Robotics and Intelligent Systems
Semester-I: 2024/2025 AC

October, 2024

Outline

- To understand advanced principles of robotic perception, planning, and control.
- To explore the application of AI techniques, such as machine learning and reinforcement learning, in robotics.
- To develop practical skills in designing and implementing intelligent robotic systems.
- To critically evaluate state-of-the-art research and emerging trends in AI and robotics.

Learning Outcomes

- Upon successful completion of this course:
  - Design and implement advanced robotic systems that integrate AI techniques.
  - Develop algorithms for robotic perception, motion planning, and control.
  - Apply machine learning and reinforcement learning to robotics.
  - Analyze and evaluate current research and applications in AI-driven robotics

Topics to be Covered

Introduction to Advanced Robotics (overview, challenges, architecture)

Robotic Perception (Sesnsor, Sensor fusion and multi-sensor integration, Simultaneous Localization and Mapping (SLAM))

Motion Planning and Control (Advanced motion planning algorithms, Optimal control and model predictive control, Planning under uncertainty)

Learning in Robotics (Supervised, unsupervised, and reinforcement learning)

Human-Robot Interaction (HRI principles and challenges, NLP in robotics)

AI in Autonomous Systems (Autonomous navigation and decision-making
Ethical considerations in autonomous robots)

Research and Emerging Trends

Administration

- Course Type: Lecturing, Lab exercises, classroom discussion and assignments
- Course schedule:
  - Lecture Session: Friday 9:00 – 11:00pm
  - Lab Session: Wednesday 1:30- 3:30 pm (Tentative schedule)
- Prerequisites: Deep Learning

Books and References

- Reference:
  - Robotics: Modelling, Planning and Control by Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo, 2nd Edition, 2010.
  - *Computational Principles of Mobile Robotics* by Gregory Dudek and Michael Jenkin, Second Edition

Administration

- Course Policies
  - Attendance: It is compulsory to attend class in time and every time. Missing more than three classes during the term causes readmission for the course.
  - Assignments: No late assignment will be accepted
  - Test/Quizzes: Rarely reexamination schedules will be arranged for those who missed the exam by accidental or uncontrollable situation.
  - Cheating/Plagiarism: No second Chance or excuses.

Getting a Grade

- Assignment 20%... Due date ….
  - Assignment-I (10%) Nov 22, 2024,
  - Assignment-II (10%)- December 22, 2024

- Exams
  - Mid-Exam 20%: Scheduled inline with the AAU Academic Calendar
  - Final-Exam 40%: Scheduled inline with the AAU Academic Calendar

- Project Demonstration– 20% – group projects will be presented by … due date January, 17 2025

Introduction to Advanced Robotics

What is a robot?

- The field of robotics has its origins in science fiction.
- The term robot was derived from the English translation of a fantasy play written in Czechoslovakia around 1920.
- It took another 40 years before the modern technology of industrial robotics began.

Definition

A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks: Robot Institute of America, 1979

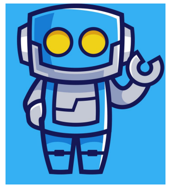
"Robotics“ is defined as the science of designing and building Robots which are suitable for real life application in automated manufacturing and other non-manufacturing environments.

Robotics is an applied engineering science that has been referred to as a combination of machine tool technology and computer science. It includes machine design, production theory, micro electronics, computer programming & artificial intelligence.

Why you Study Robotics ?

Why Study Robotics

- Make things easier/cheaper/faster for people



Factory robots

Delivery Robot

- Go places/do things people can't explore (space, deep sea, volcanos)

High performance drones

- Build 'real' artificial intelligence

Create a general learner
(artificial general intelligence)

Understand how humans became intelligent

Use of Robots

Manufacturing Applications
Manufacturing

Assembly Lines: Robots used for precision and speed in automotive, electronics, and other sectors.

Welding and Painting: Robots improve quality and safety in repetitive or hazardous tasks.

Healthcare Applications

Surgery: Surgical robots enable minimally invasive surgeries, enhancing precision and recovery times.

Rehabilitation: Robots assist in physical therapy, improving mobility for patients.

Medical Assistance: Robots deliver supplies and medications, easing hospital workloads.

Logistics and Warehousing Applications

Logistics and Warehousing
ASRS (Automated Storage and Retrieval Systems): Robots manage warehouse inventory efficiently.

Delivery Drones and Robots: Streamlining the supply chain through automated deliveries.

Use of Robots

Agricultural Applications Agriculture

- Harvesting and Planting: Robots boost efficiency in planting, watering, and harvesting.
- Weed Control and Monitoring: Robots help identify pests and monitor crop health.

- Exploration and Research Applications
  - Space Exploration: Robots like Mars rovers explore inhospitable environments.
  - Underwater Exploration: Autonomous robots collect deep-sea data.
  - Scientific Research: Robots assist in laboratories for experiments and data analysis.

- Military and Defense Applications Military and Defense Surveillance and Reconnaissance: Robots gather intelligence in dangerous zones.
  - Bomb Disposal: Robots handle explosives to protect human operators.
  - Unmanned Aerial Vehicles (UAVs): Used for reconnaissance, surveillance, and combat.

Use of Robots

Household and Consumer Applications

- Vacuum Cleaning and Lawn Mowing: Robots automate household chores.
- Personal Assistance: Virtual assistants help with daily tasks like reminders and caregiving.

Retail and Customer Service Applications

- Automated Checkout Systems: AI-powered kiosks assist in inventory and transactions.
- Customer Service Robots: Guide and assist customers in retail environments.

Construction Applications

- Bricklaying and 3D Printing: Robots build structures using innovative techniques.
- Inspection and Monitoring: Robots enhance safety by performing hazardous tasks.

Education Applications Education

- Learning Assistants: Robots teach programming and STEM in classrooms.

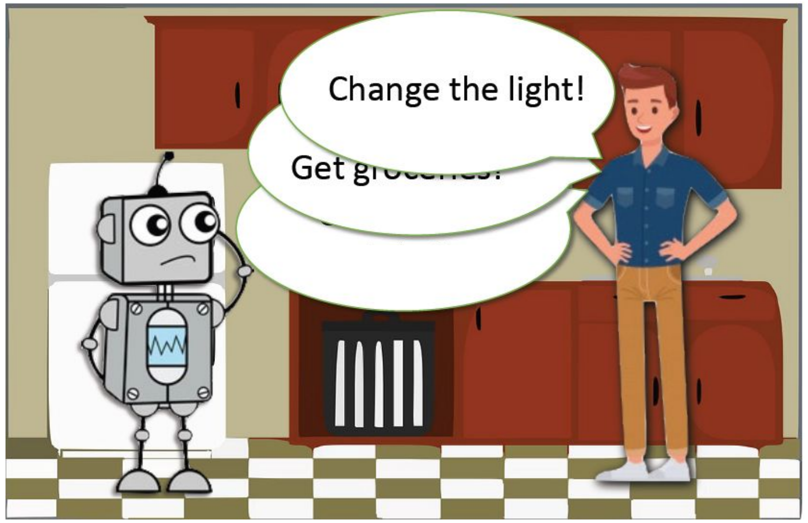
Food Industry Applications Food Industry

- Food Preparation and Cooking: Robots automate processes in restaurants and fast-food chains.

Why you Study Robotics ?

Goal of Robot Learning

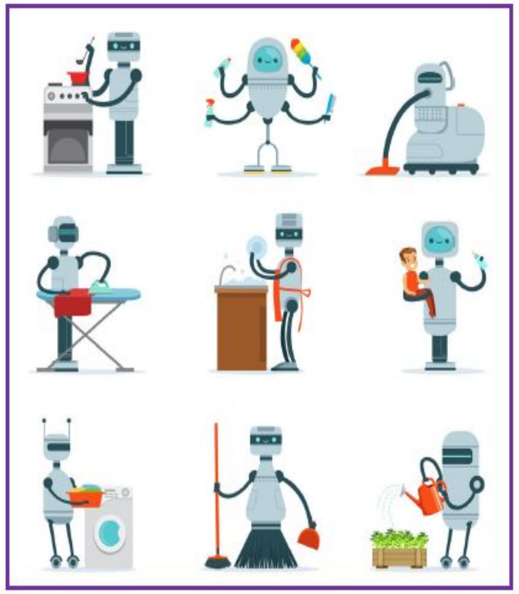
- Ultimate goal: Build general-purpose embodied intelligence by learning to make sequential decisions in the physical world

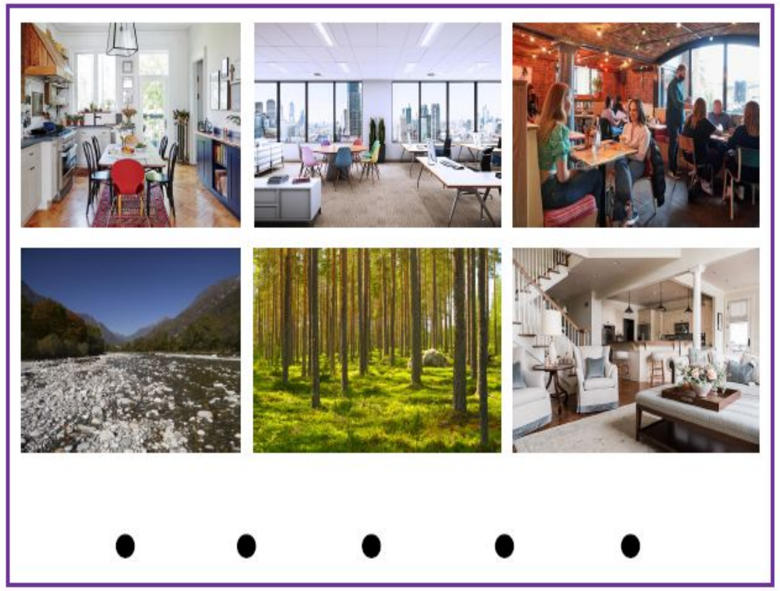


Source: Introduction to Robot Learning, Spring 2024, Carnegie Mellon University

Goal of Robot Learning

- Robots that can do thousands of tasks in thousands of environments



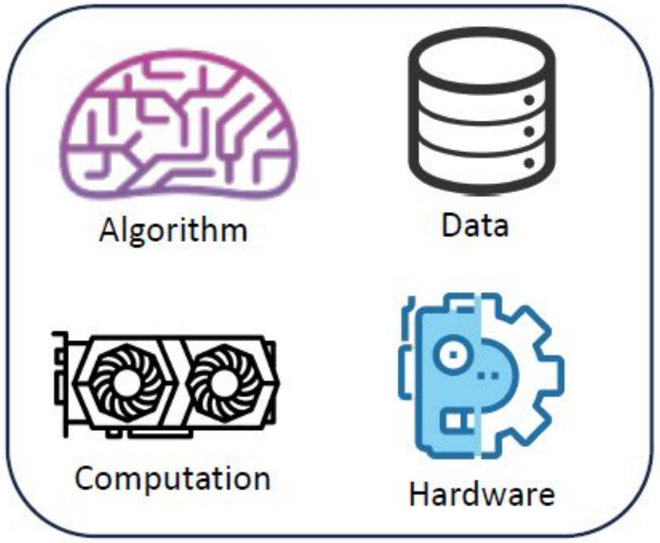


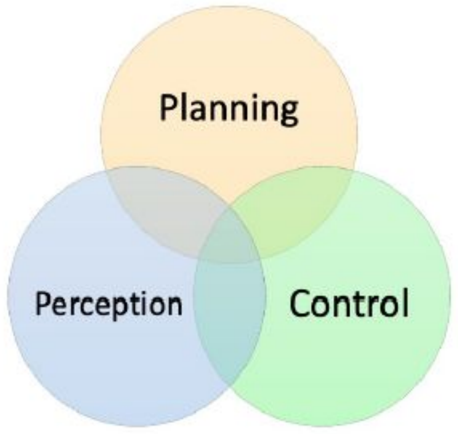
- Progressed in terms of domain-specific embodied intelligence, either learning-based or not

- Very far from general-purpose embodied intelligence!

Why Do We Need Robot Learning?

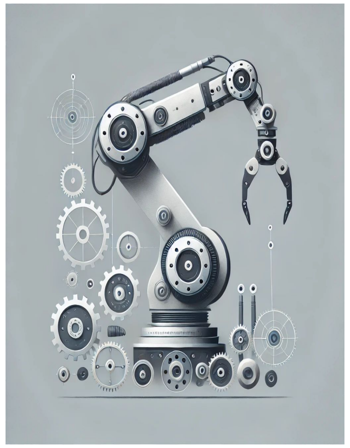
- Data increases--- challenging algorithm, computation, and hardware
- Modeling the world could be hard
- Environments/tasks might change
- The policy structure might be limited
- The optimization solver might be wrong
- Assumptions might not hold
- Non-learning methods often separately design each module



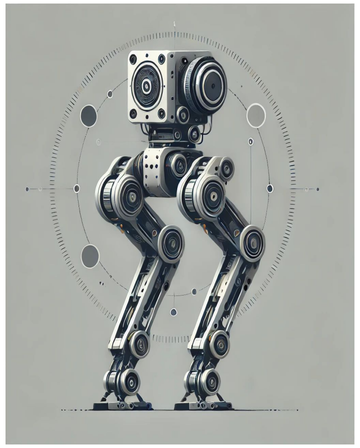


Types of Robots

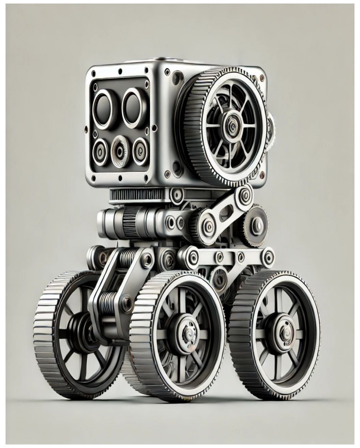
<u>Manipulator</u>



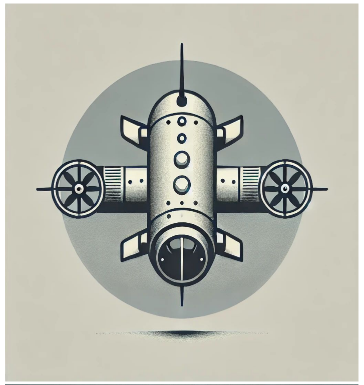
<u>Legged robot</u>

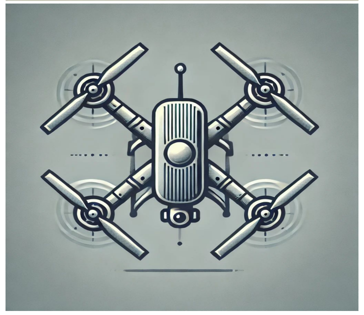


<u>Wheeled robot</u>



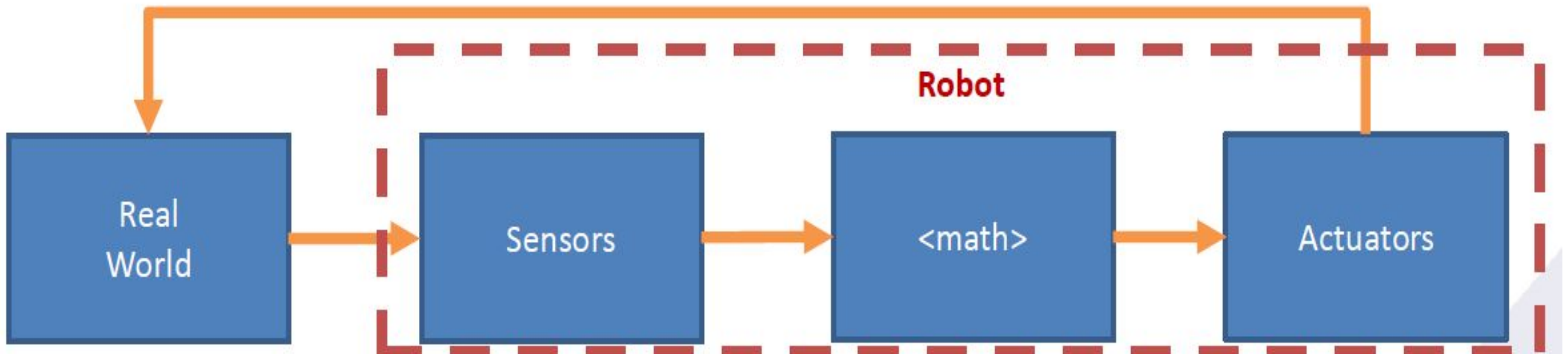
<u>Autonomous under water/Aerial Vehicle</u>





Do you know about the architecture of the listed robot types?
How the works, communicate and achieve their goal?

Robotics Architecture



Simplified Architecture

Robotics Architecture

- Robot software systems tend to be complex. Due to:
  - The need to control diverse sensors and actuators in real time, in the face of significant uncertainty and noise.
- Robot systems must work to achieve tasks while monitoring for, and reacting to, unexpected situations.
- Doing all this concurrently and asynchronously adds immensely to system complexity.
- The use of a well-conceived architecture, together with programming tools that support the architecture, can often help to manage that complexity.
- There is no single architecture that is best for all applications – different architectures have different advantages and disadvantages.

Robotics Architecture

- Definition: The term *robot architecture* is often used to refer to two related, but distinct, concepts.
  - Architectural structure refers to how a system is divided into subsystems and how those subsystems interact.
  - Architectural style refers to the computational concepts that underlie a given system
- All robotic systems use some architectural structure and style.
- However, in many existing robot systems it is difficult to pin down precisely the architecture being used.
- Major types of architectural components:
  - Behavioral control
  - Executives, and
  - Task planners

Robotics Architecture

- Robot architectures are distinguished from other software architectures because of the special needs of robot systems.
- Most importantly robot systems need to interact asynchronously, in real time, with an uncertain, often dynamic, environment.
- Many robot systems need to respond at varying temporal scopes – from millisecond feedback control to minutes, or hours, for complex tasks

Modularity and Hierarchy

- One common feature of robot architectures is modular decomposition of systems into simpler, largely independent pieces.
  - Decreases overall system complexity and increases overall reliability.
- System decomposition is hierarchical –modular components are themselves built on top of other modular components.

"While hierarchical decomposition of robotic systems is generally regarded as a desirable quality, debate continues over the dimensions along which to decompose".

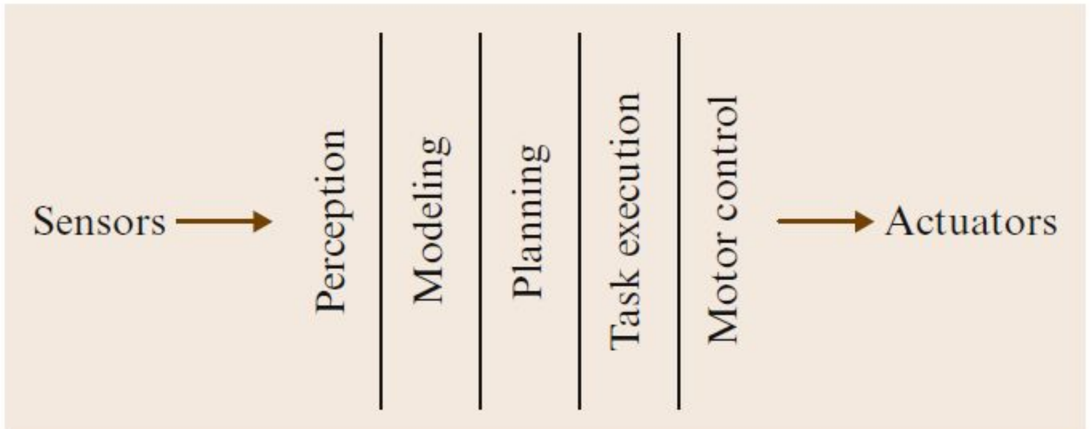
- Some architectures decompose along a temporal dimension – each layer in the hierarchy operates at a characteristic frequency an order of magnitude slower than the layer below.
  - Useful when dealing with both local and global navigation

- In other architectures , the hierarchy is based on task abstraction – tasks at one layer are achieved by invoking a set of tasks at lower levels.

Architectural styles can often be found to accommodate different needs.

Robotics Architecture: History

- Robot architectures and programming began in the late 1960s with the Shakey robot at Stanford University.
- Shakey’s architecture was decomposed into three functional elements: sensing, planning, and executing:
  - Sensing system translated the camera image into an internal world model.
  - Planner took the internal world model and a goal and generated a plan (i. e., a series of actions) that would achieve the goal.
  - Executor took the plan and sent the actions to the robot.
- This approach has been called the sense–plan–act (SPA) paradigm

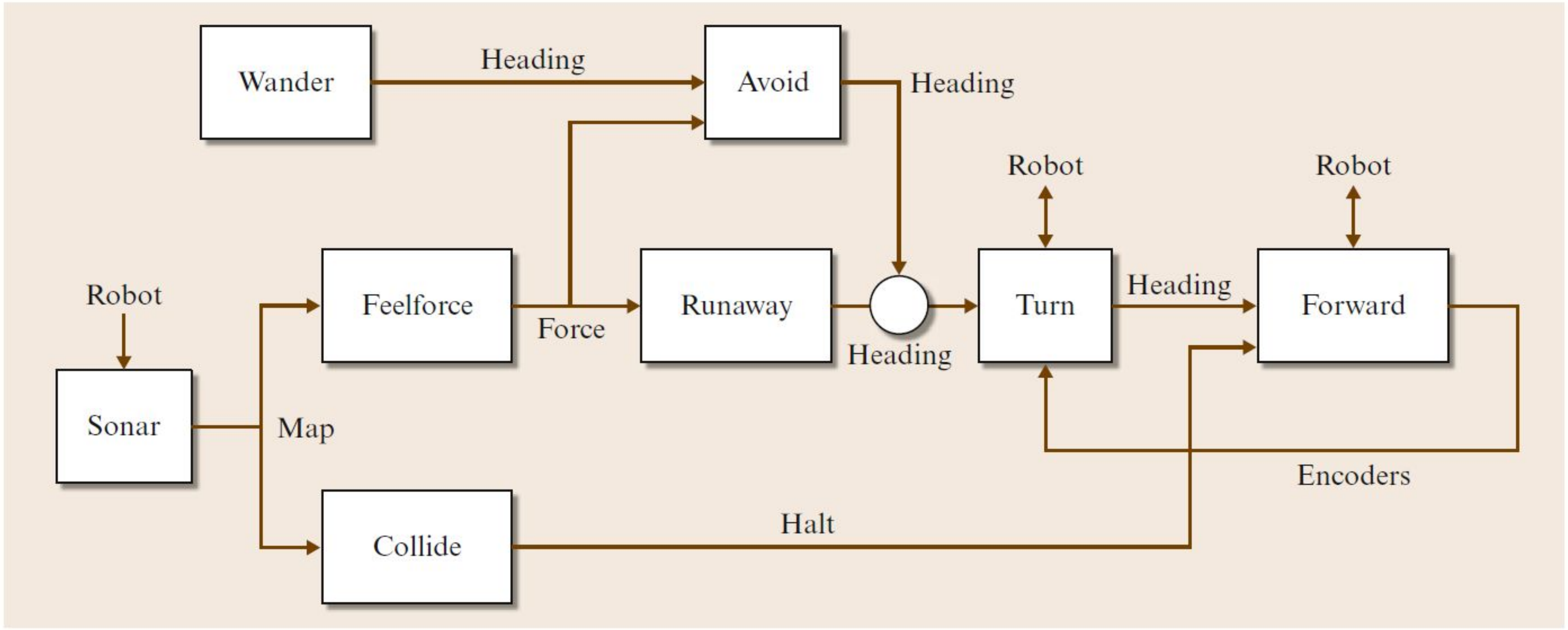


Robotics Architecture: History

- In the early 1980s, it became apparent that the SPA paradigm had problems.
  - Planning in any real-world domain took a long time, and the robot would be blocked, waiting for planning to complete.
  - Execution of a plan without involving sensing was dangerous in a dynamic world
- Several new robot control architecture paradigms began to emerge
  - The most influential work, however, was the subsumption architecture of Brooks
  - A subsumption architecture is built from layers of interacting finite-state machines – each connecting sensors to actuators directly

Represents a shift from traditional AI, which relied heavily on complex centralized planning and symbolic reasoning, to a more distributed, layered approach for controlling robots.

Robotics Architecture: History



Example of the Subsumption architecture

Explain how the submission architecture different from

Robotics Architecture: History

- Two levels:
  - Low-Level Behaviors are the basic actions or simple behaviors that a robot can perform without much processing or abstraction.
  - High-Level Behaviors are more complex actions that require coordination of multiple low-level behaviors. They are goal-oriented and can involve decision-making and planning.

- Subsumption had an arbitration mechanism that enabled higher-level behaviors to override signals from lower-level behaviors.
  - This behavior is always active and the robot is always driving somewhere.

- Higher-level behavior could take sensor input, detect obstacles, and steer the robot away from them.

Robotics Architecture: History

- However, if it detects an obstacle it overrides the lower-level behavior and steers the robot away.
- As soon as the obstacle is gone (and the higher-level behavior stops sending signals), the lower level behavior gets control again.
- Multiple, interacting layers of behaviors could be built to produce more and more complex robots.
- Many robots were built using the subsumption approach – most at MIT
- SPA robots were slow and ponderous, Subsumption robots were fast and reactive.

Robotics Architecture: History

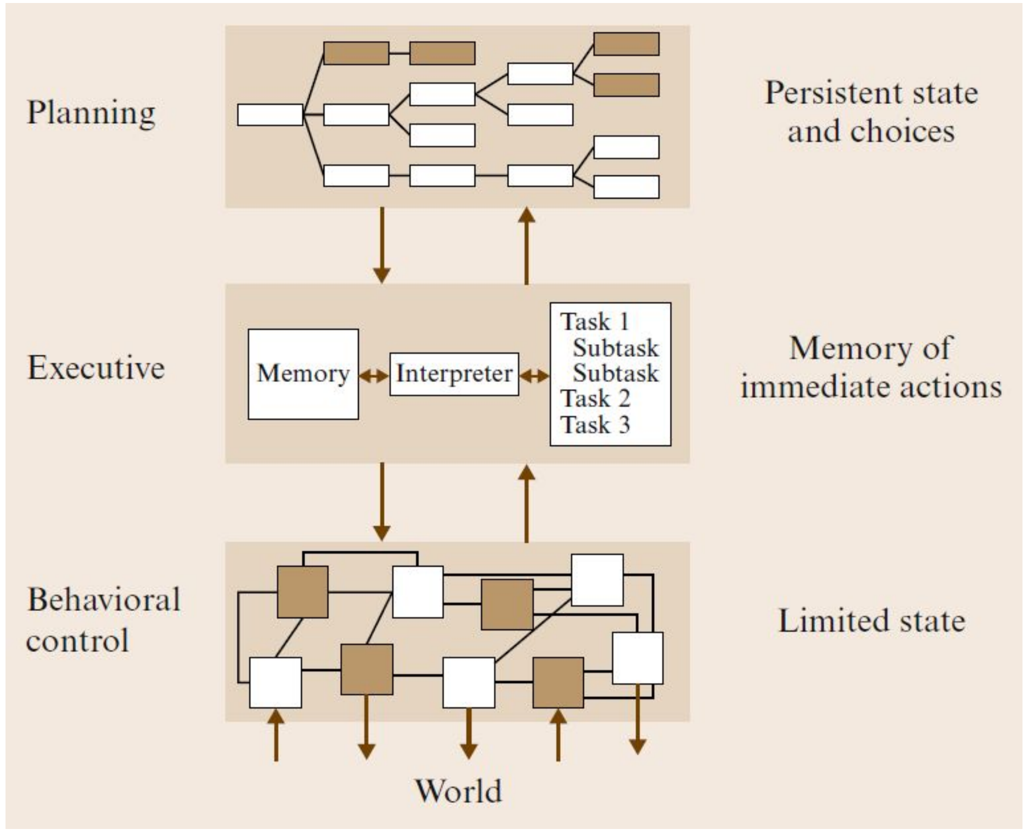
- Several behavioral architectures arose in addition to Subsumption, often with different arbitration schemes for combining the outputs of behaviors

- Behavior-based robots soon reached limits in their capabilities.
  - It proved very difficult to compose behaviors to achieve long-range goals and it proved almost impossible to optimize robot behavior.

- This realization led to the development of layered, or tiered, robot control architectures.

Layered Robot Control Architectures

- Reactive action packages (RAPs) system created by Firby, three-layer architecture



- Independently and concurrently, *Bonasso* at MITRE [8.32] devised an architecture that began at the bottom layer with robot behaviors programmed in the Rex language as synchronous circuits

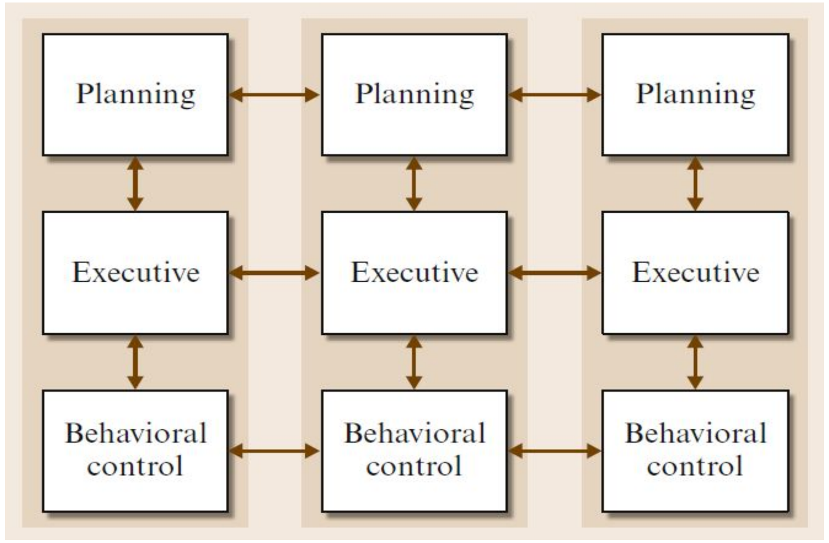
- This work culminated in the 3T architecture (named after its three tiers of interacting control processes – planning, sequencing, and real-time control), which has been used on many generations of robots

Layered Robot Control Architectures

- Remote agent is an architecture for the autonomous control of spacecraft
- Consists of four layers – a control (behavioral) layer, an executive, a planner/scheduler, and mode identification and recovery (MIR) that combines fault detection and recovery
  - The control layer is the traditional spacecraft real-time control system.
  - The executive is the core of the architecture
    - It decomposes, selects, and monitors task execution, performs fault recovery, and does resource management turning devices on and off at appropriate times to conserve limited spacecraft power
  - The planner/scheduler is a batch process that takes:
    - Goals, an initial (projected) state, and currently scheduled activities, and produces plans that include flexible ranges on start and end times of tasks.
    - The plan also includes a task to reinvoke the planner to produce the next plan segment.
- An important part of the remote agent is configuration management:
  - Configuring hardware to support tasks and monitoring that the hardware remains in known, stable states.

Layered Robot Control Architectures

- Syndicate architecture extends the 3T model to multirobot coordination
- Each layer interfaces not only with the layers above and below, as usual, but also with the layers of the other robots at the same level
- Distributed control loops can be designed at multiple levels of abstraction.



The Syndicate multirobot architecture

Layered Robot Control Architectures

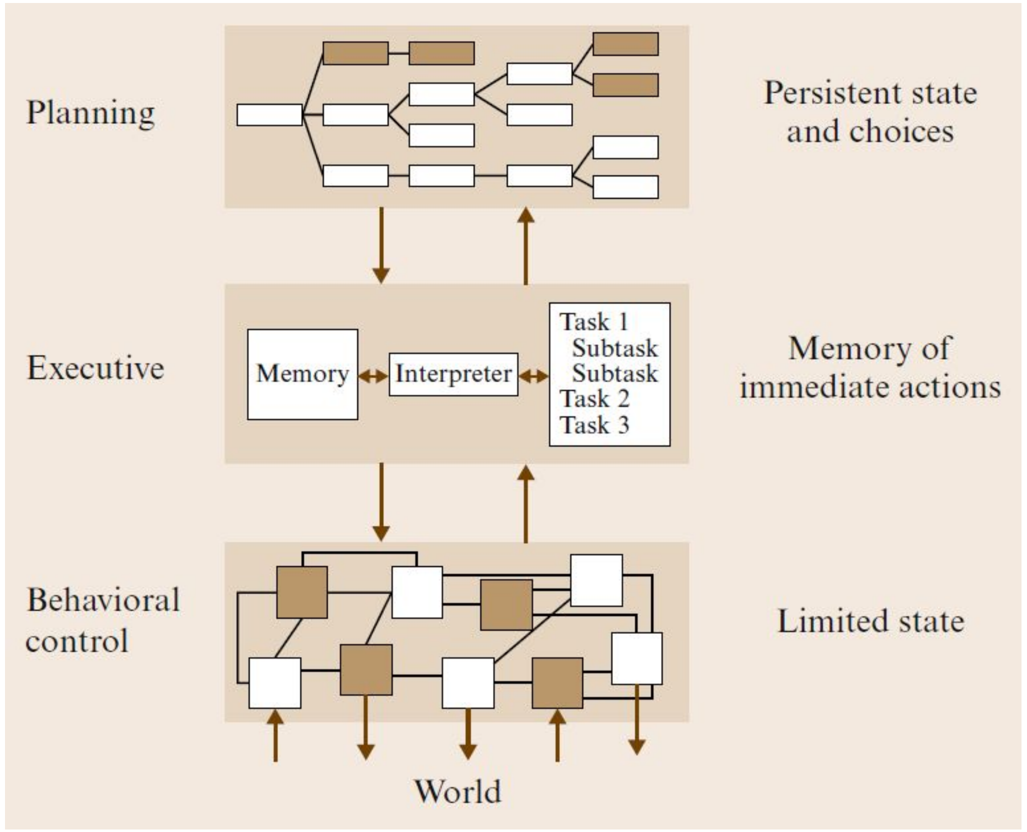
- Three-layered robot architectures are very popular, various two-layered architectures have been investigated by researchers.

- The coupled layered architecture for robot autonomy (CLARAty) was designed to provide reusable software for NASA’s space robots, especially planetary rovers

- CLARAty consists of a functional and a decision layers.
  - Functional layer is a hierarchy of object-oriented algorithms that provide more and more abstract interfaces to the robot, such as motor control, vehicle control, sensor-based navigation, and mobile manipulation.
  - Each object provides a generic interface that is hardware independent, so that the same algorithms can run on different hardware.
  - The decision layer combines planning and executive capabilities.

Architectural Components

- The three-tiered architecture as the prototype for the components



All these tiers need to work together and exchange information.

The lowest tier (or layer) -- behavioral control -- tied most closely to sensors and actuators.

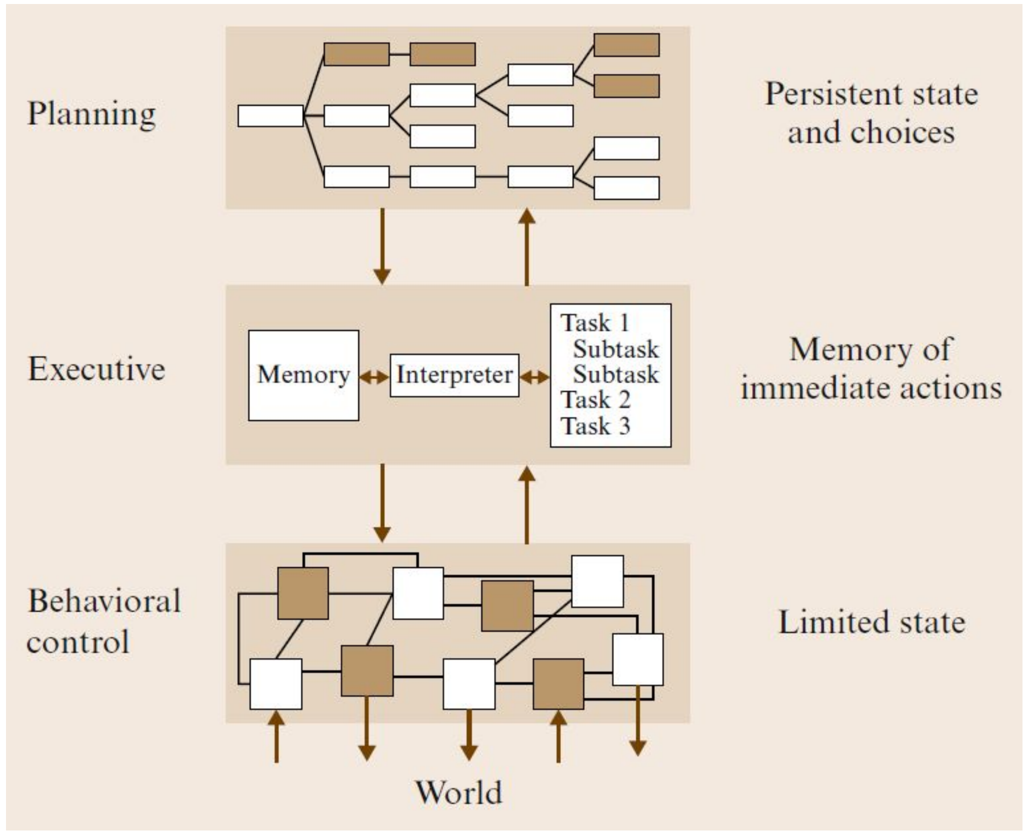
The second tier is the executive layer– responsible for choosing the current behaviors of the robot to achieve a task.

The highest tier is the task-planning layer --- responsible for achieving long-term goals of the robot within resource constraints.

“Ex: Office delivery robot,
the behavioral layer is responsible for moving the robot around rooms and hallways for avoiding obstacles, for opening doors, etc. The executive layer coordinates the behavioral layer to achieve tasks such as leaving a room, going to an office, etc. The task-planning layer is responsible for deciding the order of deliveries to minimize time, taking into account delivery priorities, scheduling, recharging, etc. The task-planning layer sends tasks (e.g., exit the room, go to office 110) to the executive.

Architectural Components

- The three-tiered architecture as the prototype for the components



All these tiers need to work together and exchange information.

“Ex: Office delivery robot,

- The behavioral layer is responsible for moving the robot around rooms and hallways for avoiding obstacles, for opening doors, etc.
- The executive layer coordinates the behavioral layer to achieve tasks such as leaving a room, going to an office, etc.
- The task-planning layer is responsible for deciding the order of deliveries to minimize time, taking into account delivery priorities, scheduling, recharging, etc.
- The task-planning layer sends tasks (e.g., exit the room, go to office 110) to the executive.”

Architectural Components

- Problem of connecting components to each other?

Connecting Components

- Architecture components need to communicate with each other, need to both exchange data and send commands.
- Choice of how components communicate (often called the middleware) is most important and most constraining of the many decisions a robot architecture designer will make.
- Majority of the debugging time in developing robot architectures have to do with communication between components.
- Once a communication mechanism is chosen it becomes extremely difficult to change, so early decisions persist for many years.
- There are two basic approaches to communication: client–server and publish–subscribe.

Architectural Components

Client–Server

- Also called a point-to-point communication protocol, components talk directly with other components.

- Example: remote procedure call (RPC) protocols in which one component (the client) can call functions and procedures of another component (the server).

  - Common object request broker architecture (CORBA) allows for one component to call object methods that are implemented by another component.

  - Available in most major object-oriented languages.

  - Disadvantage of CORBA -- introduces quite a bit of additional code into applications.

Architectural Components

- The biggest advantage of a client– server protocol the -- interfaces are very clearly defined in advance and everyone knows when the interface has changed.
- It allows for a distributed approach to communication with no central module that must distribute data.
- Disadvantage of client–server protocols --- introduce significant overhead, if many components need the same information.

Architectural Components

Publish–Subscribe

- also called a broadcast protocol, a component publishes data and any other component can subscribe to that data.

- Centralized process routes data between publishers and subscribers.

- There are several existing publish–subscribe middleware solutions.
  - A popular one for robotics is the real-time innovations' (RTI) data distribution service (DDS), formerly the network data distribution service (NDDS)
  - Another popular publish–subscribe paradigm is IPC developed at Carnegie Mellon University

- Many publish subscribe protocols are converging on using extensible markup language (XML) descriptions to define the data being published

Architectural Components

- Publish–subscribe protocols are often more difficult to debug because the syntax of the message is often hidden in a simple string type.
- Are also not as readable when it comes to sending commands from one module to another.
- Publish–subscribe protocols often use a single central server to dispatch messages to all subscribers, providing a single point of failure and potential bottleneck.

JAUS

- Recently, a standard has emerged in the defense robotics community not only for a communication protocol but also for definitions of messages that are sent via that communication protocol.
- The joint architecture for unmanned systems (JAUS) defines a set of reusable messages and interfaces that can be used to command autonomous systems

Architectural Components

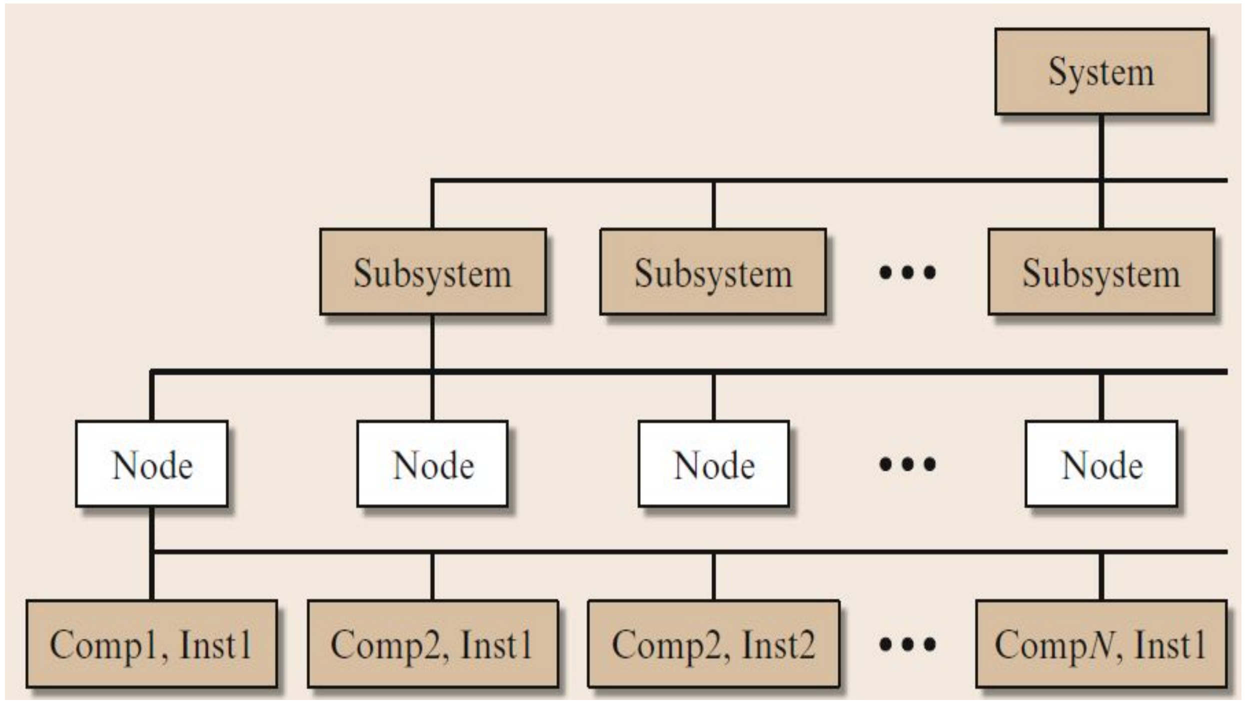
- These reusable components reduce the cost of integrating new hardware components into autonomous systems.

- Reuse also allows for components developed for one autonomous system to be used by another autonomous system.

- JAUS has two components: a domain model and a reference architecture.
  - The domain model is a representation of the unmanned systems' functions and information
  - The reference architecture provides a well-defined set of messages. Messages cause actions to commence, information to be exchanged,and events to occur.

- Everything that occurs in a JAUS system is precipitated by messages. This strategy makes JAUS a component-based, message-passing architecture.

Architectural Components



The JAUS reference architecture topology

- The topology defines the *system* as the collection of vehicles, operator control units (OCU), and infrastructure necessary to provide the full robotic capability.
  - Subsystems are individual units (e.g., vehicles or OCUs) in the system.
  - Nodes define a distinct processing capability within the architecture and route JAUS messages to components.
  - Components provide the different execution capabilities and respond directly to command messages.

- Messages have headers that follow a specific format and include message type, destination address (e.g., system, subsystem, node, and component), priority, etc.

While JAUS is primarily point to point, JAUS messages can also be marked as *broadcast* and distributed to all components.

Architectural Components

Behavioral Control

- Behavioral control represents the lowest level of control in a robot architecture…. connects sensors and actuators.
- Hand-crafted functions written in C or C++, there have been specialized languages developed for behavioral control, including ALFA, Behavioral Language, and Rex

- In architectures such as 3T, the behavioral layer functions as a Brooksian machine – that is, the layer is composed of a small number of behaviors (also called skills) that perceive the environment and carry out the actions of the robot.

- Example: Consider an office delivery robot that operates in a typical office building. The behavioral control layer contains the control functions necessary to move around in the building and carry out delivery tasks. Assuming the robot has an a priori map of the building, what are some possible behaviors for this robot include?

Architectural Components

- In 3T, not all of the behaviors are active at the same time. Typically, only a few behaviors that do not conflict would be active at a time.

- The executive layer is responsible for activating and deactivating behaviors to achieve higherlevel tasks and to avoid conflicts between two behaviors competing for the same resource (e.g., an actuator).

Situated Behaviors

- The behavior works only in very specific situations.

- The behavior is not responsible for putting the robot in the particular situation. However, it should recognize that the situation is not appropriate and signal as such.

Architectual Components

Cognizant Failure

- A key requirement for behaviors is that they know when they are not working, called cognizant failure.
- Common problem with early Subsumption robots is that the behaviors did not know they were failing and continued to take actions that were not resulting in progress.

Implementation Constraints

- The algorithms used for behaviors should be constant in state and time complexity.

- Behaviors should simply be transfer functions that take in signals (from sensors or other behaviors) and send out signals (to actuators or other behaviors), and repeat these several times a second.

Architectural Components

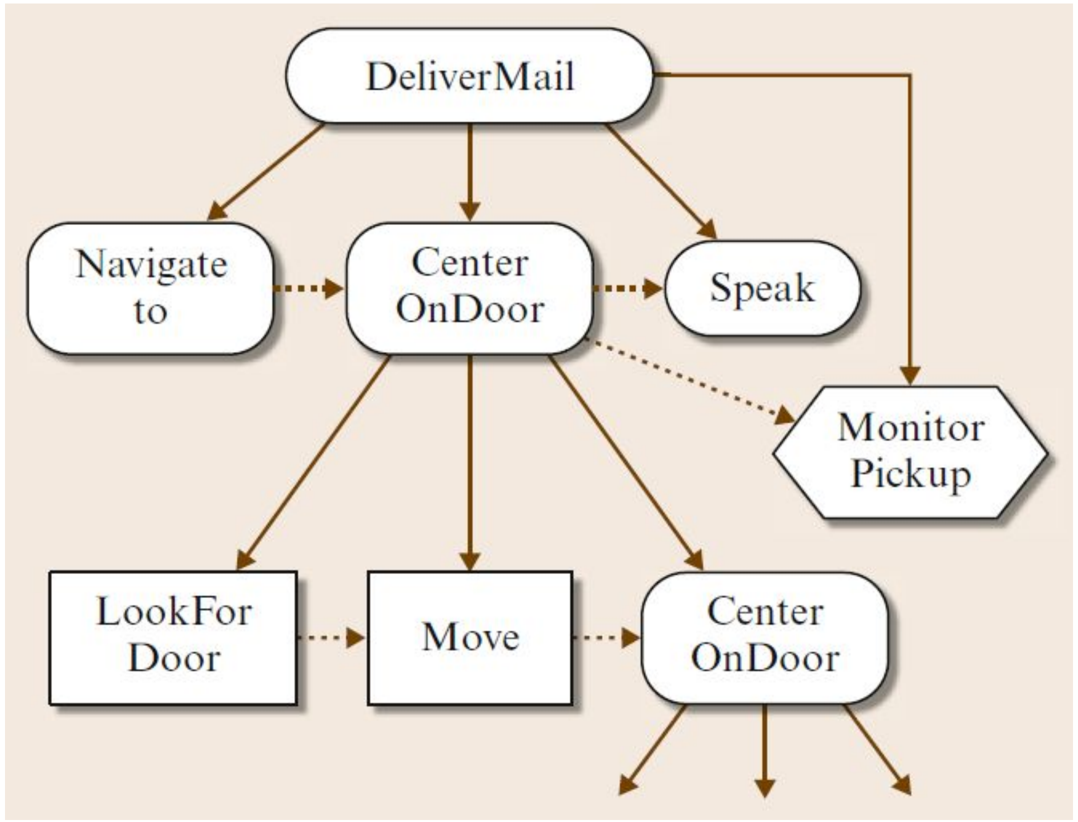
Executive

- Is the interface between the numerical behavioral control and the symbolic planning layers.
- The executive is responsible for translating high-level plans into low-level behaviors, invoking behaviors at the appropriate times, monitoring execution, and handling exceptions.
- Some executives also allocate and monitor resource usage, although that functionality is more commonly performed by the planning layer.

Continuing the example of an office delivery robot, the main high-level task would be to deliver mail to a given office. The executive would decompose this task into a set of subtasks. It may use a geometric path planner to determine the sequence of corridors to move down and intersections at which to turn. If there are doorways along the route, a task would be inserted to open and pass through the door. At the last corridor, the executive would add a concurrent task that looks for the office number.

Architectural Components

- Capability of the executive layer:
  - The executive decomposes high-level tasks (goals) into low-level tasks (behaviors).



- Done in a procedural fashion: the knowledge encoded in the executive describes *how* to achieve tasks, rather than describing *what* needs to be done and having the executive figure out the *how* by itself.

- Sometimes it uses specialized planning techniques, such as the route planner used in the example above.
- The decomposition is typically a hierarchical task tree, with the leaves of the task tree being invocations and parameterizations of behaviors.

Besides decomposing tasks into subtasks, executives add and maintain temporal constraints between tasks.

The executive is responsible for dispatching tasks when their temporal constraints are satisfied.

By managing such temporal aspects, the executive helps optimize robot performance, making sure tasks are synchronized and completed efficiently and effectively within the designated timeframe.

Architectural Components

- The final two important executive capabilities are execution monitoring and error recovery.. Why?
  - The behaviors are situated, and the situation may change unexpectedly.
  - The behavior may move the robot into a state that is unexpected by the executive.
- Implementation Constraints
  - Petri nets are a popular choice for representing executive functions
  - Various languages have been developed specifically to assist programmers in implementing executive-level capabilities.

Architectural Components

- Various languages have been developed specifically to assist programmers in implementing executive-level capabilities.
  - Reactive action packages (RAPs): are a structured approach used for representing and executing complex robotic behaviors through hierarchical procedures.
    - RAPs were notably used in the Shakey robot project
  - The procedural reasoning system (PRS): a framework designed to enable intelligent agents to reason and act in real-time.
    - PRS has been applied in space missions, air-traffic control, and other systems requiring high levels of autonomy and adaptability.
  - The execution support language (ESL): ESL is a high-level language used to define and support task execution within complex systems.
    - ESL is used to ensure that tasks are executed predictably and within predefined constraints

Architectural Components

- Various languages have been developed specifically to assist programmers in implementing executive-level capabilities.
  - The task description language(TDL): TDL is used for specifying, organizing, and controlling the execution of complex tasks and behaviors in robotic systems.
    - Commonly used in robotics and AI for dynamic task planning and management
  - The plan execution interchange language (PLEXIL): PLEXIL is a formal plan execution language developed by NASA to support autonomous spacecraft and robotic missions.
    - Used in space exploration and complex robotic missions
- All of these executive languages provide support for hierarchical decomposition of tasks into subtasks.

Architectural Components

- Planning
  - The planning component of our prototype layered architecture is responsible for determining the long-range activities of the robot based on high-level goals.

In our running example of an office delivery robot, the planning component would look at the day’s deliveries, the resources of the robot, and a map, and determine the optimal delivery routes and schedule, including when the robot should recharge.

  - The planning component is also responsible for replanning when the situation changes

Architectural Components

- Types of Planning
  - The two most common approaches used are hierarchical task net (HTN) planners and planner/schedulers.
  - HTN planners decompose tasks into subtasks, in a manner similar to what many executives do.
  - The main differences are that HTN planners typically operate at higher levels of abstraction, take resource utilization into account, and have methods for dealing with conflicts between tasks
- Planner/schedulers are useful in domains where time and resources are limited.
- They create high-level plans that schedule when tasks should occur, but typically leave it to the executive to determine exactly how to achieve the tasks.

Architectural Components

- Integrating Planning and Execution
  - There are two main approaches to the integration of the planning and execution components in robotic architectures.
    - Planning component is invoked as needed by the executive and returns a plan.
    - The planning component sends high-level tasks down to the executive as required and monitors the progress of those tasks.

The Art of Robot Architectures

- Designing a robot architecture is much more of an art than a science.
- The goal of an architecture is to make programming a robot easier, safer, and more flexible.
- The art of designing a robotic architecture starts with a set of questions:
  - What are the tasks the robot will be performing? Are they long-term tasks? Short-term? User-initiated? Robot-initiated? Are the tasks repetitive or different across time?
  - What actions are necessary to perform the tasks? How are those actions represented? How are those actions coordinated?

The Art of Robot Architectures

- Data
  - What data is necessary to do the tasks?
  - How will the robot obtain that data from the environment or from the users?
  - What sensors will capture/produce the data?
  - What representations will be used for the data?
  - What processes will abstract the sensory data into representations internal to the architecture?
  - How often does the data need to be updated? How often can it be updated?

The Art of Robot Architectures

- Computational Capability
  - What computational capabilities will the robot have?
  - What data will these computational capabilities produce?
  - What data will they consume?
  - How will the computational capabilities of a robot be divided, structured, and interconnected?
  - What is the best decomposition/granularity of computational capabilities?
  - How much does each computational capability have to know about the other capabilities?
  - Are there legacy computational capabilities (from other robots, other robot projects, etc.) that will be used?
  - Where will the different computational capabilities reside (e.g., onboard or offboard)?

The Art of Robot Architectures

- Users
  - Who are the robot’s users?
  - What will they command the robot to do?
  - What information will they want to see from the robot?
  - What understanding do they need of the robot’s computational capabilities? How will the user know what the robot is doing?
  - Is the user interaction peer to peer, supervisory, or as a bystander?

Current challenges in AI-driven robotics

- Generalization and Adaptability
    - Generalization is critical for deploying robots in diverse real-world scenarios, from hospitals to homes, where conditions and tasks change unpredictably.
- Real-time Perception and Decision-Making
    - Accurate and fast perception through sensors is challenging, especially in dynamic or cluttered environments.
- Data Scarcity and Quality
    - Training AI systems requires extensive, high-quality data, which is often expensive and time-consuming to gather, especially for specific applications.
    - Without diverse, high-quality datasets, AI models risk being biased, which can lead to poor performance and safety issues.

Current challenges in AI-driven robotics

- Energy Efficiency and Battery Life
  - Robotic systems require significant energy for computation, movement, and sensor processing, which can drain batteries quickly.

- Perception and Understanding of Complex Environments
  - Robots struggle with accurately perceiving and interpreting dynamic, cluttered, and unpredictable environments.
  - Navigation and Path Planning in Dynamic Settings

Current challenges in AI-driven robotics

- Human-Robot Interaction (HRI)
  - Developing robots that can understand and respond to complex human behaviors, emotions, and intentions is still a major hurdle.
- Safety and Reliability
  - AI-driven robots must operate reliably and safely, particularly in applications like healthcare, autonomous driving, and manufacturing.

2015 involved a worker at a Volkswagen plant in Germany who was fatally crushed by a robotic arm

- Ethics and Bias in Decision-Making
  - AI systems may inadvertently learn biases present in their training data, leading to potentially harmful decision-making, especially in sensitive applications like surveillance, law enforcement, or hiring.

This area raises significant ethical and regulatory questions regarding the deployment of autonomous lethal systems.

Term Paper : Due Date Nov 22, 2024

- AI has significantly enhanced the capabilities and performance of robots, making them more adaptable, intelligent, and efficient.

1. Enhanced Perception and Sensing
2. Autonomous Navigation and Path Planning
3. Learning from Experience
4. Improved Manipulation and Dexterity
5. Human-Robot Interaction (HRI)
6. Predictive Maintenance and Self-Diagnostics
7. Greater Adaptability in Unstructured Environments
8. Energy Efficiency and Task Optimization
9. Data-Driven Improvements and Continuous Learning
10. Collaboration with Other Robots and IoT Integration