
- Metode seperti RL-CycleGAN mengatasi kesenjangan Sim2Real dengan menerjemahkan gambar simulasi menjadi gambar realistis sambil mempertahankan

semantik. Ini dapat membantu meningkatkan kemampuan agen RL untuk melakukan generalisasi pada tugas-tugas dunia nyata.

- Pembelajaran dari simulasi menggunakan data simulasi dan data dunia nyata meningkatkan eksplorasi tugas dan perolehan keterampilan. Hal ini ditunjukkan oleh QT-Opt untuk manipulasi robot, dan proyek-proyek seperti Voyager dan MineDojo, dengan LLM dan model dasar. Simulator juga membantu dalam desain penghargaan, seperti yang terlihat pada Eureka. Selain itu, metode pemodelan dunia seperti ROSIE menambah data pelatihan untuk meningkatkan adaptabilitas kebijakan, sementara VIMA menggunakan petunjuk multimodal dan transformer untuk melatih robot pada berbagai tugas manipulasi.

Bab selanjutnya mengeksplorasi penggunaan metode pembelajaran mendalam untuk pemetaan dan lokalisasi dalam robotika, pengaturan sensor, teknik pemetaan geometris dan semantik, dan berbagai metode lokalisasi.

Tutorial PyBullet

Tutorial ini memandu Anda melalui konsep-konsep dasar di PyBullet, termasuk memuat URDF, menjalankan simulasi, dan mengontrol sendi. Untuk memulai, Anda perlu membuat file URDF dari struktur fisik robot Anda. Jika Anda belum pernah membuat file URDF sebelumnya, Anda dapat menggunakan contoh dari internet atau membuatnya sendiri menggunakan SolidWorks, yang memiliki plugin ekspor URDF.

Untuk menginstal PyBullet di sistem Anda, Anda dapat menyalin dan menempelkan yang berikut ini ke terminal Anda (Mac dan Linux):

```
sudo pip3 install pybullet
```

Untuk Windows, Anda dapat menggunakan perintah berikut di terminal Anda:

```
pip install pybullet
```

Buat file Python baru tempat URDF Anda disimpan. Beri nama `pybullet_simulation.py` dan buka di editor kode. Bagian kode pertama yang perlu ditulis dalam skrip Anda adalah pernyataan `import`. PyBullet memberi Anda akses ke API simulasinya dan modul `time` memungkinkan Anda menambahkan penundaan untuk mengontrol kecepatan simulasi.

```
import pybullet as p  
import pybullet_data  
import time
```

Anda dapat membuat koneksi ke GUI simulasi PyBullet menggunakan baris ini, yang memungkinkan Anda untuk memvisualisasikan dan berinteraksi dengan simulasi:

```
p.connect(p.GUI)
```

Anda dapat menjalankan simulasi dengan dunia kosong dengan menambahkan perulangan ini:

```
while True:  
    p.stepSimulation()  
    time.sleep(1./240)
```

Lingkungan ini harus benar-benar kosong. Anda dapat menambahkan aliran dengan memuat URDF bidang yang sudah ada yang tersedia di jalur data PyBullet:

```
p.setAdditionalSearchPath(pybullet_data.getDataPath())  
floor = p.loadURDF("plane.urdf")
```

Anda kemudian dapat memuat URDF robot dan mengatur gravitasi di lingkungan Anda. Pastikan untuk mengganti [nama_file_robot].urdf dengan nama sebenarnya dari file URDF Anda.

```
robot = p.loadURDF("my_robot.urdf")  
p.setGravity(0,0,-9.81)
```

Sebelum mengendalikan sendi robot, ada baiknya untuk memeriksa sendi-sendi tersebut guna memahaminya dengan baik. Fungsi `getNumJoints()` dan `getJointInfo()` dapat digunakan untuk tujuan ini:

```
for i in range(p.getNumJoints(robot)):  
    joint_info = p.getJointInfo(robot,i)  
    print(f"Joint {i}: {joint_info}")
```

Untuk menggerakkan beberapa sendi, Anda dapat menggunakan fungsi `setJointMotorControl2()`. Fungsi ini memungkinkan Anda untuk menentukan posisi target, kecepatan, dan parameter kontrol lainnya.

Kode ini akan mengontrol sendi pertama, yang biasanya mewakili sendi dasar robot Anda:

```
target_position = 1.57 # 90 degrees  
p.setJointMotorControl2(bodyUniqueId = robot,  
    jointIndex = 0, # can be changed  
    as needed  
    controlMode = p.POSITION_CONTROL,  
    targetPosition = target_position)
```

Anda dapat menjelajahi berbagai mode kontrol:

- **POSITION_CONTROL**: Menggerakkan sendi ke sudut tertentu.
- **VELOCITY_CONTROL**: Mengontrol kecepatan sendi.
- **TORQUE_CONTROL**: Mengontrol langsung torsi yang diterapkan pada sendi.
- **PD_CONTROL**: Menggunakan pengontrol PD untuk sendi.

Terakhir, Anda dapat menambahkan loop simulasi ke akhir skrip Anda untuk menjalankan semuanya secara bersamaan:

```
while True:  
    p.stepSimulation()  
    time.sleep(1./240)
```

Setelah Anda berhasil membuat pengaturan dasar ini, Anda dapat mulai bereksperimen dengan fitur-fitur PyBullet yang lebih canggih:

- **Deteksi tabrakan**: Tambahkan objek ke lingkungan dan periksa tabrakan.
- **Sensor**: Simulasikan sensor seperti kamera atau LiDAR.
- **Pembelajaran penguatan (RL)**: Latih robot Anda menggunakan algoritma RL di PyBullet.

Untuk debugging dan memperbaiki kesalahan apa pun, kami sarankan untuk melihat dokumentasi resmi PyBullet.

BAB 6

PEMETAAN, LOKALISASI, DAN NAVIGASI

Manusia menggunakan persepsi sensorik multimodal untuk mengidentifikasi objek dan bergerak di ruang yang kompleks. Hal ini memungkinkan kita untuk bernavigasi, melakukan tugas-tugas kompleks, dan berinteraksi dengan objek di lingkungan kita. Demikian pula, dengan menggunakan berbagai sensor dan persepsi sensorik multimodal, robot (seperti mobil otonom, drone, dan robot rumah tangga) dapat merasakan lingkungan sekitarnya dan membuat keputusan dengan memperkirakan keadaan dan lingkungan mereka saat ini. Kemampuan ini dimungkinkan melalui lokalisasi dan pemetaan, yang menjadi fokus bab ini. Lokalisasi membantu robot memahami posisinya dalam suatu adegan saat ini; ini melibatkan penentuan "keadaan" robot (lokasi, orientasi, kecepatan, dll.) relatif terhadap objek lain. Pemetaan melibatkan pengambilan model lingkungannya melalui geometri, penampilan, dan semantik ruang 2D atau 3D yang berbeda. Pemetaan merupakan pendahulu lokalisasi, karena lokalisasi robot dilakukan sehubungan dengan peta. Navigasi adalah tindakan bergerak ke lokasi yang diinginkan, dan mencakup pemetaan, lokalisasi, dan penalaran. Sebelum Anda mempelajari metode untuk masing-masing hal tersebut, penting untuk memahami mengapa dan kapan menggunakan pembelajaran mendalam untuk tugas pemetaan dan lokalisasi.

6.1 METODE LOKALISASI DAN PEMETAAN ROBOT

Banyak metode dan algoritma klasik telah dikembangkan selama bertahun-tahun untuk mengatasi tantangan dalam lokalisasi dan pemetaan. Beberapa metode ini termasuk estimasi odometri—seperti odometri visual, odometri LIDAR, SLAM, dan struktur dari gerak (SfM). Memilih antara teknik berbasis deep learning dan teknik tradisional, seperti filter Kalman, melibatkan pertimbangan trade-off. Filter Kalman efektif untuk odometri visual, lokalisasi fusi sensor, atau SLAM. Ambil contoh, implementasi open-source Comma AI, RedNose yang digunakan untuk self-driving di dunia nyata. Meskipun bab ini berfokus pada metode deep learning, pilihan metode yang digunakan bergantung pada persyaratan aplikasi, karena setiap pendekatan memiliki kelebihan dan kekurangan.

Metode Tradisional

Metode tradisional unggul dalam menangani data yang bising atau tidak konsisten melalui teknik matematika yang mapan. Metode ini dapat lebih mudah diinterpretasikan, yang sangat penting untuk aplikasi di mana pemahaman proses pengambilan keputusan sangat penting. Metode-metode ini biasanya membutuhkan lebih sedikit data berlabel untuk pelatihan, sehingga cocok untuk skenario dengan pengumpulan data yang terbatas. Namun, metode tradisional seringkali membutuhkan rekayasa fitur manual, yang dapat memakan waktu dan mungkin hanya menangkap beberapa informasi relevan dalam data sensor yang kompleks. Metode ini juga mungkin memerlukan modifikasi signifikan untuk beradaptasi dengan lingkungan baru atau yang berubah.

Metode Pembelajaran Mendalam

Metode pembelajaran mendalam dapat mencapai akurasi tinggi ketika dilatih pada kumpulan data yang besar dan beragam, karena dapat mempelajari hubungan kompleks dalam data. Fitur yang relevan dapat dipelajari langsung dari data input sensor yang kompleks dan berdimensi tinggi, mengurangi kebutuhan rekayasa fitur manual. Jika dilatih pada data representatif, metode ini dapat beradaptasi dengan lingkungan dan skenario yang berbeda, sehingga cocok untuk aplikasi dengan lingkungan yang sering berubah. Selain itu, metode ini menangkap hubungan spasial yang rumit dalam data, yang berguna untuk memahami peta kompleks atau rekonstruksi 3D.

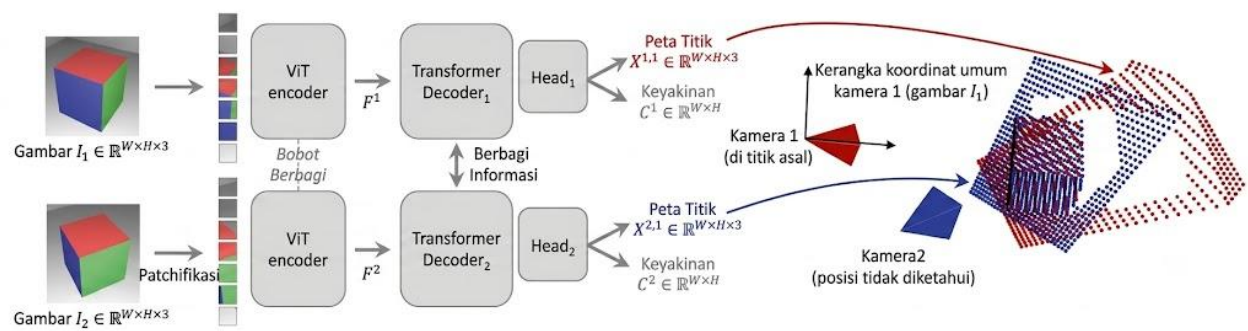
Kelemahannya, agar model pembelajaran mendalam dapat belajar secara efektif, banyak data berlabel yang dibutuhkan untuk pelatihan dan anotasi data dapat memakan waktu dan mahal. Metode ini juga membutuhkan sumber daya komputasi yang signifikan untuk pelatihan dan pengoperasian. Lebih lanjut, meskipun dapat unggul di lingkungan tempat mereka dilatih, mereka mungkin hanya dapat digeneralisasikan ke skenario baru atau yang belum pernah dilihat sebelumnya ketika dilatih pada kumpulan data yang besar dan beragam.

Pendekatan Hibrida

Secara lebih praktis, strategi hibrida sering diterapkan di dunia nyata, yang menggabungkan manfaat dari pembelajaran mendalam dan metode konvensional. Sebagai contoh, model pembelajaran mendalam dapat mengekstrak fitur dan mempelajari pola yang rumit, sementara teknik konvensional dapat digunakan untuk fusi sensor atau pengurangan kebisingan. Saat ini, tugas-tugas seperti deteksi objek dan segmentasi semantik dilakukan melalui jaringan saraf, tetapi fungsi seperti odometri dan rekonstruksi 3D sering dilakukan menggunakan metode yang lebih "tradisional". Lihat Gambar 6.1.

Namun, kita mulai melihat perubahan, karena metode seperti Dust3r menggunakan jaringan saraf untuk merekonstruksi geometri. Dust3r bekerja dengan langkah-langkah utama berikut:

- Mengambil beberapa gambar dari sudut pandang yang berbeda dan meneruskannya melalui ViT untuk mendapatkan peta fitur.
- Peta fitur ini kemudian dimasukkan ke dalam transformer yang berbagi informasi antara sudut pandang yang berbeda untuk menghasilkan peta titik dan skor kepercayaan untuk setiap gambar. Peta titik mewakili koordinat 3D titik-titik dalam adegan, dan masing-masing memiliki skor kepercayaan yang ditetapkan.
- Peta titik dari sudut pandang yang berbeda diselaraskan ke kerangka koordinat umum, memungkinkan rekonstruksi model adegan 3D.



Gambar 6.1. Jaringan menggunakan encoder ViT bersama, decoder transformer, dan kepala regresi untuk menghasilkan peta titik dan peta kepercayaan dari dua tampilan adegan.

Demikian pula, metode berbasis jaringan saraf menjadi semakin menjanjikan untuk estimasi kedalaman monokular. Perkembangan terbaru dalam model video merupakan contoh kuat dari hal ini, karena dapat dikondisikan pada gambar masukan dan dapat memungkinkan orang untuk melakukan estimasi kedalaman dari rangkaian video.

Sebuah survei tahun 2023 menyoroti bahwa meskipun estimasi kedalaman monokular NN belum sepenuhnya matang, kemajuan sedang dibuat: “Menurut pandangan kami, sebagian besar algoritma MDE yang ada mengalami keterbatasan dalam keragaman pelatihan, khususnya dalam hal berbagai area dan jenis gambar. Misalnya, ketika dataset pelatihan kekurangan jumlah gambar yang cukup yang menampilkan langit, hal itu dapat menyebabkan tantangan dalam memperkirakan kedalaman secara akurat di wilayah tertentu tersebut.” Seiring dengan semakin beragam dan detailnya dataset pelatihan, kinerja metode estimasi kedalaman berbasis jaringan saraf dapat meningkat secara signifikan. Keputusan tentang di mana dan kapan menggunakan pembelajaran mendalam untuk tugas pemetaan dan lokalisasi pada akhirnya dipengaruhi oleh variabel seperti ketersediaan data, sumber daya komputasi, dan kebutuhan aplikasi.

6.2 PENGATURAN ROBOT SELULER KHAS

Meskipun terdapat banyak variasi dalam konfigurasi dan perwujudan sensor robot, sebagian besar sistem tujuan umum (seperti mobil otonom atau robot humanoid) memerlukan sensor khusus agar efektif dalam pemetaan dan lokalisasi. Memiliki kamera RGB dan sensor IMU sangat penting, dan sebagian besar sistem kemungkinan akan memiliki beberapa kamera RGB (mungkin dalam pengaturan kamera stereo untuk estimasi kedalaman). Sensor kedalaman LiDAR dapat memberikan peta 3D yang sangat akurat, tetapi beberapa perusahaan (misalnya, Tesla) lebih memilih sistem berbasis visi karena biaya dan kompleksitas LiDAR. Terakhir, sensor GPS umumnya digunakan untuk robot yang mungkin menempuh jarak yang cukup jauh (seperti mobil otonom). Menentukan konfigurasi sensor yang tepat dan memasangkannya dengan teknik estimasi yang benar adalah masalah kompleks tanpa jawaban "satu ukuran untuk semua" dan seringkali melibatkan beberapa eksperimen.

6.3 PEMETAAN BERBASIS PEMBELAJARAN MENDALAM

Dalam robotika, pemetaan mengacu pada kemampuan robot untuk membuat model lingkungan yang andal dan detail, yang membantunya menyadari sekitarnya. Metode pembelajaran mendalam dapat digunakan untuk memahami lingkungan dan membuat peta adegan. Peta adegan ini dapat membantu dalam perencanaan dan navigasi.

Pemetaan Geometris

Pemetaan geometris merepresentasikan bentuk dan detail struktural suatu adegan. Dalam pemetaan geometris, representasi voxel adalah representasi adegan yang membagi lingkungan 3D menjadi kisi-kisi kubus kecil (*voxel*) yang dapat memberi tahu Anda apakah ruang dalam adegan pada titik tersebut ditempati atau kosong. Representasi kedalaman melibatkan pembuatan gambar dari sensor kedalaman di mana setiap piksel memiliki informasi tentang jarak ke objek terdekat. Representasi ini memberikan pemahaman tentang tata letak spasial dan geometri spasial adegan.

Representasi Kedalaman

Peta kedalaman sangat penting untuk memahami geometri dan struktur suatu adegan. Banyak metode pembelajaran mendalam, seperti metode pembelajaran terawasi, menggunakan CNN dengan gambar RGB dan peta kedalaman untuk memprediksi kedalaman. Namun, metode-metode ini membutuhkan sejumlah besar data mendalam, yang mungkin tidak tersedia dalam semua kasus. Untuk mengatasi hal ini, metode semi-supervised dan metode self-supervised yang mengandalkan data tanpa label dapat digunakan, tetapi metode-metode ini menghadapi tantangan dalam generalisasi.

Langkah-langkah utama dalam pipeline representasi kedalaman adalah sebagai berikut:

- Mengumpulkan gambar RGB menggunakan kamera monokuler atau dataset seperti NYU-v2 atau KITTI, yang menyediakan gambar RGB dengan peta kedalaman kebenaran dasar.
- Untuk mengekstrak fitur dari gambar, gunakan CNN, seperti model pra-terlatih seperti ResNet atau DenseNet.
- Berbagai model pembelajaran mendalam dapat digunakan untuk memprediksi nilai kedalaman untuk setiap piksel gambar. Dalam metode pembelajaran terawasi, CNN akan dilatih pada gambar RGB dan peta kedalaman yang sesuai untuk langsung menghasilkan informasi kedalaman. Dalam metode pembelajaran mandiri, jaringan akan dilatih menggunakan gambar tanpa label dengan mendapatkan informasi tambahan dari gambar stereo atau odometri visual. Metode semi-supervised sering menggabungkan data berlabel dan tanpa label untuk pelatihan.
- Seringkali, modul tambahan menyempurnakan peta kedalaman. Hal ini dapat dilakukan dengan menyelaraskan perkiraan kedalaman awal dengan batasan geometris yang diharapkan atau mengintegrasikan konteks global melalui perhatian mandiri.

Kami tidak membahas terlalu detail tentang metode-metode ini, tetapi gambaran umum model estimasi kedalaman berbasis pembelajaran mendalam tertentu ditunjukkan pada Gambar 6.2, dan Anda dapat mempelajari lebih lanjut di sini.

Berikut adalah konversi dari gambar tabel di atas ke dalam format tabel Markdown versi Bahasa Indonesia yang rapi, terstruktur, dan mudah dibaca secara jelas:

Metode	Arsitektur	Kategori
EMDEOM	FC	Terdireksi (<i>Supervised</i>)
ACAN	Encoder-Decoder	
DenseDepth	Encoder-Decoder	
DORN	CNN	
VNL	Encoder-Decoder	
BTS	Encoder-Decoder	
DeepV2D	CNN	
LISM	Encoder-Decoder	Mandiri (<i>Self-supervised</i>)
monoResMatch	CNN	
PackNet-SfM	CNN	
VOMonodepth	Auto-Decoder	
monodepth2	CNN	
GASDA	CNN	Semi-terdireksi (<i>Semi-supervised</i>)

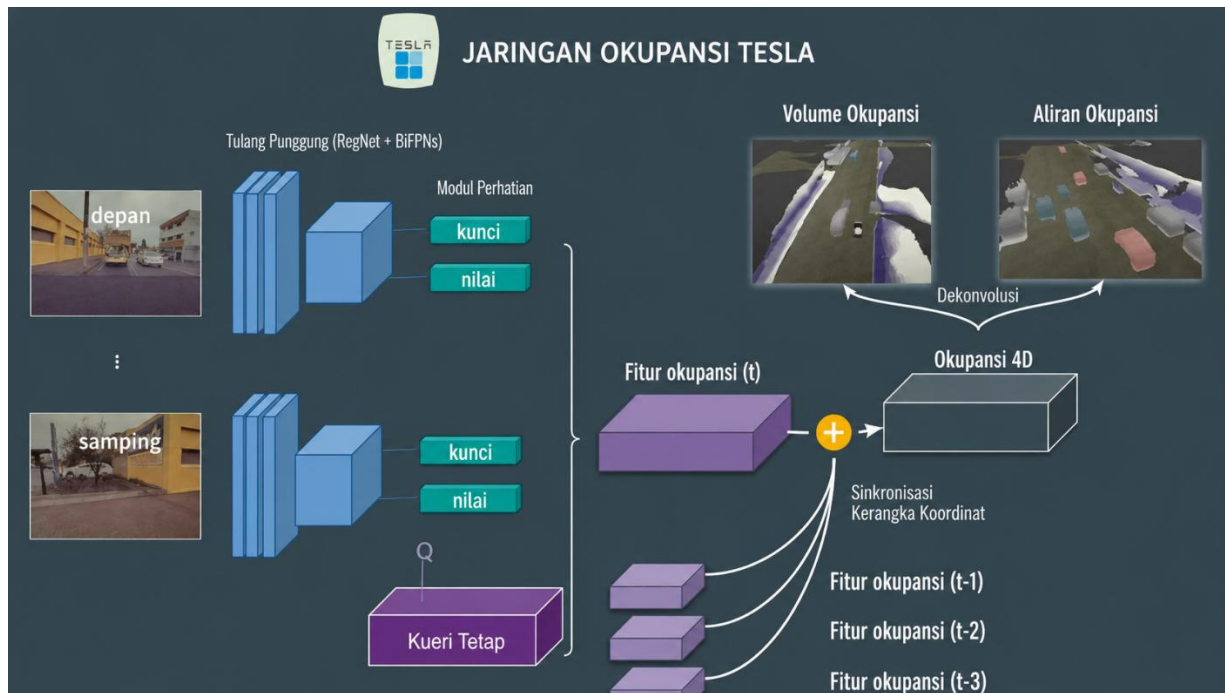
Gambar 6.2. Metode untuk estimasi kedalaman.

Representasi Voxel

Elemen volume ruang 3D dikenal sebagai voxel, dan cara alami untuk merepresentasikan geometri 3D adalah dengan formulasi berbasis voxel. Alur kerja untuk representasi voxel seringkali terlihat seperti berikut:

- Sensor seperti LiDAR, RGB-D, atau kamera stereo mengumpulkan data 3D mentah.
- Data 3D mentah ini dapat dikonversi menjadi awan titik, yaitu kumpulan titik data di mana setiap titik mewakili titik pada permukaan suatu objek.
- Dalam beberapa kasus, awan titik ini juga dapat dikonversi menjadi mesh 3D, yaitu sekelompok simpul, tepi, dan permukaan yang mewakili bentuk objek.
- Data 3D ini dapat mengalami voxelisasi, di mana, berdasarkan titik-titik dari awan titik, setiap voxel diidentifikasi sebagai ditempati oleh suatu objek atau kosong.
- Kisi voxel dapat digunakan sebagai input ke CNN untuk mengekstrak fitur yang mewakili struktur spasial dan okupansi lingkungan.

Tesla sebelumnya mempopulerkan Jaringan Okupansi, representasi voxel 3D dari dunia dari delapan aliran kamera mereka. Sistem mereka menggunakan delapan kamera, yang memberi input ke model yang dirancang untuk ekstraksi fitur (lihat Gambar 6.3), di mana modul perhatian menghasilkan volume fitur okupansi, yang dikombinasikan dengan volume sebelumnya untuk membuat kisi okupansi 4D. Akhirnya, dekonvolusi mengembalikan ukuran asli, menghasilkan volume okupansi, tampilan 3D dari atas, dan aliran okupansi, yang menunjukkan pergerakan setiap piksel dari satu frame ke frame berikutnya.



Gambar 6.3. Gambaran umum Jaringan Okupansi Tesla.

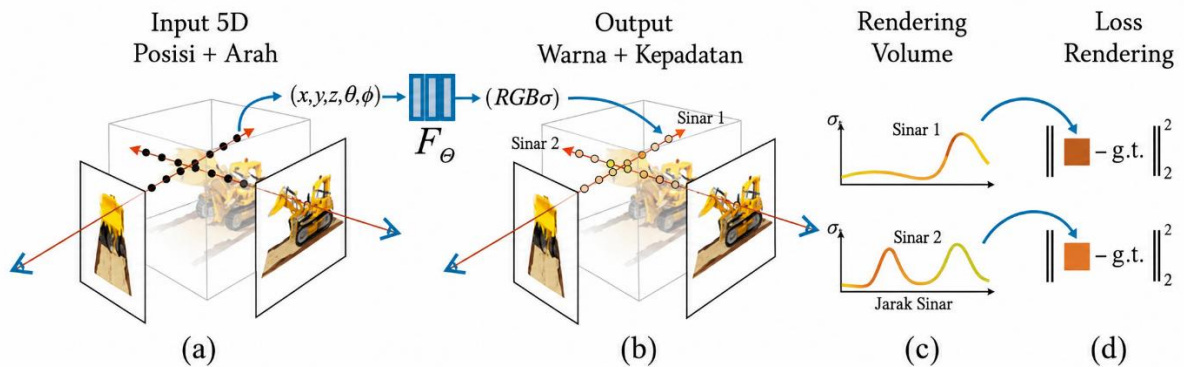
Meskipun voxel memiliki banyak kegunaan, salah satu keterbatasannya adalah tuntutan komputasi yang tinggi untuk merekonstruksi suatu adegan dalam resolusi tinggi yang dibutuhkan untuk representasi voxel.

NeRF: Neural Radiance Fields

Neural Radiance Fields (NeRFs) digunakan dalam bidang visi komputer karena memungkinkan pembuatan adegan 3D terperinci dari gambar 2D. Mereka menawarkan aplikasi yang berharga dalam robotika, khususnya untuk lokalisasi, pemetaan, dan pemahaman lingkungan 3D. NeRF hanyalah pelatihan jaringan saraf untuk memprediksi seperti apa gambar suatu adegan dari pose kamera tertentu. Jaringan tersebut sengaja "di-overfit" ke banyak gambar dari satu adegan dan kemudian dapat memprediksi gambar untuk pose kamera baru yang tidak ada dalam set pelatihan.

NeRF menggunakan jaringan feed-forward untuk menghasilkan informasi detail tentang suatu adegan. Mereka menggunakan sinar, yaitu garis-garis dalam ruang 3D yang memanjang dari sudut pandang kamera tertentu ke dalam adegan ke arah tertentu. Sinar-sinar ini adalah garis-garis dalam ruang 3D yang diproyeksikan dari sudut pandang kamera tertentu ke dalam adegan sepanjang jalur tertentu. Proses rendering kemudian menggabungkan informasi ini menjadi gambar 2D. Karena kedua komponen tersebut dapat didiferensiasi, seluruh sistem dapat dilatih secara end-to-end. NeRF mendefinisikan medan radiasi 5D sebagai input lokasi (x, y, z) dan arah 2D (ditentukan oleh dua sudut dalam ruang 3D).

Seperti yang ditunjukkan pada Gambar 6.3, output dari fungsi ini terdiri dari kepadatan volume (opasitas) dan warna (nilai RGB). Medan radiasi menggunakan jaringan saraf feed-forward dengan input 5D untuk menghasilkan informasi warna dan kepadatan volume yang sesuai. Gambaran umum langkah-langkah utama dalam NeRF ditunjukkan pada Gambar 6.4. Kita akan menguraikan setiap langkah ini secara lebih detail selanjutnya.



Gambar 6.4. Gambaran umum NeRF dan rendering.

Untuk membuat tampilan baru, sinar kamera dilewatkan melalui adegan untuk mendapatkan titik 3D. Titik-titik ini, bersama dengan arah pandang 2D-nya, dimasukkan ke dalam jaringan saraf untuk menghasilkan warna dan kepadatan. Kemudian, ini digabungkan menggunakan teknik rendering volume klasik menjadi gambar 2D.

Kode berikut mengilustrasikan MLP yang dapat digunakan untuk mengkonversi input 5D menjadi output RGB:

```
# Model architecture
```

```
def init_nerf_model(D=8, W=256, input_ch=3, input_ch_views=3,
output_ch=4, skips=[4], use_viewdirs=False):
```

```
    relu = tf.keras.layers.ReLU()
    def dense(W, act=relu): return tf.keras.layers.Dense(W,
activation=act)
```

```
print('MODEL', input_ch, input_ch_views, type(
    input_ch), type(input_ch_views), use_viewdirs)
input_ch = int(input_ch)
input_ch_views = int(input_ch_views)
```

```
inputs = tf.keras.Input(shape=(input_ch + input_ch_views))
inputs_pts, inputs_views = tf.split(inputs, [input_ch,
input_ch_views], -1)
inputs_pts.set_shape([None, input_ch])
inputs_views.set_shape([None, input_ch_views])
```

```
print(inputs.shape, inputs_pts.shape, inputs_views.shape)
outputs = inputs_pts
for i in range(D):
    outputs = dense(W)(outputs)
    if i in skips:
```

```
outputs = tf.concat([inputs_pts, outputs], -1)

if use_viewdirs:
    alpha_out = dense(1, act=None)(outputs)
    bottleneck = dense(256, act=None)(outputs)
    inputs_viewdirs = tf.concat(
        [bottleneck, inputs_views], -1) # concat viewdirs

outputs = inputs_viewdirs
# The supplement to the paper states there are 4 hidden
# layers here, but this is an error since
# the experiments were actually run with 1 hidden
# layer, so we will leave it as 1.
for i in range(1):
    outputs = dense(W//2)(outputs)
    outputs = dense(3, act=None)(outputs)
    outputs = tf.concat([outputs, alpha_out], -1)
else:
    outputs = dense(output_ch, act=None)(outputs)

model = tf.keras.Model(inputs=inputs, outputs=outputs)
return model
```

NeRF menghitung kerugian dengan membandingkan piksel gambar yang dirender dengan gambar kebenaran dasar. Dengan menggunakan proses ini, NeRF dapat membuat rekonstruksi 3D terperinci dari ruang fisik, yang dapat bermanfaat bagi robot yang menavigasi lingkungan yang kompleks. Selain itu, gambar fotorealistik yang dihasilkan oleh NeRF menangkap geometri dan penampilan objek, meningkatkan kemampuan robot untuk menafsirkan adegan dan menavigasi lingkungan sekitarnya secara akurat.

Pemetaan Semantik

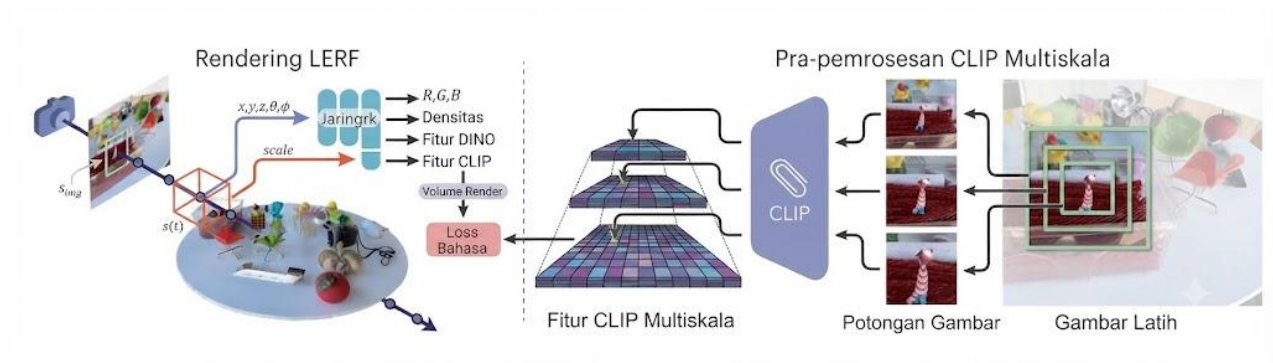
Pemetaan semantik adalah pembuatan peta sehingga objek dan fitur dalam suatu adegan diberi label dengan informasi yang menggambarkan adegan tersebut. Ini dapat mencakup identifikasi objek (mobil, anjing), atribut tentang objek (merah, tinggi) dan hubungan (kendaraan berada di sebelah pohon). Secara tradisional, pemetaan semantik didasarkan pada teknik kosakata tertutup, yang berarti menggunakan serangkaian label yang telah ditentukan sebelumnya. Karena model bahasa visi besar seperti CLIP, semantik semakin mendekati teknik kosakata terbuka. Hal ini memungkinkan pemahaman yang lebih fleksibel dan dinamis tentang adegan dan kemampuan untuk menghasilkan label berdasarkan kosakata yang dipelajari dari kumpulan data gambar-teks yang besar.

Contoh metode yang menggunakan pendekatan ini adalah *Language Embedded Radiance Fields* (LeRFs), yang menggunakan pendekatan neural untuk menggabungkan semantik kosakata terbuka, dan ConceptFusion, pendekatan eksplisit untuk pemetaan semantik.

Language Embedded Radiance Fields (LeRFs)

LeRF mengintegrasikan fitur semantik kosakata terbuka ke dalam rendering neural, memungkinkan representasi adegan 3D yang detail untuk dibuat (lihat Gambar 6.5). Langkah-langkah utama dalam LeRF adalah sebagai berikut:

- Gambar masukan diproses terlebih dahulu menjadi patch multiskala, yang merupakan bagian yang lebih kecil dari gambar yang diekstrak pada skala atau resolusi yang berbeda. Patch ini dilewatkan melalui model bahasa visi besar seperti CLIP untuk mendapatkan fitur CLIP pada skala yang berbeda.
- Model CLIP adalah kunci dalam mengekstrak informasi visual dan semantik dari patch untuk menghasilkan peta fitur pada skala yang berbeda.
- Jaringan saraf akan menerima koordinat spasial (x, y, z) , arah pandang (θ, ϕ) , dan skala sebagai input.
- Berbagai parameter dikeluarkan, seperti nilai RGB, kepadatan, dan fitur DINO, yang menangkap informasi visual dan semantik dari gambar input, dan fitur CLIP, yang menyediakan representasi yang menghubungkan konten visual dengan deskripsi teks.
- Fungsi kerugian menyelaraskan fitur yang dirender dengan informasi semantik dari CLIP. Fungsi ini penting untuk memastikan bahwa adegan 3D yang dihasilkan memiliki label semantik yang akurat.



Gambar 6.5. Rendering LeRF dan praproses CLIP multiskala.

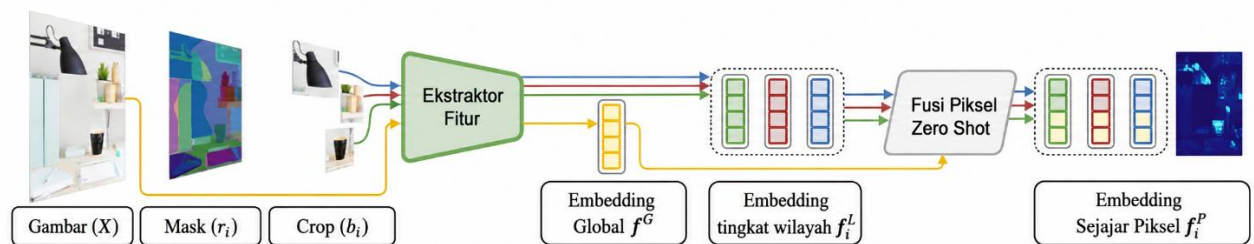
ConceptFusion

ConceptFusion menggunakan pendekatan eksplisit untuk pemetaan semantik yang menggabungkan berbagai tingkat penyematan visual untuk membuat penyematan yang sejajar piksel yang berisi informasi semantik. ConceptFusion memiliki beberapa langkah utama:

- Dengan menggunakan gambar masukan, masker dan potongan gambar dihasilkan. Masker mengidentifikasi objek yang berbeda dalam gambar, dan potongan gambar hanyalah bagian yang lebih kecil dari gambar yang sesuai dengan masker ini.
- Dua tingkat penyematan dihasilkan menggunakan ekstraktor fitur yang menerima gambar asli, masker, dan potongan gambar. Penyematan global mewakili konteks keseluruhan dari seluruh gambar, dan penyematan tingkat wilayah menangkap fitur yang lebih detail dari objek tertentu dalam gambar.

- Penyematan global dan tingkat wilayah digabungkan menggunakan fusi piksel zero-shot, yang menyelaraskan fitur-fiturnya pada tingkat piksel.
- Output dari fusi piksel zero-shot adalah embedding yang sejajar dengan piksel yang menangkap fitur visual dan informasi semantik gambar.

Dengan embedding yang dipelajari ini, Anda kemudian memiliki gambar dengan label yang menyediakan peta semantik dari seluruh adegan. Alur kerja ConceptFusion ditunjukkan pada Gambar 6.6.



Gambar 6.6. ConceptFusion menghasilkan fitur yang sejajar piksel.

6.4 LOKALISASI BERBASIS PEMBELAJARAN MENDALAM

Memahami lokasi robot sangat penting karena membantu robot melacak jalurnya dan menentukan tindakannya di masa depan. Pendekatan umum untuk mengetahui lokasi robot melibatkan penggunaan sensor seperti LiDAR dan odometri. Misalnya, mengintegrasikan data LiDAR dengan odometri dapat membantu melacak bagaimana posisi robot berubah dari waktu ke waktu, menciptakan peta lingkungan sekitar robot. Contoh sistem lokalisasi tersebut adalah *Simultaneous Localization and Mapping* (SLAM). Namun, SLAM dapat mengalami masalah ketika robot dipindahkan secara tidak terduga dan mengalami mode kegagalan karena ketergantungan pada odometri. Karena alasan ini, menggunakan data LiDAR atau kamera dengan metode pembelajaran mendalam merupakan arah lain yang menjanjikan.

Lokalisasi 2D-ke-2D

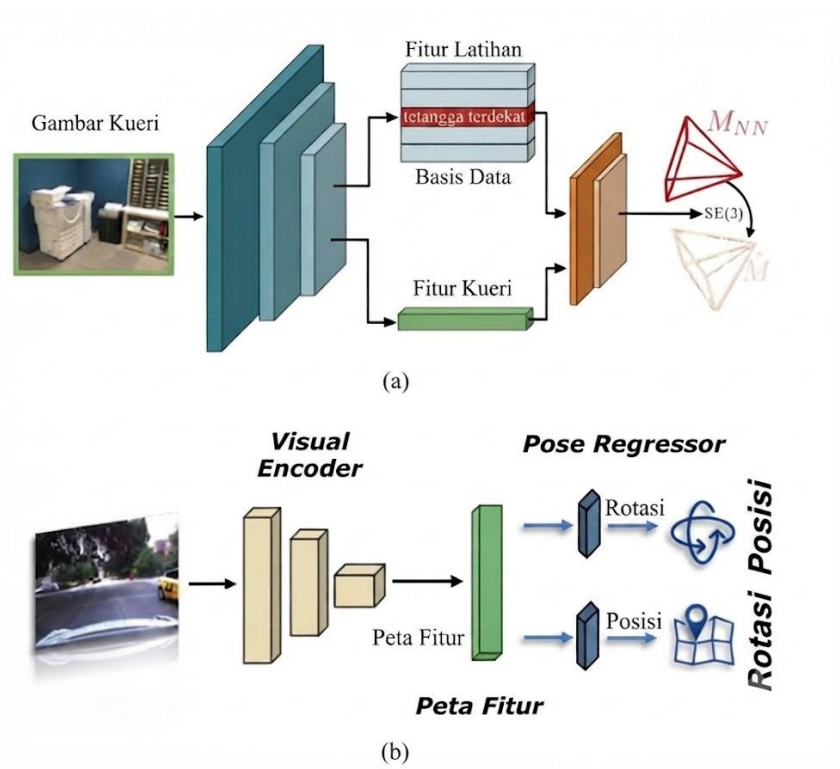
Lokalisasi 2D-ke-2D memperkirakan posisi kamera relatif terhadap peta 2D dengan mencocokkan gambar yang diambil oleh robot dengan fitur pada peta. Metode ini dapat menggunakan lokalisasi eksplisit, di mana adegan direpresentasikan oleh gambar yang diberi tag dengan koordinat geografis yang dapat langsung dibandingkan dan dicocokkan. Atau, lokalisasi implisit menggunakan jaringan saraf yang telah dilatih untuk memahami dan merepresentasikan adegan. Dalam hal ini, posisi kamera disimpulkan berdasarkan gambar yang diambil tanpa tag geografis eksplisit. Terdapat banyak metode lokalisasi 2D-ke-2D, dan meskipun kami tidak membahas semuanya di sini, Gambar 6.7 memberikan gambaran umum yang komprehensif untuk referensi lebih lanjut.

Gambar 6.8 menunjukkan arsitektur lokalisasi berbasis 2D-ke-2D yang umum menggunakan peta eksplisit dari RelocNet dan peta implisit dari PoseNet. Dalam kasus

lokalisasi berbasis peta eksplisit, gambar baru diambil, dan fitur diekstrak. Terdapat basis data fitur yang sudah ada yang diekstrak dari gambar pelatihan yang diberi tag dengan koordinat geografis.

	Model	Tahun	Agnostik	Performa (m/derajat)		Kontribusi
				7 Adegan	Cambridge	
Peta Eksplisit	NN-Net	2017	Ya	0.21/9.30	-	menggabungkan pencarian ulang dan estimasi pose relatif
	DeLS-3D	2018	Tidak	-	-	pelatihan bersama dengan semantik
	AnchorNet	2018	Ya	0.09/6.74	0.84/2.10	alokasi titik jangkar
	RelocNet	2018	Ya	0.21/6.73	-	kehilangan tumpang tindih frustum kamera
	CamNet	2019	Ya	0.04/1.69	-	pencarian gambar multi-tahap
	PixLoc	2021	Ya	0.03/0.98	0.15/0.25	lokalisasi kamera tetap sebagai pembelajaran metrik
Relokalisasi dalam Peta 2D	PoseNet	2015	Tidak	0.44/10.44	2.09/6.84	jaringan saraf tiruan pertama dalam regresi pose global
	Bayesian PoseNet	2016	Tidak	0.47/9.81	1.92/6.28	estimasi ketidakpastian Bayesian untuk pose global
	BranchNet	2017	Tidak	0.29/8.30	-	pembelajaran multi-tugas untuk orientasi dan translasi
	VidLoc	2017	Tidak	0.25/-	-	lokalisasi efisien dari urutan gambar
	Geometric PoseNet	2017	Tidak	0.23/8.12	1.63/2.86	kehilangan yang sadar geometri
	SVS-Pose	2017	Tidak	-	1.33/5.17	augmentasi data dalam ruang 3D
	LSTM PoseNet	2017	Tidak	0.31/9.85	1.30/5.52	korelasi spasial
	Hourglass PoseNet	2017	Tidak	0.23/9.53	-	arsitektur berbentuk jam pasir
	MapNet	2018	Tidak	0.21/7.77	1.63/3.64	menerapkan kendala spasial dan temporal
	SPP-Net	2018	Tidak	0.18/6.20	1.24/2.68	augmentasi data sintesis
	GPoseNet	2018	Tidak	0.30/9.90	2.00/4.60	model hibrida dengan Regresor Proses Gaussian
	LSG	2019	Tidak	0.19/7.47	-	lokalisasi berbasis odometri
	PVL	2019	Tidak	-	1.60/4.21	masker dropout model sebelumnya untuk meningkatkan ketahanan
	AdPR	2019	Tidak	0.22/8.8	-	arsitektur adversarial
	AtLoc	2019	Tidak	0.20/7.56	-	korelasi spasial berbasis perhatian
	GR-Net	2020	Tidak	0.19/6.33	1.12/2.40	membangun grafik pandangan
	MS-Transformer	2021	Ya	0.18/7.28	1.28/2.73	memperluas ke beberapa adegan dengan transformer

Gambar 6.7. Model yang digunakan untuk lokalisasi 2D-ke-2D.

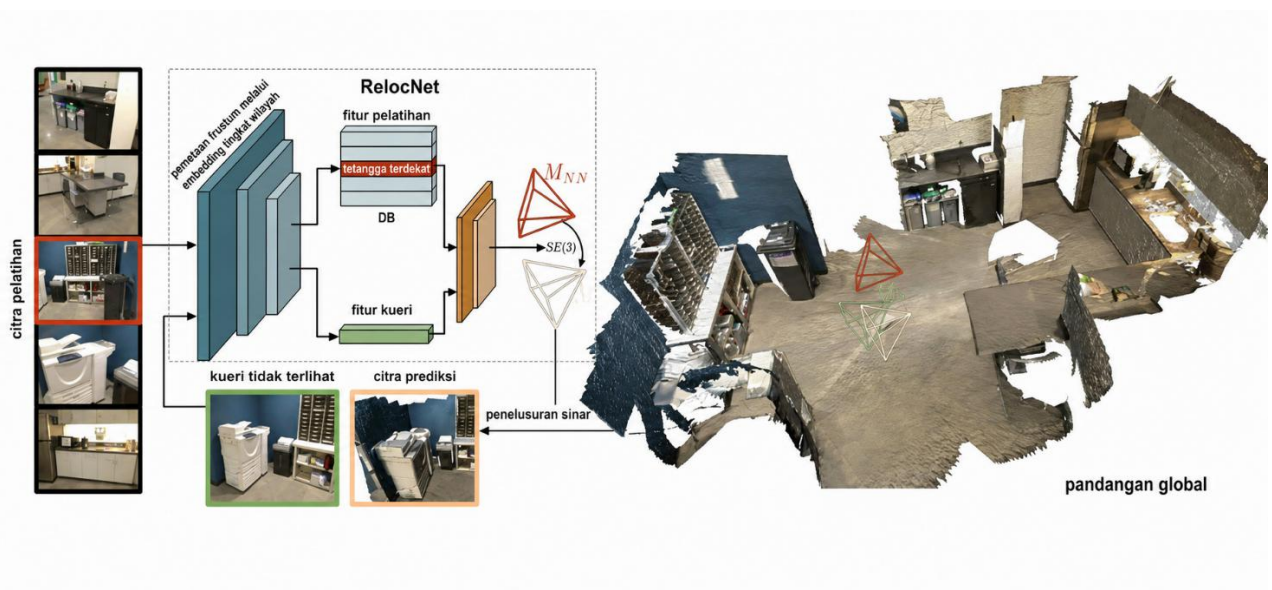


Gambar 6.8. Perbedaan antara lokalisasi berbasis peta eksplisit (a) RelocNet (https://www.robots.ox.ac.uk/~lav/Papers/balntas_etal_eccv2018/balntas_etal_eccv2018.pdf) dan implisit (b) PoseNet (<https://arxiv.org/abs/1505.07427>).

Fitur dari gambar yang diambil dibandingkan dengan fitur dalam basis data untuk menemukan tetangga terdekat. Posisi dan orientasi kamera ditentukan dengan mengidentifikasi fitur yang paling cocok dalam basis data dan kemudian menerapkan transformasi koordinat untuk memetakan fitur-fitur ini ke pose kamera.

Di sisi lain, lokalisasi berbasis peta implisit memiliki modul jaringan saraf lain yang mengambil peta fitur dari gambar masukan dan langsung memprediksi posisi dan rotasi kamera. Seluruh alur kerja dilatih secara end-to-end untuk langsung memetakan fitur gambar ke pose kamera. RelocNet (lihat Gambar 6.9) menentukan posisi kamera dengan membandingkan gambar. Ia belajar merepresentasikan gambar sehingga gambar yang serupa memiliki representasi yang serupa menggunakan pencocokan tetangga terdekat dan deskriptor fitur berbasis pembelajaran metrik kontinu.

Metode ini menggabungkan informasi dari bagian gambar yang tumpang tindih (tumpang tindih frusta kamera) untuk meningkatkan akurasi. Selain itu, RelocNet menyertakan pose regressor yang menyempurnakan estimasi posisi menggunakan informasi geometris, sehingga prediksi lokasi menjadi lebih akurat.



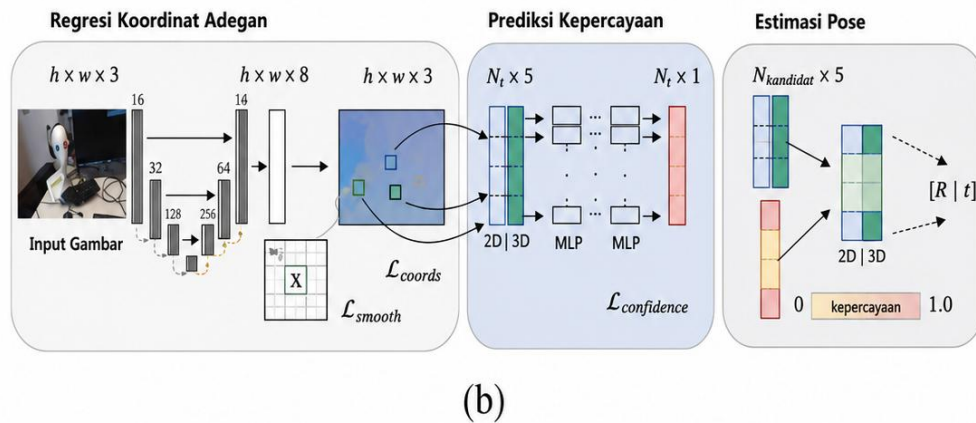
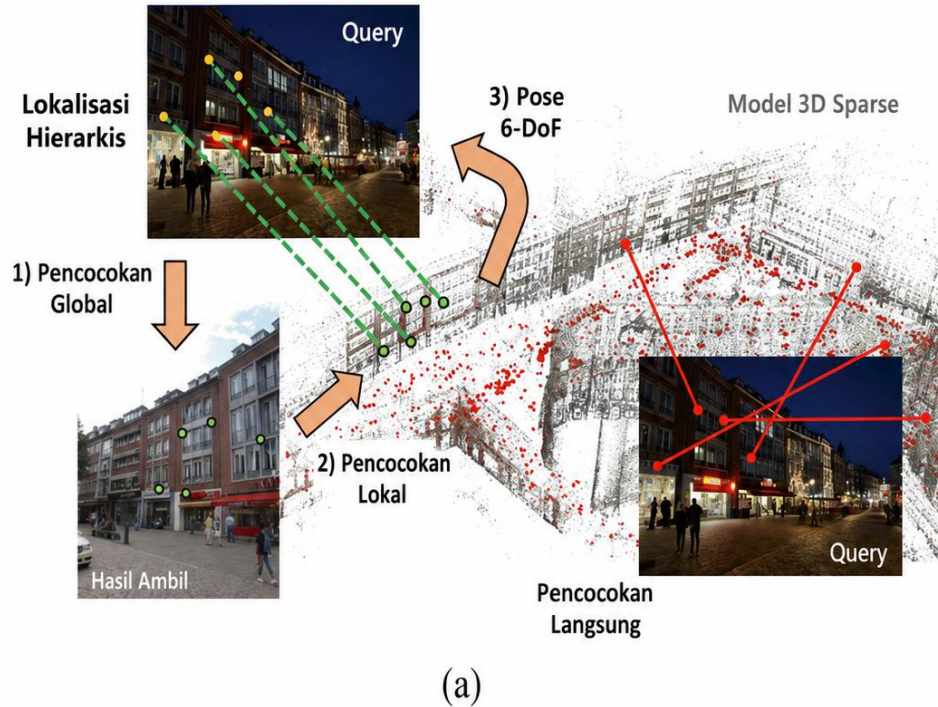
Gambar 6.10. Gambaran Umum RelocNet.

Lokalisasi 2D-ke-3D

Dua metode lokalisasi 2D-ke-3D yang umum adalah pencocokan deskriptor dan regresi koordinat adegan, seperti yang ditunjukkan pada Gambar 6.10. Pencocokan deskriptor menentukan posisi kamera dengan membandingkan fitur tingkat tinggi dan detail yang lebih halus dari gambar dengan basis data gambar yang telah disimpan sebelumnya untuk menyempurnakan pencocokan dan menghitung posisi dan orientasi yang tepat.

Sebaliknya, regresi koordinat adegan menggunakan jaringan saraf untuk memprediksi koordinat 3D untuk setiap piksel dalam gambar masukan, menilai kepercayaan prediksi ini, dan

kemudian menentukan pose kamera berdasarkan koordinat ini. Kedua metode tersebut bertujuan untuk menentukan lokasi kamera tetapi menggunakan teknik yang berbeda.



Gambar 6.10. Perbedaan antara pencocokan deskriptor (a), HF-Net (<https://arxiv.org/pdf/1812.03506>), dan regresi koordinat adegan (b), Confidence SCR (<https://arxiv.org/abs/1805.08443>).

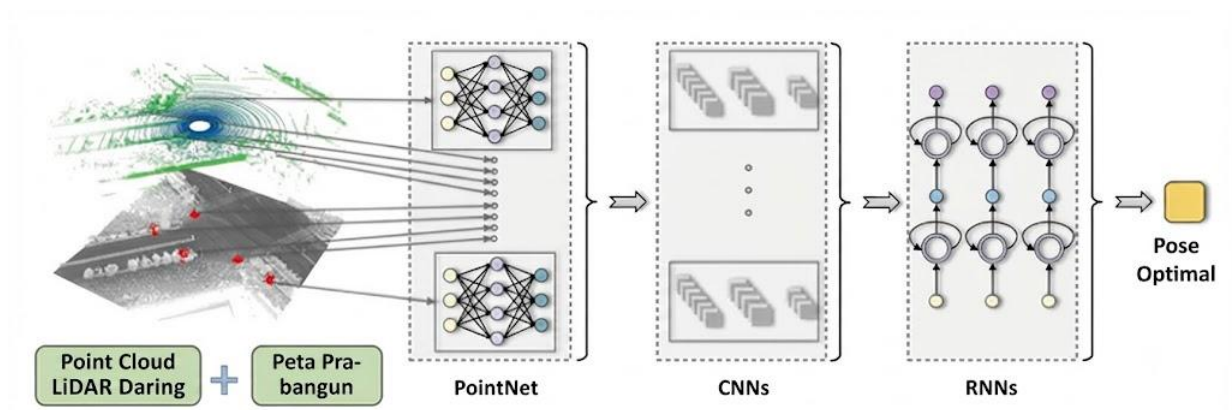
Ringkasan metode lokalisasi 2D-ke-3D umum ditunjukkan pada Gambar 6.11.

		Model	Tahun	Agnostik	Performa (m/derajat)		Kontribusi
					7Scenes	Cambridge	
Relokalisasi dalam Peta 3D	Berbasis Deskriptor	NetVLAD	2016	Ya	-	-	lapisan VLAD yang dapat dibedakan
		DELF	2017	Ya	-	-	pendeskriptor fitur lokal berbasis perhatian
		InLoc	2018	Ya	0.04/1.38	0.31/0.73	asosiasi data yang padat
		SVL	2018	Tidak	-	-	memanfaatkan model generatif untuk pembelajaran deskriptor
		SuperPoint	2018	Ya	-	-	bersama-sama mengekstrak titik minat dan deskriptor
		Sarlin et al.	2018	Ya	-	-	lokalisasi hierarkis
		NC-Net	2018	Ya	-	-	konsensus lingkungan sekitar
		2D3D-MatchNet	2019	Ya	-	-	bersama-sama mempelajari deskriptor untuk keypoint 2D dan 3D
		HF-Net	2019	Ya	0.042/1.3	0.356/0.31	lokalisasi kasar-ke-halus
		D2-Net	2019	Ya	-	-	bersama-sama mempelajari keypoint dan deskriptor
		Speciale et al.	2019	Tidak	-	-	lokalisasi yang menjaga privasi
		O2D-Net	2019	Tidak	-	-	anotasi objek-minat
		Camposceco et al.	2019	Ya	-	0.56/0.66	kompresi pemandangan hibrida untuk lokalisasi
		Cheng et al.	2019	Ya	-	-	penyaringan paralel bertingkat
		Taira et al.	2019	Ya	-	-	analisis komprehensif terhadap verifikasi pose
		R2D2	2019	Ya	-	-	mempelajari prediktor diskriminativitas deskriptor
		ASLFeat	2020	Ya	-	-	memanfaatkan jaringan konvolusional yang dapat dideformasi
		CD-VLM	2021	Ya	-	-	pencocokan cross-deskriptor
		VS-Net	2021	Tidak	0.024/0.8	0.136/0.24	pemungutan suara berdasarkan segmentasi
Koordinat Pemandangan	Regresi	DSAC	2017	Tidak	0.20/6.3	0.32/0.78	RANSAC yang dapat dibedakan
		DSAC++	2018	Tidak	0.08/2.40	0.19/0.50	tanpa menggunakan model 3D dari pemandangan
		Angle DSAC++	2018	Tidak	0.06/1.47	0.17/0.50	kehilangan reproyeksi berbasis sudut
		Dense SCR	2018	Tidak	0.04/1.4	-	regresi koordinat pemandangan penuh
		Confidence SCR	2018	Tidak	0.06/3.1	-	ketidakpastian model korespondensi
	Pemandangan	ESAC	2019	Tidak	0.034/1.50	-	integrasi DSAC dalam Mixture of Experts
		NG-RANSAC	2019	Tidak	-	0.24/0.30	pencarian hipotesis model berbasis panduan sebelumnya
		SANet	2019	Ya	0.05/1.68	0.23/0.53	arsitektur agnostik pemandangan untuk lokalisasi kamera
		MV-SCR	2019	Tidak	0.05/1.63	0.17/0.40	kendala konsistensi multi-tampilan
		HSC-Net	2020	Tidak	0.03/0.90	0.13/0.30	pembuatan koordinat pemandangan hierarkis
		KFNet	2020	Tidak	0.03/0.88	0.13/0.30	memperluas masalah ke domain waktu
		DSM	2021	Ya	0.027/0.92	0.27/0.52	prediksi koordinat yang padat

Gambar 6.11. Model lokalisasi 2D-ke-3D.

Lokalisasi 3D-ke-3D

Lokalisasi 3D-ke-3D, juga dikenal sebagai lokalisasi LIDAR, melibatkan penentuan posisi dan orientasi kamera dengan mencocokkan awan titik 3D yang dihasilkan oleh LiDAR dengan peta 3D lingkungan yang sudah ada. Proses ini biasanya berarti membandingkan gambar 3D yang diambil dengan peta yang tersimpan untuk menemukan kecocokan terbaik. Ini dapat membantu untuk lingkungan yang lebih kompleks di mana pemetaan 2D mungkin kurang memadai.



Gambar 6.12. Proses lokalisasi 3D-ke-3D.

Proses lokalisasi 3D-ke-3D digambarkan pada Gambar 6.12. Dengan menggunakan awan titik LiDAR dan peta yang sudah ada, PointNet mengekstrak fitur yang relevan. Fitur-fitur ini kemudian dimasukkan ke dalam CNN untuk ekstraksi fitur lebih lanjut. Terakhir, data yang diproses dilewatkan melalui Jaringan Neural Berulang (RNN) untuk menyempurnakan estimasi pose, mempelajari hubungan temporal, dan menghasilkan pose kamera yang optimal. Gambar 6.13 menunjukkan ringkasan metode lokalisasi 3D-ke-3D lainnya.

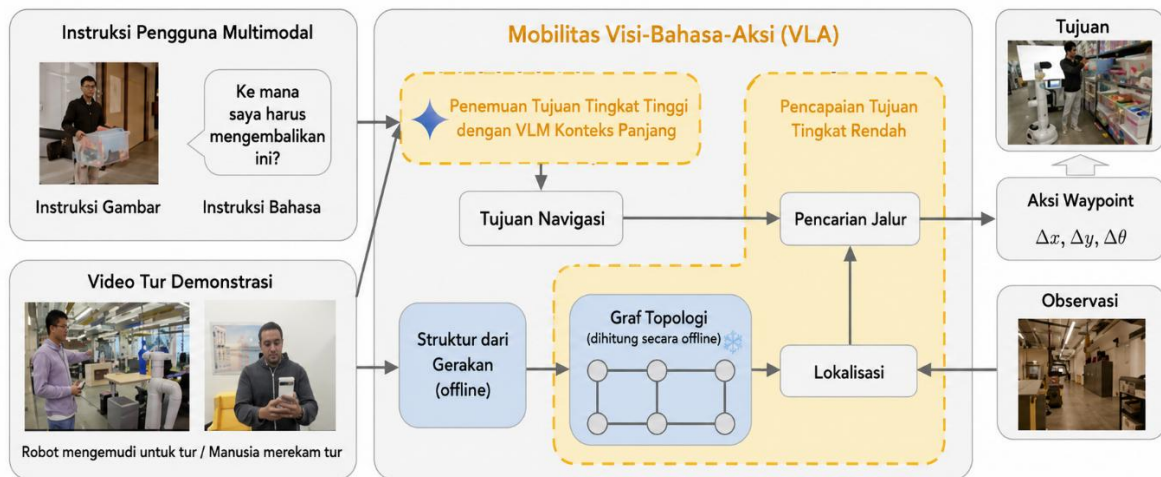
Model	Agnostik	Kontribusi
LocNet	Tidak	mengonversi titik 3D menjadi matriks 2D, pencarian di peta sebelumnya secara global
PointNetVLAD	Ya	mempelajari deskriptor global dari point cloud
Barsan et al.	Tidak	mempelajari dari peta intensitas LiDAR dan point cloud online
L3-Net	Tidak	mengeksrak fitur oleh PointNet
PCAN	Ya	memprediksi signifikansi setiap titik lokal berdasarkan konteks
DeepJCP	Ya	menghasilkan korespondensi pencocokan yang sesuai dari probabilitas pencocokan yang dipelajari
DCP	Ya	pembelajaran berdasarkan titik terdekat secara iteratif
D3Feat	Ya	belajar bersama detektor dan deskriptor untuk titik 3D

Gambar 6.13. Model lokalisasi 3D-ke-3D.

Kami merekomendasikan untuk memeriksa makalah ini, yang kami rujuk dalam bab ini, untuk bacaan lebih lanjut.

6.5 NAVIGASI DAN LOKOMOSI CERDAS PADA ROBOT

Model bahasa visi (VLM) adalah model besar yang dapat memahami dan memproses data teks dan gambar. Gemini 1.5 Pro milik Google adalah contohnya, dan menggunakan arsitektur Mixture-of-Experts untuk memproses konteks yang panjang. Ia dapat menangani hingga 1 juta token dalam penggunaan standar dan hingga 10 juta token dalam pengaturan studi. Ini setara dengan sekitar satu jam video, sebelas jam audio, dan lebih dari 30.000 baris kode. Penggunaan VLM dapat sangat bermanfaat bagi robotika dalam menyelesaikan tugas navigasi.



Gambar 6.14. Arsitektur Mobility VLA.

Secara spesifik, Navigasi Instruksi Multimodal dengan Tur Demonstrasi (MINT) adalah jenis tugas navigasi yang menggunakan demonstrasi video yang direkam untuk memandu navigasi robot. Contoh metode ini adalah Mobility VLA, yang menggunakan video tur demonstrasi lingkungan dan instruksi pengguna multimodal (gambar dan bahasa).

Sistem ini memiliki dua bagian utama: kebijakan tingkat tinggi, yang menggunakan video dan instruksi untuk mengidentifikasi tujuan, dan kebijakan tingkat rendah, yang menggunakan tujuan ini bersama dengan peta topologi yang telah dihitung sebelumnya untuk merencanakan jalur dan tindakan robot. Gambaran umum Mobility VLA ditunjukkan pada Gambar 6.14. Arsitektur Mobility VLA terdiri dari bagian-bagian utama berikut:

- **Instruksi pengguna multimodal:** Pengguna memberikan instruksi dalam bentuk gambar lingkungan atau tugas yang perlu dilakukan. Misalnya, gambar rak tempat suatu barang perlu diletakkan. Instruksi bahasa terdiri dari instruksi teks yang menjelaskan tugas tersebut. Misalnya, teks yang mengatakan "Letakkan kotak ini di rak paling atas."
- **Video tur demonstrasi:** Video yang direkam di mana manusia atau robot menavigasi lingkungan dan lokasi serta tindakan penting direkam. Misalnya, video yang menunjukkan seseorang berjalan melalui kantor dan menunjukkan di mana meja, rak, dan pintu berada. Ini berfungsi sebagai referensi bagi robot untuk memahami tata letak dan fitur penting lingkungan.

Sebuah perintah (F, N, d, I) disiapkan, yang mencakup bingkai (gambar dari video tur), narasi (deskripsi yang terkait dengan setiap bingkai), jarak (informasi kontekstual tentang jarak atau lokasi), dan instruksi pengguna multimodal yang menggabungkan teks dan gambar. Contohnya ditunjukkan pada Gambar 6.15.

Anda adalah robot yang beroperasi di dalam gedung dan tugas Anda adalah menanggapi perintah pengguna tentang pergi ke lokasi tertentu dengan menemukan bingkai terdekat dalam video tur untuk dinavigasi.
Bingkai-bingkai ini berasal dari tur gedung tahun lalu.
[Bingkai 1 Gambar f_1]
Bingkai 1. [Narasi bingkai n_1]
...
[Bingkai k Gambar f_k]
Bingkai k. [Narasi bingkai n_k]
Gambar ini adalah apa yang Anda lihat sekarang. Anda mungkin atau mungkin tidak melihat pengguna dalam gambar ini.
[Instruksi Gambar I]
Pengguna berkata: Di mana saya harus mengembalikan ini?
Bagaimana Anda akan menanggapi? Bisakah Anda menemukan bingkai terdekat?

Gambar 6.15. Demonstrasi bagaimana perintah multimodal digunakan oleh Model Bahasa Visi (VLM) untuk memandu robot dalam tujuan navigasi.

- **Pencarian tujuan:** VLM mengidentifikasi frame spesifik dalam video yang sesuai dengan tujuan yang dijelaskan oleh instruksi masukan. Misalnya, VLM dapat mengidentifikasi frame yang menunjukkan rak tempat kotak perlu diletakkan. Ini memastikan bahwa robot memahami apa yang perlu dicapainya berdasarkan konteks dari instruksi dan video.
- **Tujuan navigasi:** Frame tujuan tingkat tinggi diidentifikasi oleh VLM yang sesuai dengan instruksi pengguna. Tujuan ini bisa berupa mencapai rak yang diidentifikasi dalam video.
- **Pencarian jalur:** Robot menggunakan peta yang telah dihitung sebelumnya, yang dibuat dari video tur demonstrasi menggunakan metode yang diklasifikasikan sebagai struktur-dari-gerakan, yang menciptakan representasi 3D dari lingkungan. Dengan menggunakan peta ini, robot menemukan jalur optimal untuk mencapai tujuan navigasi.

Di sinilah kebijakan pencapaian tujuan tingkat rendah dapat digunakan. Sebagai masukan, bingkai spesifik yang diidentifikasi oleh kebijakan tingkat tinggi dan peta yang telah dibuat sebelumnya digunakan. Robot menangkap gambar baru dari pandangannya saat ini dan kemudian, dengan menggunakan ini, robot menentukan posisinya dalam grafik topologi. Robot memeriksa apakah telah mencapai simpul tujuan dan jika ya, tujuan navigasi tercapai. Jika tidak, robot menghitung jalur terpendek dari posisinya saat ini ke tujuan. Robot menghitung tindakan selanjutnya yang diperlukan untuk bergerak dari posisinya saat ini menuju simpul pertama di jalur tersebut dan kemudian robot melakukan tindakan yang telah dihitung. Ini berlanjut hingga robot mencapai tujuan atau jumlah langkah maksimum terlampaui. Lihat Gambar 6.16.

Algoritma 1 Kebijakan Pencapaian Tujuan Tingkat Rendah

```
1: Input: indeks frame tujuan  $g$ , graf topologi  $G$  yang dibangun secara offline.  
2:  
3: selama timestep  $\leq$  langkah maksimum lakukan  
4:   Dapatkan gambar observasi kamera baru  $O$   
5:   Dapatkan posisi robot  $v_t$  dan pose robot  $T$  dengan melokalisasi  $O$  di  $G$   
6:   jika  $v_t == v_g$  maka  
7:     Tujuan navigasi tercapai, hentikan  
8:   end if  
9:   Hitung  $S = [v_s, v_1, \dots, v_g]$ , jalur terpendek antara  $v_s$  dan  $v_g$ .  
10:  Hitung aksi waypoint  $a$  dari pose relatif antara  $T$  dan  $v_1$   
11:  Eksekusi aksi pada robot  
12: end while
```

Gambar 6.16. Kebijakan tingkat rendah ini menghitung tindakan titik arah pada setiap langkah waktu.

Metode ini menunjukkan kekuatan penggunaan VLM dan memanfaatkan kemampuannya untuk menangkap informasi kontekstual panjang untuk memahami lingkungan dan instruksi yang kompleks. Menggunakan VLM untuk memproses input multimodal dapat membantu robot mendapatkan pemahaman yang lebih bernuansa tentang tugas dan lingkungan sekitarnya.

Navigasi dan Eksplorasi

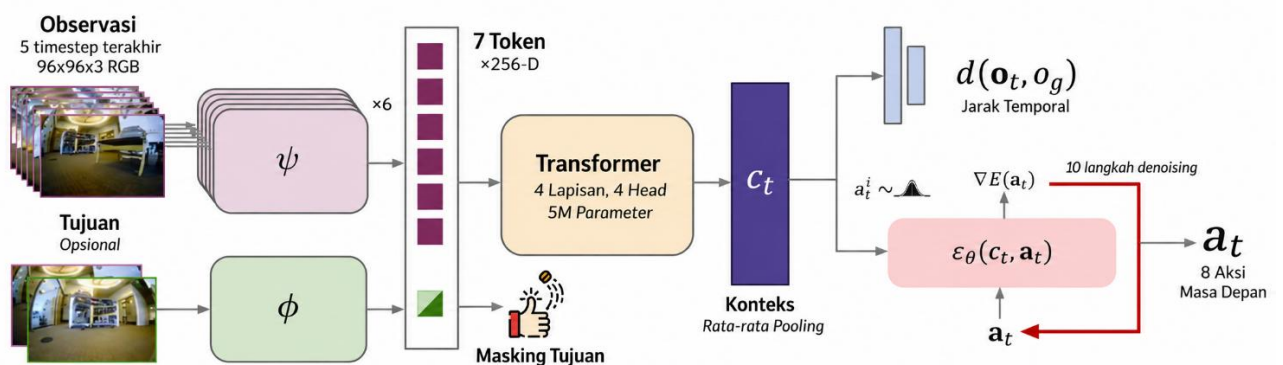
Menavigasi lingkungan baru merupakan tantangan bagi robot karena mereka perlu dapat mencapai tujuan spesifik yang telah ditentukan dan mencari tujuan di lingkungan baru. Tugas-tugas ini biasanya didefinisikan secara terpisah dan diselesaikan menggunakan model yang berbeda. NoMaD adalah teknik yang bertujuan untuk membangun pendekatan terpadu yang dapat menyelesaikan kedua tugas dan meningkatkan kinerja navigasi secara keseluruhan. Lihat Gambar 6.17.

NoMaD menggunakan kebijakan difusi tunggal yang menangani navigasi yang berorientasi pada tujuan dan navigasi eksploratif:

- Robot menerima data visual dari lingkungannya dalam beberapa saat terakhir. Secara opsional, robot dapat memiliki input tujuan visual yang menunjukkan lokasi tujuan, yang menunjukkan kepada robot ke mana ia harus pergi.
- Dua encoder EfficientNet digunakan untuk memproses data visual dan input tujuan dan mengubahnya menjadi token yang dapat digunakan oleh decoder Transformer.
- Token tersebut diteruskan ke model Transformer, yang menggunakan token tersebut untuk memahami konteks dan membuat keputusan tentang tindakan robot. Transformer menghasilkan vektor konteks dengan mengambil informasi dari token input. Vektor konteks ini mewakili pemahaman saat ini tentang lingkungan dan tugas.
- Masking tujuan adalah metode yang digunakan untuk membantu model memahami apakah ia harus fokus pada navigasi yang berorientasi pada tujuan atau navigasi eksploratif. Ini menggabungkan informasi dari input tujuan dengan konteks yang

dihasilkan dari pengamatan untuk memastikan model dapat beradaptasi berdasarkan tugas yang diberikan.

- Vektor konteks digunakan dalam model difusi aksi, yang memprediksi tindakan masa depan yang harus diambil robot. Ia menggunakan proses penghilangan derau dengan sepuluh langkah untuk menyempurnakan prediksi tindakan ini. Sepanjang proses ini, model menghitung jarak temporal antara jarak saat ini dan pengamatan tujuan (jika disediakan) untuk memahami seberapa jauh robot dari tujuan dalam hal waktu.
- Model difusi aksi menghasilkan urutan tindakan masa depan yang memandu pergerakan robot untuk menavigasi lingkungan, baik itu mencari tujuan atau bergerak menuju tujuan yang diketahui.



Gambar 6.17. Arsitektur NoMaD.

Model ini diuji di lingkungan dalam dan luar ruangan menggunakan platform LoCoBot. Dalam mode eksplorasi, robot mengidentifikasi jalur potensial untuk menemukan tujuan. Dalam mode berorientasi tujuan, ia menavigasi langsung menuju target yang ditentukan. Ini menunjukkan kemampuan pendekatan seperti ini untuk memahami lingkungannya, mengikuti rute navigasi, dan membuat keputusan yang tepat di persimpangan. Lihat Gambar 6.18.



Gambar 6.18. Peluncuran NoMaD di lingkungan dalam ruangan (atas) dan luar ruangan (bawah). Garis kuning adalah sampel tindakan masa depan dari mode tak terarah dan garis biru menunjukkan tindakan yang dipilih oleh perencana tingkat tinggi.

Lokomosi untuk Robot Berkaki

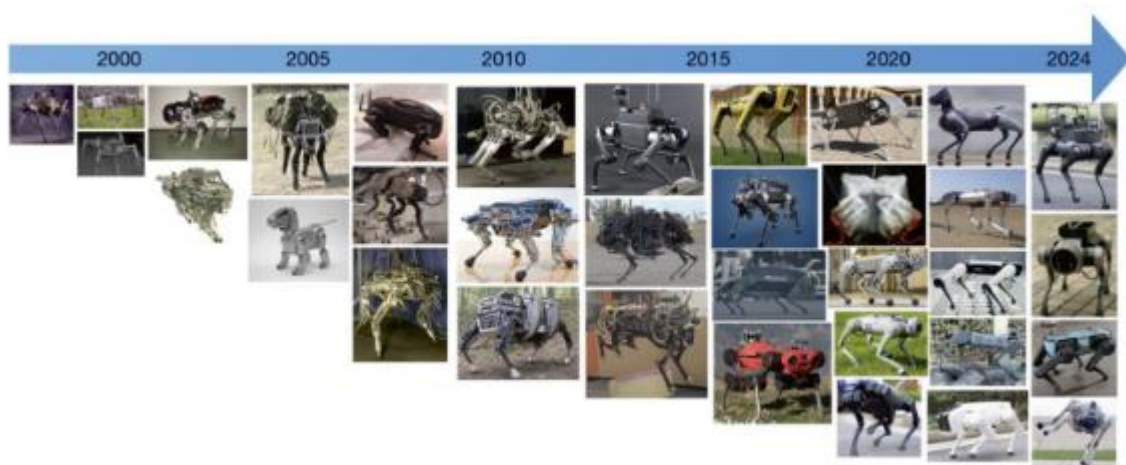
Area pengembangan penting dalam studi robotika adalah lokomosi berkaki—ketika robot menggunakan kaki untuk bergerak dan menavigasi lingkungannya. Robot ini bermanfaat untuk inspeksi dan eksplorasi, di mana mereka harus menavigasi medan yang menantang dan bermanuver di sekitar rintangan. Baru-baru ini, perkembangan untuk menurunkan biaya perangkat keras, membuat simulasi lebih realistis, dan mengembangkan metode berbasis pembelajaran telah mendorong studi di bidang ini.

Perangkat Keras

Selama bertahun-tahun, perangkat keras untuk robot berkaki empat telah berevolusi untuk memiliki peningkatan kinerja dan pengurangan biaya. Ada tiga jenis aktuasi: hidrolik, motor listrik, dan aktuator elastis seri.

- **Aktuator hidrolik:** Aktuator ini ideal untuk gerakan yang kuat dan dinamis karena memiliki rasio daya terhadap berat yang tinggi. Terlepas dari itu, aktuator ini bisa mahal dan membutuhkan perawatan khusus.
- **Motor listrik:** Aktuator ini paling umum digunakan pada robot berkaki empat. Di masa lalu, aktuator ini digunakan bersama komponen lain, seperti roda gigi rasio tinggi untuk meningkatkan torsi. Baru-baru ini, aktuator dengan kepadatan torsi tinggi telah dikembangkan yang memungkinkan kontrol sendi langsung dengan lebih sedikit atau tanpa komponen tambahan. Perkembangan ini telah mengurangi kompleksitas dan meningkatkan kinerja.
- **Aktuator elastis seri:** Aktuator ini memiliki komponen elastis yang menyerap benturan dan memungkinkan kontrol yang lebih presisi.

Baru-baru ini, bidang ini telah berfokus pada aktuator proprioseptif, yang menawarkan kepadatan torsi dan kontrol yang lebih tinggi. Aktuator ini menggunakan sensor untuk mengukur gaya dan posisi sendi, yang memungkinkan kontrol menjadi lebih adaptif. Aktuator ini pada akhirnya akan memungkinkan robot berkaki empat untuk melakukan tugas yang lebih kompleks. Gambar 6.19 menunjukkan bagaimana robot berkaki empat telah berevolusi dari waktu ke waktu.



Gambar 6.19. Bagaimana robot berkaki empat berevolusi dari waktu ke waktu.

Simulasi

Simulasi sangat penting untuk menguji algoritma kontrol untuk robot berkaki empat sebelum menerapkannya di dunia nyata. Simulasi terbaru menggunakan model benda kaku, yang dapat menangani tabrakan dengan lebih akurat dengan mempertimbangkan gesekan, interaksi elastis (dua objek bertabrakan dan memantul satu sama lain tanpa kehilangan energi dalam sistem), dan interaksi inelastis (ketika dua objek bertabrakan, mereka dapat berubah bentuk, saling menempel, atau kehilangan sebagian energi kinetik). Akselerasi GPU juga telah mempercepat simulasi dan proses pelatihan untuk tugas yang lebih kompleks menggunakan campuran data dunia nyata dari input sensor dan data simulasi.

Formulasi MDP

Proses Keputusan Markov (MDP) digunakan dalam RL sebagai kerangka kerja untuk pengambilan keputusan. Untuk pergerakan pada robot berkaki, MDP membantu merancang kebijakan kontrol. Dinamika tersebut menggambarkan bagaimana keadaan robot (posisi, kecepatan, dll.) berubah seiring waktu. Keadaan tersebut berevolusi berdasarkan keadaan robot saat ini dan tindakan yang diambilnya. Hal ini dapat direpresentasikan sebagai berikut:

$$D \equiv (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$$

Persamaan 6.1

Persamaan ini merepresentasikan keadaan robot pada langkah waktu berikutnya, yang ditentukan oleh keadaan saat ini, tindakan yang diambil, dan perubahan selama langkah waktu kecil (dt). Perubahan keadaan akibat tindakan dimodelkan menggunakan fungsi probabilitas:

$$p(s_{t+1}|s_t, a_t)$$

Persamaan 6.2

Ini memberikan kemungkinan berpindah ke keadaan berikutnya dari keadaan saat ini setelah mengambil suatu tindakan.

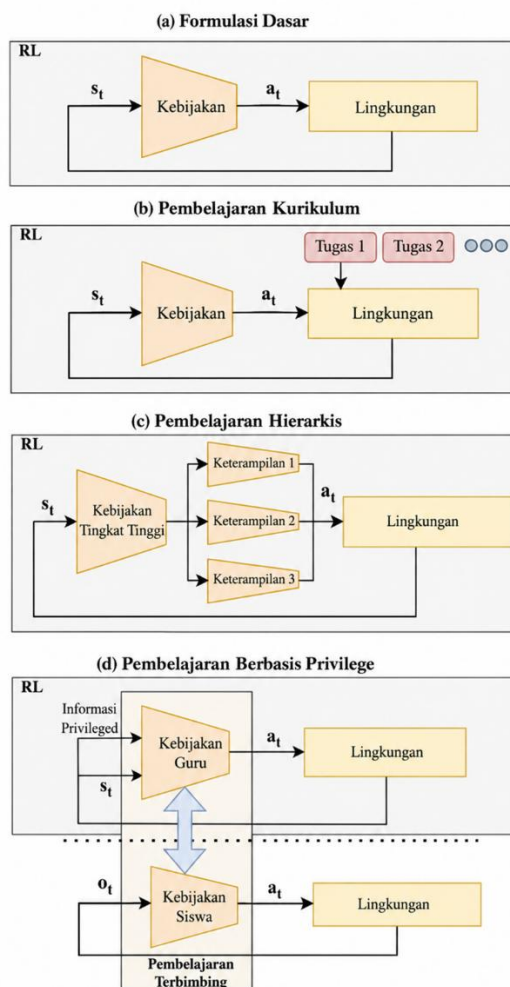
Metode Berbasis Pembelajaran

Beberapa metode berbasis pembelajaran umum yang digunakan untuk menyelesaikan MDP untuk kebijakan lokomosi ditunjukkan pada Gambar 6.20.

- **Pembelajaran ujung-ke-ujung:** Robot belajar cara bergerak dengan memperlakukan seluruh tugas sebagai satu masalah besar, yang sering dirumuskan sebagai masalah kontrol oleh MDP. Seringkali, algoritma RL seperti TRPO (*Trust Region Policy Optimization*) dan PPO (*Proximal Policy Optimization*), yang dijelaskan dalam Bab 7, digunakan untuk menyelesaikan hal ini. Intuisinya adalah bahwa algoritma ini membantu robot belajar dengan melakukan pembaruan kecil dan aman dalam pergerakan.
- **Pembelajaran kurikulum:** Tujuan pembelajaran kurikulum adalah untuk meniru cara manusia belajar dengan perlahan meningkatkan kesulitan tugas dari waktu ke waktu. Metode ini membantu robot membangun keterampilan dari waktu ke waktu, yang

dapat menjadikannya teknik yang bermanfaat di lingkungan yang kompleks di mana pembelajaran secara perlahan dengan peningkatan kesulitan tugas dapat berguna.

- **Pembelajaran hierarkis:** Tugas-tugas kompleks dipecah menjadi tugas-tugas yang lebih kecil dan lebih sederhana. Kebijakan tingkat tinggi memutuskan tindakan yang lebih besar, seperti memilih ke mana harus melangkah, dan kebijakan tingkat rendah menangani detail spesifik seperti menggerakkan kaki robot. Hal ini dapat mempermudah penanganan perilaku yang rumit dengan memfokuskan pada tindakan spesifik satu bagian pada satu waktu.
- **Pembelajaran istimewa:** Kebijakan "guru" dilatih dalam simulasi di mana ia memiliki akses penuh ke semua informasi yang dibutuhkan. Kebijakan ini membimbing kebijakan "siswa", yang beroperasi dengan informasi yang lebih terbatas, mirip dengan kondisi dunia nyata. Kebijakan "siswa" belajar untuk memanfaatkan informasi yang dimilikinya dengan menginterupsi riwayat pengamatan untuk membuat keputusan. Pendekatan ini dapat membantu menjembatani kesenjangan antara simulasi dan dunia nyata dengan membantu robot melakukan tugas di dunia nyata menggunakan model simulasi yang sepenuhnya informatif.



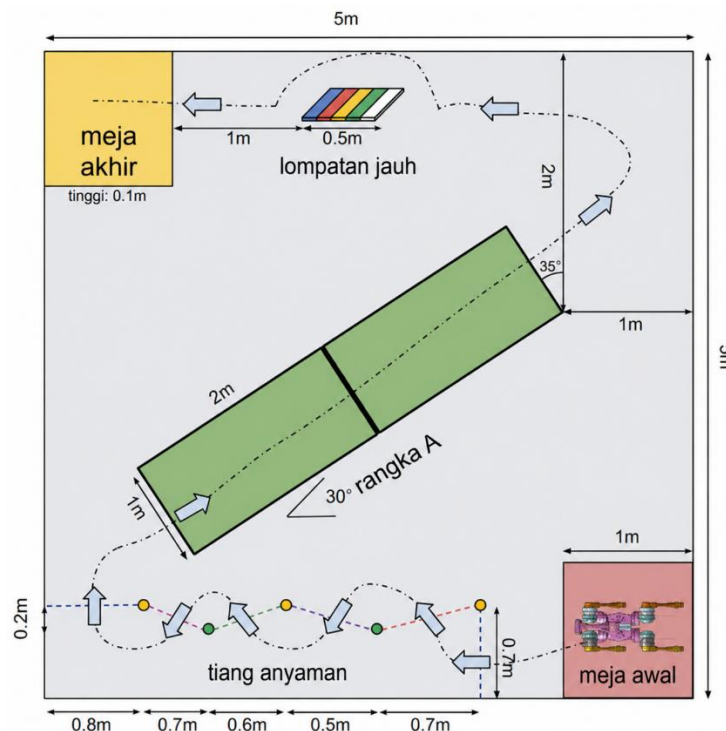
Gambar 6.20. Kerangka kerja pembelajaran populer.

Arah masa depan adalah menggunakan model dasar, yaitu model besar yang telah dilatih pada sejumlah besar data yang beragam. Untuk robot berkaki, model dasar membantu robot menafsirkan lingkungannya dan merencanakan urutan tindakan. Penyempurnaan model dasar dengan data spesifik robot, seperti untuk tugas manipulasi, juga dapat menjadi arah yang berguna untuk mengontrol tindakan robot secara langsung.

Salah satu tantangan dalam pergerakan robot berkaki adalah kebutuhan akan tolok ukur untuk mengevaluasi kebijakan kontrol dan perangkat keras. Tolok ukur Barkour bertujuan untuk menyelesaikan hal ini dengan menyediakan jalur rintangan standar untuk mengukur kelincihan dan kinerja robot berkaki (lihat Gambar 6.21). Jalur ini dirancang berdasarkan kompetisi kelincihan anjing dan mencakup empat rintangan utama:

- Meja jeda, yaitu platform tempat robot harus berhenti dan mempertahankan posisinya selama waktu yang ditentukan.
- Rintangan berupa tiang-tiang yang harus dilewati robot. Ini secara langsung menguji kemampuan robot untuk melakukan gerakan yang presisi dan lincah.
- Rintangan berbentuk A, yaitu tanjakan curam yang harus dinaiki dan dituruni robot, yang memungkinkan evaluasi seberapa baik robot dapat menangani permukaan miring.
- Rintangan berupa papan lompat, yaitu celah lebar yang harus dilompati robot, yang mengevaluasi kemampuan robot untuk melompat.

Lintasan ini mengevaluasi berbagai gerakan, seperti berlari, bergerak ke samping, memanjat, dan melompat. Area dan waktu yang dialokasikan secara khusus digunakan untuk setiap rintangan guna evaluasi yang adil dan konsisten.



Gambar 6.21. Desain untuk lintasan Barkour.

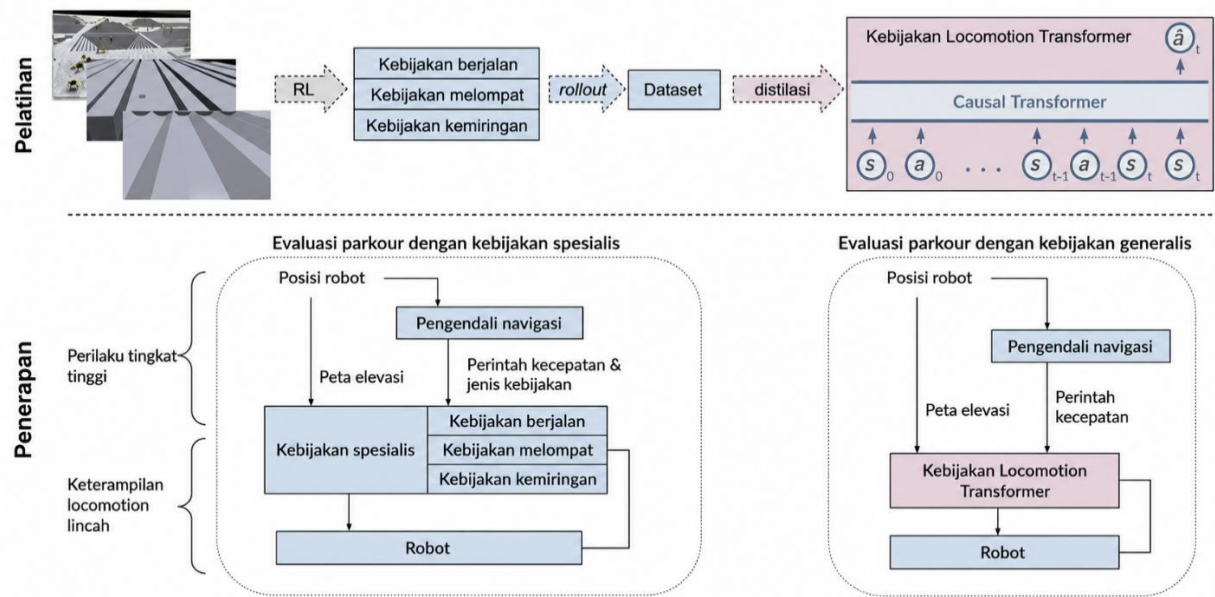
Skor mengukur seberapa cepat dan akurat robot menyelesaikan lintasan. Skor dimulai dari 1,0, dan poin dikurangi untuk setiap rintangan yang gagal atau dilewati atau karena membutuhkan waktu lebih lama dari waktu yang ditentukan.

$$R_{\text{agility}} = 1.0 - \max\left(\frac{t_{\text{run}} - t_{\text{allotted}}}{t_{\text{allotted}}}, 0\right) - \text{penalties}$$

Persamaan 6.3

Di sini, waktu yang dialokasikan adalah waktu standar berdasarkan kompetisi ketangkasan anjing. Hukumannya termasuk pengurangan 0,1 untuk setiap rintangan yang gagal atau dilewati dan 0,01 untuk setiap detik selama waktu yang ditentukan. Lihat Gambar 6.22. Untuk mengatasi tolok ukur, ada dua fase utama untuk menetapkan garis dasar:

- **Fase pelatihan:** Terdapat kebijakan spesialis yang mencakup berjalan, melompat, dan kemiringan. Dalam kebijakan berjalan, robot belajar cara berjalan ke segala arah. Kebijakan ini dilatih menggunakan RL. Kebijakan melompat adalah di mana robot belajar melompati rintangan dan kebijakan ini juga dilatih pada RL. Kebijakan kemiringan adalah di mana robot belajar menangani lereng miring dengan berlatih naik dan turun lereng. Masing-masing kebijakan ini dilatih secara terpisah dalam lingkungan simulasi. Setelah pelatihan, kinerja robot di bawah setiap kebijakan dicatat, menciptakan kumpulan data yang menangkap perilaku robot dalam kondisi yang berbeda. Kumpulan data yang dikumpulkan digunakan untuk melatih Locomotion Transformer, yang merupakan kebijakan generalis yang mengintegrasikan perilaku yang dipelajari ke dalam satu kebijakan yang beragam yang dapat beradaptasi dengan berbagai tugas di lintasan.
- **Fase penerapan:** Evaluasi memiliki dua bagian: evaluasi dengan kebijakan spesialis dan evaluasi dengan kebijakan generalis. Untuk kebijakan spesialis, pengontrol navigasi, yang merupakan pengontrol tingkat tinggi, menentukan jalur keseluruhan yang harus ditempuh robot sepanjang lintasan. Hal ini ditentukan berdasarkan posisi robot dan peta elevasi lintasan. Berdasarkan perintah dari pengontrol navigasi, robot beralih antara kebijakan spesialis (berjalan, melompat, dan kemiringan) untuk menavigasi lintasan. Mirip dengan pendekatan spesialis, pengontrol navigasi memandu robot melalui lintasan dalam evaluasi kebijakan generalis. Alih-alih beralih antara kebijakan yang berbeda, robot menggunakan kebijakan Locomotion Transformer untuk menyesuaikan gerakan secara dinamis berdasarkan lingkungan dan keadaannya sendiri.



Gambar 6.22. Metode yang digunakan untuk menetapkan garis dasar untuk tolok ukur Barkour.

6.6 RINGKASAN

Selama bab ini, Anda telah mempelajari bahwa:

- Pembelajaran mendalam dapat mempelajari pola kompleks dari data sensor berdimensi tinggi dan beradaptasi dengan lingkungan yang berubah, tetapi membutuhkan kumpulan data berlabel yang besar dan sumber daya komputasi. Metode tradisional lebih mudah diinterpretasikan dan efektif dengan data yang lebih sedikit tetapi membutuhkan rekayasa fitur manual.
- Sebagian besar robot menggunakan kamera RGB dan sensor IMU dan beberapa menggabungkan LiDAR untuk pemetaan 3D. Pilihan sensor dan teknik estimasi bervariasi berdasarkan lingkungan dan tugas.
- Berbagai representasi untuk mendefinisikan tugas pemetaan ada dalam robotika. Pemetaan geometris menciptakan peta 3D dengan data kedalaman, sementara representasi voxel membagi ruang 3D menjadi kubus untuk menunjukkan okupansi. NeRF (*Neural Radiance Fields*) menghasilkan adegan 3D dari gambar 2D. Pemetaan semantik menggunakan model seperti CLIP untuk memberi label objek dalam adegan dengan contoh pendekatan kosakata terbuka seperti LeRF dan ConceptFusion.
- Metode lokalisasi mencakup lokalisasi 2D-ke-2D, yang membandingkan gambar yang diambil dengan peta yang telah diberi label sebelumnya, dengan metode seperti RelocNet dan PoseNet yang menggunakan jaringan saraf untuk memprediksi posisi robot. Lokalisasi 2D-ke-3D mencocokkan fitur gambar atau menggunakan regresi koordinat adegan untuk memperkirakan posisi robot. Lokalisasi 3D-ke-3D menggunakan LiDAR untuk membandingkan awan titik dengan peta 3D.
- Beberapa pendekatan navigasi mencakup Mobility VLA, yang menggunakan kombinasi tur demonstrasi dan instruksi multimodal untuk memandu robot. NoMaD menyatukan

navigasi yang berorientasi pada tujuan dan eksplorasi menggunakan kebijakan difusi yang beradaptasi dengan berbagai tugas dengan menggabungkan data visual dan masukan tujuan.

- Kemajuan dalam perangkat keras, simulasi, dan metode pembelajaran (misalnya, kurikulum dan pembelajaran hierarkis) telah memungkinkan robot untuk menavigasi medan yang sulit dengan studi yang berkembang dalam lokomosi berkaki.

Bab selanjutnya membahas peran pembelajaran penguatan dalam robotika, berfokus pada bagaimana agen belajar dari interaksi untuk meningkatkan kinerja, mengatasi tantangan seperti desain hadiah dan efisiensi sampel, dan menerapkan teknik seperti transfer simulasi ke dunia nyata dan metode tanpa model seperti PPO. Bab ini juga menyoroti tren yang muncul, seperti mengintegrasikan LLM dan pembelajaran tanpa fungsi hadiah yang direkayasa secara manual atau eksplisit.

BAB 7

PEMBELAJARAN PENGUATAN DAN KONTROL

Chris Paxton adalah seorang ahli robotika yang pernah bekerja untuk laboratorium FAIR di Meta dan studi NVIDIA. Ia meraih gelar PhD di bidang Ilmu Komputer pada tahun 2019 dari Universitas Johns Hopkins di Baltimore, Maryland, dengan fokus pada penggunaan pembelajaran untuk menciptakan kemampuan perencanaan tugas dan gerakan yang kuat bagi robot yang beroperasi di lingkungan manusia. Karyanya memenangkan Penghargaan Makalah Terbaik ICRA 2021 tentang Interaksi Manusia-Robot dan dinominasikan sebagai makalah sistem terbaik di CoRL 2021, di antara penghargaan lainnya. Studinya berfokus pada penggunaan bahasa, persepsi, perencanaan, dan pembelajaran kebijakan untuk menjadikan robot sebagai asisten serbaguna.

Paradigma dominan untuk pembelajaran secara umum saat ini adalah pembelajaran terawasi: mengambil sekumpulan data yang diketahui dan menyesuaikan beberapa model besar dengannya, yang dapat digunakan untuk aplikasi selanjutnya. Tetapi ini meninggalkan banyak pertanyaan yang belum terjawab: dari mana data itu berasal? Jika modelnya tidak cukup baik—dan tidak ada model yang cukup baik untuk segalanya—bagaimana model tersebut akan meningkat, dan data apa yang diperlukan agar model tersebut meningkat? Akan sangat bagus jika robot dapat mengumpulkan data mereka sendiri dan meningkatkan kemampuan mereka sendiri. Mempelajari cara melakukan hal ini adalah ide utama dari pembelajaran penguatan (*reinforcement learning*).

Impian dari pembelajaran penguatan adalah membangun sistem cerdas yang dapat belajar seperti manusia. Mereka mengumpulkan data melalui interaksi mereka dengan dunia, secara cerdas memilih tujuan mana yang akan dicoba dan keterampilan mana yang akan digunakan dalam berbagai situasi sehingga dapat meningkatkan kemampuan dasar mereka dan, berpotensi, model mereka tentang dunia.

Dalam banyak hal, pembelajaran penguatan tampaknya merupakan prasyarat sejati untuk kecerdasan terwujud yang nyata dan tangguh. Agen yang dikerahkan di dunia nyata harus mampu pulih dan belajar dari kesalahan mereka; secara umum, orang tidak menginginkan robot yang melakukan kesalahan yang sama berulang kali. Selain itu, pembelajaran penguatan mungkin merupakan jalur paling jelas menuju kecerdasan supermanusia sejati. Ia mengoptimalkan beberapa tujuan mendasar dan mengumpulkan datanya sendiri, sehingga dapat melampaui kinerja bahkan para ahli domain dan terus-menerus mengeksplorasi penemuan. Lihat Gambar 7.1.



Gambar 7.1. Robot titik Boston Dynamics yang dilatih untuk berjalan di medan kasar melalui pembelajaran penguatan. Robot modern sering diajarkan untuk berjalan melalui pembelajaran penguatan, dan Boston Dynamics bukanlah satu-satunya perusahaan yang melakukannya.

Terlepas dari keberhasilannya yang menonjol di komunitas studi, penguatan masih merupakan bidang studi yang berkembang dan terus berubah. Tidak seperti pembelajaran terawasi, yang memiliki keberhasilan besar yang dramatis dalam aplikasi seperti visi komputer dan bahasa, pembelajaran penguatan masih memiliki banyak hal untuk dibuktikan dalam skala besar, baik di bidang robotika maupun di luarnya. Namun, metode ini memiliki banyak aplikasi yang jelas, mulai dari mempelajari cara bergerak di dunia, hingga cara bernavigasi di berbagai lingkungan, hingga cara menggenggam dan memanipulasi objek, dan bahkan telah menemukan aplikasi dalam model bahasa yang besar.

Pembelajaran penguatan memiliki tempat yang tidak biasa dalam lingkungan metode pembelajaran mesin. Metode ini setara dengan pembelajaran terawasi dan tidak terawasi sebagai salah satu dari tiga paradigma dasar pembelajaran mesin. Pembelajaran terawasi melibatkan pengambilan data berlabel dan menyesuaikan model dengannya, dan pembelajaran tidak terawasi melibatkan pengambilan data apa pun yang ada di dunia dan menggunakannya untuk mempelajari model yang berguna secara umum. Pembelajaran penguatan cenderung jauh lebih berorientasi pada tujuan, dan karenanya melibatkan pembuatan lebih banyak asumsi.

Mungkin contoh yang paling jelas adalah apa yang digunakan pembelajaran penguatan dalam robotika. Metode ini umumnya tidak digunakan untuk membangun sistem tujuan umum atau membangun fondasi untuk algoritma generalis. Ini digunakan untuk unggul dalam tugas tertentu, dan sering dipercepat oleh sistem penglihatan atau bahasa-penglihatan yang telah dilatih sebelumnya.

Hal ini telah diringkas secara terkenal oleh Profesor Yann Lecun dalam metafora kuenya. Jika diterapkan pada robotika, ini berarti:

1. Pembelajaran mandiri (varian dari pembelajaran tanpa pengawasan) digunakan untuk melatih model yang digunakan sebagai tulang punggung visual.
2. Data demonstrasi digunakan untuk memulai kebijakan, untuk menghindari minimum lokal dan secara dramatis mempercepat pembelajaran kebijakan.
3. Akhirnya, pembelajaran penguatan dapat digunakan untuk mencapai kinerja yang benar-benar luar biasa, mengungguli para ahli dalam domain apa pun yang dioperasikan robot.

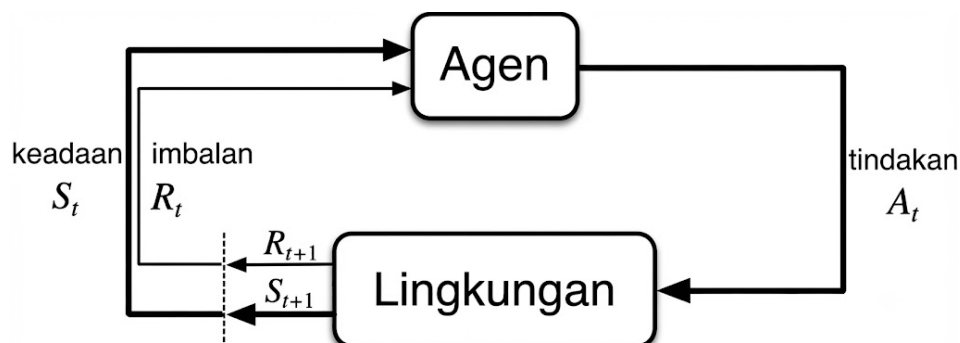
Rasio komponen-komponen ini bervariasi. Dalam banyak kasus, pengaturan masalah yang lebih jelas dan prioritas rekayasa yang kuat dapat menggantikan model yang dipelajari sebelum pembelajaran penguatan, yang menciptakan banyak kesulitan.

Bab ini mencakup tinjauan singkat tentang gagasan pembelajaran penguatan, membahas beberapa metode yang banyak digunakan, dan membahas aplikasi modern umum dari pembelajaran penguatan, baik untuk pembelajaran mesin secara umum maupun untuk robotika secara khusus.

7.1 DASAR-DASAR PEMBELAJARAN PENGUATAN

Pembelajaran penguatan umumnya diformalkan sebagai Proses Keputusan Markov, atau MDP (lihat Gambar 7.2). Proses Keputusan Markov adalah model berbentuk (S, A, T, R) , di mana:

1. S adalah ruang keadaan, representasi dunia seperti posisi objek yang mungkin perlu dimanipulasi oleh robot.
2. A adalah ruang aksi, yang mewakili tindakan apa yang dapat dilakukan agen dari setiap keadaan.
3. T adalah model transisi, dalam bentuk distribusi probabilitas bersyarat $P(S_{t+1}|S_t, A_t)$, yang berarti mewakili seberapa besar kemungkinan transisi dari satu keadaan ke keadaan berikutnya dengan mempertimbangkan tindakan tertentu.
4. R adalah fungsi hadiah, yang mewakili nilai langsung dari transisi dari keadaan S_t ke keadaan S_{t+1} .



Gambar 7.2. Ilustrasi dari loop pembelajaran penguatan dasar. Agen—seperti robot—mengambil tindakan dari himpunan tindakannya A , yang menghasilkan keadaan baru di S dan imbalan yang diamati, R .

Dari setiap keadaan $s \in S$, dalam Proses Keputusan Markov, tujuan agen adalah untuk memilih tindakan terbaik a , sehingga agen memaksimalkan akumulasi imbalan selama masa hidupnya. Pilihan a ditentukan oleh kebijakan $a = \pi(s)$; tujuan pembelajaran penguatan kemudian adalah untuk mempelajari fungsi kebijakan π , yang sering kali diwakili oleh jaringan saraf.

Anda dapat langsung melihat bagaimana ini dapat mencakup berbagai masalah: tindakan tersebut mungkin menampilkan berbagai jenis iklan kepada pengguna yang menjelajahi situs web, atau mungkin derajat kebebasan yang dapat dikontrol dari lengan robot. Imbalan positif dapat diberikan untuk klik-tayang, atau ketika genggamannya berhasil dieksekusi, atau ketika robot bergerak semakin dekat ke tujuannya.

Menyelesaikan Proses Keputusan Markov

Anggaplah untuk saat ini Anda memiliki akses ke fungsi transisi T , dan Anda dapat dengan mudah mengunjungi semua kemungkinan keadaan $s \in S$. Mulai dari keadaan S_0 , Anda dapat dengan mudah memilih keadaan berikutnya yang terbaik berdasarkan probabilitas transisi:

$$\pi(s) = \operatorname{argmax}_a \sum_{s'} P(s, s' | a) R(s, s') + \gamma V(s')$$

Di mana $P(s, s' | a)$ adalah probabilitas transisi dari keadaan s ke keadaan s' dengan tindakan a , $R(s, s')$ adalah imbalan yang terkait dengan transisi tersebut, γ adalah faktor diskon, dan $V(s')$ adalah nilai keadaan s' . Nilai tersebut adalah imbalan yang diharapkan dari titik ini dan seterusnya. Dengan kata lain, kebijakan tersebut menghitung tindakan yang Anda harapkan akan menghasilkan imbalan keseluruhan tertinggi.

Untuk Proses Keputusan Markov yang terbatas dan berukuran tetap, Anda dapat menyelesaikan ini melalui pemrograman dinamis dengan melakukan iterasi nilai Bellman:

$$V(s)_{i+1} = \max_a \sum_{s'} P(s, s' | a) R(s, s') + \gamma V_i(s')$$

untuk iterasi ke- i dari algoritma, hingga konvergen. Kemudian tindakan terbaik dapat dipilih dengan mudah. Ini adalah dasar dari pembelajaran penguatan, dan Anda akan melihat bagaimana hal itu dikembangkan untuk aplikasi yang lebih kompleks nanti di bab ini.

Pertimbangan

Ketika menerapkan ini pada robotika, ada banyak masalah yang jelas. Pertama, algoritma ini sebagian besar bekerja pada ruang diskrit yang terbatas dan relatif kecil di mana masuk akal untuk menghitung probabilitas dan melakukan iterasi atas semua kemungkinan keadaan dalam ruang keadaan. Ruang keadaan robotika cenderung besar dan kontinu, yang berarti bahwa penerapan langsung metode seperti iterasi nilai Bellman tidaklah mudah. Selain itu, ada banyak kesulitan dalam menskalakan metode ini, dan banyak pekerjaan yang dilakukan untuk merancang MDP yang mendasarinya, dengan keadaan, tindakan, dan imbalannya.

Imbalan dapat mengambil banyak bentuk; kekhawatiran utama dalam robotika adalah bahwa imbalan tersebut mungkin jarang, yang berarti bahwa seringkali sulit untuk mengamati imbalan yang bermakna dan informatif pada transisi dari S_t ke S_{t+1} . Pertimbangkan contoh labirin pada Gambar 7.2. Bayangkan sebuah fungsi hadiah yang padat untuk agen yang memecahkan labirin, yang merepresentasikan hadiah sebagai penalti, dengan diberikan:

$$-\|p - g\|_2$$

Di mana $p = (x, y)$ adalah posisi robot dan $g = (x_g, y_g)$ adalah posisi ujung lain dari labirin. Ini memiliki beberapa keuntungan: mudah dihitung, mudah ditentukan, relatif generik, dan akan berguna di semua labirin dan dalam banyak tugas terkait seperti navigasi. Namun, ini sangat serakah: jika solusinya melibatkan banyak penelusuran balik, fungsi hadiah ini dapat menyesatkan agen, karena akan menghukum agen karena menjauh dari tujuan, bahkan jika tidak ada cara untuk mendekat!

Pertimbangkan sebagai gantinya fungsi hadiah yang jarang. Ini bahkan lebih generik: Anda hanya memberikan satu poin kepada agen jika mencapai ujung, dan tidak ada poin jika tidak. Pendekatan semacam ini ideal untuk sebagian besar aplikasi, tetapi jauh lebih sulit untuk dipecahkan, karena hadiah pada dasarnya tidak memberikan informasi di sepanjang jalan. Dalam praktiknya, sebagian besar aplikasi pembelajaran penguatan berkembang dalam kasus pertama, di mana ada fungsi hadiah yang padat, dan gagal dalam kasus yang terakhir. Desain fungsi hadiah menjadi bagian utama dari banyak aplikasi pembelajaran penguatan.

7.2 RL TANPA MODEL VS BERBASIS MODEL

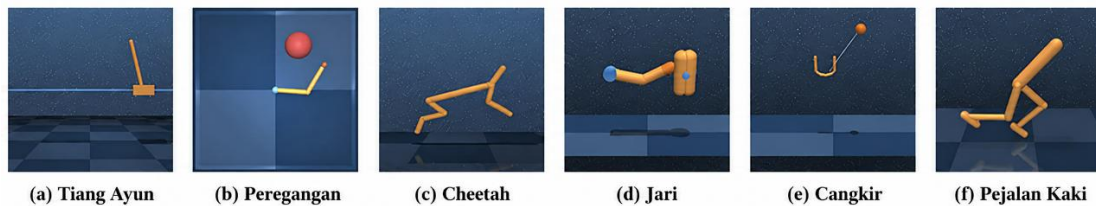
Salah satu kekhawatiran mendasar dalam semua pembelajaran penguatan, khususnya dalam robotika, adalah kompleksitas sampel. Berapa banyak data yang dibutuhkan metode Anda agar dapat bekerja dengan baik dalam skala besar? Dapatkah metode tersebut menyelesaikan masalah yang sangat kompleks?

Ambil contoh algoritma RL "tradisional", yang diwakili oleh Proses Keputusan Markov. Algoritma ini menjelajahi lingkungan—seperti labirin—berdasarkan pengambilan sampel dari kebijakan stokastik, atau dengan menambahkan beberapa noise pada output kebijakan. Kemudian, berdasarkan pengalaman ini, algoritma akan langsung mengoptimalkan kebijakan yang mendasarinya. Ini disebut pembelajaran penguatan tanpa model karena alasan yang akan kita bahas sebentar lagi.

Sekilas, ini tampak hebat. Metode ini bergantung pada sedikit asumsi tentang dunia; pada prinsipnya, metode seperti itu akan bekerja di domain apa pun, untuk aplikasi apa pun. Namun, dengan cepat menjadi jelas bahwa di bawah beberapa asumsi realistis—imbalan yang jarang seperti dalam contoh labirin, atau masalah berdimensi tinggi seperti mengendalikan lengan robot dengan derajat kebebasan tinggi—tiba-tiba Anda mungkin membutuhkan sejumlah besar informasi untuk mengkarakterisasi ruang yang mendasarinya dengan benar.

Ada banyak solusi potensial untuk ini, mulai dari pembentukan imbalan yang selalu populer—merancang sistem imbalan secara manual untuk mendorong agen agar berhasil—

hingga yang lebih berprinsip. Namun pada dasarnya, salah satu cara utama untuk meningkatkan kinerja pembelajaran penguatan adalah melalui perencanaan, membayangkan berbagai kemungkinan masa depan yang Anda yakini akan optimal, alih-alih benar-benar mengeksekusinya. Ini membutuhkan model dunia yang akurat; oleh karena itu, pembelajaran penguatan berbasis model. Lihat Gambar 7.3.



Gambar 7.3. Contoh domain visual yang digunakan dalam PlaNet. Ini adalah domain pembelajaran penguatan umum yang digunakan untuk pengembangan model. Algoritma RL berbasis model seperti PlaNet mampu melakukan pembelajaran yang lebih efisien, tetapi terkadang skalabilitasnya kurang baik.

Tentu saja, menggunakan model dunia untuk menghasilkan berbagai kemungkinan masa depan lebih mudah diucapkan daripada dilakukan. Agar hal ini bermanfaat, model tersebut harus dapat dievaluasi secara efisien secara offline, yang seringkali sangat sulit dalam domain yang kompleks! Selain itu, lingkungan tertentu hampir tidak mungkin disimulasikan (misalnya, wadah penuh objek yang dapat berubah bentuk), dan bahkan jika dapat disimulasikan, hal ini mungkin tidak mungkin dilakukan lebih cepat dari waktu nyata. Dorongan terbaru yang menunjukkan hasil yang mengesankan adalah pembelajaran penguatan berbasis model dengan model yang dipelajari. Karya seperti AlphaGo dan PlaNet menunjukkan bahwa di beberapa domain, model yang berguna dapat dipelajari, yang kemudian memberikan semua keuntungan dari RL berbasis model tanpa memerlukan model ini.

Sebagian besar, impian RL berbasis model belum terwujud. Cara paling umum untuk melakukan pembelajaran penguatan pada tahun 2024 adalah *Proximal Policy Optimization* (PPO), algoritma bebas model yang sangat terukur. PPO telah menunjukkan hasil yang bagus dalam berbagai masalah robotika yang berbeda, mulai dari mempelajari keterampilan robot yang cekatan hingga navigasi hingga manipulasi bergerak. Namun, masih banyak tantangan dan perdebatan belum benar-benar terselesaikan. Dengan konteks tersebut, bagian selanjutnya akan membahas lebih detail tentang dua keluarga utama metode pembelajaran penguatan ini.

Pembelajaran Penguatan Tanpa Model

Sebagai pengingat, masalah pembelajaran penguatan diformalkan sebagai Proses Keputusan Markov (S, A, T, R), dengan keadaan S , tindakan A , imbalan R , dan transisi keadaan

T. Dalam pembelajaran penguatan tanpa model, Anda tidak memiliki akses ke atau mencoba memperkirakan probabilitas transisi T.



Gambar 7.4. OpenAI menggunakan *Proximal Policy Optimization* (PPO), metode pembelajaran penguatan tanpa model, untuk melatih kebijakan yang dapat mengalahkan juara dunia Dota 2.

Ini berguna karena mempelajari dan memperkirakan T dengan benar dapat mahal dan sulit, dan T seringkali sangat sulit untuk dimodelkan—kadang-kadang lebih sulit untuk dimodelkan daripada sekadar memperkirakan nilai keadaan tertentu secara abstrak! Nilai $V(s)$ untuk keadaan $s \in S$ mewakili imbalan yang akan diperoleh dari keadaan tertentu s , artinya imbalan yang akan diakumulasikan untuk semua keadaan yang diharapkan agen untuk dikunjungi mulai dari titik ini. Sederhananya, ini bisa menjadi nilai yang lebih mudah dipelajari, karena seringkali merupakan kualitas langsung dari keadaan itu sendiri—jika Anda tidak ingin jatuh, terhuyung-huyung di tepi tebing jelas buruk—dan hanya memerlukan estimasi nilai itu sendiri, satu kuantitas tunggal, alih-alih memprediksi keadaan lengkap, yang bisa sangat berdimensi tinggi. Lihat Gambar 7.4.

Akibatnya, metode tanpa model secara historis lebih mudah dikelola dan lebih terukur daripada pesaing berbasis modelnya. Beberapa pendekatan pembelajaran penguatan yang paling terkenal adalah tanpa model, termasuk Deep Q Learning, yang membawa pembelajaran mendalam ke arus utama dengan hasil yang menarik dalam memainkan game Atari; AlphaGo, yang dapat memainkan Go pada tingkat yang melebihi manusia terbaik di dunia; dan Proximal Policy Optimization, metode umum yang digunakan untuk memainkan video game Dota 2 pada tingkat kompetitif.

Q Learning

Fungsi Q didefinisikan secara sederhana sebagai $Q(s, a)$ untuk keadaan $s \in S$ dan tindakan $a \in A$, di mana $Q(s, a)$ adalah nilai yang diharapkan (hadiah untuk bermain) dari keadaan yang diberikan, jika tindakan yang ditentukan diambil. Jika Anda secara berturut-turut mengambil tindakan yang memaksimalkan nilai pengembalian fungsi ini dari setiap keadaan, Anda mungkin akan menyelesaikan masalah dengan cara yang optimal. Kesulitannya,

kemudian, adalah mempelajari fungsi Q sejak awal. Inti dari Q learning adalah versi modifikasi dari algoritma iterasi nilai Bellman dari bagian “Menyelesaikan Proses Keputusan Markov”, yang dimodifikasi untuk menggunakan fungsi Q sebagai berikut:

$$Q_{i+1}(s, a) = (1 - \alpha)Q_i(s, a) + \alpha R(s, s') + \gamma Q_i(s, a)$$

di mana $0 \leq \alpha \leq 1$ adalah laju pembelajaran. Perhatikan tidak adanya probabilitas transisi dalam versi asli algoritma. Namun, formulasi ini lebih praktis dalam banyak hal, mampu menangani masalah di mana probabilitas transisi tidak diketahui atau sulit dihitung, memungkinkan agen untuk mempelajari perilaku optimal melalui interaksi langsung dengan lingkungan.

Deep Q Learning

Pada tahun 2014, Google DeepMind merilis Deep Q Learning, yang merupakan varian dari Q learning yang dimodifikasi untuk menggunakan jaringan saraf konvolusional sebagai aproksimator fungsi. Hal ini membuka beberapa masalah menarik, khususnya karena sangat rentan terhadap osilasi atau perubahan parameter yang mendasari jaringan saraf.

Sebagai gantinya, mereka mengusulkan pembelajaran penguatan mendalam melalui buffer pemutaran ulang. Saat agen menjelajah, titik data ditambahkan ke memori, dan saat menghitung pembaruan kebijakan, kumpulan pengamatan acak (s, a, r, s') diambil sampelnya dari memori. Kemudian nilai Q target saat ini dapat dihitung dan pembaruan penurunan gradien dapat dilakukan pada bobot jaringan saraf.

Berhasil untuk pertama kalinya dalam melakukan berbagai tugas hanya dari piksel, inovasi ini memicu ledakan dalam studi pembelajaran penguatan. Namun, Deep Q Learning memiliki beberapa keterbatasan ketika diterapkan pada robotika: ia masih beroperasi pada ruang aksi diskrit, sedangkan aksi robot umumnya kontinu. Ia rentan terhadap berbagai parameter hiper yang perlu disetel, dan seringkali tidak dapat diandalkan. Hal ini mengarah pada keluarga metode kedua yang kurang terbatas: metode gradien kebijakan.

Metode Gradien Kebijakan

Namun, Deep Q Learning memiliki sejumlah kelemahan. Fakta bahwa Anda masih mendekati nilai Q ini secara iteratif memperkenalkan banyak potensi ketidakstabilan saat belajar, karena perubahan kecil dalam kebijakan dapat menyebabkan perubahan dramatis dalam evaluasi, terutama sebagai respons terhadap perubahan keadaan. Saat Anda mengoptimalkan $Q(s, a)$, penting untuk diingat bahwa apa yang sebenarnya Anda optimalkan masih

$$Q_{i+1}(s, a|\theta) = (1 - \alpha) Q_i(s, a|\theta) + \alpha R(s, s') + \gamma Q_i(s, a)$$

di mana θ adalah himpunan bobot jaringan saraf. Sebaliknya, metode gradien kebijakan bertujuan untuk mengoptimalkan parameter kebijakan θ sehingga mengoptimalkan pengembalian yang diharapkan. Pengembalian yang diharapkan diberikan sebagai

$$J(\theta) = E \left\{ \sum_{k=0}^H \gamma^k r_k \right\}$$

Di mana r_k adalah imbalan pada langkah ke- k , γ adalah faktor diskon, dan seterusnya. Dengan kata lain, pengembalian adalah jumlah diskon dari semua imbalan yang diharapkan di masa depan.

Sekarang, alih-alih memperbarui Q , anggaplah Anda memiliki beberapa kebijakan dengan parameter θ , di mana θ misalnya dapat berupa parameter jaringan saraf. Anda kemudian dapat mengambil turunan terhadap parameter kebijakan ini dan mengoptimalkannya secara langsung:

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_{\theta} J(\pi_{\theta}) \big|_{\theta=\theta_k}$$

Sekali lagi untuk parameter kebijakan θ . Ini memberi Anda formulasi di mana, sekarang, Anda hanya perlu menghitung turunan dari pengembalian $J(\theta)$ dan dapat langsung mengoptimalkan kebijakan—oleh karena itu, gradien kebijakan.

Anda kemudian dapat menyatakan gradien $\delta J(\theta)$ sebagai:

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \mathbb{P}_{\pi_{\theta}}(\tau)} \left[\sum_{t=0}^H \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \underbrace{\left(\sum_{t'=t}^H \gamma^{t'-t} r(s_{t'}, a_{t'}) - b(s_t) \right)}_{\text{return estimate } \hat{A}(s_t, a_t)} \right]$$

Di mana

- $\tau \sim p_{\pi_{\theta}}(\tau)$ menunjukkan bahwa Anda mengambil sampel lintasan dari kebijakan $b(s_t)$ adalah imbalan dasar yang akan diperoleh
- $\hat{A}(s_t, a_t)$ adalah estimasi pengembalian—estimasi tentang pengembalian yang diharapkan berdasarkan (s, a) hingga cakrawala
- Istilah $\hat{A}$ dapat dihitung dengan berbagai cara; seringkali diestimasi oleh jaringan saraf kedua, yang disebut kritikus.

Secara keseluruhan, metode gradien kebijakan memungkinkan Anda secara teoritis langsung mengoptimalkan apa yang benar-benar Anda pedulikan, yaitu parameter kebijakan. Meskipun ada banyak cara untuk menghitung pengembalian dan karenanya gradien, bagian ini hanya berfokus pada beberapa opsi.

Optimasi Kebijakan Wilayah Kepercayaan

Tentu saja, seanggun kedengarannya, ada beberapa masalah serius dengan ide ini jika diterapkan secara naif. Optimasi berbasis gradien itu bagus, tetapi rentan terhadap terjebak dalam minimum lokal, dan terhadap ketidakstabilan karena mengambil langkah yang terlalu

besar. Selain itu, hanya menghitung gradien sehubungan dengan ribuan atau bahkan jutaan parameter dalam jaringan saraf dalam dapat menjadi masalah.

Trust Region Policy Optimization (TRPO) bertujuan untuk menyelesaikan masalah ini dengan membatasi divergensi *Kullback-Leibler* (KL) antara distribusi kebijakan lama dan baru selama pembaruan kebijakan. Hal ini dilakukan sambil mengoptimalkan fungsi keuntungan pengganti antara kebijakan lama dan baru.

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}(\theta_k, \theta) \quad \text{s.t.} \quad \overline{D}_{KL}(\theta || \theta_k) \leq \delta$$

Dengan kata lain, TRPO memaksimalkan keuntungan—perbaikan yang diperkirakan—antara kebijakan lama dan kebijakan baru pada setiap langkah proses pembelajaran, sambil membatasi divergensi KL antara kebijakan lama dan kebijakan baru.

Dalam praktiknya, hal ini masih terlalu sulit untuk dipecahkan, sehingga TRPO membuat beberapa perkiraan, khususnya ekspansi Taylor dari tujuan dan kendala:

$$\mathcal{L}(\theta_k, \theta) \approx g^T(\theta - \theta_k)$$

$$\overline{D}_{KL}(\theta || \theta_k) \approx \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k)$$

Hal ini menghasilkan fungsi tujuan terbatas perkiraan:

$$\theta_{k+1} = \arg \max_{\theta} g^T(\theta - \theta_k) \quad \text{s.t.} \quad \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta$$

Yang dapat diselesaikan secara analitis. Namun, Anda belum selesai di sini, karena ekspansi deret Taylor di atas mungkin telah menimbulkan kesalahan, yang berarti kendala awal tidak sepenuhnya terpenuhi. Oleh karena itu, Anda menambahkan pencarian garis mundur (*backtracking line search*) ke solusi analitis asli.

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g,$$

Di mana α dalam $(0, 1)$ adalah koefisien backtracking dan j adalah bilangan bulat non-negatif terkecil sedemikian sehingga terdapat keuntungan positif dan kendala divergensi KL terpenuhi.

TRPO mampu secara substansial mengungguli Deep Q Learning pada domain Atari dari input visi, yang berarti ini merupakan langkah maju yang signifikan untuk pembelajaran penguatan. Namun, TRPO memiliki beberapa masalah penting:

1. Seperti yang terlihat jelas, matematikanya cukup kompleks—misalnya, memerlukan penyelesaian masalah optimasi terbatas. Hal ini membuatnya lebih sulit untuk diimplementasikan.

2. Secara komputasi tidak efisien, karena memerlukan perhitungan hal-hal seperti divergensi KL.

Untungnya, TRPO segera diikuti oleh metode yang menyelesaikan masalah ini: Optimasi Kebijakan Proksimal.

Optimasi Kebijakan Proksimal

Terakhir, pembelajaran penguatan yang paling banyak digunakan saat ini—Optimasi Kebijakan Proksimal (PPO)—diperkenalkan oleh OpenAI pada tahun 2017. Ide di balik PPO sama dengan TRPO: Anda membutuhkan cara untuk memperbarui kebijakan secara substansial dengan sejumlah besar parameter tanpa menyebabkan kinerja menurun. TRPO memecahkan masalah ini dengan menerapkan batasan dengan metode orde kedua yang kompleks. PPO menambahkan beberapa batasan tambahan tetapi menghasilkan masalah optimasi yang secara keseluruhan lebih sederhana. Persamaan inti yang dipecahkan oleh PPO adalah ini:

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s, a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)],$$

Di mana pada setiap langkah, Anda mencari θ yang memaksimalkan $L(s, a, \theta, \theta_k)$ sedemikian sehingga

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A^{\pi_{\theta_k}(s, a)}, g(\epsilon, A^{\pi_{\theta_k}(s, a)}) \right)$$

Di mana g diberikan oleh

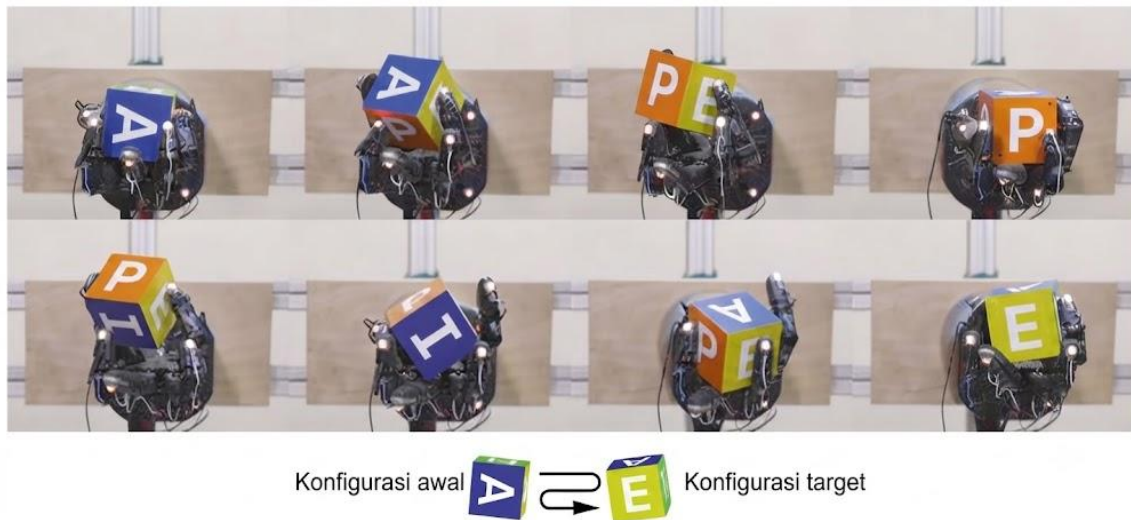
$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & \text{if } A \geq 0 \\ (1 - \epsilon)A & \text{if } A < 0 \end{cases}$$

A adalah keuntungannya.

Jadi, apa arti semua ini? Mari kita uraikan:

1. Jika keuntungannya positif, suku g akan menjadi $(1 + \epsilon)A$, dan $\min()$ dalam L berarti tidak ada keuntungan untuk menjauh terlalu jauh dari kebijakan lama, π_{θ} .
2. Jika keuntungannya negatif, tujuan meningkat jika tindakan menjadi kurang mungkin, sekali lagi dibatasi untuk tetap dekat dengan kebijakan lama.

Oleh karena itu, Anda memiliki masalah yang jauh lebih sederhana, yang dapat dioptimalkan berulang kali dengan beberapa langkah penurunan gradien (stokastik, minibatch), sebelum batch baru diambil sampelnya. Lihat Gambar 7.5.



Gambar 7.5. Tangan robot lima jari yang dilatih menggunakan PPO untuk memanipulasi kubus ke dalam konfigurasi apa pun. PPO adalah salah satu algoritma pembelajaran penguatan yang paling banyak digunakan saat ini.

Dalam praktiknya, PPO adalah algoritma yang sangat efektif, digunakan untuk segala hal mulai dari bermain game seperti Dota hingga berbagai tugas robotika, seperti lokomosi berkaki empat dan manipulasi yang cekatan.

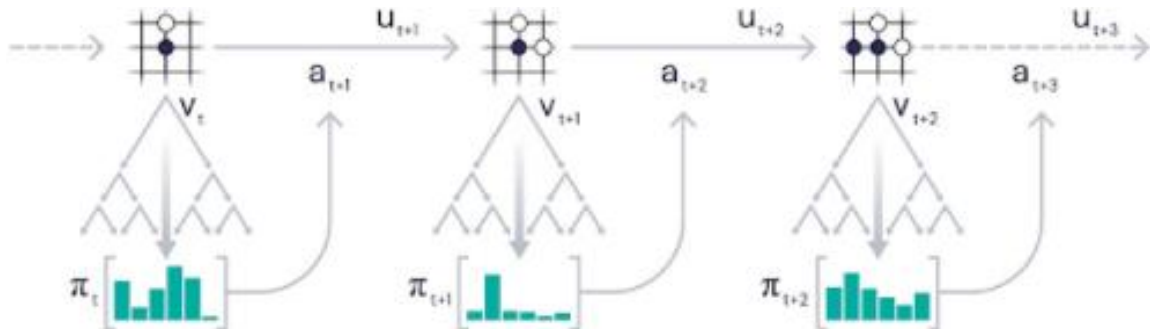
Pembelajaran Penguatan Berbasis Model

Pembelajaran penguatan berbasis model berbeda karena mengasumsikan adanya model dunia, yaitu Anda akan dapat memperkirakan distribusi probabilitas keadaan $P(s, s' | a)$ selama pembelajaran. Ini kurang umum digunakan dalam praktik dibandingkan metode tanpa model, tetapi penting untuk memahami mengapa distribusi probabilitas ini—model dunia ini—merupakan tambahan yang sangat menarik untuk kerangka kerja pembelajaran penguatan, dan bahkan kerangka kerja pembelajaran robot secara umum.

Pembelajaran penguatan berbasis model tidak memiliki banyak algoritma yang banyak digunakan di seluruh bidang seperti PPO. Secara intuitif, ini karena RL berbasis model membutuhkan model dunia; dan seringkali, asumsi ini dapat dibangun ke dalam algoritma itu sendiri. Mungkin karya paling signifikan di bidang ini berasal dari Google DeepMind, yang menggunakan RL berbasis model untuk mengalahkan juara dunia dalam permainan Go, menggunakan kombinasi pencarian pohon—yaitu, perencanaan tradisional tentang keadaan mana yang akan dieksplorasi—dan jaringan saraf untuk mempelajari area mana yang akan melakukan pencarian pohon tersebut.

Pendekatan semacam ini diperluas dalam karya-karya lanjutan, menghasilkan MuZero, sebuah sistem berbasis model yang menggunakan perencanaan untuk menentukan keadaan mana yang layak dieksplorasi. Lihat Gambar 7.6. Sistem ini mempelajari tindakan apa yang harus diambil, serta keadaan apa yang akan dihasilkan dari tindakan tersebut. Dengan membuat prediksi semacam ini, MuZero dapat menyelesaikan tugas-tugas yang jauh lebih menantang dan berjangka panjang, seperti Go yang disebutkan sebelumnya. Metode

pembelajaran penguatan berbasis model dari keluarga ini—AlphaProof dan AlphaGeometry2—meraih medali perak di Olimpiade Matematika Internasional pada tahun 2024, menunjukkan betapa mampunya metode berbasis model dalam menyelesaikan serangkaian masalah yang sangat menantang.

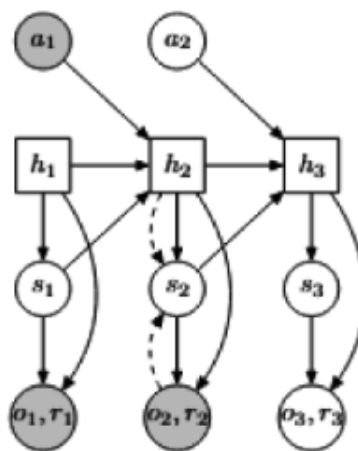


Gambar 7.6. MuZero menggunakan perencanaan untuk menentukan ke mana harus menjelajah selanjutnya. Ia mempelajari fungsi dinamika yang digunakan untuk memodelkan bagaimana keadaan masa depan akan berubah dan imbalan apa yang akan dihasilkan.

Robotika: RL Berbasis Model untuk Kontrol Kontinu

Ada beberapa upaya untuk mengembangkan metode RL berbasis model yang dapat diterapkan pada robotika. Metode-metode ini memiliki beberapa ide yang mirip dengan metode MuZero:

1. Selain mempelajari kebijakan, seseorang harus mempelajari simulator.
2. Simulator ini pada dasarnya memberi Anda fungsi transisi $T(s, a) \rightarrow s'$ dari simulator pembelajaran penguatan asli.
3. Ini memungkinkan Anda untuk merencanakan dan menghitung urutan tindakan mana yang akan memberi Anda pengembalian optimal sebelum benar-benar mengeksekusinya.

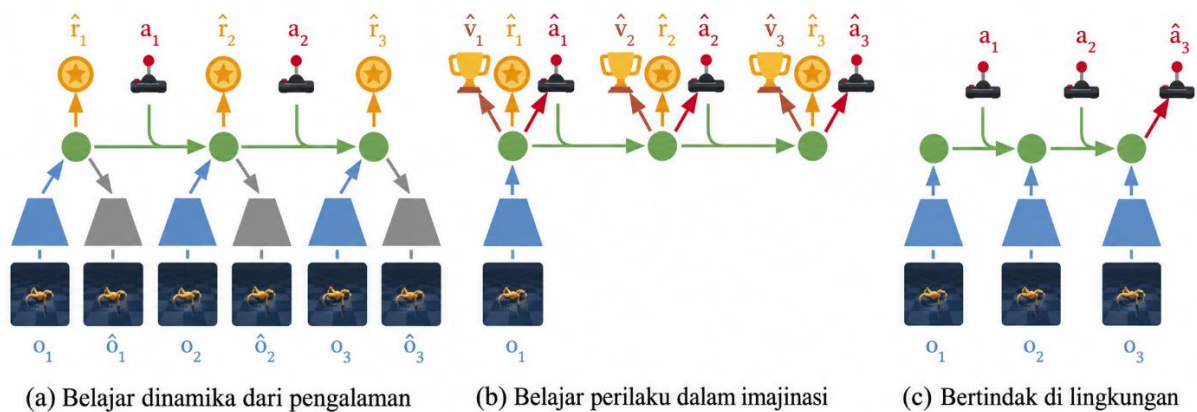


Gambar 7.7. Model Ruang Keadaan Berulang dari PlaNet. Metode RL berbasis model dapat menggunakan pencarian dan model dunia yang dipelajari untuk memfokuskan eksplorasi mereka di wilayah ruang keadaan yang lebih bermanfaat.

Salah satu metode tersebut adalah Deep Planning Network (PlaNet) yang diusulkan dalam. PlaNet menggunakan *Recurrent State Space Model* (RSSM) untuk memprediksi keadaan laten masa depan berdasarkan urutan tindakan. Lihat Gambar 7.7. Memprediksi ruang laten alih-alih observasi lengkap berarti bahwa metode ini tidak perlu mendekode dan menghasilkan gambar lengkap untuk melakukan "peluncuran" rencana masa depan; ini membuat metode secara keseluruhan lebih efisien dan lebih cepat.

Ide inti dari PlaNet dan, terkait dengan Dreamer, adalah bahwa mereka membuat prediksi di beberapa ruang laten. Lihat Gambar 7.8. Di ruang laten ini, Anda menggunakan data untuk mempelajari tidak hanya kebijakan π tetapi juga beberapa fungsi transisi:

$$s' \sim T(s, a)$$



Gambar 7.8. Pendekatan pembelajaran model dari Dreamer.

Ini memberi tahu Anda bahwa, dengan diberikan keadaan s dan tindakan a , Anda akan berakhir di keadaan s' yang dihasilkan dari keadaan ini. Model seperti PlaNet mengasumsikan dinamika keadaan deterministik (yaitu, $s' = T(s, a)$). Fungsi transisi ini sering disebut sebagai model dunia.

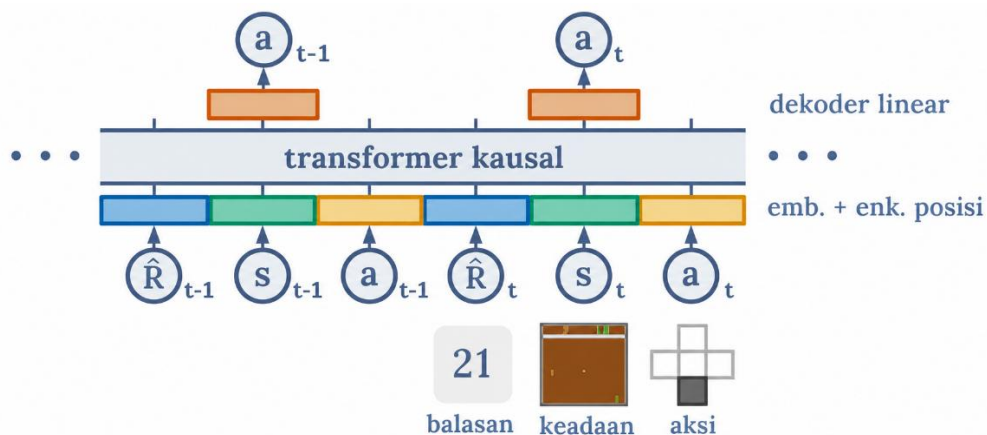
Namun, tantangan pemodelan dunia penuh ini sangat sulit. Anda dapat membandingkannya dengan teknik AI generatif untuk pembuatan video, seperti SORA—dari luar, mempelajari model dunia seperti itu adalah masalah yang bahkan lebih sulit daripada menghasilkan video! Ini karena model dunia robot harus menjadi simulator dunia yang sangat akurat. Model seperti Universal Policy menggunakan pembuatan video sebagai "model dunia" untuk memprediksi masa depan untuk tindakan robot. Ini dapat digunakan untuk merencanakan secara langsung tindakan apa yang harus diambil, tetapi juga memberikan contoh bagaimana model dunia seperti itu dapat dipelajari dan apa yang terkandung di dalamnya.

Metode berbasis model, singkatnya, memiliki janji yang luar biasa: menggunakan perencanaan dan pandangan ke depan untuk memilih tindakan yang lebih baik dan mengeksplorasi lebih efisien, memecahkan masalah inti yang dihadapi pembelajaran

penguatan tanpa model. Hal ini memungkinkan mereka untuk memecahkan masalah yang sangat sulit seperti Olimpiade Matematika Internasional. Namun, belum ada pendekatan "terbaik" yang jelas, dan hanya ada sedikit metode berbasis model yang dapat diterapkan secara luas dalam robotika tanpa pekerjaan yang substansial—bahkan lebih dari rekayasa imbalan dan lingkungan yang biasa diperlukan untuk metode tanpa model. Namun, ini adalah bidang studi yang menarik dan menjanjikan.

7.3 PEMBELAJARAN PENGUATAN OFFLINE

Pembelajaran penguatan sering kali diselesaikan sebagai proses online yang inheren, artinya dilakukan saat agen berinteraksi dengan dunia (nyata atau simulasi). Ini secara inheren membatasi di era big data; kita menginginkan algoritma yang dapat diterapkan pada kumpulan data yang sangat besar yang dikumpulkan dengan berbagai cara. Terkadang data manusia akan berguna; terkadang eksplorasi otonom; Di waktu lain, bahkan akan diperlukan penggunaan kebijakan heuristik untuk mendapatkan awal yang baik di domain-domain sulit tertentu. Di area ini, pendekatan pembelajaran penguatan sebelumnya seringkali gagal.

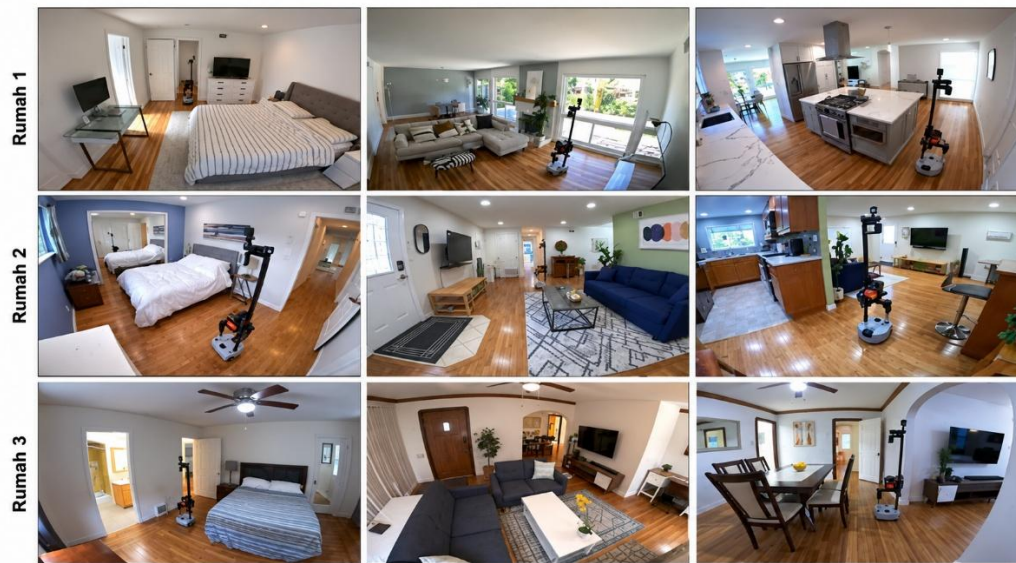


Gambar 7.9. Decision Transformer adalah contoh metode pembelajaran penguatan offline, yang bekerja mirip dengan teknik pengawasan yang digunakan untuk model bahasa besar.

Namun, formulasi masalah pembelajaran penguatan yang mendasarinya masih tetap berharga. Salah satu bidang studi lebih intensif meneliti metode pembelajaran penguatan offline, off-policy—yaitu metode yang dapat diterapkan pada dataset besar yang telah dihitung sebelumnya—dalam banyak hal, dataset ini paling mencerminkan realitas masalah pembelajaran robot.

Ambil Decision Transformer sebagai contoh, seperti yang ditunjukkan pada Gambar 7.9. Ia mengikuti struktur pembelajaran penguatan klasik, dengan aksi $a \in A$, keadaan $s \in S$, dan bahkan fungsi hadiah $R(s, s')$. Namun, Anda tidak mempelajari semua ini melalui interaksi dengan lingkungan, seperti pada formulasi pembelajaran penguatan asli. Sebaliknya, ia menggunakan kumpulan data offline yang besar, dan memiliki label hadiah sebagai bagian dari data itu sendiri.

7.4 APLIKASI DAN TANTANGAN



Gambar 7.10. Menerapkan keterampilan RL untuk navigasi robot bergerak di berbagai lingkungan rumah nyata. Keterampilan pembelajaran penguatan telah diuji pada masalah pembelajaran robot dan dalam banyak kasus bersaing dengan atau bahkan melampaui metode robotika tradisional.

Robotika telah lama menjadi salah satu contoh utama yang memotivasi pembelajaran penguatan. Sekilas, ini adalah aplikasi yang sempurna: kita tidak tahu bagaimana menyelesaikan banyak masalah robotika penting, tetapi kita tahu seperti apa masalah ini ketika berhasil.

Namun, menerapkan pembelajaran penguatan pada masalah robotika memiliki beberapa masalah besar. Yang terbesar, seperti biasa, adalah data. Data robotika cukup langka dan mahal untuk dikumpulkan, tetapi metode pembelajaran penguatan dapat membutuhkan ratusan ribu atau bahkan jutaan langkah. Ada dua cara untuk maju: berlatih dalam simulasi atau menggunakan sejumlah besar robot untuk mengumpulkan data dalam skala besar di dunia nyata. Lihat Gambar 7.10.

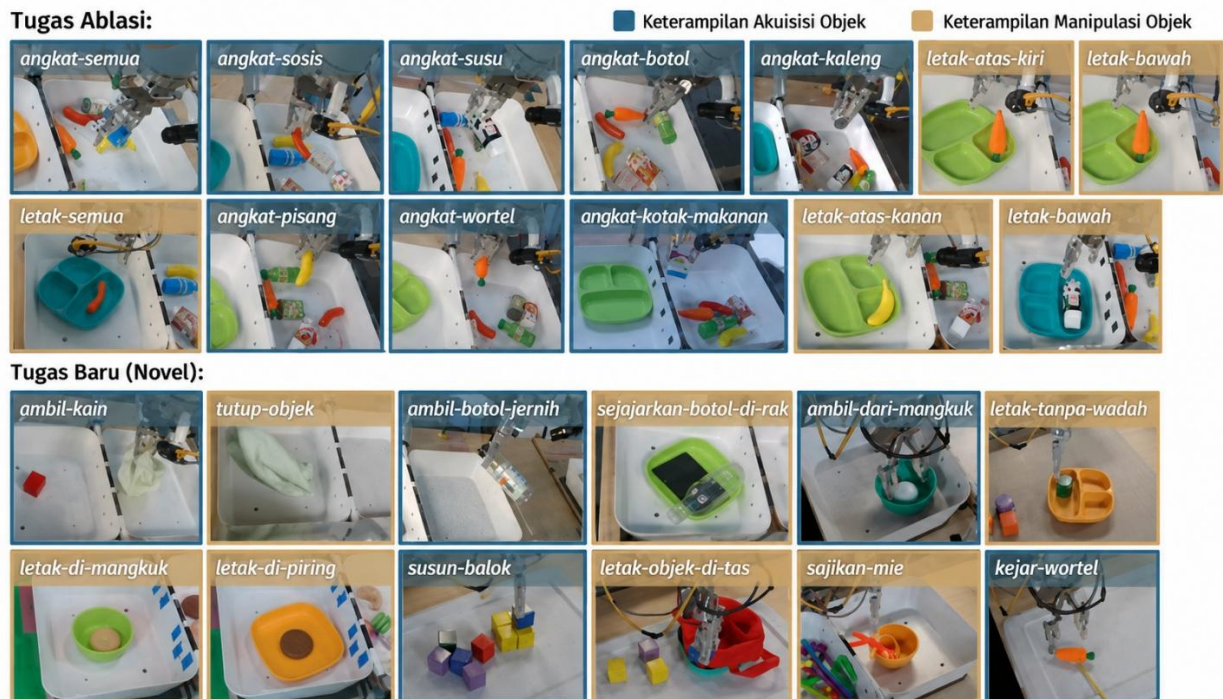
Penggunaan sejumlah besar robot jelas sangat mahal dan menimbulkan banyak masalah—armada robot harus dipelihara, pengumpulan data sulit dilakukan, dan penerapan kebijakan cukup mahal. Namun, simulasi ke dunia nyata, meskipun sangat menarik, memiliki kekurangan nyata. Studi yang melakukan pengujian simulasi ke dunia nyata telah mampu menunjukkan beberapa hasil yang mengesankan, dan untuk hal-hal tertentu seperti pembelajaran gerak robot berkaki empat dan pembelajaran gerak robot humanoid, hal ini dengan cepat menjadi standar.

Namun, untuk tugas-tugas semantik yang lebih kompleks, simulasi ke dunia nyata seringkali tidak dapat ditransfer sama sekali—sebaliknya, studi yang mempelajari keterampilan semantik seperti pencarian objek menggunakan detektor yang dilatih pada pembelajaran terawasi dan campuran kontrol robot tradisional untuk hasil terbaik dan paling

umum. Namun, transfer simulasi ke dunia nyata adalah bidang yang berkembang pesat dan menarik, dan melalui transisi nyata ke simulasi yang lebih baik, kebijakan yang sangat mengesankan dapat dilatih untuk keterampilan robot yang dinamis, reaktif, dan tangguh seperti bermain sepak bola.

Peningkatan Skala RL di Dunia Nyata

Namun, standar emasnya adalah pengumpulan data dunia nyata. Banyak pekerjaan telah dilakukan untuk meningkatkan pengumpulan data dunia nyata, khususnya yang dipelopori oleh laboratorium robotika Google di California, yang menggunakan “arm farm” yang terdiri dari tujuh robot dalam pekerjaan MT-Opt mereka. Lihat Gambar 7.11.



Gambar 7.11. Contoh tugas manipulasi dari MT-Opt. Melatih sejumlah besar keterampilan manipulasi pada sejumlah besar robot nyata berpotensi menjadi jalan menuju robot yang berguna secara umum.

Secara khusus, MT-Opt melihat bagaimana melatih kebijakan robot tujuan umum. Ini berarti bahwa ia melakukan sesuatu yang disebut pembelajaran penguatan multi-tugas, di mana kebijakan tersebut juga diparameterisasi oleh beberapa tujuan (yang dapat dianggap sebagai bagian dari keadaan dalam formulasi RL tradisional dari awal bab). Dalam hal ini, token tambahan diberikan untuk menunjukkan tugas mana yang sedang dieksekusi.

Kemudian model dapat diatur untuk melatih sejumlah besar keterampilan sekaligus. MT-Opt mengkodeifikasi bagaimana Anda dapat melakukan pembelajaran penguatan dalam skala besar dalam pengaturan seperti itu:

1. Siapkan sistem sehingga tugas dapat dengan mudah direset atau dapat mereset dirinya sendiri.
2. Latih pengklasifikasi keberhasilan untuk menentukan kapan tugas berhasil.

3. Yang terpenting, pastikan Anda memiliki tugas-tugas yang dapat diselesaikan dengan cukup mudah sehingga Anda mendapatkan sinyal penghargaan positif dengan cepat, sebelum beralih ke tugas-tugas yang lebih sulit.

Tim tersebut mampu melatih robot untuk melakukan berbagai macam tugas, seperti menyelaraskan, menata ulang, dan memindahkan objek dari satu lokasi ke lokasi lain. Secara keseluruhan, mereka melatih sistem mereka untuk melakukan 12 tugas berbeda menggunakan tujuh lengan robot.



Gambar 7.12. Pemilahan sampah skala besar oleh robot bergerak dari Google dan Everyday Robots. Dua puluh tiga robot dikerahkan untuk melakukan pemilahan sampah selama dua tahun.

Yang penting, melatih semua tugas ini sekaligus sebagai bagian dari kebijakan multi-tugas sebenarnya membantu mempercepat pembelajaran tugas tunggal dalam beberapa cara. Banyak aspek pembelajaran fitur visual akan dibagi di berbagai pengaturan, yang berarti bahwa hal-hal ini tidak perlu dipelajari berulang kali. Hal-hal seperti mendekati permukaan meja atau menghindari tabrakan dibagi di semua kebijakan. Selain itu, seluruh sub-keterampilan mungkin dibagi: misalnya, mendorong objek keluar dari jalan untuk mengakses objek lain, mungkin muncul dalam beberapa keterampilan. Bahkan mungkin untuk mempelajari keterampilan mendorong ini sebagai bagian dari beberapa tugas di mana keterampilan ini secara aktif diberi penghargaan, yang kemudian akan memungkinkan robot untuk mempelajari keterampilan lain yang tidak mungkin dilakukan dengan fungsi penghargaan yang relatif jarang.

Hal ini kemudian diperluas sebagai AW-Opt, sebuah metode yang memungkinkan robot untuk belajar menggunakan data sub-optimal baik online maupun offline dengan menggunakan beberapa ide dari regresi berbobot keuntungan. Keuntungannya di sini adalah membantu mengatasi kekurangan utama dari pembelajaran penguatan pada robot: bahwa data masih sulit dikumpulkan, bahkan dengan sejumlah robot, dan bahwa data yang buruk jauh lebih umum daripada data yang baik.

Terakhir, perlu dicatat sebuah eksperimen dramatis yang dibangun berdasarkan karya-karya ini. Para peneliti di Everyday Robots mengerahkan armada 23 manipulator bergerak,

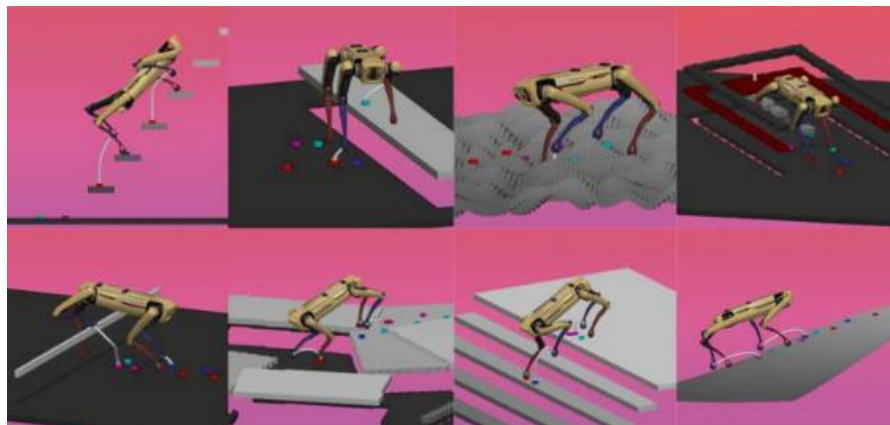
seperti yang ditunjukkan pada Gambar 7.12, untuk melakukan tugas pemilahan sampah. Selama dua tahun, mereka menggunakan pembelajaran penguatan (*reinforcement learning*) dalam skala besar untuk mengumpulkan sejumlah besar data—9.527 jam pengalaman robot—dan menggunakannya untuk melatih kebijakan pemilahan di sembilan skenario distribusi sampah yang berbeda. Robot harus menempatkan objek dengan benar ke dalam tempat sampah kompos, daur ulang, atau sampah biasa. Ini mungkin merupakan upaya RL dunia nyata terbesar yang pernah dilakukan, di mana robot harus secara berurutan mengeksekusi tugas pengambilan dan penempatan untuk berbagai macam objek, memahami objek-objek tersebut, dan memindahkannya ke lokasi yang berbeda. Untuk melakukan ini, mereka memiliki lingkungan simulasi dan lingkungan "pengajaran" di mana data dapat dikumpulkan dengan lebih efisien.

Metode ini menjanjikan, tetapi juga masih menunjukkan jebakan umum RL: data sulit didapatkan, kebijakan RL seringkali tidak dapat digeneralisasikan dengan baik di luar apa yang mereka lihat selama pelatihan dan oleh karena itu lingkungan pelatihan dibangun dengan hati-hati, dan mengumpulkan data nyata dengan imbalan yang bermanfaat itu sulit. Bagian selanjutnya membahas kasus-kasus di mana hal ini kurang relevan: pelatihan sebagian besar dalam simulasi.

Belajar Berjalan

Pembelajaran penguatan memiliki satu kisah sukses yang spektakuler: keterampilan berjalan untuk robot bergerak. Berjalan telah lama menjadi keterampilan robot yang paling penting. Ini telah menjadi cara de facto pergerakan robot berkaki empat dilakukan, dengan perusahaan seperti Unitree, Boston Dynamics, dan Anybotics mulai mengirimkan produk yang dilatih dengan pembelajaran penguatan. Lihat Gambar 7.13.

Robot-robot ini umumnya menggunakan sensor kedalaman untuk menentukan geometri dunia di sekitarnya. Masalah ini sangat cocok untuk kasus pembelajaran penguatan: tidak seperti masalah kompleks seperti pemilahan sampah atau pengambilan dan penempatan, umpan balik sering terjadi, fungsi hadiah berharga, dan mudah untuk diatur ulang dan dipulihkan. Metode RL seperti PPO dan TRPO sangat efektif dalam skenario seperti ini, di mana umpan balik terhadap kebijakan—dan dengan demikian, kemampuan untuk memperkirakan gradien yang diperlukan—tersedia dengan mudah.



Gambar 7.13. Boston Dynamics melatih robotnya untuk berjalan dalam simulasi.

Baru-baru ini, para peneliti telah beralih ke gerakan cepat dan dinamis yang dilatih menggunakan pembelajaran penguatan; pelatihan dalam simulasi dan pengujian di dunia nyata memungkinkan algoritma bebas model yang kurang efisien seperti PPO dan TRPO untuk diterapkan pada perangkat keras nyata.

Robot Bermain Sepak Bola

Melatih robot untuk bermain sepak bola menggunakan RL mendalam. Keterampilan robot dapat dipelajari dalam simulasi dan ditransfer ke robot nyata. Hal ini bahkan telah diperluas ke kasus pembelajaran penguatan multi-agen. Menggabungkan hal-hal ini, sebagai studi kasus dalam pembelajaran penguatan robot, pertimbangkan karya terbaru tentang robot yang bermain sepak bola. Ini adalah tugas yang menarik bagi robot karena:

1. Ini melibatkan banyak agen cerdas.
2. Robot harus belajar menggunakan ruang pengamatan berdimensi tinggi yang sebagian dapat diamati—gambar.
3. Robot itu sendiri memiliki derajat kebebasan yang relatif tinggi, karena merupakan humanoid kecil, dan mereka perlu dapat berlari dan pulih ketika mereka pasti jatuh.

Mereka berlatih menggunakan optimasi kebijakan a posteriori maksimum (MPO), algoritma aktor-kritik off-policy, dan mereka menghasilkan sejumlah besar data dalam simulasi untuk mencapai hal ini. Mereka melatih keterampilan secara terpisah seperti mencetak gol dan bangkit dari tanah jika robot jatuh, sebelum menggabungkannya—strategi umum dalam tugas jangka panjang yang juga terlihat di tempat lain. Kemudian mereka melatih kebijakan agen yang harus bersaing melawan lawan yang semakin sulit.

Agen mereka mampu melakukan transfer simulasi-ke-nyata tanpa data dunia nyata (*zero-shot sim-to-real transfer*), yang berarti bahwa agen tersebut bekerja tanpa data dunia nyata apa pun. Hal ini dicapai melalui penambahan gangguan dan pengacakan domain ke lingkungan simulasi. Mereka mengacak gesekan lantai dan menerapkan gaya acak pada agen saat mereka bergerak, tanpa itu transfer simulasi-ke-nyata tanpa data dunia nyata tidak mungkin dilakukan. Hal ini dilanjutkan dengan perubahan acak pada pencahayaan dan saturasi warna untuk meningkatkan transfer visual. Pada akhirnya, ini menghasilkan agen robot yang dapat bermain sepak bola satu sama lain, menendang bola, dan mencetak gol.

Pembelajaran Penguatan dan Model Bahasa Besar

Model bahasa besar (LLM) telah memberikan dampak dramatis pada bidang robotika, serta pada dunia teknologi yang lebih luas, dan sebagian dari daya tarik yang membuat LLM begitu menarik adalah teknik yang disebut Pembelajaran Penguatan dari Umpan Balik Manusia (RLHF). Karena LLM sangat relevan dengan robotika, penting untuk membahas kasus penggunaan pembelajaran penguatan yang terkait dengannya.

Pembelajaran Penguatan dari Umpan Balik Manusia (RLHF)

RLHF dilakukan setelah melatih LLM pada korpus data yang besar. Setelah melatih LLM pada semua data yang berguna (*supervised*), Anda melakukan dua langkah:

1. Melatih model penghargaan.
2. Menggunakan model penghargaan ini untuk memperbarui LLM asli.

Seperti yang telah disebutkan, mendapatkan fungsi penghargaan adalah bagian tersulit dari penerapan pembelajaran penguatan pada banyak masalah dunia nyata. LLM tidak terkecuali. Untuk mempelajari fungsi penghargaan, anotator manusia harus memberi peringkat pada output dari LLM asli (ini adalah bagian umpan balik manusia dari RLHF). Kemudian, peringkat ini digunakan untuk menyempurnakan model bahasa kedua, yang bisa memiliki arsitektur yang sama dengan yang pertama, atau, seperti yang sering terjadi, model yang lebih kecil.

Kemudian, dengan adanya model yang dapat memprediksi imbalan dari keluaran teks, Anda sekarang memiliki masalah pembelajaran penguatan. Keadaan, s , adalah teks asli, tindakan a adalah konteks, dan imbalan adalah estimator fungsi imbalan yang baru dilatih. Dari sini, Anda dapat menghitung langkah gradien kebijakan sesuai dengan algoritma PPO, memodifikasi bobot salinan model bahasa asli.

Apa hasilnya? Sebuah model yang menghasilkan teks yang lebih mirip dengan apa pun yang disukai manusia! Secara umum, model yang dilatih dengan RLHF memiliki gaya yang jauh lebih disukai daripada model yang dilatih tanpa itu, yang sangat penting untuk chatbot atau asisten studi AI! Namun, ada beberapa kelemahan. Mengumpulkan data manusia ini sangat mahal, dan Anda juga tidak dapat menggunakan fungsi imbalan Anda terlalu banyak. Anda umumnya tidak dapat mengambil terlalu banyak langkah gradien atau melakukan terlalu banyak iterasi RLHF tanpa model mulai menemukan celah yang dapat dimanfaatkannya. Jika LLM dapat menemukan contoh adversarial yang mengelabui fungsi imbalan, kinerja akan mulai menurun lagi.

Selain itu, karena didasarkan pada preferensi manusia, ia dapat terlalu fokus pada transfer gaya dan aspek visual lainnya daripada substansi. Terakhir, ada banyak pertanyaan terbuka tentang bagaimana mengatur algoritma RL, dan ada banyak potensi peningkatan pada PPO yang cocok untuk RLHF.

Optimasi Preferensi Langsung (DPO)

Optimasi Preferensi Langsung (DPO) menyederhanakan proses RLHF dengan menghilangkan kebutuhan akan model penghargaan terpisah. Sebaliknya, DPO memperlakukan masalah ini sebagai tugas klasifikasi. Model tersebut langsung disempurnakan menggunakan data preferensi manusia, yang lebih langsung menyelaraskan keluaran model dengan apa yang disukai manusia. Proses ini memiliki langkah-langkah utama berikut[38]:

- Dimulai dengan penyempurnaan terawasi (SFT) di mana model disempurnakan pada dataset berlabel. Label tersebut mewakili respons yang disukai dan keluaran model diselaraskan dengan pedoman tertentu.
- Setelah SFT, model disempurnakan lebih lanjut menggunakan data preferensi, yang terdiri dari pasangan output yang diberi peringkat oleh anotator manusia. Dalam DPO, fungsi kerugian preferensi didefinisikan yang mencerminkan seberapa baik output model selaras dengan peringkat manusia ini. Karena kerugian ini merupakan fungsi dari kebijakan itu sendiri, Anda tidak memerlukan model penghargaan terpisah.
- Model dioptimalkan dengan mengurangi kerugian preferensi sehingga, untuk setiap pasangan output, model belajar untuk menghasilkan output yang lebih sering disukai manusia.

Secara keseluruhan, DPO dapat lebih stabil dan efisien daripada RLHF, karena menghilangkan kebutuhan akan banyak pengambilan sampel dan penyetelan hyperparameter. Karena alasan ini, ini dapat menjadi metode yang berguna untuk menyempurnakan LLM dengan preferensi manusia tanpa biaya tambahan dari metode RL tradisional.

Pembelajaran Penguatan dari Umpan Balik AI (RLAIF)

RLAIF mengotomatiskan pengumpulan data preferensi manusia menggunakan LLM siap pakai untuk menghasilkan label preferensi. Alih-alih mengandalkan label biner, RLAIF menggunakan probabilitas logaritma dari berbagai keluaran preferensi, pada dasarnya menciptakan distribusi preferensi menggunakan fungsi softmax. Pendekatan ini mengurangi waktu dan biaya yang biasanya terkait dengan pengumpulan umpan balik manusia dan telah terbukti berkinerja serupa dengan RLHF, terutama ketika menggunakan model besar seperti PaLM Google.

Seiring LLM semakin terintegrasi ke dalam robotika, akan bermanfaat untuk memahami bagaimana data preferensi manusia dapat dimanfaatkan. Hal ini sangat berguna dalam aplikasi di mana robot perlu berinteraksi dengan manusia dan mematuhi kebutuhan/preferensi mereka. RLHF, DPO, dan RLAIF semuanya menyediakan jalur untuk menyempurnakan LLM agar lebih selaras dengan perilaku manusia. Namun, masing-masing metode ini memiliki kelebihan dan kekurangannya sendiri tergantung pada aplikasi, LLM yang digunakan, dan sebagainya, sehingga studi lebih lanjut diperlukan untuk menyempurnakan metode ini untuk aplikasi robotika dunia nyata.

7.5 TANTANGAN DALAM RL UNTUK ROBOTIKA

Meskipun telah ada beberapa aplikasi RL yang sukses untuk robotika dunia nyata, penting untuk mencatat beberapa tantangan utama yang masih dihadapi bidang ini. Banyak dari tantangan ini telah dibahas di sepanjang bab ini. Berikut adalah ringkasan gabungan dari tantangan-tantangan ini, beserta solusi potensialnya:

- **Efisiensi sampel:** Algoritma RL seringkali membutuhkan sejumlah besar sampel atau interaksi dengan lingkungan untuk mempelajari kebijakan yang berguna. Dalam robotika, mengumpulkan sampel-sampel ini di dunia nyata dapat memakan biaya dan waktu. Meningkatkan efisiensi sampel penting untuk mengurangi jumlah data yang dibutuhkan untuk pembelajaran sambil tetap mencapai kinerja yang baik. Salah satu cara untuk melakukan ini adalah dengan lebih mengandalkan data simulasi sambil meningkatkan metode Sim2Real, sehingga kesenjangan antara lingkungan simulasi dan dunia nyata diminimalkan. Ide-ide ini telah dibahas di Bab 5.
- **Pembelajaran transfer:** Menggeneralisasi kebijakan RL dari satu robot atau lingkungan ke robot atau lingkungan lain merupakan tantangan yang sulit. Apa yang berhasil dengan baik di lingkungan simulasi atau pada satu robot mungkin tidak dapat langsung ditransfer ke robot yang berbeda atau lingkungan dunia nyata. Mengembangkan algoritma RL yang dapat ditransfer dan beradaptasi dengan kondisi baru sangat penting untuk penggunaan praktis. Menggunakan teknik seperti meta-learning, di mana Anda

melatih agen RL untuk belajar bagaimana belajar, dan randomisasi domain dapat membantu meningkatkan kinerja di lingkungan baru.

- **Skalabilitas:** Karena tugas robotika menjadi lebih kompleks dan melibatkan ruang keadaan dan aksi berdimensi tinggi, skalabilitas menjadi perhatian. Beberapa algoritma RL yang bekerja dengan baik untuk tugas-tugas sederhana mungkin tidak dapat diskalakan untuk menangani kompleksitas aplikasi robotika dunia nyata. Beberapa arah untuk mengembangkan algoritma RL yang dapat diskalakan menggunakan RL hierarkis, di mana tugas-tugas kompleks dipecah menjadi subtugas yang lebih sederhana, dan mengembangkan strategi eksplorasi yang lebih efisien yang berfokus pada area ruang keadaan yang menjanjikan.
- **Desain imbalan:** Merancang fungsi imbalan yang sesuai yang memandu proses pembelajaran seringkali merupakan salah satu tugas tersulit dalam RL. Imbalan yang tidak selaras atau didefinisikan dengan buruk dapat menyebabkan kebijakan yang suboptimal atau agen Anda tidak mempelajari sesuatu yang berguna. Dengan demikian, mengembangkan fungsi imbalan yang secara akurat mewakili tujuan dan batasan tugas merupakan aspek penting dari desain algoritma RL. Beberapa strategi di sini termasuk menggunakan Inverse RL (IRL), di mana fungsi hadiah dapat dipelajari berdasarkan perilaku ahli yang diamati, membentuk fungsi hadiah secara bertahap, atau memiliki beberapa tujuan dalam desain hadiah.

7.6 TREN YANG MUNCUL DALAM RL UNTUK ROBOTIKA

Pembelajaran penguatan kembali populer di bidang robotika, setelah bidang ini didominasi oleh pendekatan pembelajaran imitasi selama bertahun-tahun.

- **Simulasi ke dunia nyata:** Menghasilkan data untuk pelatihan RL itu sulit; bahkan metode RL terbaik, seperti TRPO dan PPO, membutuhkan data pelatihan dalam jumlah sangat besar, yang seringkali tidak praktis untuk dikumpulkan di dunia nyata. Studi seperti dan telah menunjukkan bagaimana kita dapat menerapkan pembelajaran penguatan pada masalah yang menarik, berkat penggunaan simulasi yang cerdas.
- **Mengintegrasikan model bahasa besar:** Memasukkan model bahasa besar, model bahasa-visi besar, dan mengubahnya menjadi model aksi yang dapat bertindak di lingkungan nyata mungkin memerlukan pembelajaran penguatan, di mana agen AI berulang kali berinteraksi dengan lingkungan atau sistem virtual.
- **Pembelajaran tanpa fungsi imbalan manual:** Fungsi imbalan adalah salah satu bagian tersulit dari masalah pembelajaran penguatan. Karya seperti Eureka telah meneliti penggunaan LLM dan agen AI untuk merancang dan menyetel fungsi imbalan; karya seperti VIP telah meneliti ekstraksi fungsi imbalan ini dari video. Menemukan cara untuk menggunakan data skala internet untuk mengurangi atau menghilangkan kebutuhan penyetelan imbalan akan sangat penting untuk meningkatkan skala RL untuk robotika dunia nyata.

7.7 MASA DEPAN PEMBELAJARAN PENGUATAN DALAM ROBOTIKA

Pembelajaran penguatan adalah kerangka kerja yang ampuh dan serbaguna untuk menggambarkan masalah robotika, dan memiliki janji besar untuk memungkinkan kita mempelajari keterampilan secara otomatis untuk berbagai masalah yang sulit dikumpulkan datanya, dan untuk meningkatkan kinerja ahli. Pembelajaran penguatan sebelumnya telah diterapkan pada tugas manipulasi multi-langkah, pada lokomosi, dan pada model bahasa. Ini adalah perangkat yang ampuh yang muncul dalam produk robot nyata seperti robot berkaki empat dan chatbot AI. Namun, ada banyak pertanyaan terbuka yang perlu dipecahkan. Kita membutuhkan algoritma RL robotika yang lebih mampu melakukan generalisasi pada objek dan lingkungan yang belum pernah dilihat sebelumnya. Memanfaatkan lebih banyak data offline, baik melalui pencampuran pembelajaran imitasi seperti pada AW-Opt, metode RL offline, atau pra-pelatihan, akan menjadi bagian besar dari apa yang membuat ini berhasil, mengingat keterbatasan pengumpulan semua data di lokasi.

Pekerjaan di masa depan dalam pembelajaran penguatan pasti akan memanfaatkan pembelajaran dalam simulasi dan transfer simulasi ke dunia nyata, yang terus menghasilkan hasil yang luar biasa dan mengatasi beberapa masalah efisiensi data terbesar yang dihadapi pembelajaran penguatan. Secara keseluruhan, tampaknya pasti bahwa robot masa depan akan menggunakan pembelajaran penguatan untuk menjadi lebih cepat, lebih kuat, dan lebih cerdas—pertanyaannya adalah bagaimana caranya.

7.8 RINGKASAN

Dalam bab ini, Anda mempelajari ide dan konsep berikut:

- Pembelajaran terawasi bergantung pada data berlabel, tetapi memiliki keterbatasan. Pembelajaran penguatan mengatasi tantangan ini dengan memungkinkan robot untuk secara mandiri mengumpulkan data dan belajar saat mereka berinteraksi dengan lingkungan mereka.
- Tujuan utama RL adalah untuk menciptakan sistem cerdas yang belajar seperti manusia, dengan berinteraksi dengan lingkungan mereka, menetapkan tujuan, dan menyempurnakan keterampilan mereka. RL sangat penting untuk mencapai kecerdasan terwujud yang kuat pada robot dengan membiarkan mereka belajar dari kesalahan dan meningkatkan diri.
- Dasar-dasar RL, termasuk formulasinya sebagai Proses Keputusan Markov (MDP), dan perbedaan antara metode tanpa model dan berbasis model.
- Proyek-proyek seperti MT-Opt dan AW-Opt dari Google meningkatkan pembelajaran dengan melatih robot pada banyak tugas, memungkinkan generalisasi yang lebih baik dan pembelajaran tugas tunggal yang lebih cepat karena fitur visual dan sub-keterampilan yang sama.
- Pembelajaran Penguatan dari Umpan Balik Manusia (RLHF) digunakan dalam model bahasa besar untuk menyempurnakan output berdasarkan preferensi manusia, tetapi menghadapi kekurangan seperti biaya pengumpulan data yang tinggi dan potensi perilaku yang merugikan. Pendekatan alternatif termasuk Optimasi Preferensi

Langsung (DPO) dan Pembelajaran Penguatan dari Umpan Balik AI (RLAIF), yang dapat menjadi metode yang lebih efisien untuk menyelaraskan LLM dengan preferensi manusia tanpa model penghargaan eksplisit.

- RL di dunia nyata bergumul dengan kelangkaan data, generalisasi yang buruk, dan kesulitan desain penghargaan yang dalam beberapa hal dapat diatasi dengan metode seperti transfer simulasi ke dunia nyata.
- Di masa depan, RL dalam robotika kemungkinan akan berfokus pada peningkatan efisiensi sampel, transfer learning, skalabilitas, dan desain reward, menggunakan lebih banyak data offline dan metode simulasi-ke-nyata.

Bab selanjutnya mengeksplorasi teknologi swakemudi sebagai aplikasi utama robotika, berfokus pada potensi ekonominya dan kerangka kerja perangkat lunak dan perangkat keras yang memungkinkan persepsi, prediksi, perencanaan, dan keselamatan.

BAB 8

KENDARAAN TANPA PENGEMUDI

Bab ini membahas teknologi pengemudian otonom dan menjelaskan bagaimana mobil otonom dibangun. Tujuan bab ini adalah untuk membantu Anda memahami bagaimana sistem tersebut dirancang dan dibangun serta metrik apa yang digunakan untuk mengevaluasi kinerja berbagai komponen.

8.1 INDUSTRI DAN MODEL BISNIS KENDARAAN OTONOM

Mobil tanpa pengemudi menghadirkan peluang ekonomi yang sangat besar, karena industri transportasi bernilai sekitar Rp 7 kuantiliun. Industri transportasi juga terbagi menjadi pengemudian di jalan raya untuk pengangkutan barang dan operasi lainnya, yang membentuk tulang punggung rantai pasokan, dan pengemudian di perkotaan untuk mengangkut manusia.

Secara historis, inovasi swakemudi meningkat pesat selama tantangan perkotaan DARPA yang didanai oleh Departemen Pertahanan Amerika Serikat pada tahun 2007. Beberapa tim berkompetisi dalam tantangan ini, yang akhirnya melahirkan beberapa perusahaan rintisan, termasuk Zoox yang muncul dari Stanford, Waymo yang dipimpin oleh orang-orang dari Berkeley dan Universitas Carnegie Mellon (CMU), Argo AI yang lahir di CMU, dan Cruise Automation, yang kemudian diakuisisi oleh *General Motors* (GM). Setelah pandemi COVID-19 dan krisis pendanaan yang dipicu resesi, terjadi konsolidasi yang meluas di pasar, dengan Amazon mengakuisisi Zoox, Uber menjual ATG ke Aurora, Cruise mengakuisisi Voyager, Apple mengakuisisi drive.ai, dan beberapa perusahaan rintisan lainnya bangkrut.

Terdapat dua model bisnis utama untuk kendaraan otonom. Salah satunya adalah model mobil pribadi, yang diupayakan oleh Tesla dan Comma AI, di mana mobil otonom dijual kepada pelanggan individu. Mengingat penggunaan mobil pribadi mencakup area geografis yang luas yang dapat memakan biaya mahal untuk dipetakan dalam resolusi tinggi, perusahaan yang bertaruh pada model bisnis ini berinvestasi sesedikit mungkin dalam pemetaan. Karena mobil dijual kepada individu, ada persyaratan untuk menjaga biaya kendaraan otonom, baik perangkat lunak maupun rangkaian sensor, tetap terjangkau.

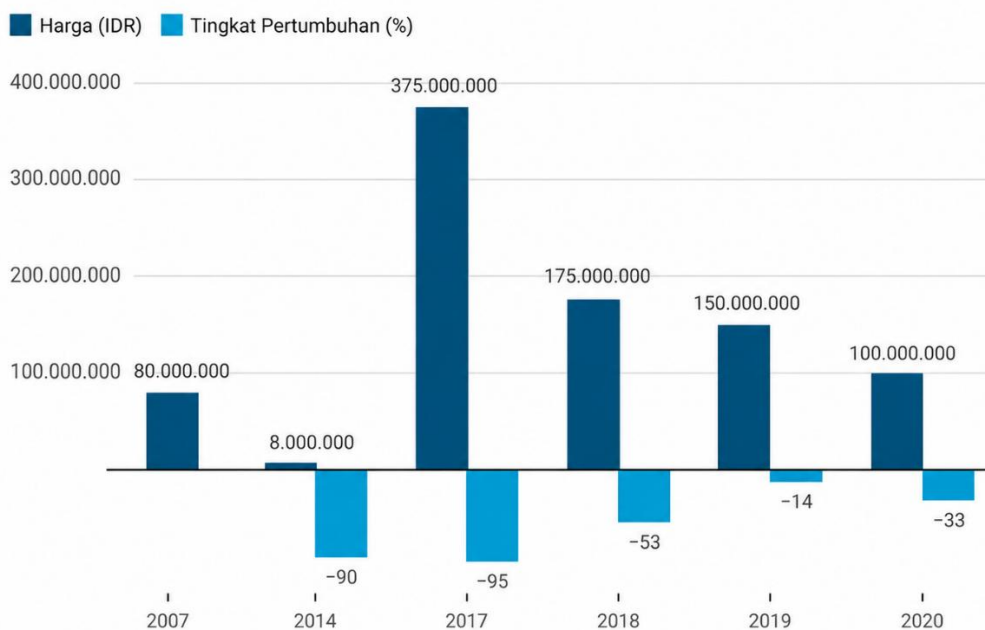
Model kedua adalah model bisnis robotaxi, di mana perusahaan menjalankan mobil otonom sebagai robotaxi, mirip dengan Uber dan Lyft, di area yang dibatasi secara geografis yang telah dipetakan dan diuji secara menyeluruh. Ini diupayakan oleh Waymo dan Cruise. Biasanya, biaya taksi diamortisasi sepanjang masa pakainya, sehingga memungkinkan harganya sedikit lebih mahal daripada kendaraan pribadi. Kedua model bisnis ini juga menghasilkan filosofi teknis yang sedikit berbeda:

1. **Sensor dan redundansi:** Karena pertimbangan biaya, model bisnis kendaraan pribadi yang dapat mengemudi sendiri cenderung kurang bergantung pada sensor mahal seperti LiDAR, yang mengarah pada mobil yang hanya mengandalkan penglihatan dan didukung oleh kamera multiview. Namun, dalam beberapa tahun terakhir, harga LiDAR telah turun drastis, seperti yang ditunjukkan pada Gambar 8.1.

2. **Gaya intervensi dan tanggung jawab:** Dalam model bisnis kendaraan pribadi, karena kendaraan dimiliki dan dioperasikan oleh pelanggan, intervensi dalam skenario yang rumit, atau yang disebut pelepasan (ketika sistem otonom dinonaktifkan untuk mendukung pengemudian manual oleh manusia), disediakan oleh pengemudi di balik kemudi. Ketika kecelakaan terjadi akibat perilaku buruk pengemudi sendiri, tanggung jawab jatuh pada pelanggan. Sebaliknya, dalam model bisnis robotaxi, intervensi disediakan oleh teleoperasi dan manusia jarak jauh, memungkinkan mobil untuk sepenuhnya menghilangkan pengemudi di dalamnya.

Harga & Tingkat Pertumbuhan LiDAR

Harga dalam Rupiah (IDR) & Tingkat Pertumbuhan dalam Persentase



(Harga dalam Rupiah (IDR) & Tingkat Pertumbuhan dalam %)

Sumber: Market.us Scoop

Gambar 8.1. Penurunan biaya LiDAR.

Mulai tahun 2023, Cruise dan Waymo memperoleh izin untuk mengenakan biaya kepada pelanggan untuk perjalanan taksi swakemudi di kota San Francisco serta di beberapa bagian Bay Area.

8.2 ARSITEKTUR SISTEM KENDARAAN OTONOM

Sistem swakemudi, dalam interpretasi paling sederhana, dirancang dengan input berupa input sensor dan output berupa akselerasi, kemudi, dan sudut mobil. Tujuan sistem ini adalah untuk berhasil berpindah dari titik A ke titik B di peta sambil mengurangi biaya seperti waktu dan tanpa melanggar peraturan lalu lintas atau batasan lainnya. Arsitektur tingkat tinggi dan blok bangunan sistem ini ditunjukkan pada Gambar 8.2.



Gambar 8.2. Arsitektur sistem pengemudian otomatis.

Untuk berkendara di perkotaan dan jalan raya, sebuah mobil diharapkan memiliki frekuensi respons lebih dari 10 Hz, yang berarti bahwa seluruh tumpukan ujung ke ujung dan berjalan pada perangkat/remote perlu memberikan output setidaknya sekali setiap 100 milidetik.

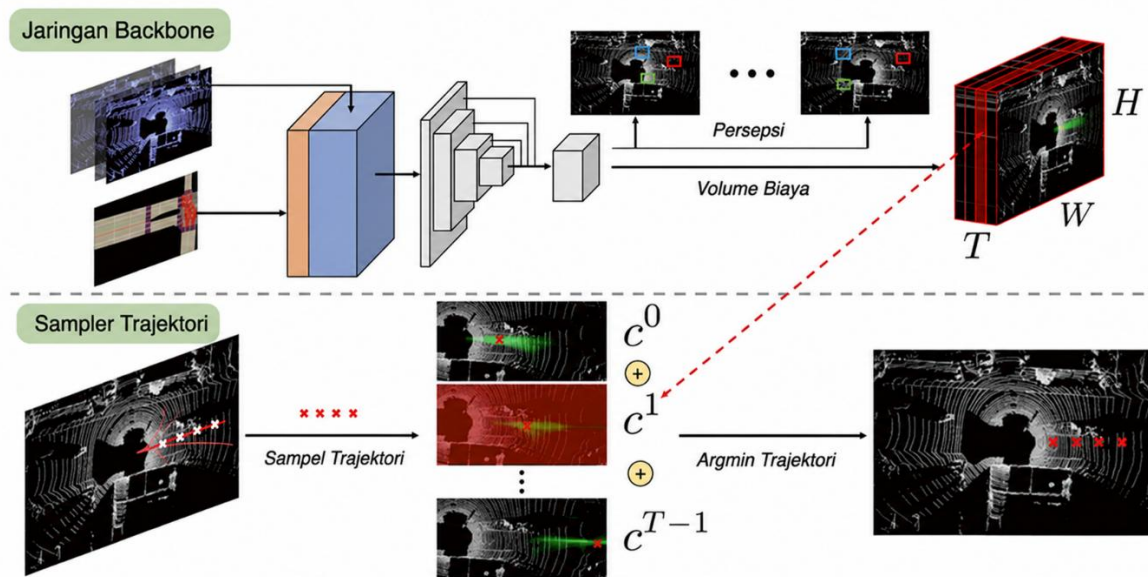
Komponen utama dari sistem ini adalah:

1. Persepsi: Rangkaian sensor, algoritma persepsi, dan lokalisasi
2. Prediksi
3. Perencanaan dan Kontrol

Banyak dari komponen ini telah dibahas secara mendalam di bab-bab sebelumnya, jadi bab ini hanya membahas detail terkait pengemudian otomatis yang mendorong desain sistem ini.

Sistem Pengemudian Mandiri Ujung-ke-Ujung (E2E)

Beberapa perusahaan mendekati atau telah melakukan pekerjaan dalam mendorong sistem pengemudian mandiri yang merupakan jaringan saraf ujung-ke-ujung. Comma AI, Wayve, dan Waabi adalah beberapa contohnya. Perencana saraf ujung-ke-ujung biasanya mengambil sensor dan peta sebagai input dan menghasilkan lintasan dengan biaya terendah atau tindakan saat ini pada setiap langkah waktu. Dibandingkan dengan desain klasik, perencana E2E adalah jaringan saraf neuron tunggal, dengan output perantara yang sangat mudah diinterpretasikan untuk verifikasi keselamatan. Dibandingkan dengan perencana tradisional, perencana E2E bisa jadi lebih tidak aman, karena jaringan saraf memiliki kendali penuh atas tindakan mobil. Karena masalah keselamatan, teknologi ini lebih sulit untuk diproduksi. Meskipun sebagian besar mobil pengemudian mandiri di jalan saat ini adalah perencana klasik, peningkatan pembelajaran mendalam yang berkelanjutan menunjukkan bahwa perencana E2E akan menjadi kemungkinan nyata di masa depan. Gambar 8.3 menunjukkan perencana neural E2E awal yang diusulkan oleh kelompok riset Uber ATG.



Gambar 8.3. Perencana neural yang dapat diinterpretasikan dari ujung ke ujung.

8.3 SISTEM PERSEPSI PADA KENDARAAN OTONOM

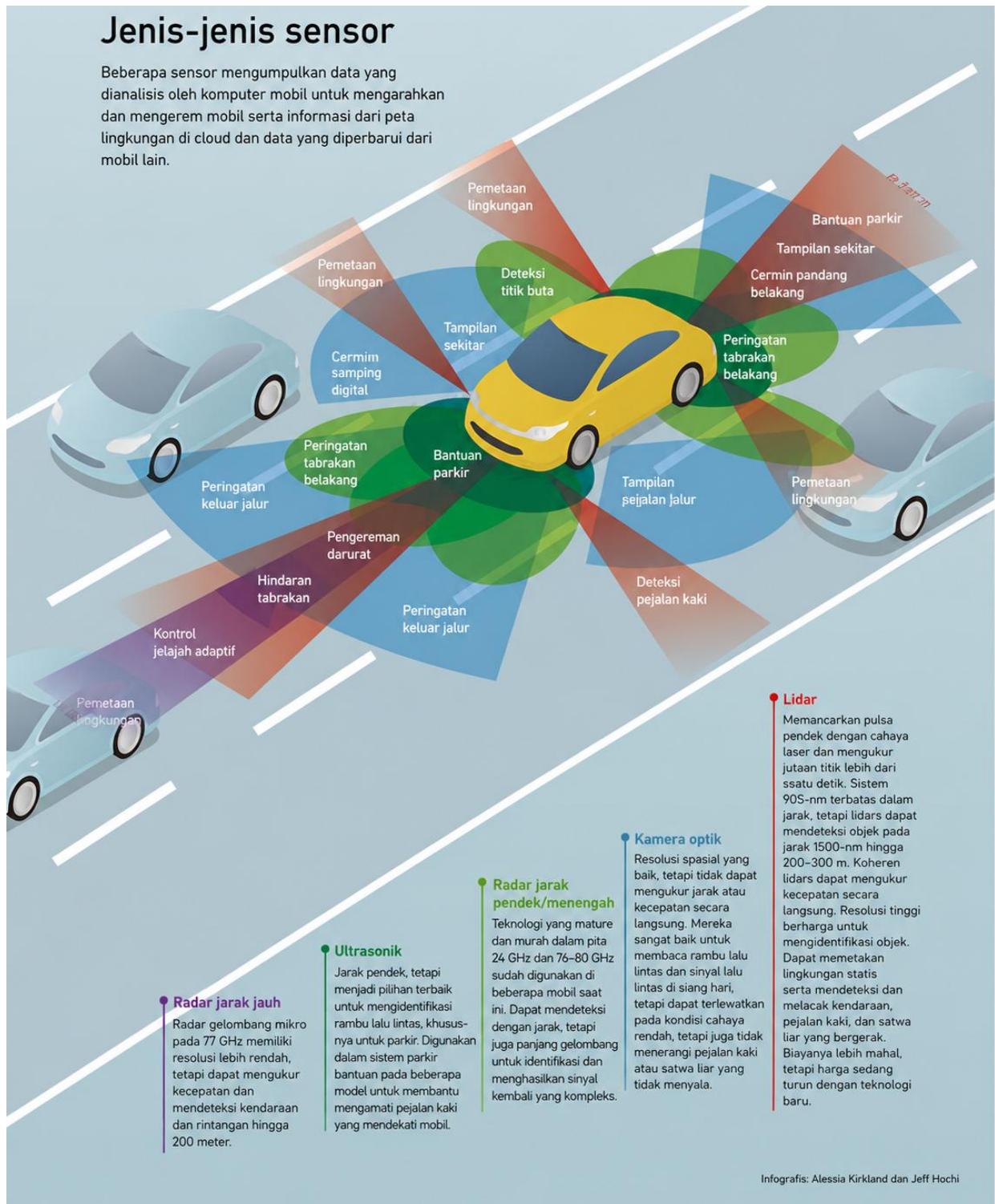
Tujuan persepsi mengemudi sendiri adalah untuk membangun model lingkungan di sekitar kendaraan yang menyadari semua objek statis, dinamis, dan informasi kontekstual adegan apa pun yang diperlukan untuk membuat keputusan mengemudi.

- Objek statis meliputi entitas yang tidak bergerak seperti lampu lalu lintas, garis jalur, bangunan, pohon, dan sebagainya. Entitas statis dari persepsi dapat digunakan untuk melokalisasi pada peta yang telah dimuat sebelumnya dengan mencocokkan artefak statis yang diamati oleh kendaraan dengan artefak yang disimpan dalam peta definisi tinggi yang telah dimuat sebelumnya. Mereka juga dapat digunakan untuk menentukan apakah peta tersebut salah, seperti jika ada konstruksi atau jika peta tersebut sudah usang, untuk merencanakan ulang secara dinamis atau meminta bantuan jarak jauh.
- Objek dinamis meliputi agen yang bergerak seperti orang, kendaraan lain seperti mobil, sepeda, dan truk, hewan hidup, dan sebagainya. Niat objek juga dideteksi menggunakan pelacakan dan model prediksi niat spesifik.
- Informasi kontekstual adegan mencakup hubungan antar objek. Berikut beberapa contohnya:
 - Seorang pengendara sepeda dan sepedanya bergerak bersama
 - Jalur mana yang berisi objek mana
 - Hewan mana yang diikatkan pada orang mana
 - Apakah truk trailer merupakan objek artikulasi atau bukan

Untuk membantu persepsi lingkungan yang baik, ada tiga tahap yang terlibat:

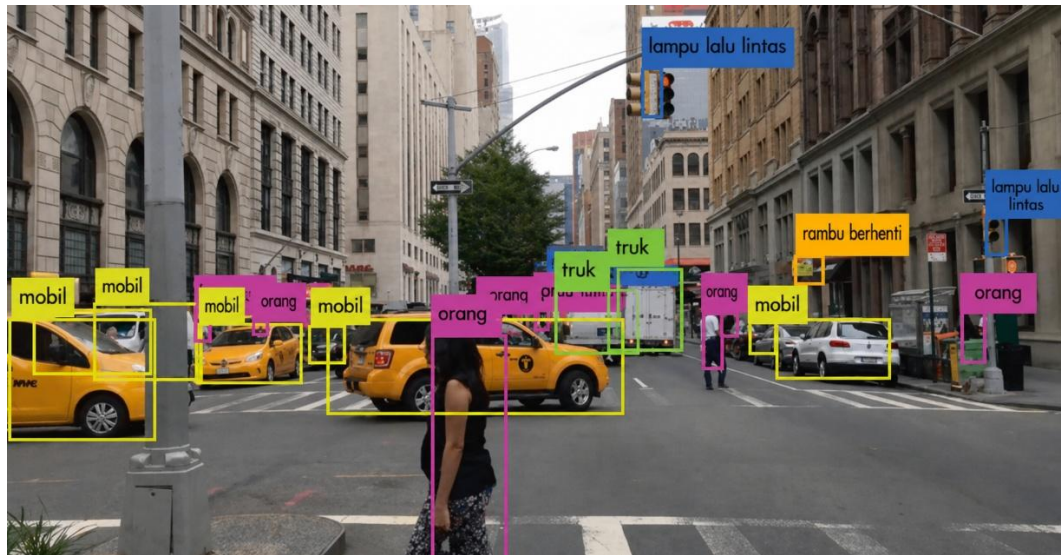
1. Penginderaan: Untuk mobil otonom, pertimbangan utama untuk rangkaian penginderaan adalah memastikan bahwa hanya ada sedikit titik buta, dan resolusi gambar kamera dan LiDAR padat. Mobil otonom modern juga dilengkapi dengan sensor audio (untuk mendeteksi kendaraan darurat yang datang, misalnya), IMU untuk odometri, radar untuk estimasi kecepatan, dan sebagainya. Gambar 8.4 menunjukkan

contoh rangkaian sensor pada sebuah mobil. Biasanya, kamera pandangan sekeliling digunakan untuk informasi jarak dekat, seperti penumpang yang naik ke pesawat dan trotoar, sedangkan LiDAR jarak jauh dengan jangkauan padat digunakan untuk informasi jarak jauh.



Gambar 8.4. “Jenis sensor” dari artikel oleh J. Hecht, “LiDAR untuk Mobil Otonom,” *Optics & Photonics News* 29(1), 26-33 (2018), dibuat oleh Alessia Kirkland, Manajer Senior dan Direktur Kreatif OPN.

2. Visi komputer: Algoritma visi komputer 2D dan 3D diterapkan pada citra sensorik yang digabungkan atau tidak digabungkan untuk memperoleh informasi relevan dari adegan dan objek, seperti yang telah dibahas sebelumnya di bagian ini. Bab 2 membahas jenis algoritma yang digunakan untuk mendeteksi informasi ini. Gambar 8.5 menunjukkan keluaran visi komputer tipikal di mana objek telah dideteksi oleh kotak pembatas.

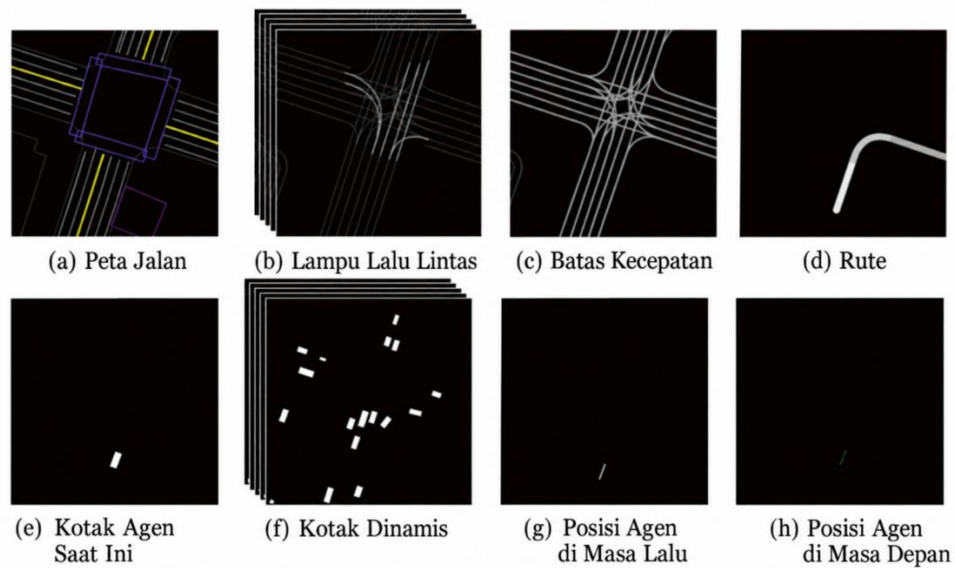


Gambar 8.5. Output visi komputer tipikal.

3. Fusi sensor: Output dari berbagai sensor digabungkan untuk menghilangkan objek hantu, meningkatkan kepercayaan objek sebenarnya, dan menggabungkan fitur objek. Misalnya, kedalaman LiDAR jauh lebih presisi daripada kedalaman stereo, jadi setelah suatu objek diklasifikasikan menggunakan gambar kamera, atribut kedalamannya diperbarui dari instance yang sesuai dalam gambar LiDAR.

8.4 PREDIKSI NIAT PADA KENDARAAN OTONOM

Sangat penting bagi mobil otonom untuk mengantisipasi tindakan agen lain. Hal ini dilakukan melalui prediksi niat. Dalam tumpukan perangkat lunak otonom terkini, prediksi niat dilakukan melalui jaringan saraf besar. Makalah ChauffeurNet Waymo mengusulkan salah satu sistem prediksi niat pertama, dengan input yang melibatkan peta, lampu lalu lintas, informasi batas kecepatan, dan agen lainnya, serta lintasan masa lalu untuk memprediksi lintasan masa depan, di mana semua input diubah menjadi gambar untuk diproses oleh jaringan saraf konvolusional. Input ini ditunjukkan pada Gambar 8.6.



Gambar 8.6. Memprediksi posisi agen di masa depan.

Arsitektur ChauffeurNet Waymo memprediksi tindakan agen jalan lainnya di masa depan menggunakan kombinasi jaringan saraf konvolusional dan rekuren. Inti dari model ini adalah FeatureNet yang memproses input (peta, lampu lalu lintas, dll.) dan mengubahnya menjadi fitur. Fitur-fitur ini kemudian dimasukkan ke dalam AgentRNN, yang memprediksi parameter mengemudi seperti arah, kecepatan, dan titik arah. Selain itu, Road Mask Net dan Perception RNN digunakan untuk memprediksi area yang dapat dilalui dan mendeteksi agen lain, masing-masing. Model itu sendiri dilatih dengan berbagai fungsi kerugian—kerugian arah, kerugian kecepatan, dan kerugian tabrakan—yang berfokus pada meminimalkan perbedaan antara prediksi model dan hasil aktual.

Arsitektur ini juga memiliki komponen memori, yang melacak lokasi agen di masa lalu. Ini meningkatkan prediksi model secara keseluruhan tentang posisi dan lintasan di masa depan. Dengan menggunakan jaringan saraf konvolusional untuk pemrosesan gambar dan jaringan saraf berulang untuk menangani data urutan, ChauffeurNet dapat memanfaatkan berbagai jenis data untuk memahami lingkungan mengemudi dan memprediksi tindakan agen di masa depan dalam skenario lalu lintas yang kompleks.

Meskipun ChauffeurNet memprediksi masa depan pengemudi, logika serupa dapat diterapkan untuk memprediksi niat agen lain juga. Prediksi niat sangat penting untuk situasi mengemudi seperti:

1. Persimpangan lampu lalu lintas, seperti persimpangan empat arah dan dua arah, di mana banyak agen menunjukkan niat mereka melalui sinyal tangan, lampu, dan cara lain.
2. Selama situasi perubahan jalur dan penggabungan, di mana penting untuk mengetahui apakah pihak lain memberi Anda hak jalan atau tidak.

8.5 PERENCANAAN JALUR PADA KENDARAAN OTONOM

Mengingat lingkungan sekitar agen, seperti lokasinya di peta, rintangan statis dan dinamis di sekitarnya, tujuannya, dan peraturan lalu lintas yang relevan, perencanaan adalah masalah menemukan lintasan optimal menuju pencapaian tujuan yang aman. Biaya yang terlibat selama perencanaan bermacam-macam:

1. Biaya tabrakan: Kecelakaan dengan agen atau rintangan statis apa pun memiliki biaya tertinggi dalam perencanaan dan harus dihindari sebagai prioritas utama.
2. Pelanggaran aturan lalu lintas: Melanggar batas kecepatan, lampu lalu lintas, aturan jalur HOV, dan aturan lalu lintas lainnya merupakan biaya tambahan. Dibandingkan dengan biaya tabrakan, biaya ini lebih rendah, yang berarti bahwa pelanggaran aturan lalu lintas atau batas kecepatan mungkin dilakukan untuk menghindari tabrakan.
3. Biaya jarak dan waktu: Mencapai tujuan dalam jalur dan waktu optimal.

Jenis rencana yang paling umum dalam kendaraan otonom adalah rencana berbasis pencarian. Ada juga rencana neural dan rencana pembelajaran penguatan dalam pengaturan studi. Kelemahan utama rencana neural adalah ketidakmampuan untuk menginterpretasikan pengambilan keputusan serta kurangnya kendali atas tindakan akhir yang dipilih oleh mobil.

Segmen kode berikut menunjukkan bagaimana rencana berbasis pencarian (pencarian A* yang digunakan dalam contoh di sini) dapat digunakan. Perhatikan bahwa ini adalah versi rencana yang sangat disederhanakan. Segmen jalur mengacu pada elemen dalam peta, dan segmen jalur yang mungkin ditempati oleh agen. Untuk contoh ini, peta diinterpretasikan sebagai grafik segmen jalur, dengan titik awal dan tujuan. Perhatikan bahwa segmen kode ini dimodifikasi berdasarkan kode pencarian A* yang disediakan di sini.

```
// A* search algorithm for self-driving to go from lane segment start
// to lane segment destination
1. Initialize an to_explore with the first lane segment where the
   ego is, and set its f = 0
2. Initialize the explored list
3. while the to_explore list is not empty
   a) find the lane segment with the least f on the to_explore
      list, call it "q"
   b) pop q off the to_explore list
   c) generate q's successors, ie, lane segments that can be
      traversed to from q and set their parents to q
   d) for each successor
      I. if successor is the goal, stop search
      II. else, compute both g and h for successor
          successor.g = q.g + cost to go from q to successor
          successor.h = cost to go from goal to successor distance
          (eg Manhattan distance cost + other costs discussed
           above)
```

- successor.f = successor.g + successor.h
- III. if a node with the same position as successor is in the to_explore list which has a lower f than successor, skip this successor
- IV. if a node with the same position as successor is in the explored list which has a lower f than successor, skip this successor otherwise, add the node to the to_explore list end (for loop)
- e) push q on the explored list end (while loop)

Demikian pula, persimpangan dapat dimodelkan sebagai berikut:

```
// Modeling a traffic intersection in C++
class Intersection
{
    public:
    Type type_of_intersection; // type of intersection,
                               // say traffic
                               // light, 4 way stop, 2 way stop
    LaneSegment[] lanesegments; // list of lane segments
                               // that are
                               // part of the intersection
                               // and their
// relationships to each other
    Crosswalk[] crosswalks; // crosswalks that are
                             // part of the
                             // intersection and agent IDs
                             // belonging to them
    TrafficLights[] trafficlights; // traffic light
                                   // elements and
                                   // their states that are
                                   // part of the intersection
    Agent[] order_of_arrival; // order of arrival
                              // of agents
    Agent[] agents;           // all agents and their
                              // attributes
                              // that are relevant to the
                              // intersection
};
```

8.6 KESELAMATAN DAN KEANDALAN KENDARAAN OTONOM

Keselamatan rekayasa merupakan konsep inti dalam membangun produk robotika yang tangguh. Hal ini sangat penting terutama untuk sistem otonom, karena kendaraan

beroperasi di antara manusia, terkadang dengan kecepatan tinggi, dan kesalahan dapat berakibat fatal.

Keselamatan dan Sistem AI

Standar keselamatan sebelumnya dirancang dengan mempertimbangkan sistem tradisional dan tidak begitu cocok untuk sistem otonom berbasis AI. Misalnya, sistem ML dapat tidak terduga, menggunakan data yang tidak lengkap, dan seringkali sulit untuk diinterpretasikan, sehingga sulit untuk menjamin keselamatannya menggunakan standar yang ada. Salah satu mode kegagalan utama sistem ML adalah kerusakan saat berinteraksi dengan manusia atau lingkungannya dengan cara yang tidak terduga. Karena metode ML sangat bergantung pada kelengkapan data, memastikan bahwa skenario dan data penting diperhitungkan sangatlah penting. Jika data hilang untuk skenario penting yang mungkin dihadapi kendaraan, perilakunya bisa tidak terduga dan lebih mungkin mengalami kerusakan.

Sebuah kelompok di ISO (Organisasi Internasional untuk Standardisasi) berfokus pada pengembangan standar untuk memastikan bahwa sistem AI yang dikembangkan aman dan andal untuk aplikasi yang melibatkan interaksi manusia, seperti kendaraan otonom. Untuk memperhitungkan tantangan unik yang dihadapi sistem ML dalam kendaraan otonom, kita juga dapat mengambil prinsip-prinsip dari komunitas keselamatan AI. Tujuan keselamatan AI adalah untuk memastikan bahwa sistem AI bertindak adil dan tidak melakukan diskriminasi dalam bentuk apa pun. Beberapa ide kunci keselamatan AI, seperti yang dijelaskan dalam, termasuk memastikan bahwa AI adil (tidak melakukan diskriminasi), transparan (mudah dipahami), dan aman (terlindung dari peretas).

Pertimbangan Keselamatan

Bagian ini membahas beberapa aspek keselamatan umum yang perlu dipertimbangkan dalam hal desain perangkat keras dan algoritma mobil otonom. Banyak dari poin-poin ini berfokus pada perusahaan yang membangun mobil otonom yang akan digunakan secara publik, tetapi ide-ide ini juga berlaku dalam lingkungan studi.

Keandalan Sensor

Mobil otonom dapat menggunakan banyak sensor, termasuk kamera, radar, dan LiDAR, untuk memahami lingkungannya. Untuk memastikan sensor bekerja dengan andal, akan sangat membantu jika menggunakan beberapa jenis sensor untuk menutupi berbagai kelemahan, melakukan kalibrasi secara berkala untuk memastikan sensor bekerja dengan baik, dan menggunakan teknik fusi sensor untuk memvalidasi informasi dari berbagai sensor. Dengan begitu, jika satu sensor mengalami gangguan, sensor lain dapat menyeimbangkannya.

Ketahanan Algoritma

Algoritma yang digunakan dalam mobil otonom harus mampu bekerja dalam berbagai skenario mengemudi, baik itu jalanan kosong maupun lingkungan perkotaan yang kompleks. Untuk menangkap kasus-kasus ekstrem dan mengukur adaptasi, algoritma ini harus diuji dalam berbagai lingkungan dunia nyata dan simulasi. Penting juga untuk menerapkan metode pengaman dan metode pemantauan untuk mendeteksi dan mengatasi anomali atau kasus ekstrem yang dihadapi kendaraan yang tidak diperhitungkan dalam desain algoritma.

Keamanan Siber

Penggunaan sensor dan teknologi lain dalam mobil otonom membuatnya lebih rentan terhadap serangan siber, yang dapat membahayakan keselamatan dan privasi. Untuk melindungi dari serangan ini, penting untuk memiliki metode enkripsi untuk melindungi data dan membangun sistem deteksi untuk memantau dan menanggapi potensi ancaman. Perusahaan harus melakukan audit keamanan secara berkala dan berkolaborasi dengan pakar keamanan siber untuk mengidentifikasi dan mengurangi risiko terkait data.

Pengambilan Keputusan Etis

Penting bagi perusahaan untuk memiliki pedoman etika yang jelas yang menentukan bagaimana keputusan etis harus dibuat oleh mobil otonom dan memastikan bahwa pedoman ini tercermin dalam cara kendaraan menangani kasus-kasus ekstrem. Untuk membangun kepercayaan pada perusahaan dan publik, memiliki dokumentasi yang jelas dan berkomunikasi secara terbuka tentang topik-topik ini sangat penting.

Interaksi Manusia-Mesin

Interaksi antara mobil otonom dan orang-orang yang berada di dalamnya harus intuitif dan ramah pengguna. Saat mendesain kendaraan ini, akan sangat berguna untuk memiliki sinyal visual dan auditori yang jelas untuk memberi tahu penumpang tentang tindakan dan tujuan kendaraan. Ini dapat mencakup mendesain kontrol dan antarmuka yang mudah dipahami dan dioperasikan, serta memungkinkan penumpang untuk mengambil alih kendali dalam keadaan darurat. Memastikan interaksi manusia-mesin ini mudah dipahami sangat penting untuk kepercayaan publik dan menjaga keselamatan.

Mekanisme Pengaman

Mobil otonom harus memiliki mekanisme untuk menangani kegagalan yang tidak terduga dengan aman. Ini termasuk mengembangkan protokol darurat komprehensif yang memungkinkan kendaraan untuk berhenti dengan aman atau kembali ke keadaan aman jika terjadi masalah. Memiliki sistem kontrol manual atau cadangan yang dapat mengambil alih jika sistem utama gagal dapat berguna selama penerapan untuk memastikan pengoperasian yang aman berlanjut bahkan jika terjadi kerusakan.

Privasi Data

Mengingat banyaknya data yang dikumpulkan oleh mobil otonom, melindungi privasi pengguna sangat penting. Saat melatih model atau berbagi data, anonimisasi data tersebut dapat berguna untuk mencegah identifikasi individu. Mengembangkan standar tentang penggunaan data dan kebijakan privasi, serta membagikan standar ini secara publik dapat membangun kepercayaan dengan orang-orang yang menggunakan mobil otonom.

8.7 RINGKASAN

Bab ini membahas poin-poin berikut:

- Terdapat peluang ekonomi yang sangat besar untuk membangun mobil otonom, karena industri transportasi bernilai Rp 7 kuantiliun.
- Desain sistem untuk kendaraan ini dapat dipecah menjadi algoritma persepsi, prediksi, dan perencanaan, bersama dengan pertimbangan perangkat keras. Terdapat

peningkatan minat pada perencana neural ujung-ke-ujung, tetapi mereka menghadapi tantangan dalam hal keselamatan dan komersialisasi.

- Tujuan persepsi dalam mobil otonom adalah untuk membangun model lingkungan yang menyadari objek statis, dinamis, dan informasi kontekstual di sekitar kendaraan.
- Prediksi berfokus padaantisipasi tindakan agen lain di lingkungan menggunakan jaringan neural besar.
- Perencanaan melibatkan pencarian lintasan optimal bagi kendaraan untuk mencapai tujuannya dengan aman sambil meminimalkan berbagai biaya, termasuk biaya tabrakan, biaya pelanggaran peraturan lalu lintas, dan biaya waktu.
- Keselamatan adalah aspek penting dari mobil otonom, termasuk keselamatan AI, ketahanan sistem ML, dan pertimbangan keselamatan umum untuk perusahaan dan studi.

Bab selanjutnya membahas tugas-tugas umum dalam robotika industri, termasuk perakitan pasak-dalam-lubang, pengelasan, dan otomatisasi gudang. Bab ini membahas berbagai jenis robot, pertimbangan desain sistem utama untuk perangkat keras, penggunaan perangkat lunak (seperti CNN untuk menggenggam), penskalaan pembelajaran penguatan untuk menggenggam, serta metrik kinerja dan keselamatan.

BAB 9

PEMBELAJARAN MENDALAM DALAM ROBOT INDUSTRI

Robot industri melakukan tugas-tugas di lingkungan industri dan manufaktur seperti perakitan, pengelasan, dan pengemasan. Tugas-tugas ini berulang dan padat karya serta membutuhkan tingkat presisi dan konsistensi. Dengan menggunakan pembelajaran mendalam (*deep learning*), kita dapat meningkatkan kemampuan robot untuk memahami dan mengerti lingkungannya serta melakukan tugas-tugas kompleks ini. Dengan menggunakan data sensor, robot dapat mengenali objek, merencanakan jalur, dan menjalankan tugas. Pembelajaran mendalam sangat penting, karena memungkinkan robot untuk beradaptasi dengan lingkungan yang berubah dan tidak terstruktur serta melakukan tugas-tugas dalam pemeliharaan dan operasi prediktif bersama manusia.

Teknik pembelajaran mendalam digunakan oleh beberapa bisnis di lingkungan manufaktur dan industri. Salah satunya adalah Amazon, yang menggunakan metode visi komputer untuk mengarahkan armada robot di pusat pemenuhan pesanan mereka. Menurut Amazon, “Pada tahun 2022, 1 miliar paket, atau seperdelapan dari semua pesanan yang kami kirimkan kepada pelanggan di seluruh dunia, disortir oleh Robin, salah satu sistem penanganan robot Amazon.” Dengan menciptakan lingkungan kolaboratif antara robot dan manusia, mereka telah melihat peningkatan efisiensi, pengurangan kesalahan, dan peningkatan jumlah pesanan yang dapat mereka proses di pusat pemenuhan pesanan mereka. Sebelum membahas metode pembelajaran mendalam yang digunakan, bagian berikut membahas tugas-tugas yang umum dilakukan dalam lingkungan manufaktur.

9.1 TUGAS UTAMA ROBOT INDUSTRI

Robotika manufaktur dan industri digunakan dalam berbagai tugas untuk mengotomatisasi proses, meningkatkan efisiensi, dan meningkatkan presisi. Beberapa tugas umum dalam robotika manufaktur dan industri meliputi pengambilan dan penempatan, pemasangan pasak, dan pengelasan.

Pengambilan dan Penempatan

Dalam otomatisasi industri, mengambil suatu objek dari satu posisi dan menempatkannya di posisi lain dikenal sebagai "pengambilan dan penempatan". Di bidang seperti otomotif, elektronik, dan logistik, tugas ini sangat penting bagi robot untuk diselesaikan, karena membantu menempatkan komponen pada jalur perakitan dan secara umum memanipulasi barang-barang berat. Sistem pengambilan dan penempatan biasanya terdiri dari lengan robot atau manipulator dengan efektor ujung seperti penjepit, cangkir vakum, atau lampiran magnetik. Untuk penempatan dan kontrol, ia juga memiliki sensor, sistem penglihatan, dan pengontrol. Umumnya, lengan robot FANUC digunakan dalam perakitan otomotif, robot ABB digunakan untuk manufaktur elektronik, dan robot Amazon digunakan di pusat pemenuhan. Gambar 9.1 adalah ilustrasi dari sistem robot pengambil dan penempat.



Gambar 9.1. Sistem pick-and-place oleh Covariant AI.

Pasak di Lubang

Dalam tugas pasak di lubang, suatu barang harus dimasukkan dengan hati-hati ke dalam lubang atau wadah yang benar. Produksi elektronik, perakitan mobil, dan manufaktur pesawat terbang semuanya bergantung pada pekerjaan ini. Misalnya, dalam manufaktur elektronik, Anda mungkin perlu menempatkan chip pada papan sirkuit, yang membutuhkan presisi ekstrem yang dapat dibantu oleh robot seperti seri IRB ABB.

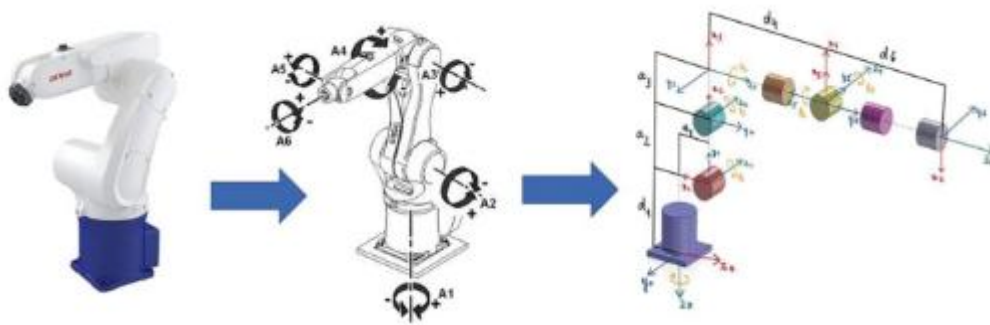
Di industri otomotif, produsen pesawat terbang menggunakan robot seperti M-20iA FANUC atau robot KR Quantec KUKA untuk merakit komponen. Komponen sistem pasak di lubang meliputi lengan robot, efektor ujung (biasanya penjepit atau alat khusus lainnya), sensor, dan sistem penglihatan untuk penyelarasan dan pemosisian. Gambar 9.2 menggambarkan sebuah sistem dengan robot pasak-dalam-lubang sebagai contoh.



Gambar 9.2. Robot melakukan tugas memasukkan pasak ke dalam lubang.

Pengelasan

Mesin las robot digunakan untuk mengelas material bersama-sama dengan cara melelehkan dan kemudian menggabungkannya menggunakan panas. Ini adalah tugas penting di berbagai sektor termasuk industri pesawat terbang, konstruksi, dan otomotif. Di industri pesawat terbang, Anda membutuhkan las yang presisi dan berkualitas tinggi, dan di sektor konstruksi, robot digunakan untuk tugas-tugas seperti pengelasan balok baja. Di industri otomotif, produsen seperti Tesla, General Motors, dan Toyota menggunakan robot untuk mengelas bodi mobil. Hal ini dapat mempercepat waktu produksi dan, dalam beberapa kasus, meningkatkan kualitas las. Robot umum yang digunakan untuk tugas-tugas ini adalah KR Quantec dari KUKA, seri FANUC ARC Mate, dan robot IRB 6700 dari ABB. Sistem robot pengelasan menggabungkan lengan robot dengan obor atau alat, sumber daya pengelasan, sensor, dan sistem penglihatan untuk pelacakan. Gambar 9.3 menunjukkan ilustrasi robot pengelasan.



Gambar 9.3. Contoh robot pengelasan yang menunjukkan struktur mekanik dan rotasi sambungan.

Tugas Gudang

Selain meningkatkan proses industri tertentu, robot juga dapat digunakan untuk meningkatkan tugas pergudangan secara keseluruhan. Robot dapat melakukan pergerakan dan pemenuhan inventaris, pelacakan inventaris, dan pemindaian, yang dapat membantu gudang dan pusat distribusi modern beroperasi seefisien mungkin.

- **Pergerakan dan pemenuhan inventaris:** Manajemen dan penanganan barang yang efektif di dalam gudang atau pusat distribusi diperlukan untuk tugas pergerakan dan pemenuhan inventaris. Untuk memenuhi pesanan klien dan mempertahankan tingkat stok, robot digunakan untuk memilih, mengemas, menyortir, dan mengirimkan produk. Biasanya, kendaraan berpemandu otomatis (AGV)—yaitu robot bergerak yang memiliki konveyor, rak, penjepit, dan sensor—digunakan untuk transportasi dan pemenuhan inventaris. Mereka dapat dihubungkan ke sistem komando terpusat yang mengontrol pesanan dan informasi inventaris.
- **Pelacakan dan pemindaian inventaris:** Robot digunakan untuk pelacakan dan pemindaian inventaris untuk memantau dan mengontrol tingkat inventaris,

memastikan akurasi, dan melacak pergerakan barang di seluruh gudang. Robot ini memiliki berbagai sensor, termasuk kamera, pemindai kode batang, pembaca RFID, dan sistem penglihatan komputer untuk pemantauan, pengenalan objek, navigasi, dan penghindaran rintangan.

9.2 JENIS DAN PLATFORM ROBOT INDUSTRI

Robot yang digunakan dalam lingkungan industri hadir dalam berbagai bentuk dan ukuran, tergantung pada lingkungan dan tugas yang diselesaikan.

Robot Industri Mandiri

- **Robot artikulasi:** Robot ini memiliki sambungan putar dan antara tiga hingga enam derajat kebebasan. Karena alasan ini, mereka fleksibel dan dapat berputar ke berbagai arah. Mereka memiliki alas, lengan, pergelangan tangan, efektor ujung, sistem kontrol, dan sensor/aktuator, yang memungkinkan untuk mengontrol pergerakan. Umumnya jenis robot ini digunakan dalam perakitan dan pengelasan. Contoh robot seperti ini adalah seri Kuka KR Quantec, yang umumnya digunakan dalam manufaktur untuk pengelasan bodi mobil.
- **Robot Paralel:** Robot ini memiliki tiga lengan, yang terhubung ke platform dasar menggunakan sambungan universal. Lengan-lengan itu sendiri ditempatkan dalam konfigurasi jajaran genjang, yang memungkinkan mereka untuk bergerak cepat. Karena alasan ini, robot paralel ideal untuk pengemasan dan perakitan presisi tinggi. Robot FlexPicker ABB adalah contohnya; robot ini telah digunakan di pabrik pengolahan makanan untuk pengambilan dan penempatan serta untuk menyiapkan pizza beku.
- **Robot Gantry:** Dengan tiga sumbu kontrol linier yang ditempatkan pada sudut 90 derajat satu sama lain, robot ini dikenal mudah dikendalikan dan berguna ketika ruang terbatas. Robot gantry terdiri dari aktuator linier, rel pemandu, dan sistem kontrol. Robot Gantry TMF Güdel adalah contohnya; robot ini umumnya digunakan di jalur perakitan otomotif untuk memanipulasi bagian-bagian berat.

Robot Kolaboratif (Cobot)

Seperti namanya, robot ini bekerja dengan manusia dan dirancang dengan mempertimbangkan keselamatan. Robot kolaboratif ini berguna dalam tugas-tugas di mana fleksibilitas dan bekerja di lapangan bersama manusia sangat penting.

- **Perakitan:** Robot kolaboratif ini dapat melakukan tugas-tugas seperti memasang sekrup, memasang, dan menyambung bagian-bagian, seringkali bekerja bersama pekerja manusia.
- **Penanganan material:** Mereka membantu dalam memindahkan, menyortir, dan mengemas material, yang mengurangi beban fisik pada pekerja manusia dan meningkatkan throughput.
- **Pengambilan dan penempatan:** Robot kolaboratif ini digunakan untuk mengambil barang dari satu lokasi dan menempatkannya di lokasi lain, yang berguna dalam logistik dan pergudangan.

- **Inspeksi kualitas:** Dengan menggunakan sistem penglihatan, cobot ini dapat memeriksa produk untuk mengetahui adanya cacat. Mereka sangat efektif dalam melakukan tugas-tugas rutin, yang membebaskan operator manusia untuk tugas-tugas lain yang dibutuhkan.
- **Keselamatan:** Dengan menggunakan sensor, cobot ini dapat mendeteksi kontak dengan manusia atau benda dan berhenti atau melambat dalam skenario ini untuk menghindari cedera. Mereka sering dirancang untuk menerapkan gaya yang kecil, yang dapat membantu dalam kasus di mana terjadi kontak yang tidak disengaja dengan manusia.
- **Perangkat keras:** Sensor gaya digunakan untuk mendeteksi gaya dan torsi yang diterapkan oleh cobot dan menyesuaikan pergerakan sesuai kebutuhan. Kamera dan sensor lain yang disebutkan dalam Bab 2 digunakan untuk mengenali manusia dan benda, menavigasi di lingkungan mereka, dan melakukan tugas-tugas seperti inspeksi kualitas. Seringkali, ada GUI atau antarmuka yang mudah digunakan bagi manusia untuk dengan mudah memprogram atau berinteraksi dengan robot jika terjadi perubahan pada lingkungan atau tugas.

Cobot populer adalah seri UR dari Universal Robots, yang sering digunakan di ruang yang lebih kecil. Robot ini ringan dibandingkan robot lain, cepat, dan dapat menangani berbagai aplikasi dengan mudah. Robot ini umumnya digunakan untuk pengelasan dan penumpukan objek.

Robot Bergerak

Terdapat dua jenis robot bergerak yang umum: Kendaraan Berpemandu Otonom (*Autonomous Guided Vehicles/AGV*) dan Robot Bergerak Otonom (*Autonomous Mobile Robots/AMR*). Perbedaan utamanya adalah, sementara AGV mengikuti jalur tetap yang telah ditentukan sebelumnya menggunakan pita, kabel, atau rel di lingkungan fisik, AMR dapat menggunakan sensor seperti kamera dan LiDAR untuk menavigasi lingkungannya secara lebih dinamis, tanpa memerlukan panduan fisik.

- **AGV:** Ini adalah robot bergerak yang mengikuti jalur tetap dan rute yang telah ditentukan sebelumnya untuk mengangkut material dari satu lokasi ke lokasi lain. Mereka seringkali memerlukan modifikasi lingkungan fisik menggunakan jalur panduan atau penanda agar dapat beroperasi. Mereka umumnya digunakan di gudang/pusat distribusi, di mana Anda mengikuti jalur yang sama untuk mengangkut suatu objek dari satu lokasi ke lokasi lain. Mereka terdiri dari sistem navigasi yang dapat menavigasi lingkungan mereka dan kemudian mengikuti jalur yang telah ditentukan, menggerakkan motor untuk pergerakan, dan berbagai sensor untuk mendeteksi rintangan apa pun. Contohnya adalah Robotika Amazon, yang biasanya mengangkut barang di dalam pusat pemenuhan mereka. AGV ini mengikuti jalur yang telah ditentukan untuk menerima dan mengirimkan barang inventaris dan mengurangi tugas berulang yang perlu dilakukan manusia.
- **AMR:** Robot ini menggunakan kamera, LiDAR, dan algoritma untuk menavigasi lingkungan mereka dengan cara yang lebih fleksibel. AMR dapat digunakan dalam logistik dan manufaktur untuk mengangkut material dan mengambil serta

menempatkan barang dan mengirimkannya di dalam fasilitas. AMR memiliki sistem navigasi dengan pemetaan, kemampuan lokalisasi, sensor untuk mendeteksi dan menghindari rintangan, dan perencanaan jalur. AMR Fetch Robotics umumnya digunakan di gudang untuk mengambil dan mengangkut barang secara otonom.

Robot Humanoid

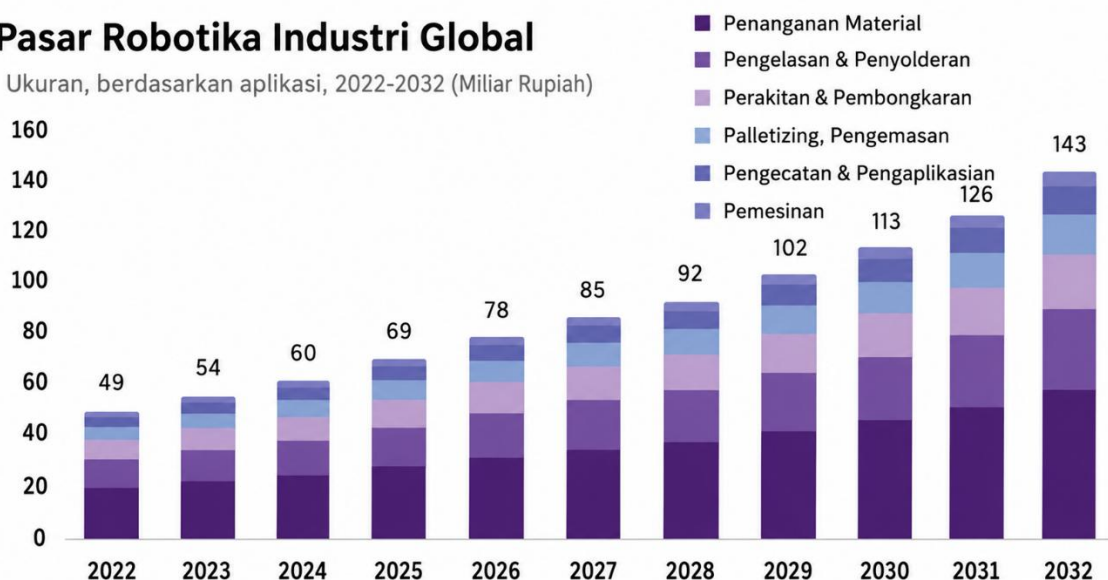
Robot humanoid dirancang untuk menyelesaikan tugas seperti yang dilakukan manusia, dengan kepala, badan, lengan, dan kaki. Robot ini terdiri dari aktuator yang membantu menggerakkan anggota tubuh dan persendian robot, sensor yang digunakan untuk memberikan umpan balik tentang lingkungan, sistem kontrol yang memproses data sensor, mengelola aktuator yang bergerak, dan efektor akhir yang berinteraksi dengan benda untuk melakukan tugas-tugas tertentu seperti menggenggam atau memanipulasi benda. Baru-baru ini perusahaan Figure membangun robot humanoid dan menandatangani perjanjian dengan BMW untuk menggunakan robot mereka dalam manufaktur otomotif. Demikian pula, perusahaan seperti Tesla sedang membangun robot humanoid untuk bekerja bersama manusia di pabrik mereka. Untuk informasi lebih lanjut mengenai desain perangkat keras dan perangkat lunak robot humanoid ini, kami sarankan untuk membaca Bab 10.

9.3 TREN DAN PERTUMBUHAN INDUSTRI ROBOTIKA

Menurut laporan McKinsey, “Pasar otomatisasi secara keseluruhan berkembang pesat: setidaknya beberapa sumber ahli memperkirakan pengiriman robot akan meningkat hingga 50 persen setiap tahun hingga tahun 2030, dengan otomatisasi gudang tumbuh lebih dari 10 persen per tahun”.

Pasar Robotika Industri Global

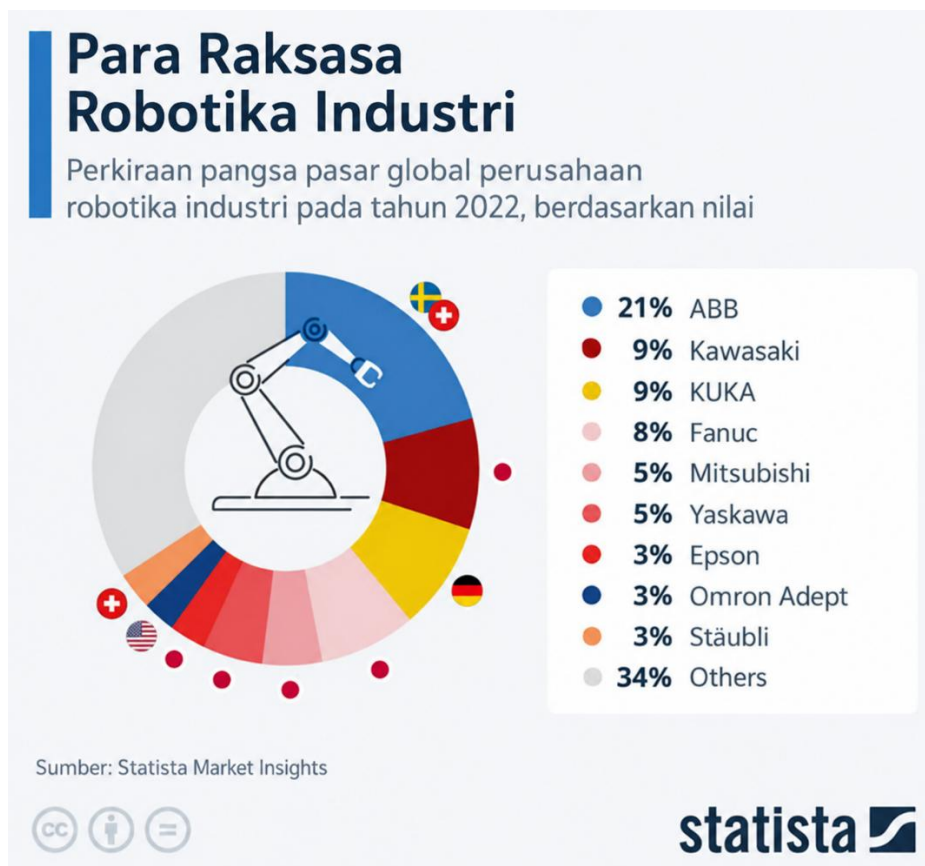
Ukuran, berdasarkan aplikasi, 2022-2032 (Miliar Rupiah)



Gambar 9.4. Pertumbuhan pasar robot industri dan aplikasi utama dalam pengelasan, penyolderan, perakitan, dan sebagainya.

Pasar robotika industri di seluruh dunia bernilai 54 miliar pada tahun 2023 dan diperkirakan akan tumbuh menjadi 142,8 miliar. Pertumbuhan pasar ini dan rincian aplikasinya ditunjukkan pada Gambar 9.4. Ada beberapa alasan meningkatnya minat pasar, termasuk:

- **Penurunan harga:** Karena biaya produksi yang lebih rendah di wilayah dengan tenaga kerja yang lebih murah, robot menjadi lebih mudah diakses.
- **Robot:** Terdapat peningkatan variasi robot dan perangkat lunak yang lebih baik dengan presisi dan mobilitas yang lebih tinggi, yang memungkinkan robot untuk digunakan di berbagai industri manufaktur.
- **Tenaga kerja:** Meningkatnya biaya tenaga kerja secara global juga membuat investasi dalam robotika menjadi lebih menarik karena perusahaan berupaya melakukan otomatisasi untuk mempertahankan daya saing dan mengatasi kekurangan keterampilan.



Gambar 9.5. Perusahaan yang memproduksi robot yang umum digunakan dalam tugas-tugas industri.

Menurut data terbaru dari Market Insights Statista, ABB memimpin pasar dengan pangsa pasar global sebesar 21 persen pada tahun 2022, KUKA dan Kawasaki Jepang memiliki pangsa pasar sebesar 9 persen, dan pemain penting lainnya adalah Fanuc dengan pangsa pasar 8 persen, Mitsubishi, dan Yaskawa, yang masing-masing memiliki pangsa pasar 5 persen. Rinciannya ditunjukkan pada Gambar 9.5.

Perusahaan manufaktur robot ini mendesain lengan atau platform yang digunakan oleh perusahaan lain. Ada juga perusahaan seperti Boston Dynamics, yang memanfaatkan metode pembelajaran mendalam untuk robot mereka, termasuk robot mereka, Spot, yang dapat digunakan untuk inspeksi industri.

9.4 DESAIN SISTEM UNTUK ROBOT PENGAMBIL DAN PENEMPATAN

Setelah Anda memahami tugas-tugas umum dalam robotika industri, bagian ini menjelaskan cara mendesain sistem untuk menyelesaikan tugas-tugas ini, khususnya tugas pengambil dan penempatan. Tugas pengambil dan penempatan sangat penting karena merupakan tugas spesifik yang biasanya merupakan bagian dari pencapaian tujuan manufaktur yang lebih besar, seperti merakit komponen, mengemas produk, dan memilah material. Robot pengambil dan penempatan yang paling umum meliputi:

- **Lengan robot:** Lengan robot lima sumbu ini digunakan untuk sebagian besar tugas pengambil dan penempatan umum yang melibatkan pengambilan dan penempatan objek dalam satu bidang. Untuk memutar objek, Anda memerlukan lengan enam sumbu atau lengan tujuh sumbu untuk memutar objek sebelum menjatuhkan/menempatkannya di suatu tempat.
- **Cartesian:** Robot ini dapat bergerak pada sumbu x, y, dan z dan menggunakan aktuator linier untuk mengontrol sabuk, bola, dan sebagainya, yang dapat membantu dalam pemosisian. Robot-robot ini sering digunakan untuk penanganan material, operasi CNC, elektronik, dan makanan karena andal dan memberikan manipulasi yang lebih presisi.
- **Delta:** Robot ini memiliki tiga atau empat lengan yang terhubung pada sebuah alas dan memberikan kecepatan dalam tugas pengambilan dan penempatan.

Komponen Perangkat Keras

Komponen utama robot pengambilan dan penempatan meliputi lengan, efektor ujung, sistem kontrol, dan sensor.

- **Lengan:** Lengan itu sendiri dapat berkisar dari desain lima sumbu untuk tugas-tugas dasar, hingga lengan enam sumbu dan tujuh sumbu, yang dapat membantu melakukan rotasi dan perubahan orientasi yang lebih kompleks. Lengan memiliki kombinasi motor dan aktuator, yang memberikan kontrol dan sensor yang digunakan untuk mengelola tekanan/gaya.
- **Efektor ujung:** Efektor ujung terpasang di ujung lengan robot dan merupakan bagian dari lengan yang digunakan untuk menggenggam, menahan, dan memanipulasi objek menggunakan pengambilan dan penempatan. Beberapa efektor ujung yang umum meliputi:
 - *Gripper vakum:* Gripper ini menciptakan vakum yang memungkinkan Anda untuk mengangkat objek. Gripper ini ideal untuk objek yang relatif datar, seperti kaca dan lembaran logam. Gripper ini umumnya digunakan dalam bidang elektronik dan pengemasan karena alasan tersebut.

- **Gripper mekanis:** Ini bisa berupa gripper dua atau tiga jari yang biasanya dirancang untuk menangani berbagai bentuk dan ukuran objek. Seringkali gripper ini digunakan untuk mengambil dan menempatkan komponen otomotif dan elektronik konsumen. Karena mampu memanipulasi berbagai objek, gripper ini umumnya digunakan untuk menggenggam dan memanipulasi.
- **Sistem kontrol:** Ini adalah "otak" dari robot pick-and-place karena digunakan untuk mengelola gerakan dan tindakan yang dilakukan lengan robot. Input diambil dari sensor dan perintah dikirim ke aktuator untuk mengeksekusi tindakan. Beberapa pengontrol umum meliputi:
 - **PID:** Pengontrol ini terus menerus menyesuaikan pergerakan robot berdasarkan umpan balik dari lingkungan. Mereka menggunakan tiga komponen: komponen proporsional, yang mengoreksi perbedaan ketika robot keluar jalur dengan menerapkan koreksi yang proporsional terhadap kesalahan, komponen integral, yang menghilangkan kesalahan berulang yang umum dengan mengakumulasi kesalahan masa lalu dan menyesuaikan jalur untuk mengoreksi offset ini, dan komponen derivatif, yang memprediksi perubahan dan menyesuaikan posisi sehingga Anda tidak melewati posisi target.
 - **ML:** Pengontrol berbasis jaringan saraf dilatih pada kumpulan data yang lebih besar tentang pergerakan dan hasil robot. Robot dapat mempelajari berbagai tindakan pengambilan dan penempatan yang berbeda dan idealnya beradaptasi dengan lingkungan apa pun tempat ia beroperasi. Pada kenyataannya, pelatihan model ini membutuhkan data yang ekstensif dan dapat sulit untuk digeneralisasikan. Banyak studi sedang berupaya untuk meningkatkan hal ini melalui data simulasi dan pembelajaran berbasis agen melalui RL.
- **Sensor:** Sensor penglihatan dan sensor gaya biasanya digunakan dalam tugas pengambilan dan penempatan. Sensor penglihatan membantu mengidentifikasi dan menemukan objek sebelum mengambilnya, dan sensor gaya memastikan bahwa robot menggenggam objek dengan jumlah gaya yang sesuai.

Semua komponen ini berinteraksi bersama untuk menyelesaikan tugas pengambilan dan penempatan. Sensor digunakan pada awalnya untuk memahami lingkungan, sistem kontrol memberikan perintah untuk mengeksekusi tugas berdasarkan lokasi awal dan target yang terdeteksi, dan sensor digunakan selama penggenggamannya untuk memastikan cengkaman yang cukup diterapkan untuk mengeksekusi tugas.

Komponen Perangkat Lunak

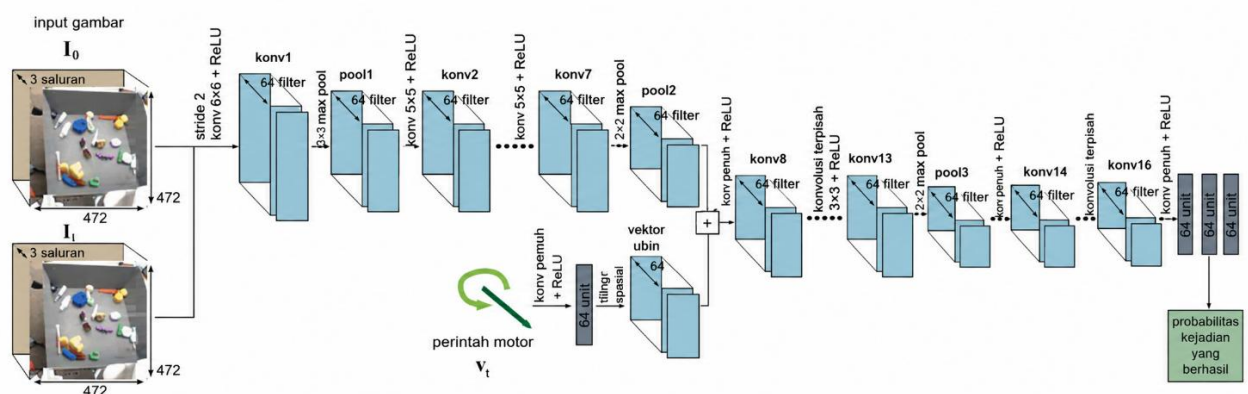
Selama bertahun-tahun, telah banyak metode pembelajaran mendalam yang dikembangkan untuk melakukan penggenggamannya. Mengapa pembelajaran mendalam bahkan baik untuk tugas seperti pengambilan dan penempatan? Mengingat sifat berulang dari tugas pengambilan dan penempatan, banyak demonstrasi dapat dikumpulkan. Ini menyediakan kumpulan data yang beragam untuk melatih model pembelajaran mendalam. Model-model ini perlu digeneralisasi ke berbagai morfologi objek dan lingkungan, yang merupakan tantangan bagi metode tradisional. Misalnya, menangani objek lunak membutuhkan kehati-

hatian untuk menghindari kerusakan, dan model pembelajaran mendalam dapat mempelajari cara-cara spesifik untuk menggenggam objek tersebut dan memastikan objek tersebut mendarat dengan benar di tempat yang ditentukan. Kemampuan beradaptasi dan presisi ini menjadikan pembelajaran mendalam sebagai pendekatan ideal untuk meningkatkan efisiensi dan akurasi tugas pengambilan dan penempatan di lingkungan visual yang berubah dan berbeda. Pendekatan umum adalah menggunakan Jaringan Saraf Konvolusional (CNN), yang dijelaskan dalam Bab 2 untuk menjalankan tugas menggenggam.

Jaringan Saraf Konvolusional untuk Menggenggam

Metode populer yang menggunakan CNN untuk menggenggam disajikan dalam makalah ini. Sebagai input, mereka menerima gambar yang diambil sebelum robot mencoba menggenggam objek apa pun (tanpa penjepit) dan gambar input yang diambil pada waktu tertentu t selama proses menggenggam. Gambar-gambar ini digabungkan dan diproses menggunakan beberapa lapisan filter konvolusional. Identy adalah agar setiap lapisan mengekstrak fitur yang berbeda dari gambar, termasuk tepi, bentuk, tekstur lingkungan, dan objek spesifik yang akan digenggam. Ukuran filter cukup kecil (3x3, 5x5, dll.) dan banyak yang diterapkan per lapisan.

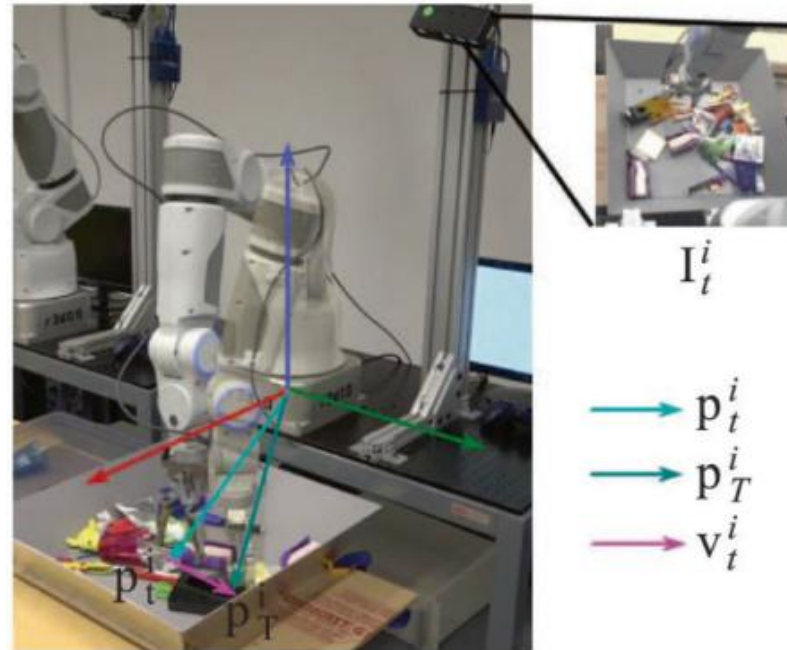
Setelah lapisan konvolusi, terdapat lapisan pooling untuk mengurangi ukuran peta fitur dan lapisan terhubung penuh di bagian akhir yang menerima peta fitur yang telah diratakan sebagai vektor dan membuat prediksi. Di sini, jaringan menghasilkan probabilitas yang menunjukkan keberhasilan menggenggam suatu objek. Aspek penting dari pendekatan ini adalah bahwa ia menerima perintah motor yang diusulkan (menggambarkan pergerakan lengan robot yang direncanakan) dan mempertimbangkannya bersamaan dengan gambar masukan dari adegan untuk menentukan keberhasilan genggaman. Arsitektur ini ditunjukkan pada Gambar 9.6.



Gambar 9.6. Arsitektur yang digunakan untuk CNN untuk pengambilan benda oleh robot dalam.

Metode ini menggunakan data pelatihan dari robot fisik nyata yang mencoba mengambil benda, merekam gambar, pose, dan gerakan pada setiap langkah waktu, dan kemudian mengevaluasi keberhasilan pengambilan benda untuk menghasilkan sampel pelatihan berlabel.

Setiap sampel mencakup gambar, pose saat ini hingga pose akhir, dan keberhasilan pengambilan benda secara keseluruhan. Pengaturan untuk mengumpulkan data pelatihan pengambilan benda ditunjukkan pada Gambar 9.7.



Gambar 9.7. Pengaturan genggam mencakup beberapa langkah waktu, yang sesuai dengan gambar dan pose.

Aspek penting lain dari metode ini adalah mekanisme penyajian. Metode ini menggunakan jaringan terlatih dari langkah sebelumnya untuk menentukan perintah motorik, yang akan meningkatkan kemungkinan keberhasilan genggam.

Secara konseptual, bagian metode ini mengoptimalkan perintah melalui pendekatan berbasis pengambilan sampel, memastikan sambil tetap berada dalam batasan, ia menyesuaikan posisi penjepit berdasarkan probabilitas keberhasilan yang diprediksi. Pada akhirnya, pendekatan ini mampu menggenggam objek secara efektif, bahkan di lingkungan yang berantakan.

Banyak studi telah dibangun di atas ini atau mengambil arah lain menggunakan CNN untuk menggenggam, tetapi pada dasarnya semua ini melakukan hal yang sama—mereka menggunakan CNN untuk memproses gambar adegan, mengekstrak fitur yang relevan, memprediksi perintah motorik optimal untuk menggenggam, dan kemudian mengevaluasi probabilitas perintah ini untuk berhasil memandu tindakan robot.

Penskalaan RL untuk Penggenggam Robot

Pembelajaran Penguatan (*Reinforcement Learning*/RL) telah menunjukkan keberhasilan dalam aplikasi robot individual seperti menggenggam dan manipulasi. Salah satu tantangan utama dalam meningkatkan skala sistem ini adalah memastikan bahwa sistem tersebut dapat digunakan secara andal dalam kehidupan sehari-hari. Sebuah makalah oleh Google[28] mengatasi masalah ini dengan menempatkan 23 robot yang diaktifkan RL selama

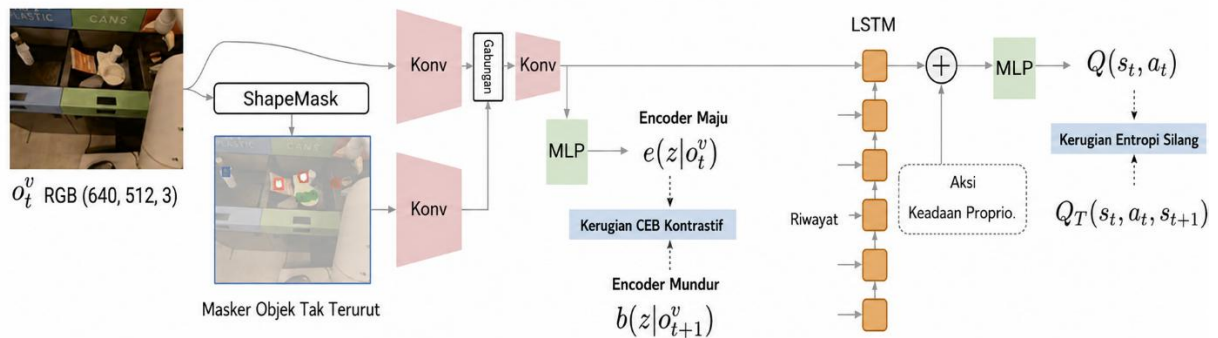
dua tahun di kantor mereka untuk memilah sampah/daur ulang dan kemudian mengumpulkan data ini. Sistem mereka menggabungkan RL mendalam dari data dunia nyata dengan data simulasi untuk melakukan penggengaman. Meskipun ini dilakukan dalam konteks pemilahan sampah, ini bertindak sebagai studi kasus tentang bagaimana ide-ide yang telah Anda pelajari di seluruh buku ini tentang simulasi, RL, dan metode persepsi dapat digabungkan untuk memecahkan tantangan manipulasi robot di dunia nyata.

Alur kerja mereka menggunakan tiga komponen utama:

- Kebijakan terprogram di mana mereka mendeteksi objek, merencanakan posisi genggam, dan menghasilkan lintasan untuk mencapai objek sebagai cara untuk mulai mengumpulkan data dalam simulasi dan dunia nyata.
- Pelatihan RL dimulai dalam simulasi dan metode simulasi-ke-nyata digunakan untuk menjembatani kesenjangan antara simulasi dan dunia nyata.
- Mereka mulai melatih tugas-tugas dengan berbagai tingkat kesulitan, dengan memilah sampah sebagai tugas tersulit. Awalnya tugas-tugas dimulai dengan lebih mudah dan tingkat keberhasilannya lebih tinggi, yang secara bertahap berubah menjadi tugas yang lebih sulit. Dengan memperkenalkan tugas-tugas yang lebih mudah di awal pelatihan, tujuannya adalah agar model dapat belajar dan mampu beradaptasi dengan tugas-tugas yang lebih menantang.

PI-QT-Opt digunakan untuk melatih kebijakan akhir pada dataset lengkap yang dikumpulkan dari simulasi dan dunia nyata. Arsitektur fungsi Q menerima gambar RGB dari adegan yang memiliki objek yang tidak terurut dan masker bentuk objek dalam adegan tersebut dihasilkan. Fitur diekstrak dari gambar dan masker menggunakan lapisan konvolusi.

Fitur dari masker dan gambar asli digabungkan. Satu encoder, yang disebut forward encoder, digunakan untuk memproses gambar saat ini dan background encoder memproses gambar berikutnya sebagai cara untuk memprediksi keadaan di masa mendatang. Lapisan LSTM digunakan untuk menjaga konteks dari waktu ke waktu pada pengamatan dan tindakan. Fungsi Q menggunakan keadaan dan tindakan saat ini untuk memprediksi imbalan di masa mendatang dan menggunakan enam langkah waktu terbaru. Output akhir diberikan sebagai nilai prediksi, yang memberi tahu Anda keberhasilan yang diharapkan dari tindakan tertentu. Arsitektur ini ditunjukkan secara lebih detail pada Gambar 9.8.



Gambar 9.8. Gambaran umum sistem PI-QT-Opt yang digunakan untuk memilah sampah.

Aspek penting yang disorot dalam karya ini adalah kebutuhan akan pengumpulan data skala besar dan kombinasi data online dan offline untuk meningkatkan kinerja model untuk pengambilan benda oleh robot.

Sistem Pengambilan Benda Multimodal

Arah yang menarik untuk membangun model dasar untuk robotika disajikan oleh Covariant AI, dalam metode mereka yang disebut RFM-1. Metode ini sangat cocok untuk aplikasi manufaktur, logistik, dan pergudangan, karena dapat beradaptasi dengan perubahan dan berkolaborasi serta berkomunikasi dengan pekerja manusia. Aspek penting adalah bahwa Covariant telah mampu mengumpulkan beragam data robot mereka yang melakukan tugas di lingkungan dinamis yang menantang dengan banyak bentuk dan jenis objek yang berbeda, seperti yang ditunjukkan pada Gambar 9.9. Beberapa aspek unik dari RFM-1 meliputi:

- **Multimodal:** Model itu sendiri adalah transformer yang melakukan tokenisasi modalitas yang berbeda seperti teks, data sensor, dan data kamera ke dalam ruang bersama, yang memungkinkan prediksi token berikutnya. Ini berarti model dapat menerima modalitas input apa pun dan menghasilkan output dalam modalitas apa pun juga.
- **Pemahaman fisika:** Dengan menggunakan teknik yang memungkinkan pembelajaran model dunia, metode ini dapat memperoleh pemahaman yang lebih dalam tentang fisika, yang dapat membantu dalam melakukan tindakan realistis dan bereaksi terhadap perubahan di lingkungan. Hal ini dilakukan dengan menghasilkan video di mana token input dari gambar awal dan tindakan robot digunakan untuk memprediksi token video di masa mendatang.
- **Pemrograman ulang bahasa:** Tantangan besar dalam robotika adalah memprogram ulang robot ketika terjadi perubahan. Seringkali, ini adalah tugas yang sulit yang hanya dapat dilakukan dengan sukses oleh orang yang awalnya memprogram robot tersebut. Melalui perintah bahasa alami, seseorang dapat menginstruksikan robot untuk melakukan suatu tugas dan robot dapat meminta bantuan ketika membutuhkannya.

Dialog bolak-balik ini berguna di dunia nyata, di mana perubahan lingkungan memerlukan perubahan tugas atau pemrograman ulang robot dengan cara apa pun.



Gambar 9.9. Sistem RFM-1 oleh Covariant AI melakukan berbagai tugas.

Metode ini menunjukkan kekuatan pengumpulan data skala besar dan manfaat yang dapat diberikan oleh data multimodal untuk meningkatkan interaksi dengan robot di lingkungan pick-and-place. Untuk mempelajari lebih lanjut, kami sarankan membaca rilis ini tentang RFM-1.

Metrik Kinerja

Keberhasilan robot pick-and-place yang didukung pembelajaran mendalam terutama diukur dengan berbagai metrik kinerja, efisiensi, dan keandalan. Metrik ini membantu menilai seberapa baik robot melakukan tugas yang dirancang untuknya. Beberapa metrik penting meliputi:

- **Akurasi deteksi objek:** Seberapa akurat robot dapat mendeteksi objek sangat penting karena hal itu menentukan kemampuan robot untuk mengenali dan mengkategorikan benda-benda di sekitarnya. Akurasi deteksi objek yang tinggi meningkatkan kemampuan robot untuk bekerja secara efisien dalam berbagai lingkungan.
- **Tingkat keberhasilan pengambilan:** Kemampuan robot untuk mengambil dan memindahkan barang tanpa menjatuhkannya atau merusaknya harus diukur. Operasi pengambilan dan penempatan harus andal dan tepat, yang membutuhkan algoritma yang memiliki tingkat keberhasilan pengambilan yang tinggi.
- **Waktu siklus:** Waktu yang dibutuhkan robot untuk mengeksekusi tugas pengambilan dan penempatan lengkap, termasuk mendeteksi objek, perencanaan, pengambilan, dan penempatan, diukur dalam hal waktu siklus. Mengurangi waktu siklus ini penting untuk meningkatkan output dan produktivitas di lingkungan industri.
- **Tingkat penghindaran tabrakan:** Kemampuan robot untuk menghindari menabrak objek atau rintangan saat menavigasi lingkungannya diukur dengan tingkat

penghindaran tabrakannya. Untuk keselamatan dan efektivitas, mengurangi tabrakan dan kecelakaan nyaris celaka sangat penting.

- **Tingkat penyelesaian tugas:** Ini mengukur proporsi tugas pengambilan dan penempatan yang berhasil diselesaikan dari semua tugas yang dicoba. Ini mengukur seberapa andal dan efisien robot dalam menyelesaikan tugas yang diberikan kepadanya.

9.5 ASPEK KESELAMATAN DALAM ROBOTIKA INDUSTRI PEMBELAJARAN MENDALAM

Saat menggunakan pembelajaran mendalam untuk robotika industri, keselamatan harus selalu diutamakan untuk melindungi manusia dan menghindari kecelakaan. Berikut beberapa poin keselamatan yang perlu diingat:

- Gunakan sensor keselamatan untuk mengawasi lingkungan sekitar robot dan mendeteksi gerakan atau rintangan yang tiba-tiba. Jika diperlukan, sensor ini dapat memulai penghentian darurat.
- Saat menggunakan cobot, pastikan mereka mengikuti peraturan keselamatan dengan merancang fitur keselamatan seperti pemantauan kecepatan dan sambungan pembatas gaya untuk mengurangi kemungkinan kecelakaan saat berinteraksi dengan manusia.
- Terapkan kemampuan perangkat lunak yang terkait dengan keselamatan, seperti pelacakan jalur dan deteksi tabrakan. Fitur-fitur ini dapat membantu robot mengidentifikasi dan bereaksi terhadap objek atau perubahan yang tidak terduga di sekitarnya.

9.6 RINGKASAN

Dalam bab ini, Anda telah mempelajari hal-hal berikut:

- Robot industri melakukan tugas-tugas seperti pengambilan dan penempatan, pemasangan pasak, pengelasan, dan operasi gudang. Pembelajaran mendalam memungkinkan robot untuk beradaptasi dengan lingkungannya, meningkatkan kemampuan mereka untuk mengidentifikasi objek, merencanakan jalur, dan menjalankan tugas di lingkungan yang tidak terstruktur.
- Robot industri hadir dalam berbagai bentuk tergantung pada desain dan aplikasinya. Robot mandiri meliputi robot artikulasi dengan sambungan putar, robot paralel untuk perakitan kecepatan tinggi, dan robot gantry. Robot kolaboratif (cobot) bekerja bersama manusia dengan fitur keselamatan seperti sambungan pembatas gaya. Robot bergerak, seperti AGV dan AMR, menangani pergerakan inventaris, sementara robot humanoid meniru gerakan manusia.
- Robotika di fasilitas manufaktur berkembang karena penurunan biaya, peningkatan kemampuan, dan kebutuhan untuk mengotomatisasi tugas-tugas kompleks.
- Bab ini menjelaskan desain sistem untuk robot pengambilan dan penempatan yang berguna untuk perakitan, penyortiran, dan pengemasan. Desain ini mencakup lengan robot yang dilengkapi dengan berbagai efektor ujung, seperti penjepit dan cangkir

vakum. Pembelajaran mendalam digunakan untuk meningkatkan akurasi genggaman melalui visi komputer dan pembelajaran penguatan, yang memungkinkan robot untuk beradaptasi dengan beragam lingkungan dan objek. Sistem multimodal menggunakan berbagai data sensorik, yang dapat berguna bagi robot untuk memproses informasi visual dan taktil.

- Metrik kinerja seperti akurasi deteksi objek, tingkat keberhasilan genggaman, waktu siklus, penghindaran tabrakan, dan tingkat penyelesaian tugas sangat penting untuk meningkatkan efisiensi dan keandalan robot industri secara real-time. Memastikan keselamatan sama pentingnya dengan mewajibkan robot untuk memiliki sensor untuk deteksi tabrakan, penghentian darurat, dan sambungan pembatas gaya untuk mencegah kecelakaan saat bekerja bersama manusia.

Bab selanjutnya membahas pengaturan perangkat keras dan perangkat lunak untuk robot humanoid, termasuk pendekatan untuk manipulasi, berjalan, teleoperasi, dan manipulasi seluruh tubuh.

BAB 10

ROBOTIKA HUMANOID

Robotika humanoid berkaitan dengan studi tentang robot yang mengambil bentuk faktor manusia. Istilah ini secara longgar diterapkan pada robot yang mencoba untuk meniru bentuk faktor manusia dalam berbagai cara, dan peniruan yang tepat tidak diperlukan. Biasanya, ini berarti robot bergerak (berjalan dan terkadang menggunakan roda), dengan tinggi manusia (5 hingga 6 kaki, meskipun terkadang hanya setinggi anak-anak), dan platform bimanual, dengan penjepit atau tangan multi-jari yang cekatan sebagai efektor akhir. Menggerakkan humanoid lengkap—dengan kaki, tangan multi-jari, dan mobilitas badan dan kepala—mungkin lebih sulit, karena alasan itulah para ahli robotika dapat memilih untuk menyederhanakan platform dengan berbagai cara (roda sebagai pengganti kaki, penjepit sebagai pengganti tangan, dll.).

10.1 ALASAN PENGEMBANGAN ROBOT HUMANOID

Ada beberapa alasan mengapa robotika humanoid merupakan strategi populer dalam robotika:

1. **Dunia fisik sering dirancang untuk bentuk faktor manusia.** Pegangan cangkir dibuat agar dapat dimanipulasi oleh jari, kaleng Coca-Cola pas di tangan, meja dapur berada pada ketinggian yang dapat dijangkau oleh manusia, dan sebagainya. Ini menunjukkan mengapa memiliki faktor bentuk manusia dapat memungkinkan robot untuk melakukan banyak tugas berbeda di lingkungan yang dirancang untuk manusia tanpa modifikasi, dan secara teoritis dapat mencapai kemahiran manusia dalam keterampilan fisik.
2. **Model skala internet mentransfer kecerdasan umum ke robotika.** Namun, sebagian besar data yang mendukung model skala internet adalah internet itu sendiri dan internet adalah kumpulan pengalaman manusia. Kebijakan kontrol optimal mungkin berbeda berdasarkan tubuh dan kecerdasan mungkin lebih mudah ditransfer dari tubuh manusia ke faktor bentuk humanoid daripada ke faktor bentuk lainnya, karena kesenjangan perwujudan lebih kecil.
3. **Banyak pekerjaan fisik yang berharga di dunia dilakukan oleh manusia.** Humanoid menunjukkan peluang untuk melengkapi pasar tenaga kerja dan memberikan nilai ekonomi tanpa perlu menciptakan pasar baru.

10.2 PENDEKATAN HUMANOID DALAM ROBOTIKA

Pendekatan humanoid terhadap robotika lebih terintegrasi secara vertikal. Hal ini kontras dengan pendekatan lintas-wujud yang ekstrem, yang membangun kecerdasan fisik umum yang dapat ditransfer ke tubuh robot mana pun. Kemungkinan besar, penyelesaian masalah robotika membutuhkan kedua pendekatan tersebut:

1. Kepemilikan ujung-ke-ujung pada penyebaran dan data dengan integrasi vertikal dan optimasi tubuh robot.
2. Model dasar lintas-wujud besar yang menyediakan kecerdasan umum dan mengurangi kebutuhan data untuk melatih tubuh yang digerakkan dengan baik.

Humanoid juga merupakan tubuh umum yang dapat melakukan berbagai macam tugas dengan tubuh yang sama. Hal ini kontras dengan pendekatan membangun robot khusus untuk tugas-tugas khusus yang berulang. Sebagian besar robot yang digunakan di industri saat ini adalah robot khusus:

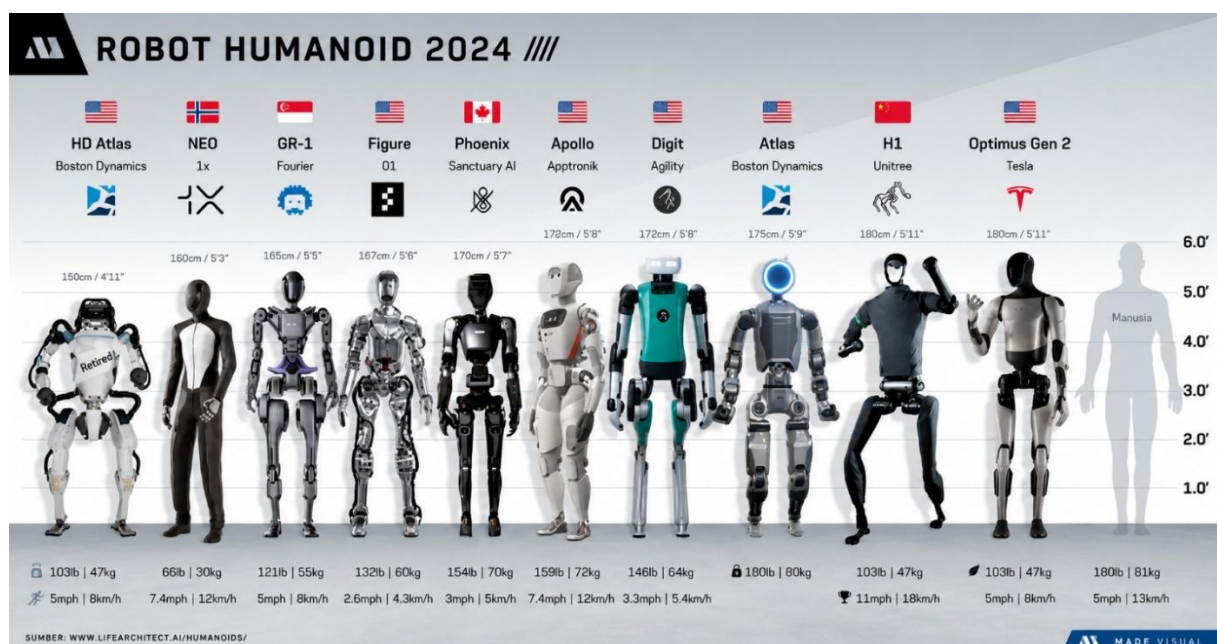
1. Spesialisasi tugas melalui lengan manipulator tujuan umum; misalnya, lengan ABB generik yang merakit Papan Sirkuit Cetak (PCB).
2. Spesialisasi perwujudan untuk tugas; misalnya, robot ular untuk inspeksi pipa.

Strategi humanoid melintasi spesialisasi tugas dan perwujudan menuju generalisasi.

10.3 EKOSISTEM DAN PASAR ROBOT HUMANOID

Ruang lingkup humanoid saat ini memiliki banyak pemain, seperti yang ditunjukkan pada Gambar 10.1:

1. Penyedia kecerdasan: Perusahaan yang mengirimkan model dasar, seperti Google, OpenAI, Mistral, Meta, dan sebagainya. Perusahaan-perusahaan ini membangun model dasar yang digunakan untuk pengembangan humanoid, memiliki tim yang melakukan studi humanoid, atau mereka mendanai/bermitra dengan perusahaan untuk melakukan studi humanoid.
2. Penyedia perangkat keras: Perusahaan yang membangun perangkat keras humanoid, seperti Boston Dynamics, Fourier Intelligence, Unitree, Apptronik, dan Agility.
3. Penyedia terintegrasi penuh: Perusahaan yang memecahkan masalah kecerdasan dan perangkat keras: Tesla, Figure AI, 1x Technologies, dan sebagainya.



Gambar 10.1. Pasar perangkat keras humanoid.

Perusahaan-perusahaan ini memiliki beberapa bahan utama yang berkontribusi pada kesuksesan mereka dalam pengembangan humanoid, termasuk:

1. **Modal:** Menciptakan robot canggih bisa mahal, jadi memiliki dana untuk mendukung studi, pengembangan, dan produksi robot humanoid sangat penting. Mengingat dampak dan pertumbuhan industri ini, investor meningkatkan pendanaan untuk perusahaan humanoid. Misalnya, Figure AI, sebuah startup yang berbasis di California, mengumpulkan Rp 6,75 triliun dalam putaran Seri B termasuk LG, Samsung, dan Microsoft. Demikian pula, Sanctuary AI, yang berbasis di Vancouver, telah mengumpulkan dana yang signifikan, termasuk investasi baru-baru ini dari Accenture Ventures dan Magna. Dana ini akan digunakan untuk memajukan pekerjaan mereka dalam robot humanoid serbaguna. Investasi yang meningkat dalam teknologi humanoid menyoroti meningkatnya kepercayaan pada potensi mereka dan sumber daya yang dibutuhkan untuk membawa robot-robot ini ke pasar.
1. **Model Dasar:** Model AI skala besar yang memungkinkan robot humanoid untuk memahami dan belajar dari sejumlah besar data sangat penting. Model dasar ini menunjukkan kemampuan kognitif robot humanoid, memungkinkan mereka untuk melakukan tugas secara otonom dan beradaptasi dengan situasi baru. Lebih lanjut, model umum skala besar ini membentuk dasar untuk disempurnakan untuk tugas-tugas khusus.
2. **Data:** Secara umum, semakin banyak data yang dapat diakses robot, semakin baik robot tersebut dapat belajar untuk melakukan tugas, mengenali objek, dan memahami perilaku manusia. Untuk robot humanoid, data ini dapat mencakup apa saja, mulai dari gambar dan video hingga teks dan data sensor dari lingkungan dunia nyata.
3. **Robot:** Konstruksi fisik robot humanoid melibatkan perangkat keras dan biomekanik seperti sensor dan aktuator untuk menciptakan mesin fisik yang dapat meniru tindakan manusia. Perusahaan seperti Boston Dynamics dan Tesla berinvestasi besar-besaran di bidang ini, mengembangkan robot yang dapat mengangkat benda berat atau melakukan gerakan kompleks lainnya dengan lincah.
4. **Komputasi:** Seiring bertambahnya ukuran model dan data, infrastruktur komputasi sangat penting untuk mendukung pelatihan dan pengujian model pada sejumlah besar data.

Pasar robot humanoid diproyeksikan tumbuh secara signifikan, dengan nilai perkiraan Rp 4,85 kuantiliun pada tahun 2035. Hal ini sebagian besar didorong oleh meningkatnya adopsi robot humanoid di lingkungan industri dan rumah tangga. Di industri seperti manufaktur, robot humanoid diperkirakan akan mengambil alih sekitar 35 persen tugas manusia saat ini, yang akan berkontribusi pada pasar senilai Rp 1,75 kuantiliun. Untuk rumah tangga, robot ini dapat menjadi hal yang umum untuk tugas-tugas seperti pembersihan dan perawatan lansia dan tumbuh menjadi pasar senilai Rp 2,8 kuantiliun. Seiring semakin banyaknya kemajuan teknologi dan penurunan harga, robot humanoid kemungkinan akan mengikuti jalur adopsi

yang serupa dengan kendaraan listrik, memungkinkan mereka menjadi bagian sehari-hari dari kehidupan kita dan meningkatkan industri serta rumah tangga.

10.4 DESAIN DAN PENGENDALIAN ROBOT HUMANOID

Bagian ini berisi gambaran umum tentang teknik yang digunakan untuk membangun dan mengendalikan humanoid.

Perangkat Keras

Pilihan desain perangkat keras:

1. **Mobilitas:** Ada dua pendekatan yang umum diterima untuk mobilitas humanoid:
 - a. *Humanoid berkaki:* Humanoid ini dilengkapi dengan kaki untuk berjalan. Keuntungan dari metode ini adalah humanoid dapat melintasi berbagai medan, lingkungan dengan trotoar atau tangga, dan sebagainya. Keuntungan kedua adalah jejak humanoid berkaki lebih kecil, dan ketangkasannya terhadap manipulasi seluruh tubuh lebih tinggi. Misalnya, ia dapat melintasi ruang yang berantakan dan mengarahkan kakinya dengan cara yang mendistribusikan momen untuk mengangkat benda berat, memanipulasi benda yang jauh di atas meja, dan sebagainya, yang dapat meningkatkan ruang kerja dan muatan humanoid. (Ketika manusia mengangkat kotak berat, mereka sering mengambil posisi berdiri yang lebar atau meletakkan satu kaki di depan untuk mendistribusikan beban.) Kekurangannya termasuk fakta bahwa berjalan adalah masalah kontrol yang lebih sulit dan memperkenalkan ketidakstabilan ke dalam sistem. Hal ini membuat kegagalan menjadi lebih dahsyat (jika humanoid jatuh, ia dapat merusak dirinya sendiri, benda-benda di sekitarnya, dan menyebabkan cedera). Pilihan desain yang menarik untuk kaki adalah lutut terbalik dari kelincahan.
 - b. *Humanoid beroda:* Bagian bawah tubuh humanoid adalah platform beroda, mirip dengan Eve dari 1X technologies. Keuntungan utama dari pilihan desain ini adalah menstabilkan robot menjadi lebih mudah. Roda juga memungkinkan robot menjadi lebih berat karena mendistribusikan berat tidak memerlukan kontrol khusus. Kekurangannya adalah robot mungkin tidak berguna di medan yang lebih keras dan lingkungan yang tidak rata. Selain itu, kemampuannya untuk bergerak melalui ruang yang berantakan dan melakukan manipulasi seluruh tubuh mungkin terbatas, sehingga membatasi ruang kerja dan fleksibilitasnya.
2. **Efektor ujung:** Humanoid dapat dilengkapi dengan berbagai efektor ujung:
 - a. *Gripper:* Gripper umumnya digunakan dalam robotika dan cukup mudah dikendalikan karena derajat kebebasan yang lebih rendah. Hal ini menjadikannya pilihan populer untuk humanoid.
 - b. *Tangan yang cekatan:* Humanoid juga dapat dilengkapi dengan tangan multi-jari. Tangan tiga jari dan lima jari adalah pilihan populer. Setiap jari menambahkan derajat kebebasan tambahan, yang kemudian harus dimodelkan dan dikendalikan, sehingga menambah kompleksitas kontrol.

- c. *Efektor ujung dengan umpan balik taktil*: Umpan balik gaya pada tangan memungkinkan kontrol proprioseptif, selain visual, dan berguna untuk memanipulasi objek yang mudah berubah bentuk dan rapuh. Ada konsekuensi berupa peningkatan biaya.
3. **Kamera pergelangan tangan dan kepala vs hanya kamera kepala**: Telah ditunjukkan bahwa penggabungan kamera yang dipasang di pergelangan tangan dapat meningkatkan kinerja manipulasi dibandingkan dengan hanya menggunakan kamera yang dipasang di kepala. Memiliki kamera pergelangan tangan memberikan umpan balik visual tambahan, meningkatkan kesadaran spasial dan akurasi dalam teleoperasi dan interaksi manusia-robot.
4. **Hanya kamera vs kamera dan kedalaman stereo/LiDAR**: Sistem hanya kamera lebih sederhana dan lebih murah tetapi menghadapi tantangan dalam persepsi kedalaman dan pengenalan objek. Teknik ML tingkat lanjut dapat membantu mengatasi beberapa keterbatasan ini. Sebagian besar desain humanoid di pasaran tidak menggunakan LiDAR untuk menjaga biaya material tetap rendah. Oleh karena itu, metode yang hanya menggunakan kamera sebagai sensor lebih disukai dan mungkin memerlukan pengembangan algoritma yang lebih banyak.
5. **Aktuator linier/rotasi, aktuator hidrolik**: Aktuator linier/rotasi lebih akurat daripada aktuator hidrolik. Namun, pengembangan dengan aktuator hidrolik juga lebih rumit karena perawatan sistem hidrolik dan potensi kebocoran cairan, dan sebagainya. Oleh karena itu, sebagian besar perusahaan robot humanoid, seperti Boston Dynamics, beralih ke motor listrik sepenuhnya dan menjauh dari sistem hidrolik.

Robot humanoid dapat memiliki parameter kontrol berikut:

1. **Posisi kepala/pinggul**: Parameter ini mengontrol posisi vertikal dan horizontal kepala dan pinggul, yang memungkinkan robot untuk mengorientasikan diri dan berinteraksi dengan lingkungannya dari berbagai sudut.
2. **Rotasi kepala**: Parameter ini mengontrol gerakan rotasi kepala robot, yang memungkinkan robot untuk melihat ke berbagai arah. Rotasi kepala penting untuk fokus pada objek atau area tertentu.
3. **Ekstensi/rotasi bahu**: Parameter ini mengatur gerakan ekstensi dan rotasi bahu robot. Parameter ini memposisikan lengan dengan benar untuk berbagai tugas seperti menjangkau atau mengangkat objek.
4. **Fleksi siku**: Parameter ini mengontrol pembengkokan dan pelurusan siku robot. Fleksi siku berguna untuk menyesuaikan panjang lengan dan posisi selama tugas manipulasi.
5. **Rotasi sendi pergelangan tangan**: Parameter ini memantau gerakan rotasi sendi pergelangan tangan. Kontrol sendi pergelangan tangan penting untuk tugas motorik halus.
6. **Buka-tutup sendi jari/gripper**: Parameter ini mengontrol gerakan masing-masing sendi jari atau pembukaan dan penutupan gripper robot. Ini memungkinkan robot untuk menggenggam, menahan, dan melepaskan objek dengan berbagai tingkat kekuatan dan presisi.

7. **Rotasi/ekstensi sendi badan:** Parameter ini memantau gerakan rotasi dan ekstensi badan robot. Ini memungkinkan penyesuaian postur tubuh bagian atas robot, yang penting untuk menjaga keseimbangan dan menjangkau berbagai area.
8. **Sendi pinggul:** Parameter ini mengontrol gerakan pinggul robot, seperti rotasi dan kemiringan. Kontrol sendi pinggul penting untuk berjalan, duduk, dan menyesuaikan cara robot berdiri.
9. **Fleksi lutut:** Parameter ini mengontrol pembengkokan dan pelurusan lutut robot. Fleksi lutut digunakan saat berjalan, berjongkok, dan menjaga stabilitas saat melakukan aktivitas.
10. **Rotasi sendi pergelangan kaki:** Parameter ini mengatur gerakan rotasi pergelangan kaki robot. Ini membantu robot beradaptasi dengan permukaan yang tidak rata dan menjaga keseimbangan saat berdiri atau bergerak.
11. **Posisi kaki:** Parameter ini mengontrol penempatan dan orientasi kaki robot. Menyesuaikan posisi kaki penting untuk stabilitas berjalan, postur, dan navigasi di berbagai medan.

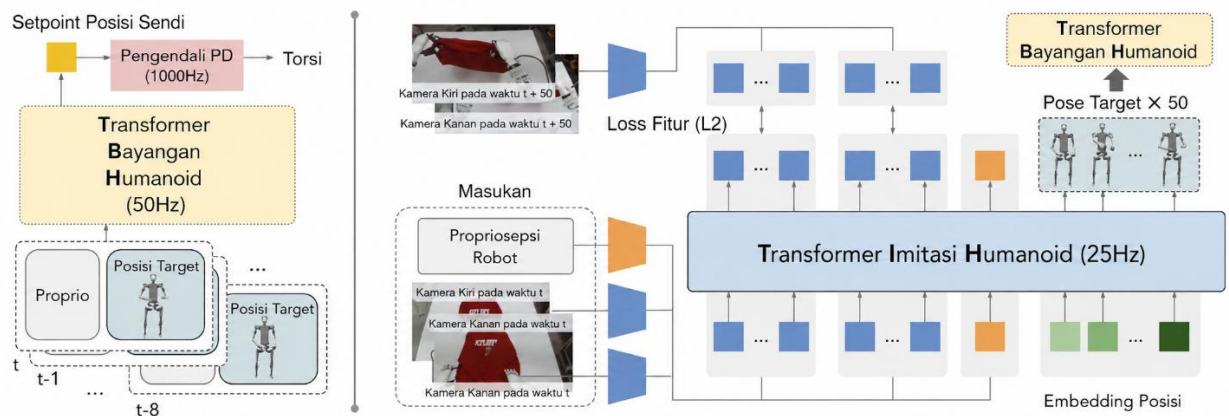
Meskipun kebanyakan orang mengharapkan robot humanoid dimodelkan berdasarkan bentuk manusia, desain robot humanoid dapat terjadi dengan cara di mana mereka memiliki parameter kontrol yang lebih sedikit atau lebih banyak. Argumen untuk derajat kebebasan yang lebih sedikit adalah bahwa hal itu disederhanakan dan lebih mudah dikontrol. Argumen untuk membuatnya lebih kompleks adalah bahwa mesin tidak terikat pada keterbatasan bentuk manusia: bahwa mereka dapat memiliki rotasi penuh di berbagai sendi. Beberapa perusahaan yang bereksperimen dengan yang terakhir adalah Booster Robotics dan Boston Dynamics.

Perangkat Lunak

Bagian ini menjelaskan tumpukan perangkat lunak dari humanoid tipikal dan pendekatan untuk mempelajari atau membuat skrip kontrol humanoid.

Pendekatan untuk Manipulasi

Manipulasi humanoid adalah masalah manipulasi cekatan bimanual dan berbagai algoritma dan teknik kontrol, yang dibahas dalam Bab 4 dan 5. Dengan demikian, penyebaran humanoid skala besar di luar demonstrasi masih merupakan masalah yang belum terpecahkan, yang menunjukkan pertanyaan studi yang belum terpecahkan tentang cara-cara yang dapat digeneralisasikan untuk mengontrol humanoid secara penuh. Pendekatan pembelajaran imitasi dengan pembangkitan data yang terukur umum dalam studi, contoh terbaru adalah Humanoid Shadowing Transformer, yang mengambil gambar dari kamera humanoid dan proprioepsi mereka (posisi sendi) sebagai input dan menggunakan transformer decoder saja untuk mempelajari parameter kontrol dari manusia yang meniru. Gambaran umum arsitektur ini ditunjukkan pada Gambar 10.2.



Gambar 10.2. Arsitektur transformer bayangan humanoid.

Pembelajaran dari data video besar masih menjadi tujuan utama dalam hal membuka kunci kecerdasan humanoid serbaguna dan metode yang dibahas dalam Bab 4 dan 5. Penggunaan data simulasi seharusnya mengarah ke arah itu. Data manipulasi manusia memiliki kesenjangan perwujudan terendah terhadap kontrol humanoid, yang membuat transfer lebih mungkin dan lebih mudah daripada perwujudan lainnya karena kesamaan struktural kebijakan.

Pendekatan untuk Berjalan

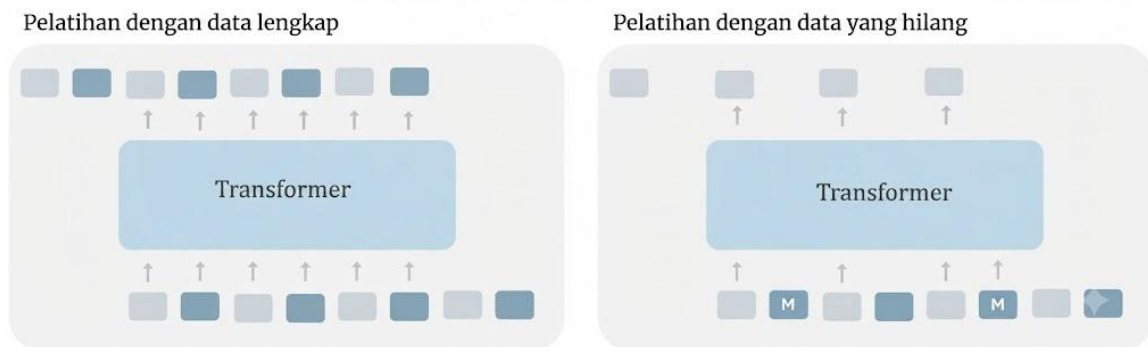
Pendekatan tradisional untuk memecahkan masalah berjalan mencakup pendekatan skrip yang tidak menggunakan pembelajaran mesin. Karena buku ini berfokus pada pendekatan pembelajaran mesin, bagian ini membahas beberapa pendekatan baru dan eksperimental untuk berjalan. Pendekatan terbaru yang patut diperhatikan berasal dari Radosavovic, yang memperlakukan pergerakan humanoid sebagai prediksi token berikutnya.

Mereka melatih pengontrol jaringan saraf berjalan pada berbagai jenis data, di mana beberapa di antaranya menyertakan keluaran aksi dan beberapa tidak. Empat jenis dataset yang mereka gunakan untuk pelatihan adalah:

1. **Kebijakan jaringan saraf:** Pasangan aksi observasi yang dihasilkan dalam simulasi oleh kebijakan RL yang dilatih dalam simulasi.
2. **Data dari pengontrol berbasis model:** Trajektori tanpa aksi, yang dihasilkan oleh pengontrol perusahaan humanoid (dalam hal ini, Agility Robotics).
3. **Data Mocap (penangkapan gerak):** Data manusia dengan penanda di tubuh mereka, seperti KIT. Data ini ditargetkan ulang ke humanoid dengan menggunakan model kinematika invers humanoid.
4. **Data YouTube tentang pose manusia:** Video YouTube tentang manusia yang melakukan berbagai hal, dengan estimasi pose diterapkan pada video dan kemudian ditargetkan ulang ke humanoid dengan model kinematika invers.

Untuk melatih model pada data dengan banyak modalitas berbeda, para penulis menggunakan token masker, yang diinisialisasi sebagai vektor acak untuk mengganti modalitas yang hilang (misalnya, tindakan). Mereka juga menggunakan masking kasual, yang berarti setiap token hanya memperhatikan input sebelumnya. Gambar 10.3 menunjukkan skema dari

hal ini. Karena metode ini dan sumber data yang digunakan, mereka mampu mengungguli metode pembelajaran penguatan terkini dan melakukan zero-shot walking dalam skenario yang belum pernah dilihat sebelumnya. Pelatihan dengan cara ini menunjukkan penskalaan dengan data, panjang konteks yang lebih besar, dan ukuran model, yang menunjukkan bahwa model yang sangat besar dengan dataset besar dan kemampuan generik dapat dibangun dengan strategi penskalaan yang mirip dengan yang diterapkan dalam pelatihan model dasar yang besar.



Gambar 10.3. Pelatihan dengan data yang hilang.

Pendekatan Teleoperasi dan Data

Salah satu cara untuk mengendalikan humanoid adalah melalui teleoperasi, yang menyediakan banyak cara untuk meningkatkan fungsionalitas humanoid. Teknik teleoperasi untuk operasi bimanual humanoid mencakup beberapa pilihan desain yang mungkin relevan:

1. **Teleoperasi headset VR vs garis pandang:** Dalam mode teleoperasi headset VR, operator yang mengendalikan humanoid hanya dikenai gambar dari kamera humanoid. Ini memastikan bahwa pengamatan untuk melatih kebijakan yang dipelajari memiliki informasi sebanyak yang dimiliki operator untuk melakukan tugas tersebut. Karena operator dapat berhasil dalam tugas tersebut, dapat diasumsikan bahwa sejumlah fitur minimum untuk menyelesaikan tugas tersebut ditangkap dalam dataset. Namun, aliran kamera robot mungkin memiliki beberapa kekurangan, termasuk persepsi kedalaman yang tidak akurat dibandingkan dengan manusia. Hal ini akan menurunkan keberhasilan tugas dan throughput pengumpulan data untuk pembelajaran imitasi. Salah satu cara untuk mengatasi persepsi kedalaman yang tidak akurat ini adalah dengan melakukan teleoperasi garis pandang, seperti pada ALOHA dan RT_1. Meskipun teleoperasi garis pandang sangat alami bagi manusia, hal itu mungkin tidak menangkap gerakan kepala (yaitu, informasi tentang ke mana harus melihat/ke mana harus mengarahkan kamera). Dengan demikian, teleoperasi garis pandang lebih berguna untuk pengaturan stasioner dan operasi di atas meja daripada untuk robot bergerak atau manipulasi seluruh tubuh.
2. **Pengoperasian boneka vs VR:** Dalam pengaturan pengoperasian boneka, terdapat dua set lengan robot—satu adalah pemimpin dan yang lainnya adalah pengikut (boneka). Dalam praktiknya, data dari pengoperasian boneka lebih bersih karena pemimpin dan

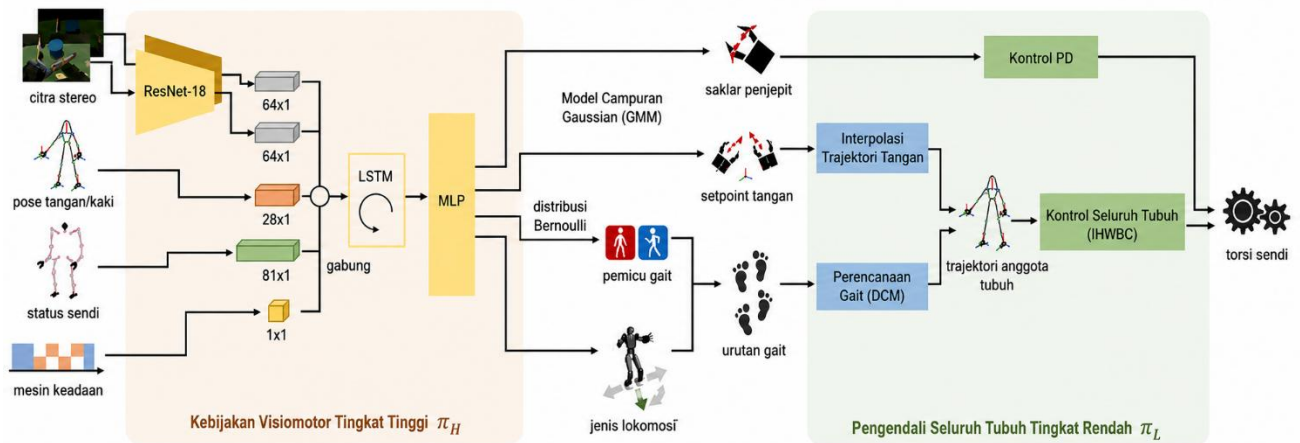
pengikut seringkali identik, seperti pada ALOHA, dan pemimpin digerakkan secara fisik oleh manusia yang melakukan teleoperasi. Pergerakan dan kesamaan kedua lengan memastikan bahwa target yang dirancang oleh pemimpin dapat dicapai oleh pengikut. Dalam pengaturan VR, target yang ditetapkan oleh pengontrol VR yang dioperasikan oleh manusia ditargetkan ulang ke robot dan mungkin termasuk kesalahan pelacakan atau kinematika invers (IK) yang tidak layak karena lengan manusia tidak persis identik dengan lengan robot. Namun, karena pengoperasian boneka melibatkan dua set robot (boneka dan dalang), hal ini dapat menjadi lebih mahal tergantung pada pengaturannya.

3. **Penangkapan gerak:** Dalam teleoperasi penangkapan gerak, algoritma visi komputer dijalankan pada aliran video manusia yang melakukan berbagai hal, seperti mendeteksi pose bagian tubuh, dan target ini ditargetkan ulang ke robot untuk mengendalikan robot. Beberapa algoritma ini cukup mirip dengan yang dibahas dalam Bab 4.

Pendekatan untuk Manipulasi Seluruh Tubuh

Manipulasi seluruh tubuh (berjalan dan melakukan tugas secara bersamaan) dapat diajarkan sepenuhnya dengan mempelajari/memodelkan semua sendi robot. Namun, teleoperasi seluruh tubuh jauh lebih sulit dan membutuhkan kerangka luar (*exoskeleton*) lengkap. Bagian ini memperkenalkan pendekatan hibrida untuk manipulasi seluruh tubuh—pendekatan yang menggunakan pembelajaran untuk tubuh bagian atas/lengan dan skrip untuk tubuh bagian bawah. Di dunia di mana kita mengharapkan robot untuk melakukan banyak pekerjaan manual, manipulasi mungkin merupakan keterampilan yang lebih berharga daripada navigasi, dan pendekatan hibrida memungkinkan manipulasi atau penempatan tangan/lengan untuk menggerakkan pemodelan bagian robot lainnya. Jadi, misalnya, penempatan kaki yang akurat didorong oleh tujuan membuka pintu, dan penempatan serta gerakan tangan yang benar untuk itu, tetapi di sini peran tubuh bagian bawah hanyalah untuk memungkinkan gerakan lengan/tangan ini sambil menjaga tubuh tetap stabil dan bergerak.

Salah satu pendekatan tersebut adalah TRILL (*Teleoperation and Imitation Learning for Loco-manipulation*) dari UT Austin, di mana manipulasi lengan robot dipelajari melalui imitasi mendalam lengan/tangan manusia melalui teleoperasi VR. Metode ini bersifat hierarkis dan terdiri dari kebijakan tingkat tinggi yang menghasilkan lintasan berdasarkan tujuan bahasa dan pengamatan gambar, dan pengontrol seluruh tubuh tingkat rendah yang mengubah lintasan yang dipelajari ini menjadi torsi sendi. Gambar 10.4 menunjukkan skema TRILL. Pemodelan perilaku tingkat tinggi melalui kebijakan yang dipelajari memungkinkannya untuk memodelkan semantik (menggerakkan tangan) dibandingkan dengan memodelkan sendi individual, dan membuat robot efisien dalam mempelajari perilaku horizon yang lebih panjang, bahkan dengan kapasitas pemodelan yang lebih kecil.



Gambar 10.4. Gambaran umum cara kerja TRILL.

10.5 PROSPEK ROBOT HUMANOID

Humanoid yang cerdas dan lincah, yang digunakan secara luas, dapat mengubah pasar tenaga kerja. Kemajuan terbaru dalam AI yang terwujud memberikan peluang untuk membuka teknologi futuristik ini. Hal ini meliputi:

1. Potensi kasus penggunaan pasar untuk humanoid dan pertimbangan biaya-manfaat
2. Pilihan desain perangkat keras untuk membangun humanoid yang efektif
3. Pendekatan pembelajaran dan data untuk meningkatkan kecerdasan humanoid

Para penulis percaya bahwa humanoid akan menjadi bidang robotika yang penting di masa depan dan kami merekomendasikan untuk menggali lebih dalam menggunakan referensi dan makalah yang dikutip dalam bab ini.

10.6 RINGKASAN

Bab ini membahas poin-poin berikut:

- Kegunaan robot humanoid dengan menyoroti kemampuannya untuk beroperasi di lingkungan yang dirancang untuk manusia. Ini termasuk tugas-tugas seperti menangani objek yang dirancang untuk tangan manusia atau menavigasi ruang seperti dapur dan kantor. Pasar robot humanoid terbagi menjadi tiga kategori utama: penyedia model dasar (misalnya, Google, OpenAI); penyedia perangkat keras (misalnya, Boston Dynamics, Unitree) yang membangun robot fisik; dan perusahaan yang terintegrasi penuh (misalnya, Tesla, Figure AI) yang menggabungkan AI dan perangkat keras untuk robot humanoid lengkap.
- Keputusan desain perangkat keras utama meliputi pemilihan antara mobilitas berkaki dan beroda, penjepit sederhana versus tangan yang cekatan, dan pengaturan kamera yang berbeda (dipasang di kepala, di pergelangan tangan, atau sensor tambahan). Aktuator linier dan putar seringkali lebih disukai daripada aktuator hidrolik karena presisi dan perawatannya yang lebih mudah.
- Metode seperti pembelajaran imitasi dan algoritma berjalan tingkat lanjut dapat meningkatkan kinerja humanoid. Teknik teleoperasi meliputi headset VR, kontrol garis

pandang, dan penangkapan gerakan, yang masing-masing memiliki manfaat dan tantangan unik.

- Sistem hibrida seperti TRILL menggabungkan kontrol berbasis pembelajaran untuk manipulasi tubuh bagian atas dengan stabilitas terprogram untuk tubuh bagian bawah dan membantu menyeimbangkan fleksibilitas dan kontrol.

Bab selanjutnya menggabungkan semua konsep yang telah kita bahas dengan mengeksplorasi infrastruktur data, strategi pelatihan dan penerapan, serta pengumpulan data robot skala besar, yang semuanya penting untuk robotika di dunia nyata.

BAB 11

ROBOTIKA BERBASIS DATA DALAM PRAKTIK

Robotika berbasis data didukung oleh lebih banyak aktivitas daripada pengembangan algoritma dan pemrograman robot. Bahkan, sebagian besar taruhannya adalah tentang penskalaan dataset robot untuk mengakumulasi keterampilan yang seluas mungkin. Bab ini membahas aspek praktis robotika berbasis data. Roda penggerak data adalah jantung dari robotika berbasis data, tetapi ini adalah kumpulan karya yang relatif kecil yang telah dipublikasikan. Kami mengundang pembaca kami untuk merujuk ke, dan untuk mempelajari aspek praktis dari roda penggerak pembelajaran imitasi dan ke untuk roda penggerak RL.

11.1 OPERASI DAN KESELAMATAN ROBOT

Operasi robot mencakup manajemen dan kontrol sehari-hari robot di lingkungan kerjanya. Untuk pembelajaran robot berbasis data, data robot adalah sumber kehidupan operasi. Operasi robot melibatkan pengelolaan hal-hal berikut:

1. **Penyebaran robot:** Memasang, mengkonfigurasi, dan memberi daya pada perangkat keras robot di lingkungan target. Untuk studi navigasi pada robot berkaki, ini juga melibatkan pemasangan gantry untuk menangkap robot yang tidak stabil.
2. **Pemantauan dan pemeliharaan:** Melibatkan pemantauan terus-menerus terhadap kinerja robot dan kesehatan sistem. Juga melibatkan kalibrasi sensor dan penggantian suku cadang/perbaikan robot.
3. **Pengorganisasian:** Penjadwalan operator jarak jauh untuk melakukan pengumpulan data dengan bantuan manusia.

Sebagian besar tim robotika memiliki operasi khusus yang bertanggung jawab untuk memelihara perangkat keras dan perangkat lunak. Pengoperasian robot yang aman merupakan komponen penting dari operasi. Prosedur keselamatan umum yang digunakan dalam robotika meliputi:

1. E-stop (penghentian darurat) untuk membekukan robot di tempat melalui intervensi manual selama keadaan darurat atau operasi rutin
2. E-stop keras untuk memutus daya ke aktuator robot selama keadaan darurat
3. Modifikasi perangkat keras untuk dampak rendah saat terjadi tabrakan
4. Batasan perangkat lunak onboard pada kecepatan dan akselerasi maksimum aktuator, dan gaya maksimum yang diizinkan pada efektor ujung
5. Intervensi manusia dalam loop secara real-time untuk menangani skenario rumit yang tidak dapat ditangani oleh model onboard
6. Keselamatan model dasar untuk evaluasi kontekstual skenario keselamatan

Organisasi yang memprioritaskan keselamatan menginvestasikan banyak sumber daya untuk memastikan penyebaran robot yang aman. Hal ini dapat memiliki konsekuensi besar bagi lintasan program dan seluruh industri. Insiden masa lalu dalam keselamatan mengemudi sendiri telah merenggut nyawa manusia, menyebabkan cedera serius, dan melumpuhkan

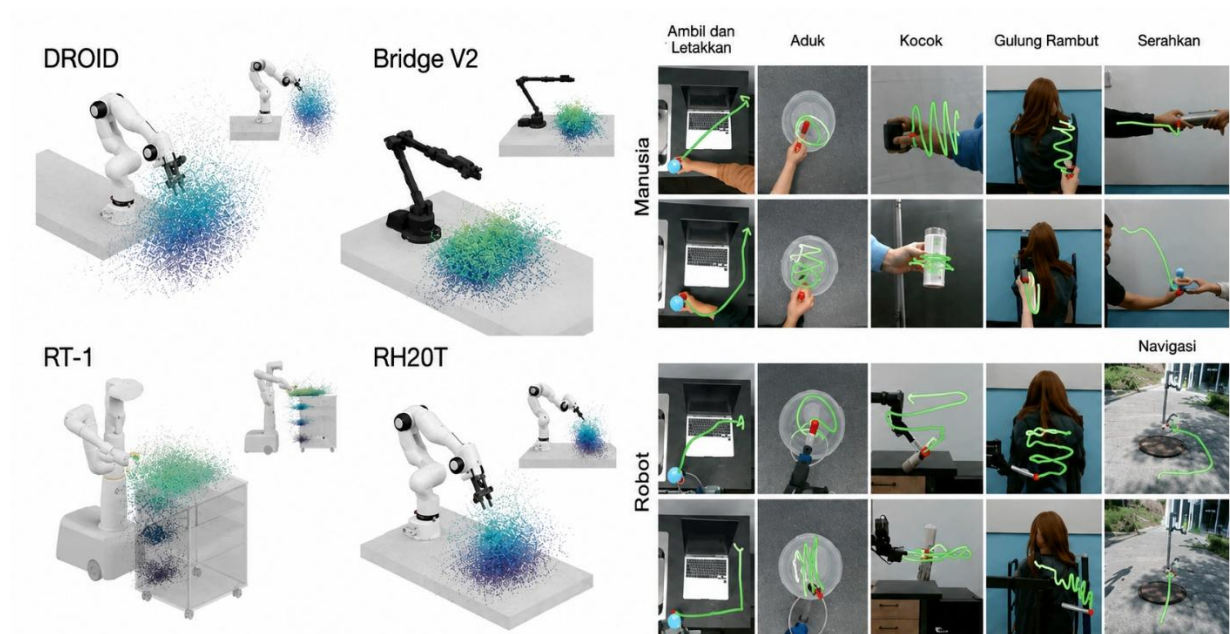
perusahaan serta menghambat seluruh industri. Membawa teknologi baru ke pasar dan membiarkannya memenangkan kepercayaan konsumen membutuhkan tindakan yang bertanggung jawab.

11.2 INFRASTRUKTUR DATA ROBOTIKA

Infrastruktur data memainkan peran penting dalam mengumpulkan, menyimpan, memproses, dan menganalisis data yang dihasilkan oleh robot, mulai dari pembacaan sensor hingga metrik kinerja (lihat Gambar 11.1). Komponen utama infrastruktur data meliputi:

1. **Pengumpulan dan pencatatan data:** Memastikan bahwa data yang berasal dari robot disinkronkan untuk merepresentasikan MDP/POMDP dalam tuple (keadaan, tindakan, imbalan). Terkadang hal ini mengharuskan sensor yang memperbarui data dengan kecepatan lebih cepat untuk di-downsample agar sesuai dengan frekuensi kontrol.
2. **Penyimpanan data:** Melibatkan pembangunan kerangka kerja untuk menyimpan sejumlah besar data robot. TFExamples adalah format penyimpanan yang umum digunakan.
3. **Pelabelan data:** Ini melibatkan pembangunan pipeline untuk mendapatkan data berlabel dari sumber seperti Mechanical Turk atau Scale AI yang menggunakan pelabel manusia untuk memeriksa dan menilai data robot. Pelabelan manusia digunakan dalam pembelajaran robot dengan berbagai cara:
 - a. Pelabelan deteksi keberhasilan untuk data episodik. Pelabel dapat bertindak sebagai penengah independen atas keberhasilan dan menentukan mengapa suatu episode gagal dengan dasar visual (bingkai mana yang memuat insiden tersebut).
 - b. Pelabelan interleaved untuk segmentasi video. Ini dapat mengajarkan model tentang apa yang terjadi selama bagian mana dari eksekusi kebijakan, di mana kebijakan salah selama eksekusi, bagaimana menalar tentang kualitas data, kelancaran lintasan, dan sebagainya.
 - c. Dataset tanya jawab visual untuk mempelajari penalaran atas data robotika.
4. **Pemrosesan dan pemuatan data:**
 - a. Memuat data yang dikumpulkan teleoperator untuk pelatihan, dan memungkinkannya untuk dilatih pada partisi spesifik dari dataset untuk eksperimen ablasi
 - b. Mengintegrasikan pelabelan ulang hindsight kembali ke dalam dataset
 - c. Memungkinkan pemrosesan pasca dataset untuk ekstraksi fitur tambahan, seperti yang dilakukan di RTTrajectory (di mana lintasan ditambahkan kembali ke dalam dataset untuk pelatihan) dan RobotMOO (di mana label berpusat pada objek ditambahkan kembali ke dalam pelatihan)
5. **Analisis data:** Menggunakan teknik pembelajaran mesin dan analitik data untuk mengekstrak wawasan dari data robot.
 - a. Analisis data offline mencakup area seperti menentukan penyebaran data pada objek, tindakan, dan lintasan 3D

- b. Metrik offline menyediakan tujuan proksi untuk mengevaluasi kapasitas suatu dataset untuk menghasilkan generalisasi
- c. Selain itu, metrik offline dapat digunakan untuk memandu pengumpulan data online (lebih lanjut tentang tugas mana yang perlu dikumpulkan, konfigurasi objek mana yang hilang, dll.)



Gambar 11.1. Atas: Rentang interaksi 3D dalam banyak dataset

Infrastruktur data yang dirancang dengan baik menghemat banyak waktu pengembang dengan mempermudah pengumpulan wawasan berharga tentang kinerja robot, serta menjalankan eksperimen.

11.3 INFRASTRUKTUR PELATIHAN ROBOT

Melatih robot untuk melakukan tugas-tugas kompleks membutuhkan infrastruktur khusus yang menyediakan sumber daya yang diperlukan untuk simulasi, pembelajaran, dan evaluasi. Infrastruktur pelatihan meliputi:

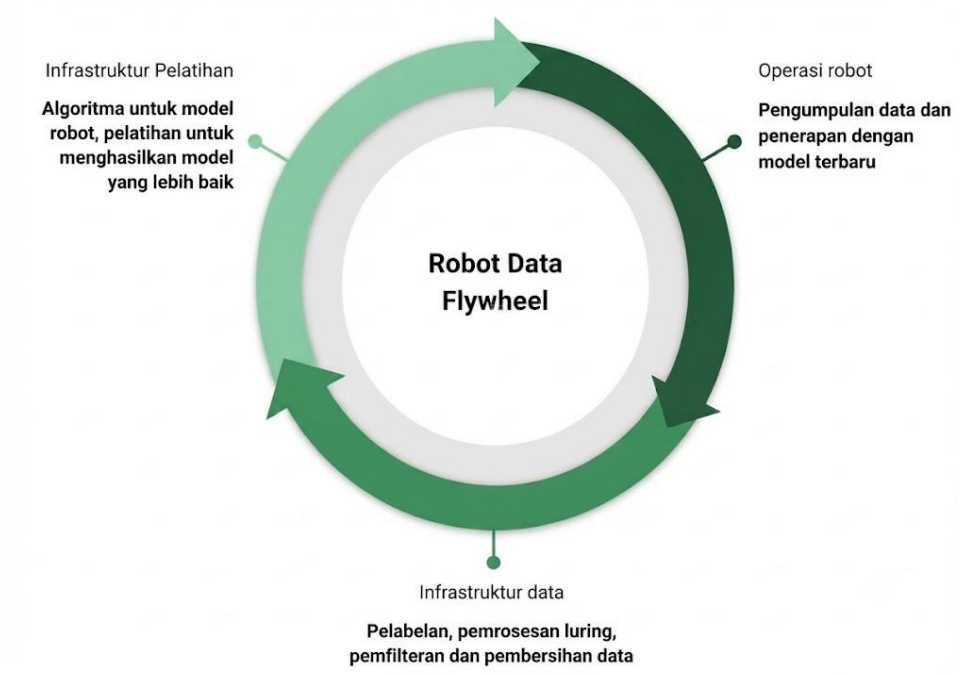
1. **Algoritma pelatihan:** Ini melibatkan pengembangan dan implementasi algoritma pembelajaran mesin yang memungkinkan robot untuk belajar dari data dan pengalaman. Umumnya, organisasi mengembangkan algoritma berbasis pembelajaran imitasi atau algoritma berbasis pembelajaran penguatan, yang keduanya membutuhkan infrastruktur yang serupa. Kedua jenis tersebut membutuhkan pemecahan data robotika menjadi tuple (keadaan, tindakan, imbalan), di mana tindakan menjadi target untuk dilatih untuk imitasi dan imbalan sebagian besar hanya digunakan dalam pengaturan pembelajaran penguatan. Karena RL membutuhkan peningkatan dari pengalaman, bootstrapping menjadi prasyarat. Mengumpulkan data yang baik membutuhkan kebijakan yang baik, dan kebijakan ini seringkali pertama kali

dilatih dalam simulasi, sebelum diterapkan di dunia nyata, seperti pada QtOpt. Oleh karena itu, simulasi memiliki peran yang lebih besar dalam kerangka kerja RL dibandingkan kerangka kerja pembelajaran imitasi.

2. **Evaluasi kinerja:** Ini melibatkan penilaian kinerja robot melalui metrik dan tolok ukur untuk mengukur kemajuan dan mengidentifikasi area untuk perbaikan. Evaluasi melibatkan jenis-jenis berikut:
 - a. *Evaluasi offline:* Ini disimpulkan dengan menggunakan kerugian pada set validasi atau dengan menjalankan inferensi pada pembagian data uji dan membandingkan hasilnya dengan kebenaran sebenarnya.
 - b. *Evaluasi dalam simulasi:* Jika simulasi sangat mirip dengan kinerja dunia nyata, simulasi dapat digunakan untuk meningkatkan skala evaluasi. AI generatif dapat digunakan untuk membangun lingkungan simulasi yang sangat realistis untuk menguji kinerja model.
3. **Infrastruktur terdistribusi:** Untuk alokasi beban selama pelatihan dan untuk inferensi yang lebih cepat selama penerapan, penggunaan komputasi yang dialokasikan secara efisien diperlukan untuk memastikan optimalitas model dasar yang dilatih. Hal ini telah dibahas di Bab 4.

11.4 RODA PENGGERAK DATA ROBOT

Integrasi yang sukses antara operasi robot, infrastruktur data, dan infrastruktur pelatihan/penyebaran sangat penting untuk memaksimalkan nilai robot. Gabungan sistem ini disebut roda penggerak data robot (lihat Gambar 11.2) dan dapat menghasilkan model dengan kemampuan yang meningkat melalui perolehan data baru. Roda penggerak data adalah jantung dari setiap laboratorium studi robotika yang digerakkan oleh AI.



Gambar 11.2. Diagram skematik roda data robot

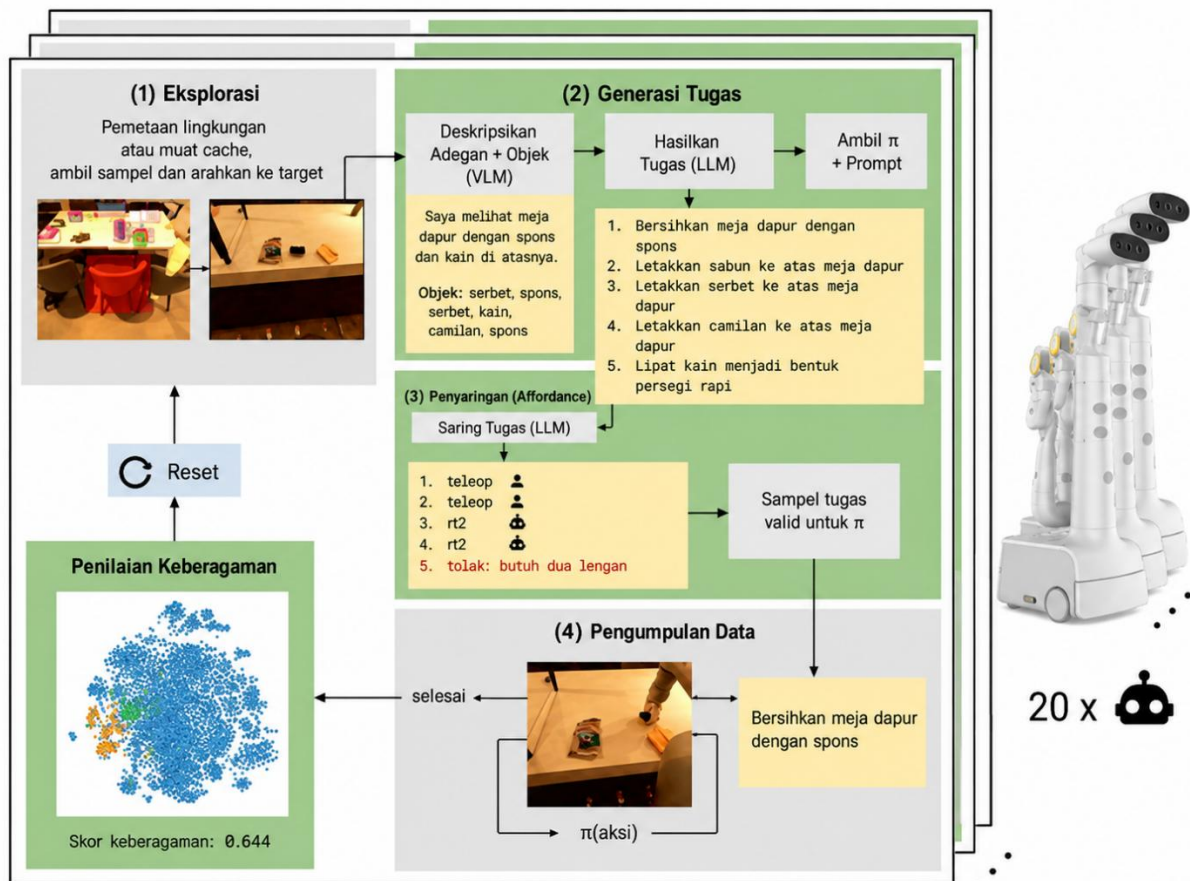
Dengan roda data robot, organisasi dapat:

1. **Terus meningkatkan kinerja robot:** Memanfaatkan wawasan data untuk mengidentifikasi area yang perlu ditingkatkan dan menyempurnakan proses pelatihan robot.
2. **Beradaptasi dengan lingkungan yang berubah:** Mengumpulkan lingkungan baru untuk dengan cepat mengadaptasi model dasar ke situasi baru dan tugas yang belum pernah dilihat sebelumnya.
3. **Mengoptimalkan pemanfaatan sumber daya:** Menganalisis data untuk mengoptimalkan strategi penyebaran, penjadwalan, dan pemeliharaan robot.

Kesimpulannya, integrasi operasi robot, infrastruktur data, dan infrastruktur pelatihan membentuk dasar untuk penyebaran dan pemanfaatan robot yang sukses. Dengan mengelola robot secara efektif, memanfaatkan wawasan data, dan mendukung proses studi yang selalu berubah, organisasi dapat bergerak cepat dalam bidang robotika. Bagian selanjutnya akan membahas beberapa proyek robotika berbasis data.

11.5 PENGUMPULAN DATA ROBOT SKALA BESAR

AutoRT adalah sistem terbaru dari Google DeepMind yang menggunakan LLM dan VLM dalam loop untuk orkestrasi robot skala besar dan pengumpulan data untuk skenario di alam liar (lihat Gambar 11.3). Metode ini menggunakan varian eksplorasi yang didorong oleh rasa ingin tahu, di mana robot menjelajahi area yang mungkin dihuni oleh manusia, dan bernalar menggunakan VLM dan LLM untuk tugas-tugas yang layak dilakukan, kemudian memanggil model aksi manusia/onboard jarak jauh untuk melakukan inferensi berdasarkan estimasi keterjangkauan dari tugas tersebut dibandingkan dengan kemampuan model otonom. Setelah pengumpulan episode tersebut, episode tersebut mendapatkan peringkat keragaman berdasarkan gambar pertama dan terakhirnya.

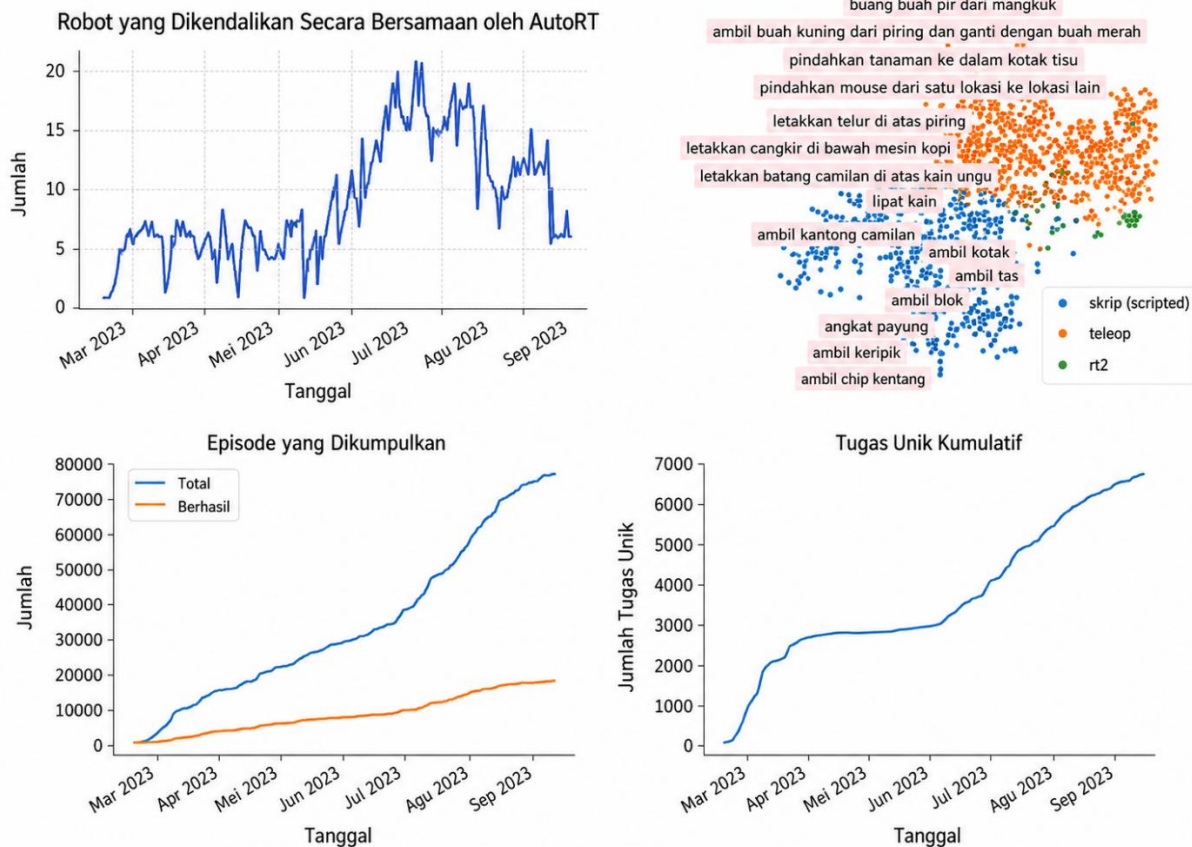


Gambar 11.3. Diagram skematik AutoRT

Untuk memutuskan tugas mana yang layak dan bermanfaat untuk dilakukan, AutoRT memperkenalkan panduan perilaku konstitusi robot. Konstitusi tersebut memiliki tiga bagian:

1. Aturan fundamental: Diturunkan dari hukum Asimov, aturan-aturan ini membahas bagaimana robot tidak boleh melakukan apa pun yang dapat membahayakan manusia.
2. Aturan keselamatan: Menjelaskan tugas-tugas yang tidak aman untuk dilakukan berdasarkan kemampuan saat ini dalam penerapan, seperti tidak berurusan dengan benda tajam, peralatan listrik, atau makhluk hidup.
3. Aturan spesifik yang diwujudkan: Aturan-aturan ini memberi tahu robot tentang kemampuannya sendiri; misalnya, bahwa itu adalah robot unimanual, yang berarti bahwa ia harus menolak tugas-tugas yang membutuhkan dua lengan.

Penskalaan dataset robot membutuhkan pengawasan manusia satu lawan satu, yang berarti bahwa jumlah teleoperator manusia merupakan hambatan untuk penskalaan. AutoRT mampu mencapai rasio manusia-robot satu banding lima, dengan manusia hanya bertindak untuk melakukan intervensi. Hal ini memperkenalkan cara untuk meningkatkan skala penyebaran robot dan belajar dari data di lapangan. Lihat Gambar 11.4.



Gambar 11.4. Penskalaan tugas unik dalam dataset, jumlah episode, jumlah robot yang dikerahkan secara bersamaan. AutoRT mengendalikan 55 robot selama enam bulan masa pakainya dengan beban puncak 20.

Meskipun AutoRT berhasil mengumpulkan data yang sangat beragam, metode ini memberikan informasi untuk upaya pengumpulan data di masa mendatang dengan cara berikut:

1. Dataset terbaik bukanlah dataset terbesar atau dataset yang paling beragam, melainkan dataset yang menghasilkan peningkatan kebijakan terbaik. Meskipun keragaman dan ukuran adalah tujuan proksi, penting untuk memverifikasi bahwa Anda mengumpulkan data yang tepat dengan melakukan pelatihan dan evaluasi data yang dikumpulkan secara berkala.
2. Meskipun data yang dikumpulkan secara otonom mungkin merupakan cara yang menjanjikan untuk mengumpulkan data tanpa meningkatkan keterlibatan manusia secara linier, pembelajaran dari data tersebut secara efisien masih belum sepenuhnya terpecahkan. Sebagian besar model dasar bertumpu pada algoritma pembelajaran imitasi yang diskalakan, tetapi pembelajaran imitasi menunjukkan kinerja yang lebih buruk ketika data berkualitas rendah ditambahkan, dan pembelajaran imitasi kehilangan kemampuan RL untuk menyambung lintasan dari lintasan suboptimal. Menggabungkan imitasi dan penguatan secara komplementer dalam robotika mungkin menjadi resep kemenangan.

3. Data yang sangat beragam mungkin memiliki distribusi yang terlalu luas untuk dipelajari oleh model saat ini. Konstruksi dataset harus bergantung pada efisiensi sampel dari kebijakan yang mendasarinya. Pelatihan dari dataset yang lebih kecil, tetapi berkualitas tinggi dan dikurasi dengan cermat, dapat menghasilkan peningkatan yang lebih besar daripada pelatihan dari sejumlah besar data yang lemah. Itu berarti bahwa pelacakan kualitas sambil menskalakan data menjadi penting.

11.6 MASA DEPAN ROBOTIKA BERBASIS MODEL DASAR

Di ambang titik di mana robotika berada saat ini, terdapat cukup banyak bukti yang meyakinkan bahwa kunci untuk memecahkan masalah robotika dan membangun robot serbaguna dan cerdas terletak pada pembangunan model dasar robot yang sangat baik. Dengan demikian, menemukan cara untuk membuat robotika semakin menyerupai masalah visi-bahasa akan diperlukan untuk menyatukan kedua dunia secara koheren.

- Generalisasi gerak mungkin merupakan masalah studi fundamental terakhir yang tersisa dalam robotika. Seperti yang dikemukakan dalam dan, VLM saat ini tidak cukup memahami lintasan, dan kemampuan mereka untuk menghasilkan gerakan baru terbatas pada kumpulan data mereka, yang mahal untuk diperoleh. Meningkatkan penalaran tentang gerakan melalui representasi visual yang berpusat pada gerakan atau melalui pembelajaran dari video Internet mungkin diperlukan untuk membuka dan meningkatkan generalisasi gerak, dan oleh karena itu memberikan VLM kemampuan untuk bernalar tentang gerakan semudah mereka bernalar tentang gambar dan bahasa.
- Memahami bagaimana VLA (*Virtual Language Agent*) berkembang dan hukum penskalaan empiris apa yang mengaturnya sangat penting untuk memahami dan memproyeksikan bagaimana bidang ini berkembang.
- Era berikutnya juga akan menyaksikan robot dan model dasar robot semakin banyak digunakan di dunia nyata untuk manipulasi yang cekatan. Tidak ada penskalaan tanpa skala itu sendiri, dan kita perlu membawa robot ke dunia nyata untuk menangkap keragaman dan skala.
- Keamanan dan keselarasan pada robot serbaguna akan menjadi syarat yang diperlukan untuk penerapan skala besar, dan membuat terobosan di bidang ini akan sangat penting untuk mendorong penggunaan di dunia nyata.
- Otonomi melalui pendakian bukit dengan sistem semi-otonom yang berskala besar, dengan kesenjangan kinerja yang dijabatani melalui intervensi dengan manusia dalam lingkaran, mungkin merupakan tren kunci lain yang akan kita lihat di masa depan.

Bertindak cerdas di ruang fisik adalah properti yang muncul dari agen bahasa audio-visual besar, dan kecerdasan pada tingkat yang paling mendasar adalah sama, baik ekspresinya digital maupun fisik. Para penulis sangat antusias untuk melihat bagaimana bidang ini berkembang, dan bagaimana Anda, pembaca, akan berperan serta dan membentuknya!

DAFTAR PUSTAKA

- Akerkar, R. (2026). Robots with a Sense of Self: Exploring Embodied Intelligence. In *Artificial Intelligence: Transcending Traditional Paradigms* (pp. 417-444). Cham: Springer Nature Switzerland.
- Barrett, L., & Stout, D. (2024). Minds in movement: Embodied cognition in the age of artificial intelligence. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 379(1911), 20230144.
- Bousetouane, F. (2025). Physical AI agents: Integrating cognitive intelligence with real-world action. *arXiv preprint arXiv:2501.08944*.
- Chitra, P., & Saleem Raja, A. (2025). Artificial intelligence (ai) algorithm and models for embodied agents (robots and drones). In *Building Embodied AI Systems: The Agents, the Architecture Principles, Challenges, and Application Domains* (pp. 417-441). Cham: Springer Nature Switzerland.
- Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., ... & Farhadi, A. (2020). Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3164-3174).
- Dong, W., Li, S., & Zheng, P. (2025). Toward embodied intelligence-enabled human–robot symbiotic manufacturing: A large language model-based perspective. *Journal of Computing and Information Science in Engineering*, 25(5), 050801.
- Duan, J., Yu, S., Tan, H. L., Zhu, H., & Tan, C. (2022). A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2), 230-244.
- Feng, Z., Xue, R., Yuan, L., Yu, Y., Ding, N., Liu, M., ... & Wang, G. (2026). Multi-agent embodied ai: Advances and future directions. *Science China Information Sciences*, 69(5), 151202.
- Friston, K., Northoff, G., Prescott, T. J., Cross, E. S., & Markose, S. M. (Eds.). (2026). *Narrow and General Intelligence: Embodied, Self-Referential Social Cognition and Novelty Production in Humans, AI and Robots*. Frontiers Media SA.
- Gaba, S., Rana, K., Sai, S., Chamola, V., & Niyato, D. (2026). A Comprehensive Review of Physical Artificial Intelligence. *Authorea Preprints*.
- Gildert, S., & Rose, G. (2023). Building and testing a general intelligence embodied in a humanoid robot. *arXiv preprint arXiv:2307.16770*.

- Huang, G. (2026). Embodied Intelligence. In *Handbook of Human-Centered Artificial Intelligence* (pp. 341-367). Singapore: Springer Nature Singapore.
- Imran, A., & Gopalakrishnan, K. (2025). *AI for Robotics: Toward Embodied and General Intelligence in the Physical World*. Apress.
- Kwon, W., Baek, S., Baek, J., Shin, W., Gwak, M., Park, P., & Lee, S. (2025). Reinforced intelligence through active interaction in real world: A survey on embodied AI. *International Journal of Control, Automation and Systems*, 23(6), 1597-1612.
- Liu, Y., Chen, W., Bai, Y., Liang, X., Li, G., Gao, W., & Lin, L. (2025). Aligning cyber space with physical world: A comprehensive survey on embodied ai. *IEEE/ASME Transactions on Mechatronics*.
- Long, X., Zhao, Q., Zhang, K., Zhang, Z., Wang, D., Liu, Y., ... & Dai, Q. (2025). A survey: Learning embodied intelligence from physical simulators and world models. *arXiv preprint arXiv:2507.00917*.
- Nurutdinov, A. R. (2024). Neural network architecture of embodied intelligence. *Automatic Documentation and Mathematical Linguistics*, 58(Suppl 4), S241-S264.
- O'Hara, P. A. (2022). Artificial General Intelligence and Autonomous Humanoid Robotics. In *Principles of Institutional and Evolutionary Political Economy: Applied to Current World Problems* (pp. 195-250). Singapore: Springer Nature Singapore.
- Paolo, G., Gonzalez-Billandon, J., & Kégl, B. (2024, July). Position: A call for embodied AI. In *Forty-first International Conference on Machine Learning*.
- Peng, Y., Han, J., Zhang, Z., Fan, L., Liu, T., Qi, S., ... & Zhu, S. C. (2024). The tong test: Evaluating artificial general intelligence through dynamic embodied physical and social interactions. *Engineering*, 34, 12-22.
- Ray, P. P. (2026). Physical AI: bridging the sim-to-real divide toward embodied, ethical, and autonomous intelligence. *Machine Learning for Computational Science and Engineering*, 2(1), 1.
- Ren, L., Dong, J., Liu, S., Zhang, L., & Wang, L. (2024). Embodied intelligence toward future smart manufacturing in the era of AI foundation model. *IEEE/ASME transactions on mechatronics*, 30(4), 2632-2642.
- Roli, A., Jaeger, J., & Kauffman, S. A. (2022). How organisms come to know the world: Fundamental limits on artificial general intelligence. *Frontiers in Ecology and Evolution*, 9, 806283.

- Sable, S., & Ikar, M. (2025). Demystifying embodied AI. In *Building Embodied AI Systems: The Agents, the Architecture Principles, Challenges, and Application Domains* (pp. 25-60). Cham: Springer Nature Switzerland.
- Sandini, G., Sciutti, A., & Morasso, P. (2025). Mutual human-robot understanding for a robot-enhanced society: the crucial development of shared embodied cognition. *Frontiers in Artificial Intelligence*, 8, 1608014.
- Song, H., Wang, L., Qiao, X., Chen, Y., Sun, D., & Sun, Z. (2025). Embodied intelligence for robot manipulation: development and challenges. *Vicinagearth*, 2(1), 8.
- Sun, F., Chen, R., Ji, T., Luo, Y., Zhou, H., & Liu, H. (2024). A comprehensive survey on embodied intelligence: Advancements, challenges, and future perspectives. *CAAI Artificial Intelligence Research*, 3(9150042), 1.
- Tao, S., Xiang, F., Shukla, A., Qin, Y., Hinrichsen, X., Yuan, X., ... & Su, H. (2024). Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*.
- Team, G. R., Abeyruwan, S., Ainslie, J., Alayrac, J. B., Arenas, M. G., Armstrong, T., ... & Zhou, Y. (2025). Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*.
- Wang, M., Niu, Y., Wang, B., Zhang, W., & Wang, C. (2026). A survey on learning motion planning and control for mobile robots: Toward embodied intelligence. *IEEE Transactions on Neural Networks and Learning Systems*.
- Xu, P. (2025, June). Embodied AI: Bridging Simulation and Reality in Robotics. In *2025 4th International Symposium on Robotics, Artificial Intelligence and Information Engineering (RAIIE)* (pp. 244-248). IEEE.
- Yifan, C., Wei, M., Wang, X., Liu, Y., Wang, J., Song, H., ... & Liu, T. (2025). Embodied ai: A survey on the evolution from perceptive to behavioral intelligence. *SmartBot*, 1(3), e70003.
- Yuan, F., Huang, X., Wang, L., Ding, J., Tian, Z., Wang, Y., ... & Mao, Z. (2026). Towards general embodied intelligence: Integrating large language models, knowledge bases, and reasoning capabilities to build the next generation of ai agents. *International Journal of Hydromechatronics*, 9(2), 250-316.
- Zhao, W., & Yuan, Y. (2025). Development of intelligent robots in the wave of embodied intelligence. *National Science Review*, 12(7), nwaf159.

AI untuk Robotika: Menuju Kecerdasan Terwujud

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

BIODATA PENULIS



Dr. Joseph Teguh Santoso, M.Kom memiliki Jabatan Akademik Lektor Kepala dan praktisi industri yang berpengalaman. Saat ini menjabat sebagai Rektor Universitas Sains dan Teknologi Komputer (Universitas STEKOM), salah satu universitas terkemuka di Jawa Tengah, Indonesia. Dengan pengalaman lebih dari 13 tahun di dunia bisnis dan praktisi industri di China, beliau membawa perspektif global dan inovasi yang signifikan ke dalam dunia akademis. Sebagai seorang entrepreneur, penulis adalah pencipta TopLoker.com, sebuah platform inovatif yang merevolusi cara mencari dan menawarkan pekerjaan. TopLoker.com adalah portal lowongan bursa kerja terbesar di Indonesia, khusus untuk pendidikan SMA/SMK sederajat.

TopLoker.com telah mendapatkan penghargaan sebagai juara 1 Startup4Industry 2022 oleh Kementerian Perindustrian Republik Indonesia. Kontribusi Dr. Joseph dalam menyediakan akses pekerjaan yang luas bagi lulusan SMA/SMK telah membantu banyak individu menemukan peluang kerja yang sesuai dengan keahlian mereka. Selain itu, Dr. Joseph Teguh Santoso, M.Kom adalah pendiri dari dua organisasi yaitu (1) organisasi guru/pendidik PTIC (Perkumpulan Teacherpreneur Indonesia Cerdas) yang bertujuan untuk meningkatkan kualitas pendidikan dan kesejahteraan guru/pendidik dengan wawasan entrepreneurship, serta (2) organisasi industri PERKIVI (Perkumpulan Komunitas Industri dan Vokasi Indonesia) yang berfokus pada pengembangan link and match antara industri dan dunia pendidikan. Sebagai Rektor, Dr. Joseph Teguh Santoso, M.Kom memiliki kepemimpinan yang berorientasi pada hasil, dan berkomitmen untuk mendorong kemajuan Universitas Sains dan Teknologi Komputer (Universitas STEKOM). Saat ini Universitas STEKOM telah mengalami transformasi positif dalam peningkatan kualitas pendidikan, perluasan fasilitas, serta penguatan kemitraan Perguruan Tinggi Nasional dan Internasional. Beliau memprioritaskan pengembangan sumber daya manusia dan penelitian, serta memastikan bahwa universitas berada di garis depan dalam inovasi dan teknologi untuk mencapai tujuan akhir, yaitu lulusan yang mampu bekerja dan sukses setelah lulus. Dr. Joseph Teguh Santoso, M.Kom sering diundang sebagai pembicara di berbagai konferensi nasional maupun internasional dan telah menerima berbagai penghargaan atas dedikasinya dalam bidang pendidikan, industri, dan kewirausahaan.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-634-7695-40-6 (PDF)



9 786347 695406