Robotics

12

12 Robotics

12.1 Robots

12.2 Robotic perception

12.3 Motion planning

12.4 Controller

12.5 Computer vision

12.6 LLM-based agents^{+}

12.7 Embodied AI*

12.8 Autonomous vehicles

Robots

Robots are physical agents that perform tasks by manipulating the physical world

Wide application

- Industry, Agriculture, Services, Transportation, Environments, Exploration
- Home care, Health care, Entertainment, Human augmentation, etc. $\Leftarrow$ Robotic age

$\Rightarrow$ Intelligent robots
more AI technique

Types of robots

- Manipulators: physically anchored to their workplace
  e.g., factory assembly line, the International Space Station
- Mobile robots: move about their environment
  – Unmanned ground vehicles (UGVs), e.g., The planetary Rover (in Mars), intelligent vehicles
  – Unmanned air vehicles (UAVs), i.e., drone
  – Autonomous underwater vehicles (AUVs)
  – Autonomous fight unit
- Mobile manipulator: combined mobility with manipulation
  – Humanoid robot: mimic the human torso
  — Android robot: a specific type of humanoid robot, mimics human features as closely as possible

Other: prosthetic devices (e.g., artificial limbs), intelligent environments (e.g., house equipped with sensors and effectors), multibody systems (swarms of small cooperating robots)

Hardware

A diverse set of robot hardware comes from interdisciplinary technologies

- Controllers (processors)
  Task planning (i.e., automated planning) ⇒ motion planning ⇒ controllers
- Sensors
- Effectors
  – Manipulators

Sensors

Passive sensors: do not need to send out a signal (light) to sense (see)

active sensors: sends out a signal (such as radar or ultrasound) and senses a reflection

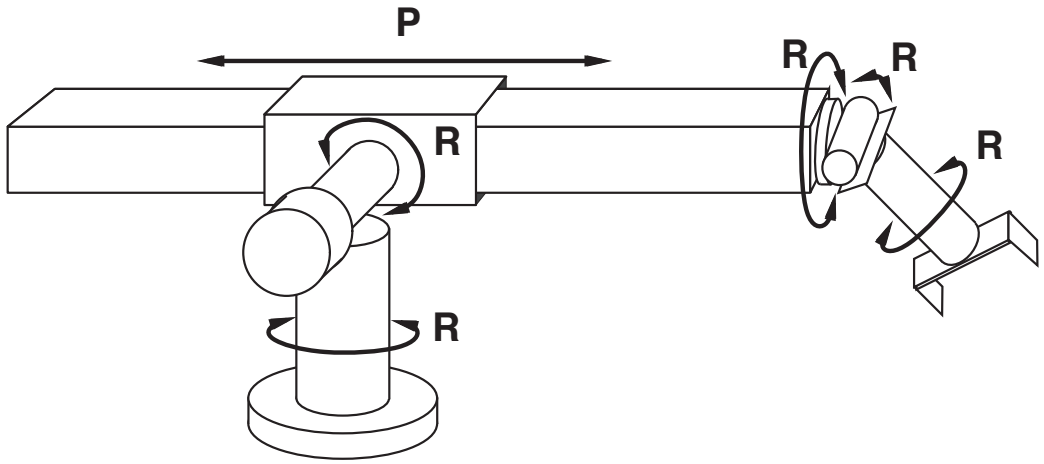
Examples

– Range finders: sonar (land, underwater), laser range finder, radar (aircraft), tactile sensors, GPS

– Imaging sensors: cameras (visual, infrared)

– Proprioceptive sensors: shaft decoders (joints, wheels), inertial sensors, force sensors, torque sensors

Manipulators



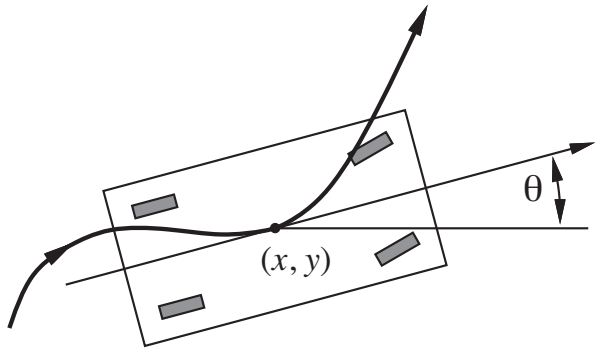
Configuration of robot specified by 6 numbers

$\Rightarrow$ 6 degrees of freedom (DOF)

6 is the minimum number required to position the end-effector arbitrarily.

For dynamical systems, add velocity for each DOF.

Non-holonomic robots



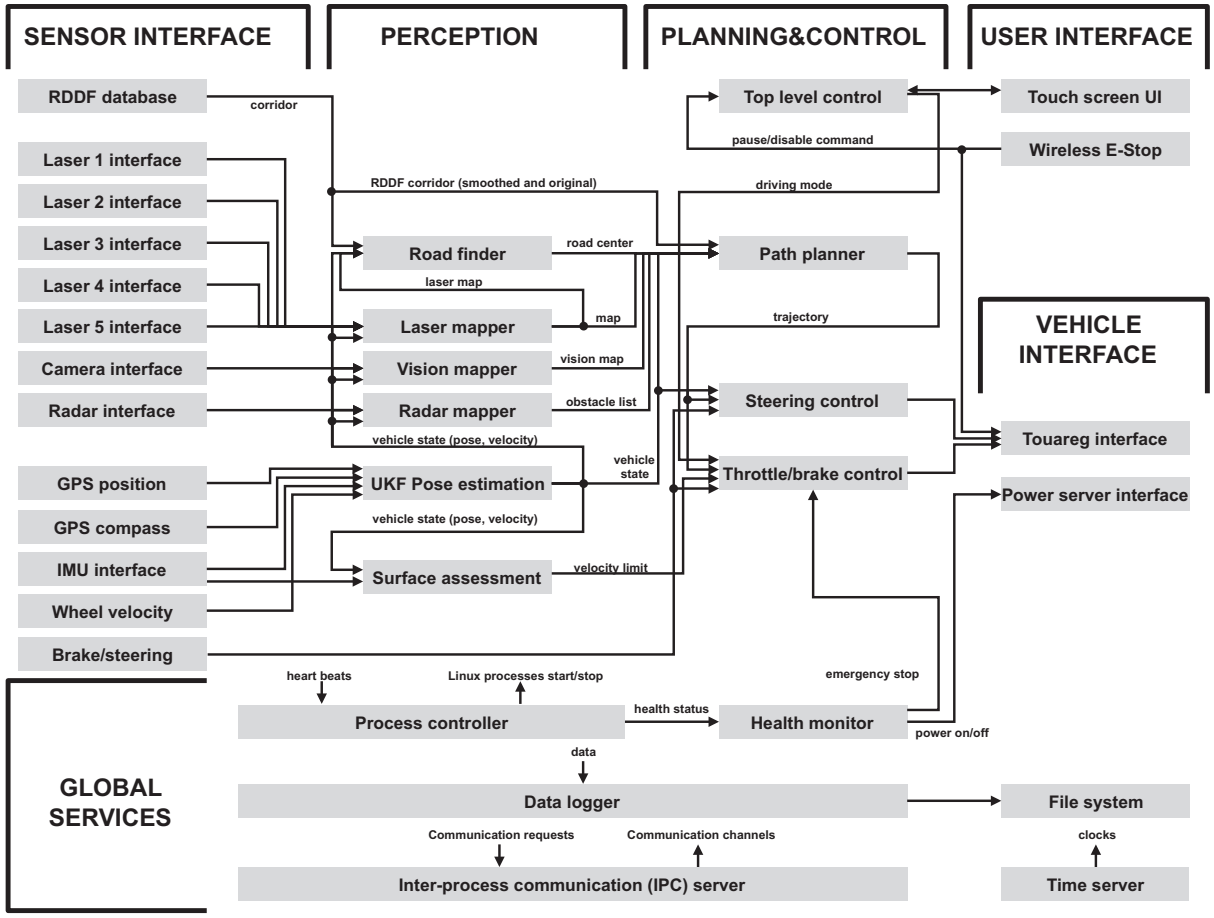
A car has more DOF (3) than controls (2), so is non-holonomic; cannot generally transition between two infinitesimally close configurations

Software

Pipeline architecture: execute multiple processes in parallel

- sensor interface layer
- perception layer
- planning and control layer
- vehicle interface layer

Example: A robot car



Robotic perception

Robotic perception: robots map sensor measurements into internal representations of the environment

- computer vision
- more, such as lidar and tactile sensors
- continuous rather than discrete variables

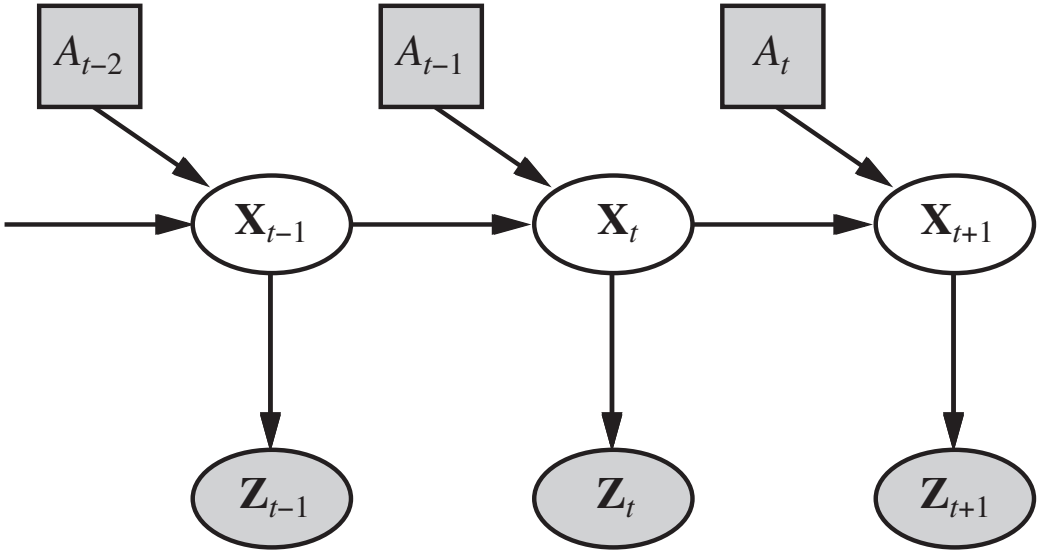
$$\mathbf{P}\left(\mathbf{X}_{t+1} \mid \mathbf{z}_{1:t+1}, a_{1:t}\right)$$
$$= \alpha \mathbf{P}\left(\mathbf{z}_{t+1} \mid \mathbf{X}_{t+1}\right) \int \mathbf{P}\left(\mathbf{X}_{t+1} \mid \mathbf{x}_t, a_t\right) P\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}, a_{1:t-1}\right) d\mathbf{x}_t$$

- Transition (motion) model: $\mathbf{P}\left(\mathbf{X}_{t+1} \mid \mathbf{x}_t, a_t\right)$
- Sensor model: $\mathbf{P}\left(\mathbf{z}_{t+1} \mid \mathbf{X}_{t+1}\right)$

- $\mathbf{X}_t$: the state of the environment (including the robot) at time t
- $\mathbf{Z}_t$: the observation received at time t
- A_t: the action taken after the observation is received

Localization

Compute current location and orientation (pose) given observations (DBN)



- Treat localization as a regression problem
- Can be done in DNNs

Mapping

Localization: given map and observed landmarks, update pose distribution

Mapping: given pose and observed landmarks, update map distribution

SLAM (simultaneous localization and mapping): given observed landmarks, update pose, and map distribution

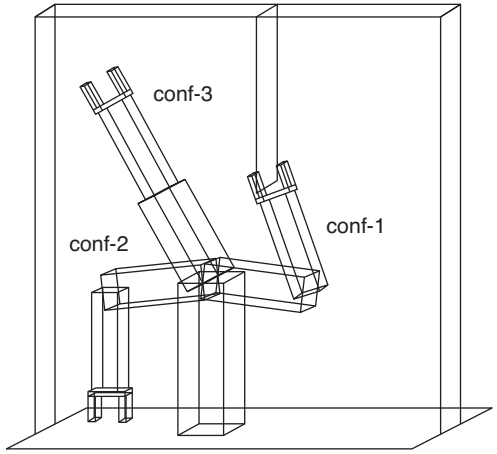
Probabilistic formulation of SLAM

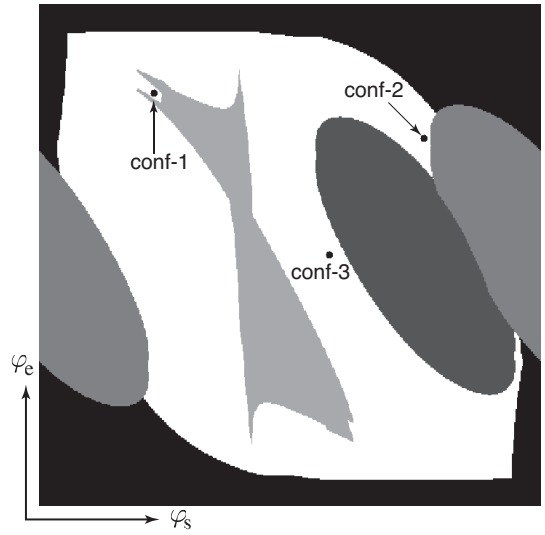
add landmark locations $L_1, \ldots, L_k$ to the state vector,
proceed as for localization

Motion Planning

Path planning: find a path from one configuration to another
– various path planning algorithms in discrete spaces

Continuous spaces: plan in configuration space (C-space, set of points) defined by the robot's DOFs





Solution is a point trajectory in free C-space
only white regions are configurations (free of collisions)

Configuration space planning#

Basic problem: ∞^d states (space of functions)

Convert to **finite** state space

Cell decomposition

divide up space into simple cells

each of which can be traversed "easily" (e.g., convex)

become discrete graph search problem

Skeletonization

reduce the free space to a one-dimensional representation

identify a finite number of easily connected points/lines

that form a graph s.t. any two points are connected

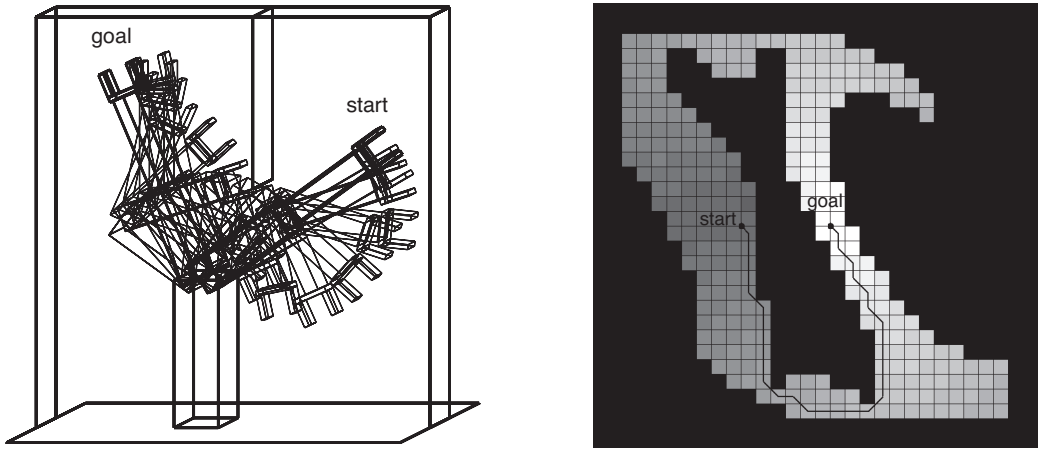
by a path on the graph

Motion planning: finding a plan that takes a robot from one configuration to another without colliding with an obstacle

Cell decomposition#

2DOFs robot arm

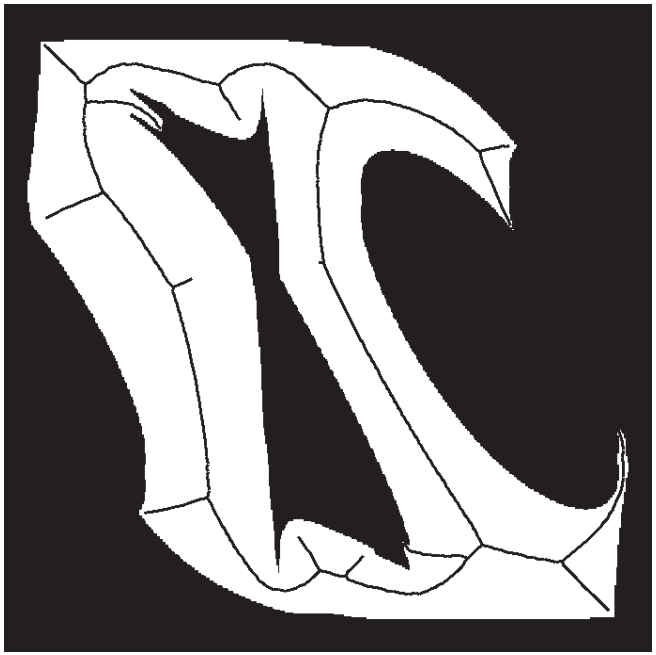
workspace coordinates configuration space



Problem: There may be no path in pure freespace (white area) cells
Solution: recursive decomposition of mixed (free+obstacle) cells

Skeletonization: Voronoi diagram#

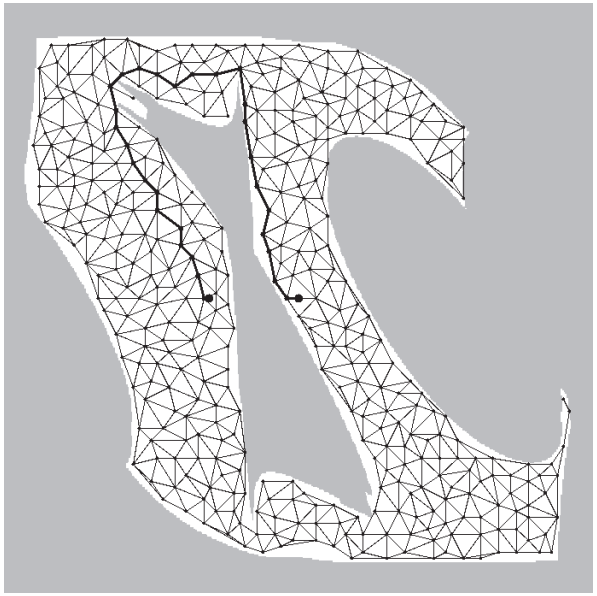
Voronoi diagram: locus of points equidistant from obstacles



Problem: doesn't scale well to higher dimensions

Skeletonization: probabilistic roadmap#

Random points generate a probabilistic roadmap in C-space and keeping those in freespace; create a graph by joining pairs by straight lines



Problem: need to generate enough points to ensure that every start/goal pair is connected through the graph

Controller

Can view the motor control problem as a search problem
in the dynamic rather than kinematic state space

– state space defined by $x_1, x_2, \ldots, \dot{x}_1, \dot{x}_2, \ldots$

– continuous, high-dimensional

Deterministic control: many problems are exactly solvable
esp. if linear, low-dimensional, exactly known, observable

Simple regulatory control laws are effective for specified motions

Stochastic optimal control: very few problems exactly solvable

$\Rightarrow$ approximate/adaptive methods

Biological motor control#

Motor control systems are characterized by massive redundancy

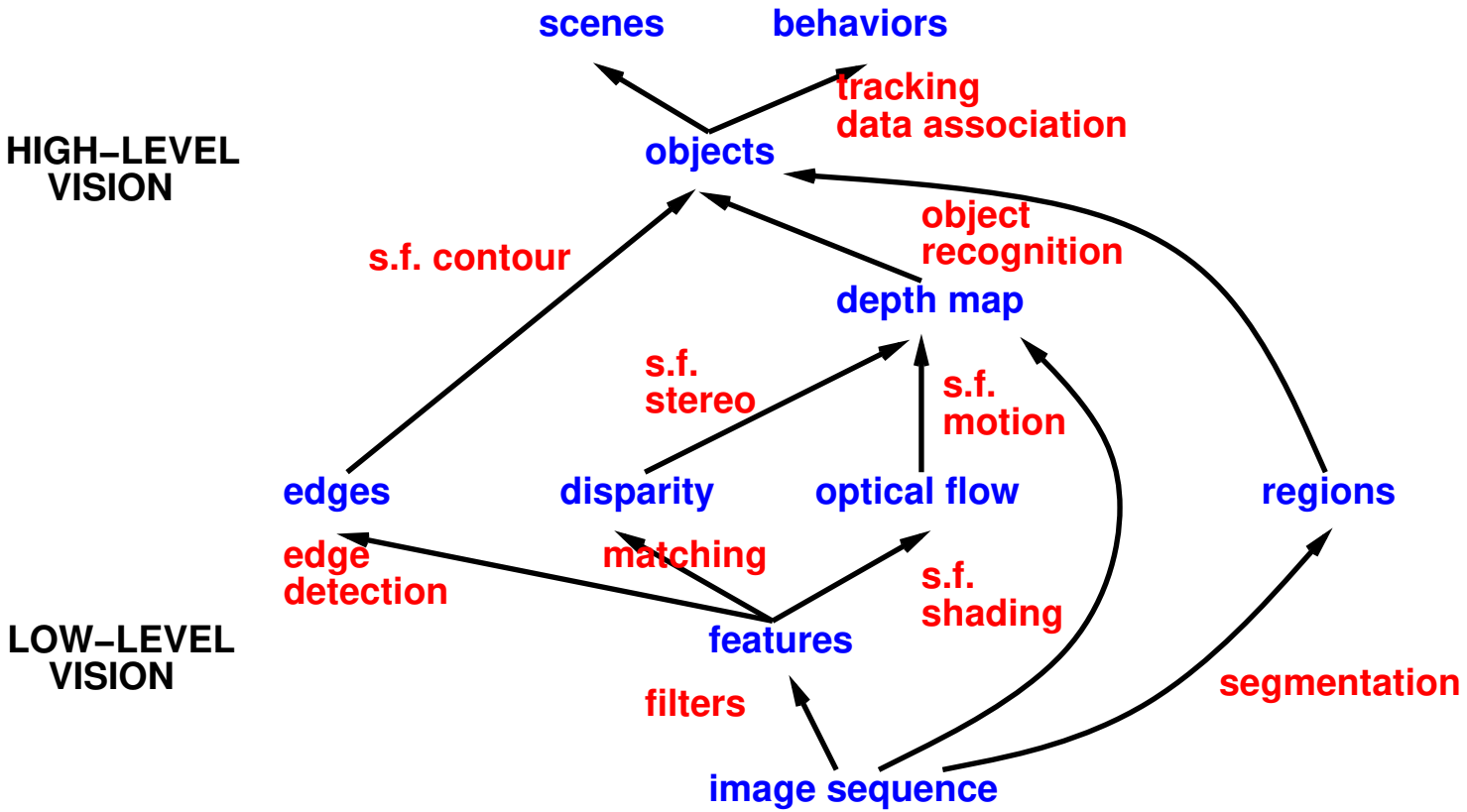
Infinitely many trajectories achieve any given task

E.g., 3-link arm moving in a plane throwing at a target
    simple 12-parameter controller, one degree of freedom at target
    11-dimensional continuous space of optimal controllers

Idea: if the arm is noisy, only "one" optimal policy minimizes error at the target

I.e., noise tolerance might explain actual motor behavior

Computer vision



Vision contributes over 80% to human perception

Vision requires combining multiple cues and commonsense knowledge

Perception

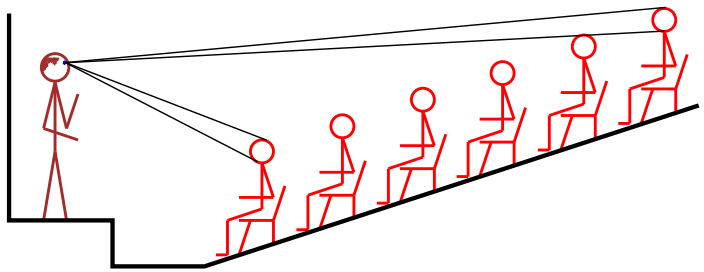
Stimulus (percept) S, World W

$$S = g(W)$$

E.g., $g =$ "graphics". Can we do vision as inverse graphics?

$$W = g^{-1}(S)$$

Problem: massive ambiguity



Perception

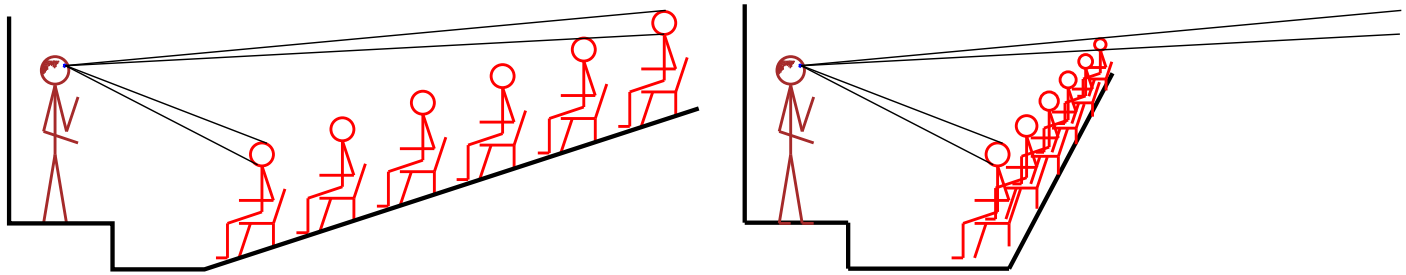
Stimulus (percept) S, World W

$$S = g(W)$$

E.g., $g =$ "graphics." Can we do vision as inverse graphics?

$$W = g^{-1}(S)$$

Problem: massive ambiguity



Perception

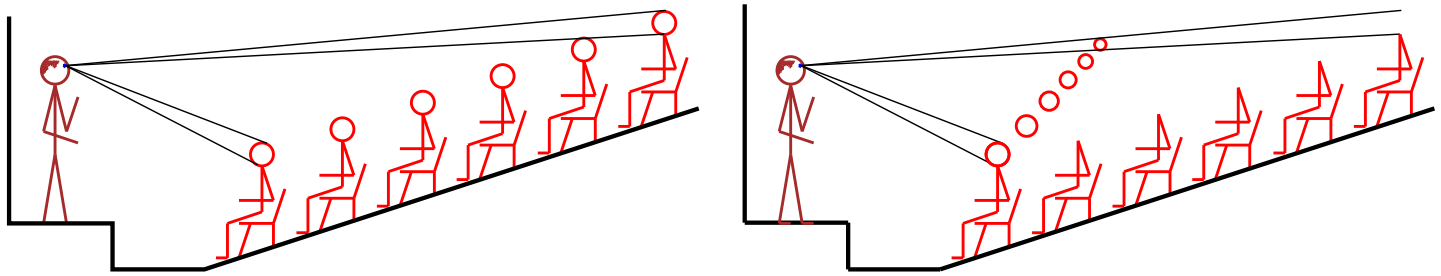
Stimulus (percept) S, World W

$$S = g(W)$$

E.g., $g =$ "graphics." Can we do vision as inverse graphics?

$$W = g^{-1}(S)$$

Problem: massive ambiguity



Visual recognition

Computer vision (CV)

– Visual recognition

– – Image classification $\Leftarrow$ (deep) learning

the whole image belonging to one class

– – – Optical flow: motion in the image (a video sequence)

$\Leftarrow$ 2D$\rightarrow$3D

Object detection: detecting instances of semantic objects of a certain class in images and videos (e.g., face detection)

what class each object is

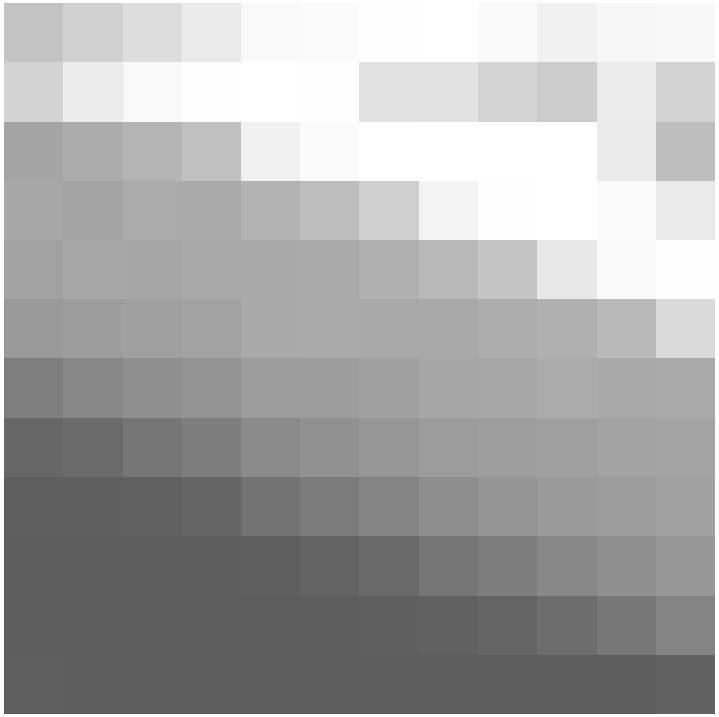
Segmentation: breaking an image into groups of similar pixels (detecting boundaries or regions)

Deep learning became a tool for computer vision

e.g., CNNs, such as PixelCNN

E.g., ImageNet: large-scale visual recognition challenge

Images



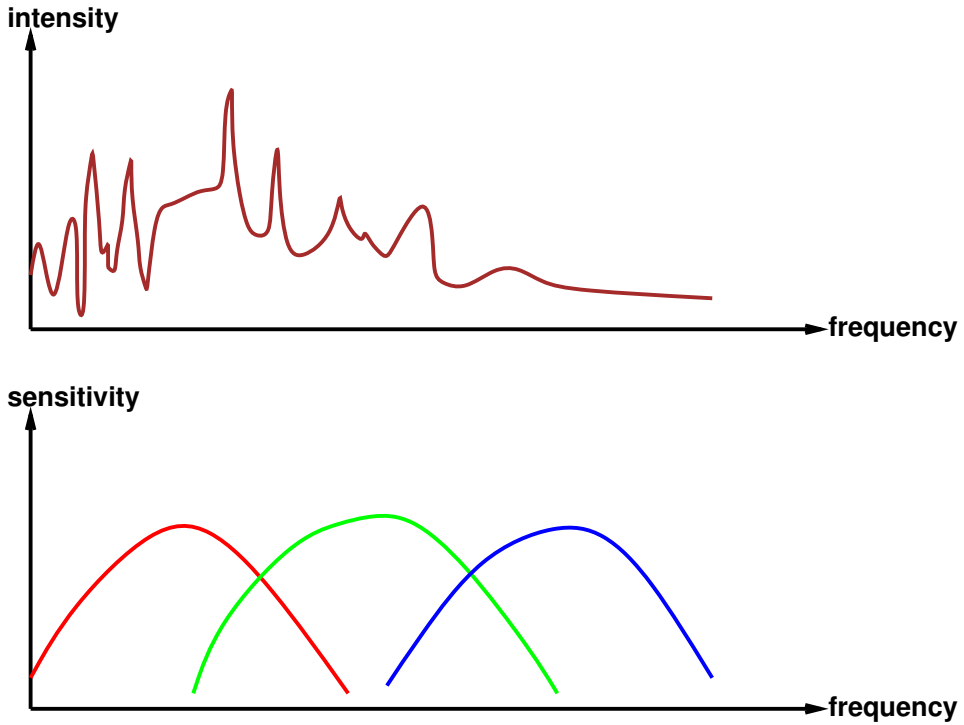
195	209	221	235	249	251	254	255	250	241	247	248
210	236	249	254	255	254	225	226	212	204	236	211
164	172	180	192	241	251	255	255	255	255	235	190
167	164	171	170	179	189	208	244	254	255	251	234
162	167	166	169	169	170	176	185	196	232	249	254
153	157	160	162	169	170	168	169	171	176	185	218
126	135	143	147	156	157	160	166	167	171	168	170
103	107	118	125	133	145	151	156	158	159	163	164
095	095	097	101	115	124	132	142	117	122	124	161
093	093	093	093	095	099	105	118	125	135	143	119
093	093	093	093	093	093	095	097	101	109	119	132
095	093	093	093	093	093	093	093	093	093	093	119

$I(x, y, t)$ is the intensity at (x, y) at time t

CCD camera $\approx$ 1,000,000 pixels; human eyes $\approx$ 240,000,000 pixels
i.e., 0.25 terabits/sec

Color vision

Intensity varies with frequency $\rightarrow$ infinite-dimensional signal



RGB: Human eye has three types of color-sensitive cells; each integrates the signal $\Rightarrow$ 3-element vector intensity

Image features

Edges: straight lines or curves in the image plane across which there is a change in image brightness

Edge detection: a compact abstract representation

Noise — changes to the value of a pixel that doesn't have to do with an edge

Solution: smoothing — using surrounding pixels to suppress noise

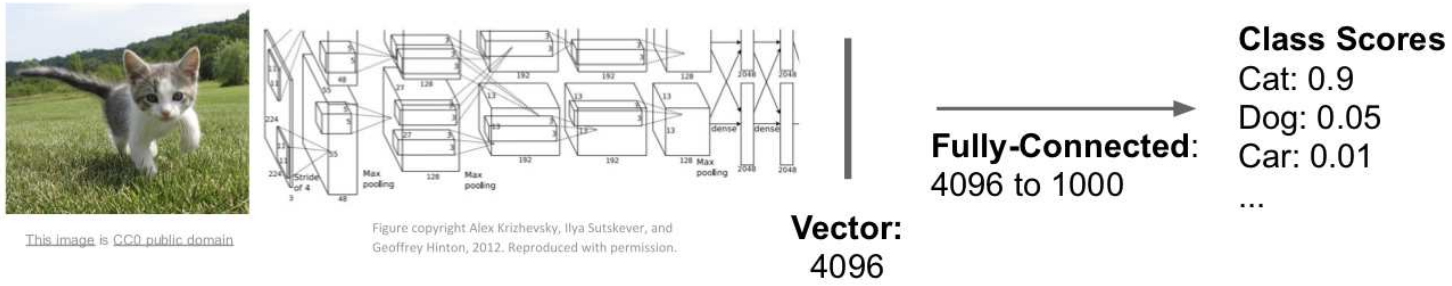
Gaussian filter: predict the true value of the pixel as a weighted sum of nearby pixels, with more weight for the closest pixels

$\Rightarrow$ convolution of two functions $\Leftarrow$ CNNs

Texture: a pattern on a surface that can be sensed visually

$\Leftarrow$ similarly, CNNs

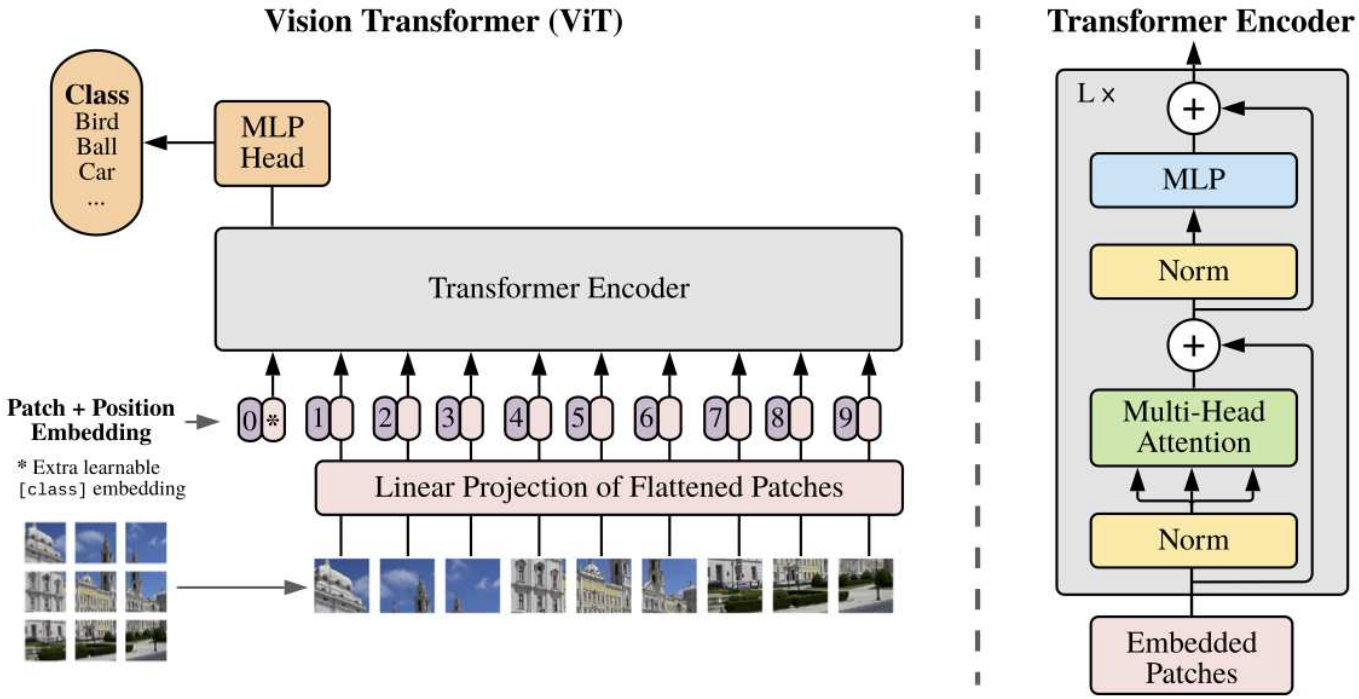
Image classification



Previously, CNNs were SOTA results

$\Leftarrow$ Transformer/LLM

ViT: Vision transformer



Ref: Dosovitskiy A et al., An image is worth 16×16 words: transformers for image recognition at scale, arXiv, 2020

ViT: Patch embeddings

Reshape the image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N \times \left(P^2 \cdot C\right)}$

- (H, W): the resolution of the original image
- C: the number of channels
- (P, P): the resolution of each image patch
- $N = HW/P^2$: the resulting number of patches (input sequence length for the Transformer)

- use constant latent vector size D through all of the layers, flatten the patches, and map to D dimensions with a trainable linear projection as the patch embeddings
- prepend a learnable embedding to the sequence of embedded patches $\left(\mathbf{z}_0^0 = \mathbf{x}_{\text{class}}\right)$, whose state at the output of the Transformer encoder $\left(\mathbf{z}_L^0\right)$ serves as the image representation $\mathbf{y}$

ViT: Formulations[#]

$$\mathbf{z}_0 = \left[\mathbf{x}_{\text{class}}\,; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots\,; \mathbf{x}_p^N\mathbf{E}\right] + \mathbf{E}_{pos}, \quad \mathbf{E} \in \mathbb{R}^{\left(P^2 \cdot C\right) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$$

$$\mathbf{z}'_\ell = \text{MultiHead}\left(\text{Layernorm}\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \ldots L$$

$$\mathbf{z}_\ell = \text{MLP}\left(\text{Layernorm}\left(\mathbf{z}'_\ell\right)\right) + \mathbf{z}'_\ell, \quad \ell = 1 \ldots L$$

$$\mathbf{y} = \text{Layernorm}\left(\mathbf{z}_L^0\right)$$

Vision-language models

VLMs: bridging the gap between images (CV) and text (NLP), e.g., GPT-4V

Vision-language pretraining (VLP)

- learn to associate images with textual descriptions in a shared latent space
- training objectives capturing semantic alignment between visual and textual modalities
- training on image-text pairs using self-supervised learning

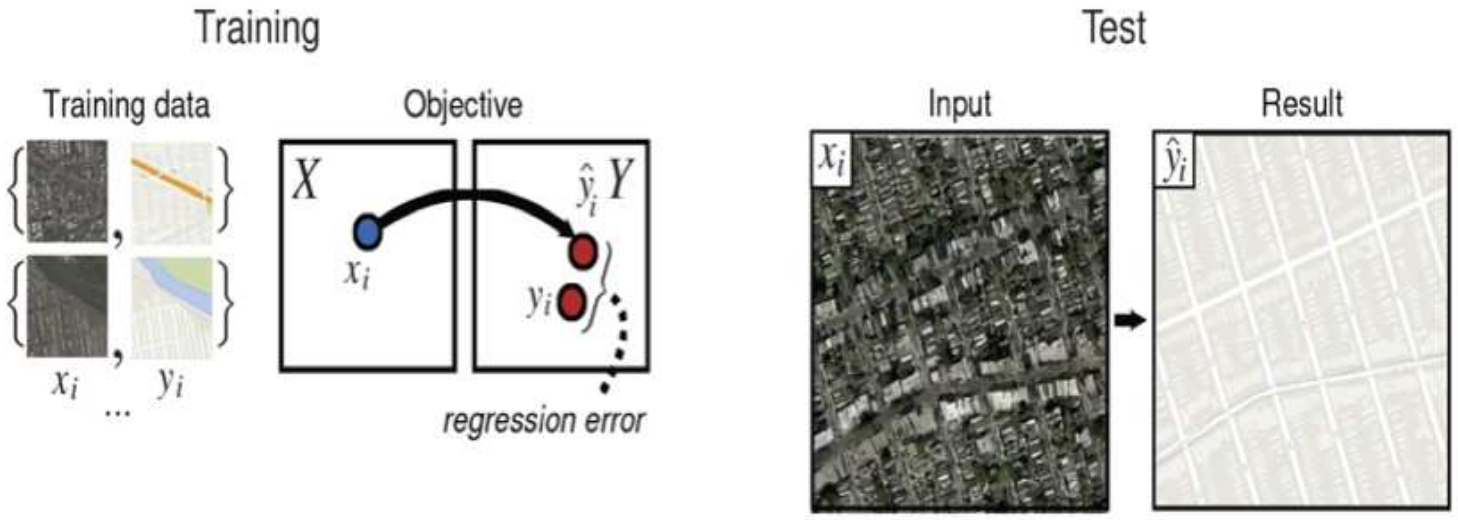
$\Rightarrow$ MM-LLMs

Downstream

- VL: alignment, understanding, (text/image) generation
- Captioning, object detection
- Video tasks, etc.

Example: image transformation

Mapping images from one type to another



Learning by comparing $\hat{y}_i$ and y_i

Example: deepfake

An image or video that looks like a particular person, but is generated from a model, in which a person in an existing image or video is replaced with someone else

Synthetic media manipulation

- Face swapping: in videos making it appear as though someone else is speaking or acting
- Voice synthesis: generating realistic-sounding speech from text input, allowing for audio manipulation
- Image synthesis: creating realistic images of people who do not exist or altering existing images to change appearances

Ethical consequences

Example: Video generation from text

Turn visual data of all types into a unified representation (i.e. patches) that enables large-scale training of MM-LLMs

– Training text-to-video generation requires a large number of videos with corresponding text captions

E.g., Sora, generated a minute of high-fidelity video from text prompting, spanning diverse durations, aspect ratios, and resolutions

Object detection

Single object: classification + localization

e.g., single-stage object detectors: YOLO/SSD/RetinaNet

Multiple objects: each image needs a different number of outputs

– apply a CNN to many different crops of the image, CNN classifies each crop as object or background

e.g. lots of object detectors

3D object detection: harder than 2D

e.g., a simple camera model

Object detection + Captioning = Dense captioning

Objects + relationships = Scene graphs

Zero-shot object detection allows robots to identify and locate objects they have never encountered previously $\Leftarrow$ VLMs

VLMs for object detection & 3D classification

VLMs revolutionize object detection by integrating both visual and textual information

Process

1. **Input image/3D data**: start with an input image containing various objects/3D data representing objects or scenes in three-dimensional space
2. **Textual query**: VLMs accept a textual query describing the objects of interest/the characteristics or attributes of objects to be classified (this query provides contextual information to guide the detection process)
3. **Detection/Classification**: Using the combined visual and textual information, VLMs analyze the image to detect and localize objects/the 3D data to classify objects into appropriate categories based on the provided query

Segmentation

Segmentation: is all needed

– Semantic segmentation: no objects, just pixels

label each pixel in the image with a category label; don't differentiate instances, only care about pixels

e.g., CNN

– Instance segmentation + object detection $\rightarrow$ multiple object

VLMs for segmentation tasks offer improved contextual understanding and precise segmentation results

Controlling movement with vision

Vision provides information both for manipulating objects and for navigating while avoiding obstacles

E.g., self-driving vehicle

- Lateral control, e.g., changing lane
- Longitudinal control, e.g., safe distance
- Obstacle avoidance, e.g., maneuver
- Obey traffic signals, etc.

LLM-based agents+

LLM-A: Combining the advantages of both LLMs and agents

- unveiling LLMs' obstacles

  e.g., knowledge update, tool utilization

- incorporating goal, action, memory, rethink, tool, environment, etc.

- Planning: derive primarily from LLMs

- Memory: preserve knowledge, experiential data, and historical information during problem-solving processes

  e.g., memory retrieval by retrieval-augmented generation

- Rethink: encompass evaluating prior decisions and subsequent environmental feedback

- Environments: computer environment (Web, API), real-world Environment, etc.

LLM-based multi-agents

LLM-MA: Leveraging the collective intelligence and specialized profiles and skills of multiple agents

- specializing LLMs into various distinct agents, each with different capabilities

- enabling interactions among these diverse agents to simulate complex real-world environments effectively

Architecture

- Agents-environment interface
- Agents profiling
- Agents communication
- Agents' capabilities acquisition

Applications

- software development, multi-robot system, society simulation, policy simulation, game simulation, etc.

Embodied AI*

Focus on endowing AI systems with physical bodies or virtual avatars equipped with sensors and actuators

Embodied agents possess physical or virtual bodies and autonomously interact with their surroundings

e.g., humanoid robots, autonomous vehicles, virtual avatars

Embodied multimodal large language models (EM-LLMs) incorporate real-world sensor and actuation modalities into pretrained LLMs.

Directions

- Seamless integration of AI into everyday environments
- Lifelong learning and continual adaptation
- Ethical and societal implications

Generalist AI

Design to perform various tasks across multiple domains with a single, unified system

E.g., Gato is a generalist agent that works as a multi-modal, multi-task, multi-embodiment generalist policy

Characteristics

- Versatility: tasks across different domains
- Adaptability: learn from experience and improve performance across diverse tasks and environments
- Integration: knowledge and skills from various sources

Autonomous vehicles

Autonomous vehicles (AVs) (intelligent cars, self-driving cars) began to migrate from testing conditions to driving on public roads
$\Leftarrow$ still a long way to run

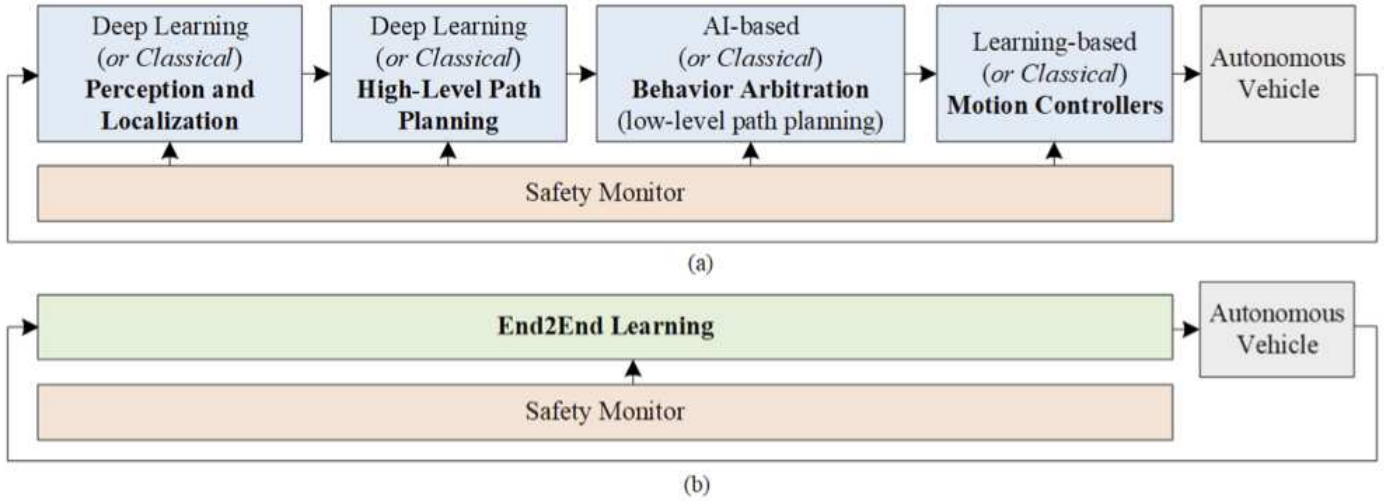
SAE Levels (J3016 standard, SAE Committee, 2014): 0-5
- lower levels (0-3): driver assistance
- higher levels (4-5) move towards requiring no human interaction
- – level 5 requires no human input and no steering wheels or foot pedals

End-end self-driving systems

(a) A conventional method as a sequential perception-planing-action pipeline

– the components can be designed either using deep learning or based on classical non-learning approaches

(b) Neural end-end system



Example: Tesla autopilot

A safety monitor is usually designed to ensure the safety of each module

- High-level path planning
- Behavior arbitration, or low-level path planning
- Motion controllers

AI-driven or software-defined cars

MM-LLMs for autonomous driving#

Multimodal large language-driven models

- Architecture

– Visual processing: the surrounding environment, detect objects, and perceive spatial relationships

– Language understanding: instructions from human operators or contextual information from traffic signs and road markings

- Advantages

– Improved contextual understanding: rich contextual information from both visual and textual inputs

– Enhanced reasoning abilities: predicting the intentions of other road users, anticipating future actions, and planning optimal trajectories

– Robustness to uncertainties: sensor noise, occlusions, and dynamic changes

MM-LLMs for autonomous driving#

- Perception

  - improve perception by integrating information from multiple sensors, such as cameras, LiDAR, radar, and GPS, with textual descriptions and contextual information

  – Scene understanding

  – Object detection

- Planning

  - enhance planning and decision-making by incorporating high-level semantic information from textual inputs, such as route descriptions, traffic regulations, and human commands

- Interaction

  - facilitate interaction between autonomous vehicles and human operators, pedestrians, and other road users

Humans and robots

Robots work in human environments

• Coordination, e.g., an autonomous car merges on the highway and needs to negotiate the maneuver with the human driver coming in the target lane

Modeling as an incomplete information game between the robot and the human

– humans as approximately rational agents

— robot predicts human action, e.g., not know other drivers, but know a bit more if someone is trying to merge in front of

— human predictions about the robot, say, the robot can act in a way to make it easier for the human to make correct predictions

• Collaboration: the human and the robot are on the same team working toward the same goal

Modeling as a joint agent whose actions are tuples of human-robot actions