AI for Mobile Robots
- CSIP5202 -

Architecture & Behaviour

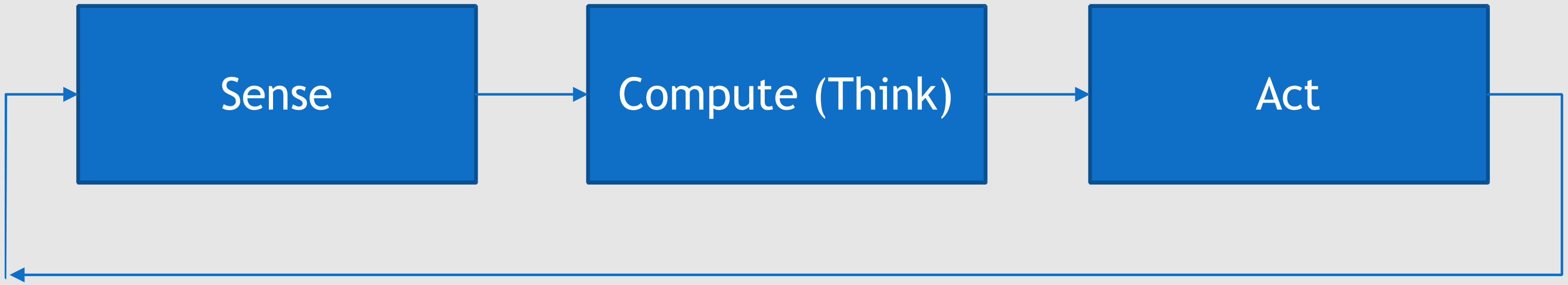
Overview

- Control Models
  - The sense-think-act control cycle
  - Model-based controllers
- Reactive Controllers
  - Behaviour-based controllers
- Other Approaches
  - Other Reactive Controllers
  - Hybrid Controllers
  - Learning Robots
- Practicalities

Control Models

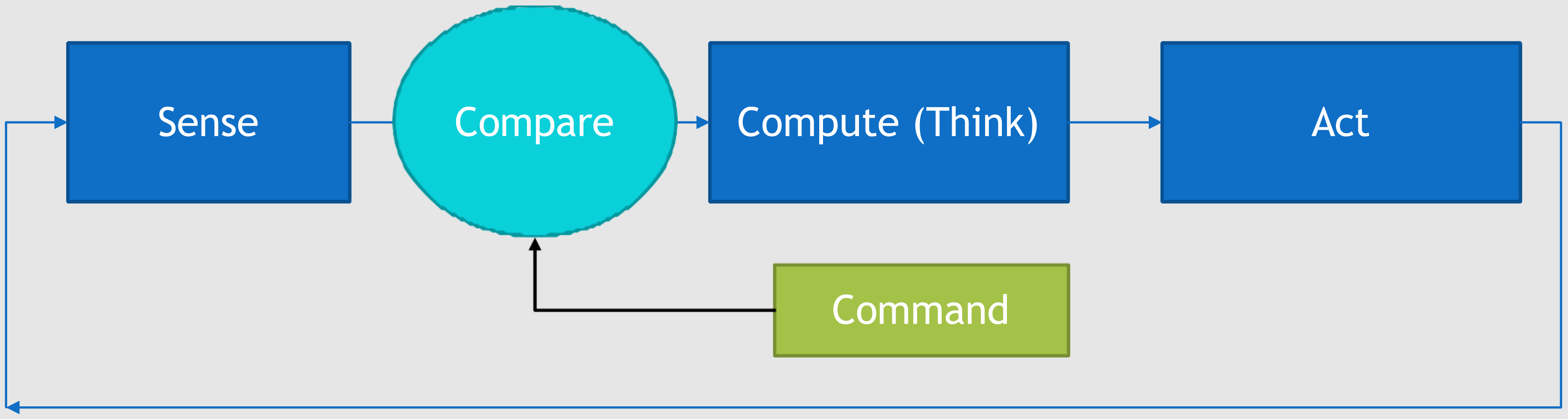
The Control Cycle

- A fundamental methodology derived in the early days of robotics from engineering principles is the ‘sense-think-act’ cycle
  - The basic idea is to use feedback to attempt to make the actual state equal to the desired state



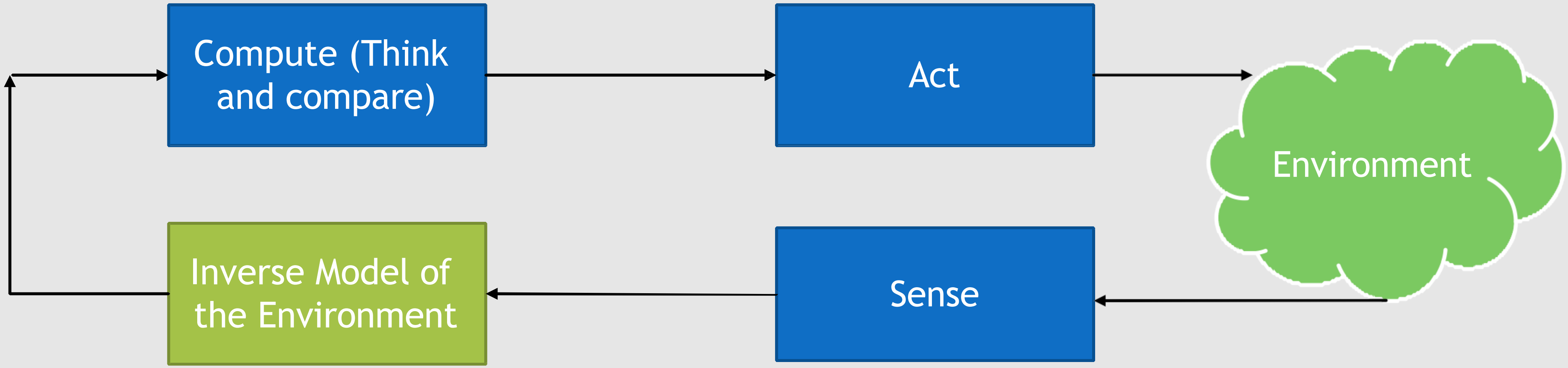
The Control Cycle

- This is based on classic control theory
  - The principle is to continuously attempt to minimise the error between the actual state and the desired state (as defined by a command, plan or instruction).



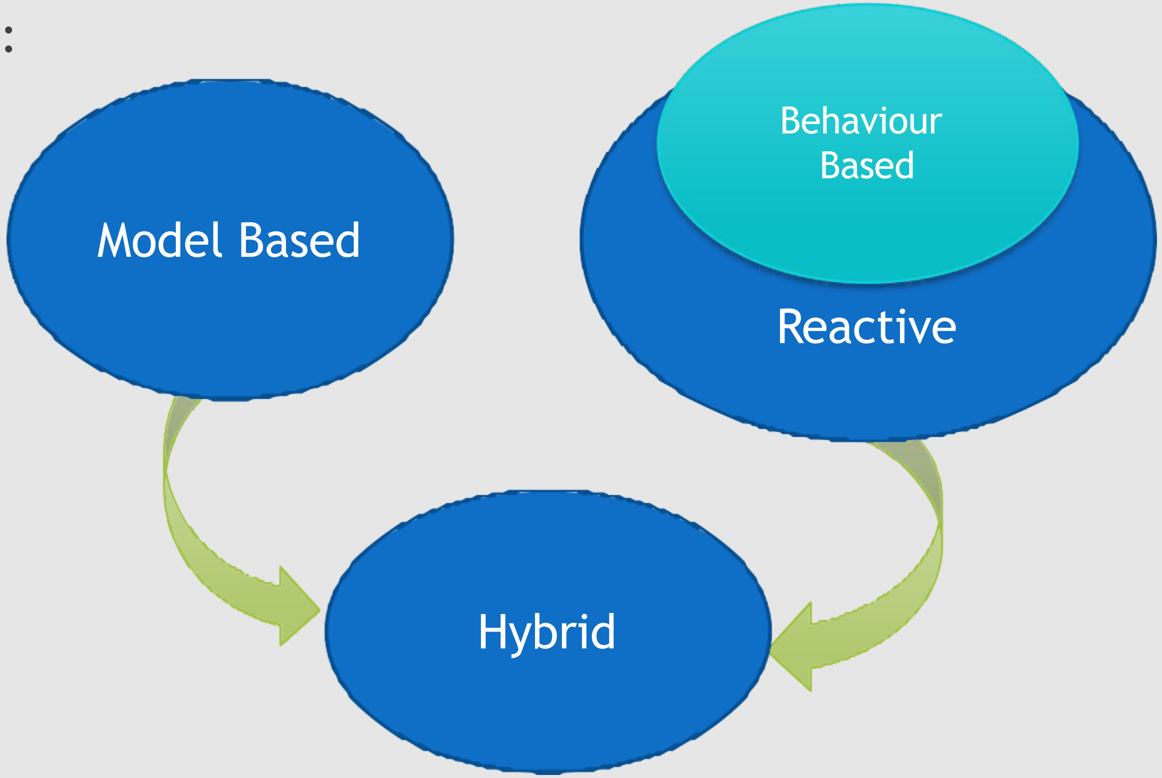
The Control Cycle

- Classic control theory uses a mathematical model to aid in the comparison.
  - An ‘inverse’ model is used to compare the set-point with the actual behaviour of the system.



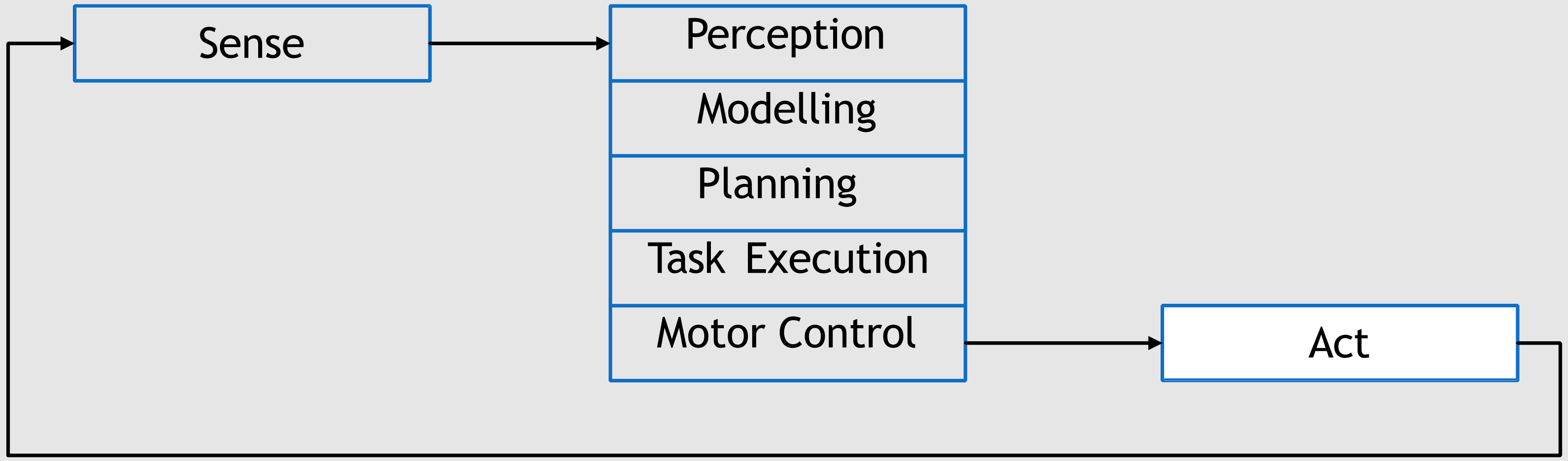
Control Architectures

- A variety of different approaches have been tried for implementing the sense-think-act control cycle.
- These approaches can be categorised as:
  - Model-based
  - Reactive
  - Hybrid



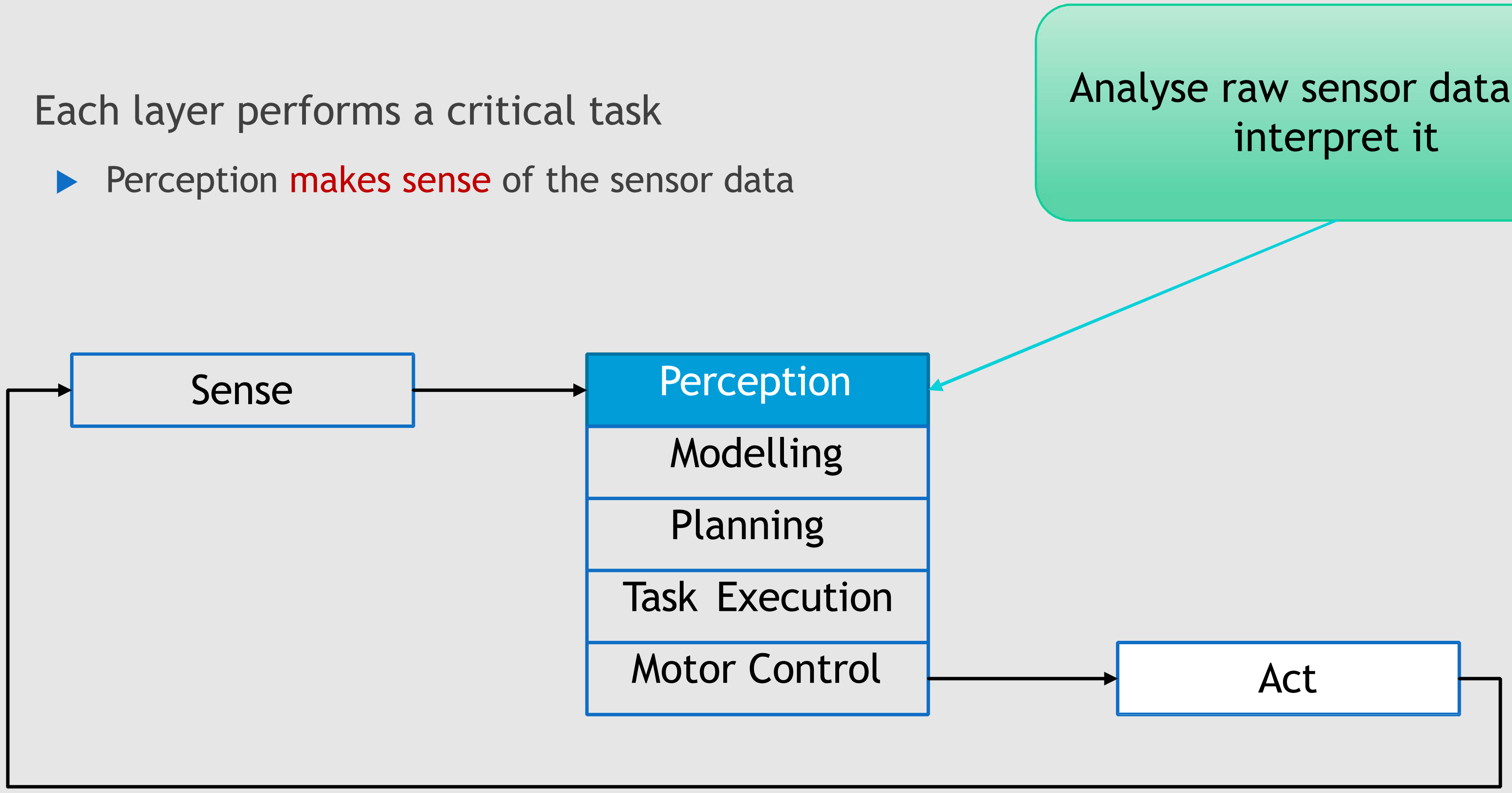
Model Based Control

- A symbolic internal 'world-model' is maintained
  - The sub-tasks are decomposed into functional layers
  - Similar to 'classical' artificial intelligence approaches



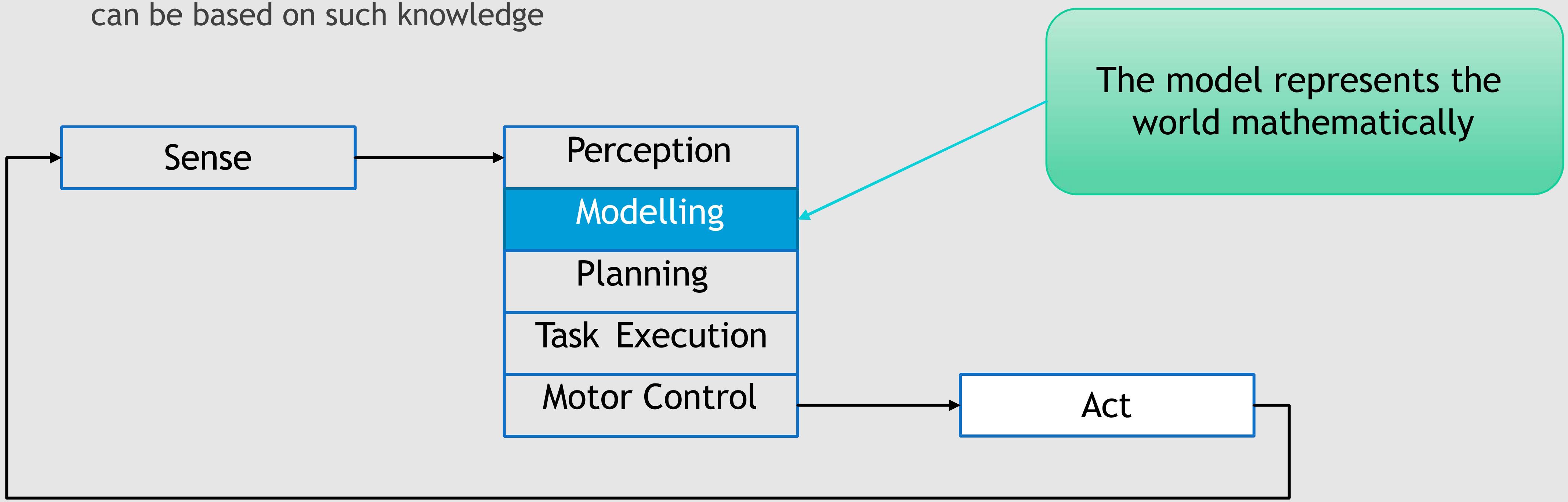
Model Based Control

- Each layer performs a critical task
  - Perception makes sense of the sensor data



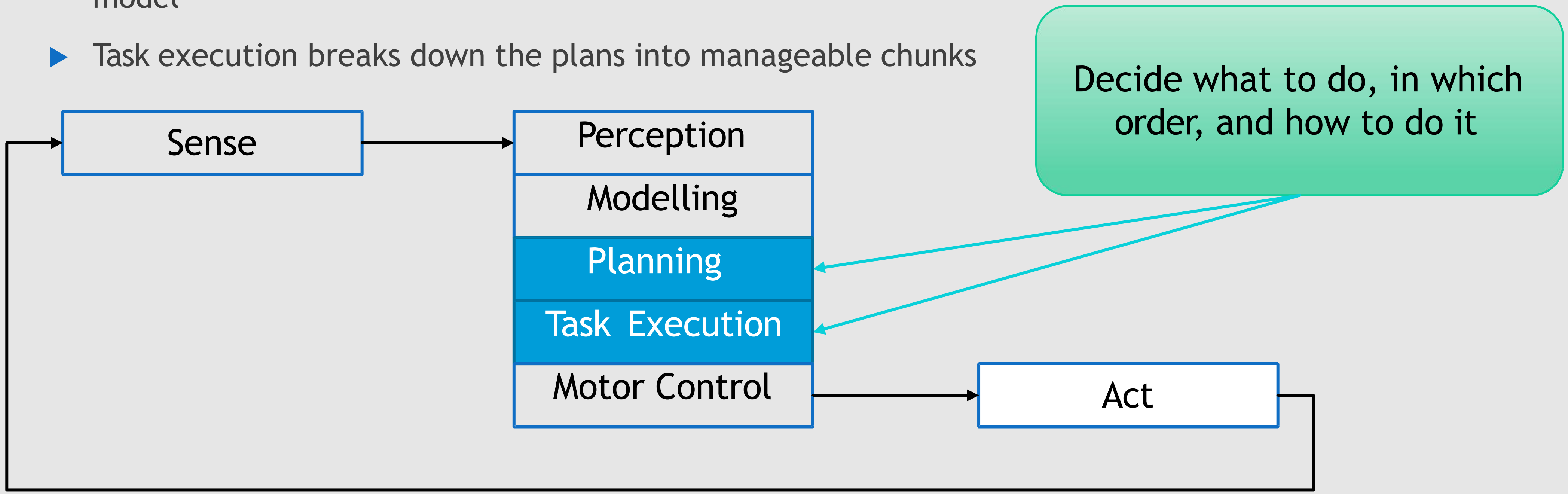
Model Based Control

- Each layer performs a critical task
  - The model keeps information on the real world so that computations and planning can be based on such knowledge



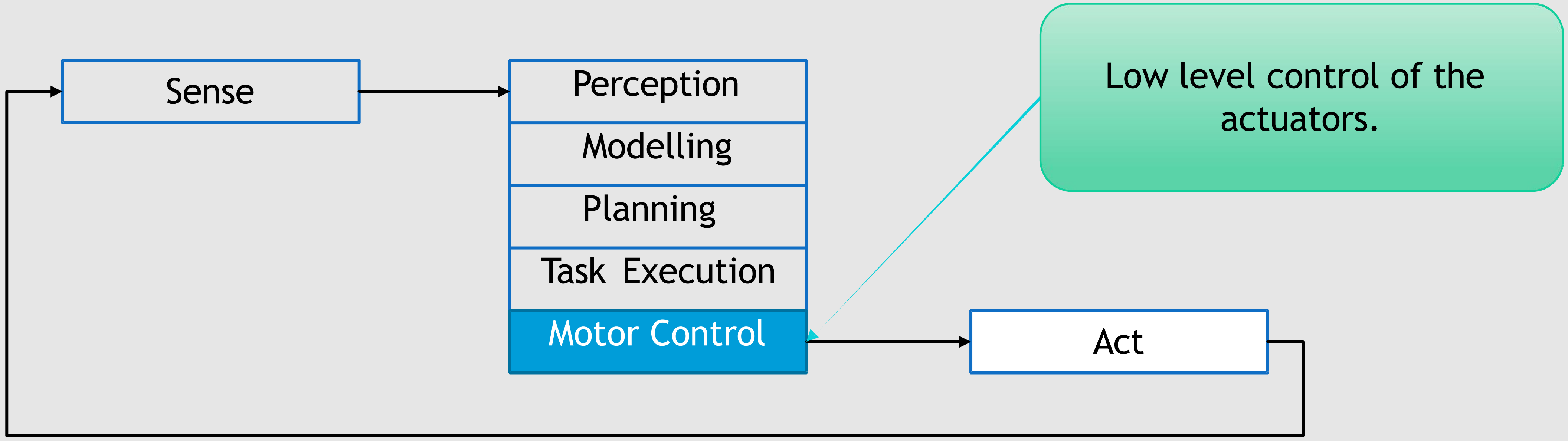
Model Based Control

- Each layer performs a critical task
  - Planning makes the scheduling of tasks and decisions depending on the state of the model
  - Task execution breaks down the plans into manageable chunks



Model Based Control

- Each layer performs a critical task
  - Motor control executes the low level management of the actuators (not necessarily only motors)



Problems with Models

- An adequate, accurate, and up-to-date model must be maintained at all times
  - This might be very difficult in practice!
  - What if sensors detect an object that hasn’t been defined?
- A model-based system is brittle
  - If on of the functional layers fails (e.g hardware issues, software bugs), then the whole system fails
- Significant processing power is required
  - Maintaining the model takes time, so would there be slow responses?

Problems with Models

- For many applications it works very well, and lots of progress has been made in creating efficient models
- High quality hardware and formal programming methodologies increase reliability
- AI methodologies help make the systems robust to inaccuracies or changes in the models

- But it doesn't always solve the problem or is robust enough.

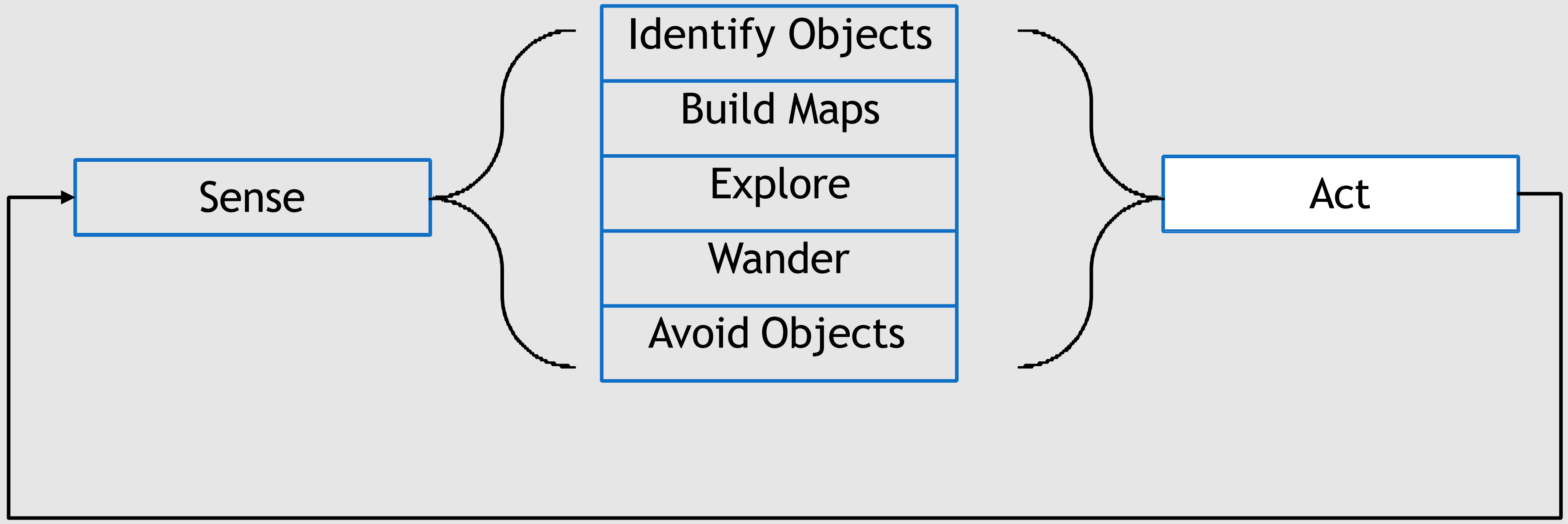
Reactive Robotics

Reactive Controllers

- In order to try to overcome the shortcomings of model-based robots, modern approaches have centred predominantly on simple reactive systems with minimal amounts of computation
  - ‘model-free systems’
- More correctly, the models are simple and implicit
  - the systems do not use symbolic models but, for example, a rule-set which tells a robot how to react to a corner when following a wall may be considered to be a simple, implicit model fragment
    - It implicitly encodes assumptions about the environment

Behaviour Based

- The control system is broken down into horizontal modules, or behaviours, that run in parallel
  - Each behaviour has direct access to sensor readings and can control the robot's motors directly

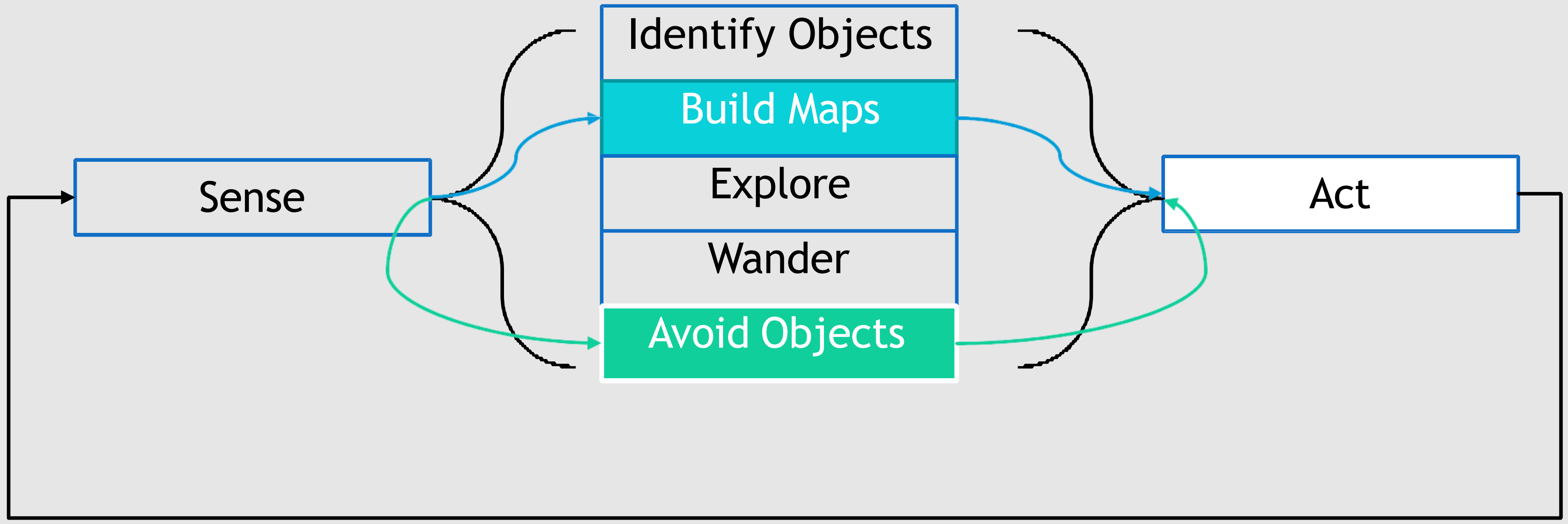


Behaviour Advantages

- It supports multiple goals and is more efficient
  - There is no functional hierarchy between layers
    - One layer does no call another layer
  - Each layer can work on different goals in parallel
  - Communication between layers is achieved via message passing which need not be synchronised
- The system is easier to design, debug, and extend
  - Each module can be designed and tested individually
- The system is robust
  - If one module fails, e.g. wander, then other layers, e.g. avoid obstacles, still function and behave correctly.

Behaviour Advantages

- The behaviours that run in parallel can work independently
  - Robust and easier to test



Behaviour Limitations

- It is extremely difficult to implement plans
  - In pure form a behaviour-based robot has no memory (not even an internal state memory) and so is unable to follow an externally specified sequences of actions
- It can be very hard to predict how a large number of multiple behaviours may interact
  - Emergent behaviour is the term given to unexpected behaviour that comes about through these interactions.
- The robot can get trapped in a limit cycle
  - Trapped in a dead-end, repeatedly turning left then right

Other Approaches

Other Reactive Approaches

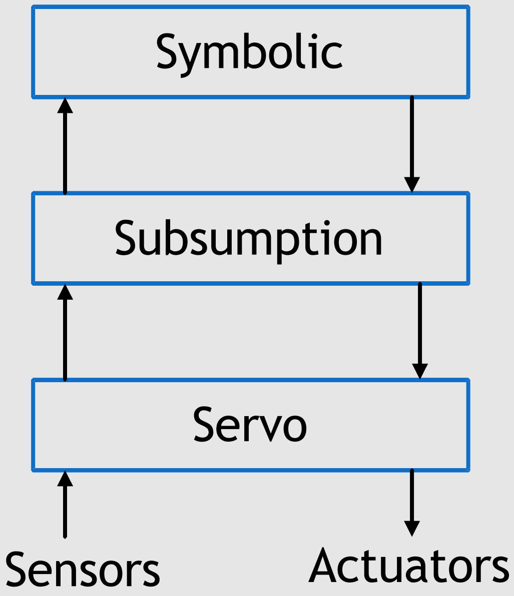
- Two other reactive approaches that are popular
- Potential field methods
  - A potential field is a concept from physics
    - E.g. the gravitational field
      - You do not need to be told which way to fall
      - Planets do not need to plan how to move around the sun
    - Obstacles exert hypothetical repulsive forces on the robot
- Motor schema navigation
  - Multiple, concurrent schema generate separate behaviours which are summed to produce output
    - Schema are dynamically created/destroyed as needed

Hybrid Approaches

- No single approach is perfect for every application
  - Model based is not very flexible or reliable, and requires lots of computing power
  - Reactive approaches are not very efficient for carrying out specific tasks or plans
- It is possible to combine the advantages of different approaches in a single system

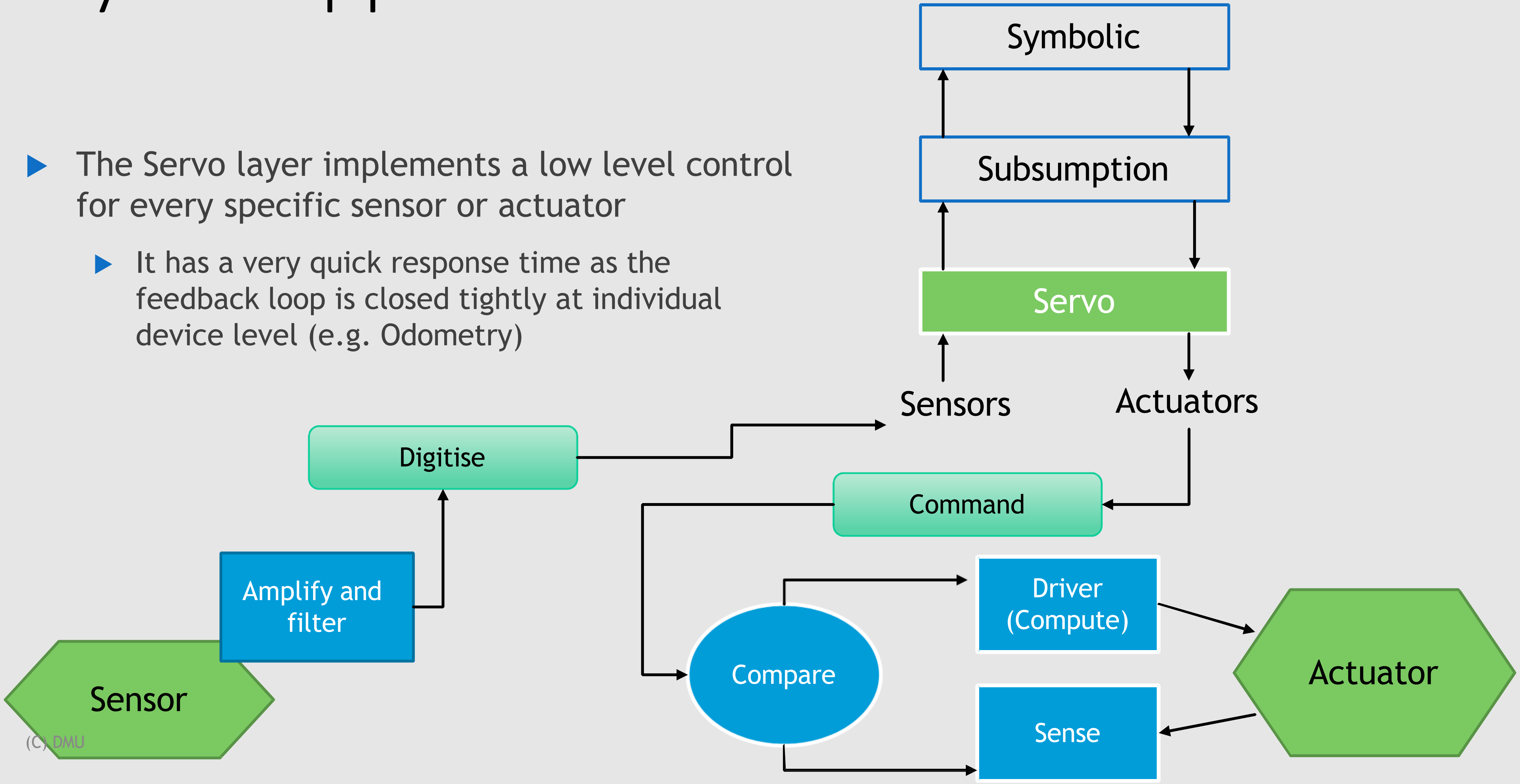
Hybrid Approaches

- The SSS three-layer architecture
  - The servo-subsumption-symbolic architecture combines a reactive architecture with a lower-level servo control layer and a higher-level symbolic system [Connell]



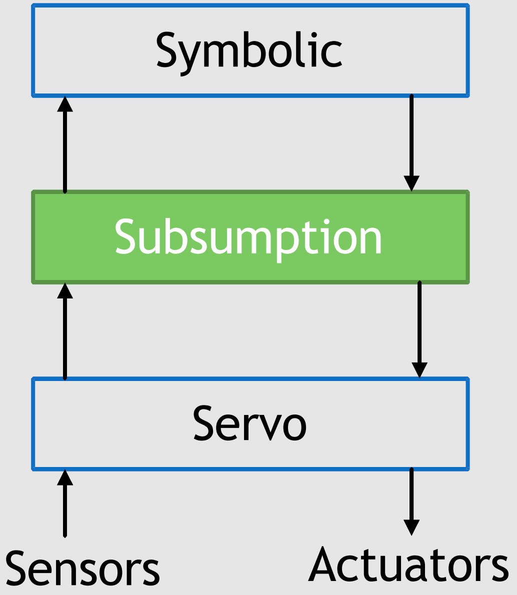
Hybrid Approaches

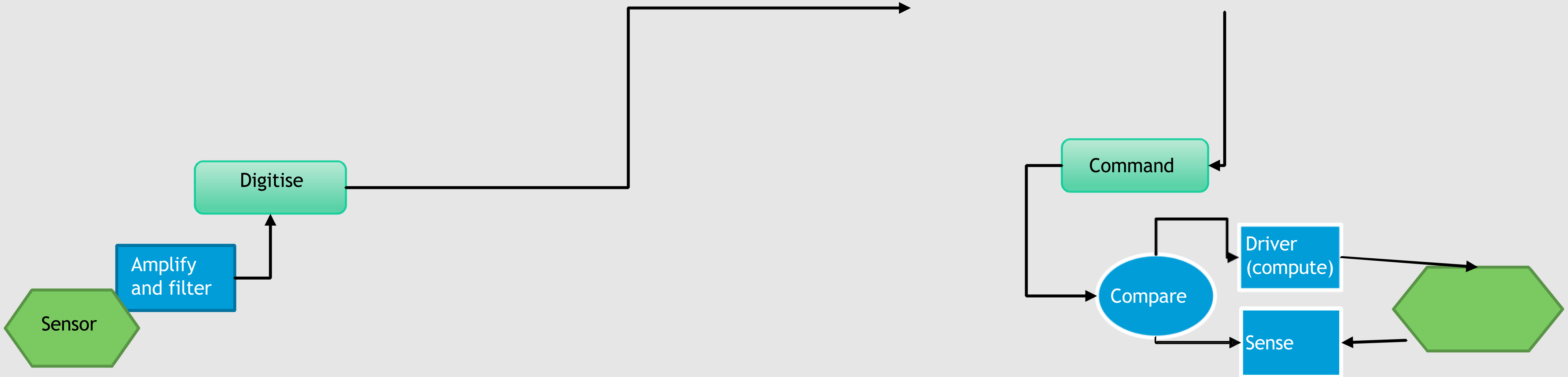
- The Servo layer implements a low level control for every specific sensor or actuator
  - It has a very quick response time as the feedback loop is closed tightly at individual device level (e.g. Odometry)



Hybrid Approaches

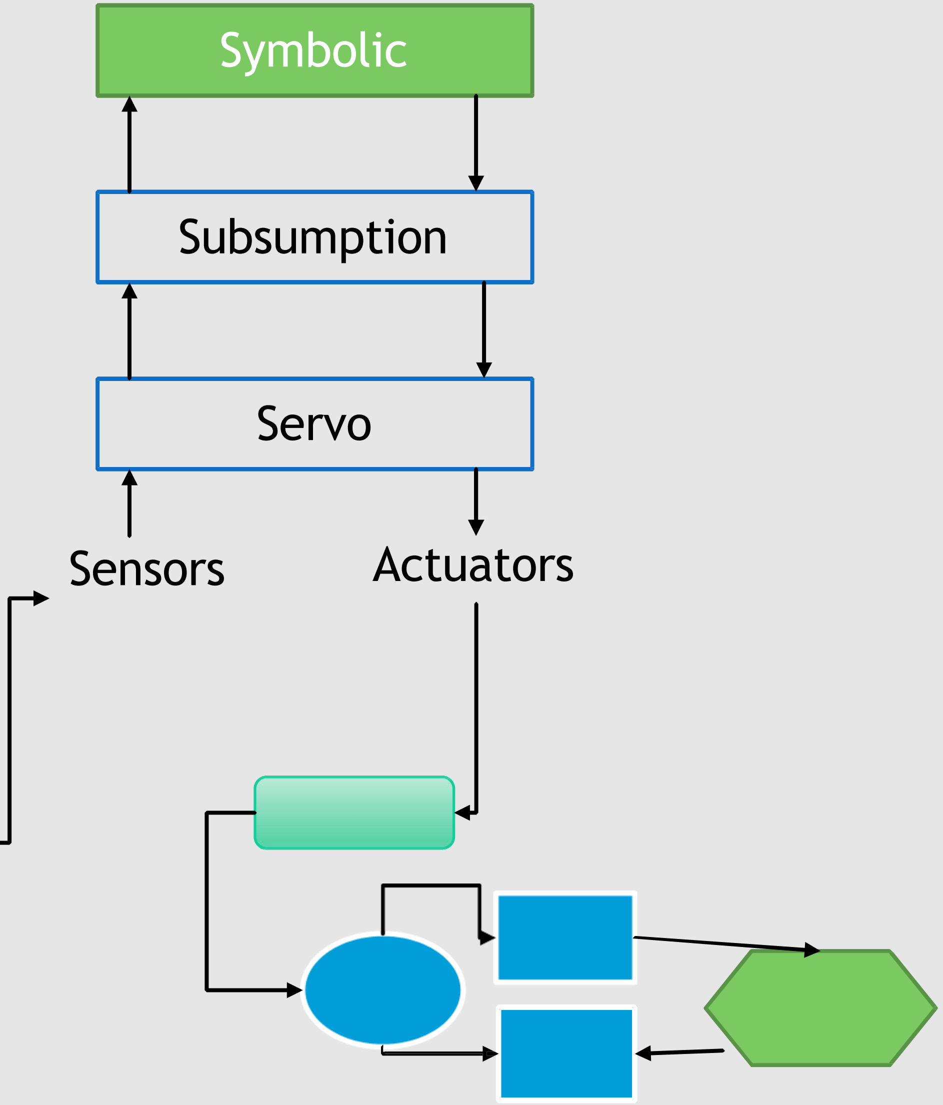
- The Subsumption layer implements a medium level reactive controller for the basic behaviour of the robot
  - It has a quick response time to the information from the sensors and unexpected situations
  - And robust use of actuators

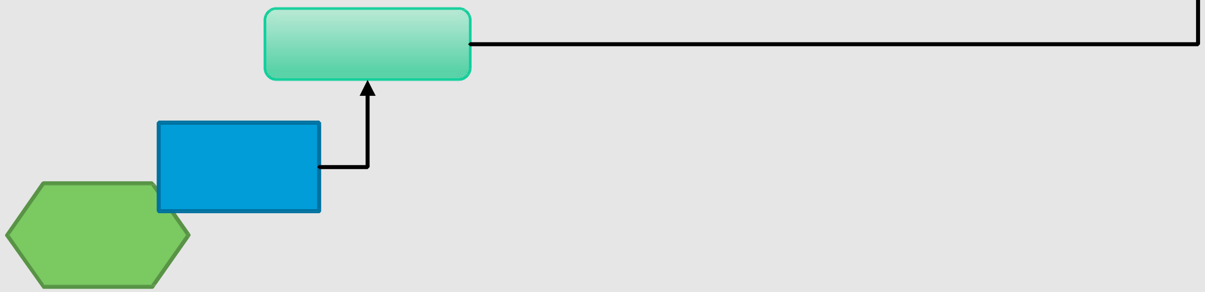




Hybrid Approaches

- The Symbolic layer uses a high-level model to control the mission and planning tasks of the robot
  - It has an overall view of the system, memory, sequence, and complex decision-making capability
  - It has time to execute as the other layers take care of immediate actions





Learning Approaches

- Traditional Learning Techniques
  - Rather than attempt to predefine and predict a symbolic model of the 'real-world', the robot learns how to operate and how to behave by:
    - Supervised Learning
      - Desired output is known for each set of input settings
    - Reinforcement Learning
      - Learning by trial and error through performance feedback
- Evolutionary Algorithms (EAs)
  - Using EAs to find good models or controllers
  - Evolving real solutions in reasonable time requires lots of computing power (not always available in a mobile robot)

Practicalities

Practicalities

- For the control cycle to be implemented the robot requires some information
- Internal state information is very important for the controller to work properly. This is information about the robot itself and can include:
  - Motors’ Speed (including power, temperature…)
  - Position of actuators/effectors
  - Energy Reserve
  - Time

Practicalities

- External state information provides additional feedback for the controller to adjust, e.g.
  - Absolute position (x, y, heading)
  - Obstacles and objects (mapping – partial or complete)
  - Information about waypoints/landmarks/beacons
  - Information about goals (e.g. plans as part of the programming or other external input/communications)
  - Distance to obstacles
  - Other sensor inputs (e.g. color sensor, bumpers, etc…)

Practicalities

- Programming the control cycle
- Your program needs to have a clear structure
  - Gather sensor readings
  - Update the internal state and absolute position if available
  - Decide on the control actions using the chosen methodology
  - Output the commands to the actuators
  - Wait for the action to occur (as required) and “close the loop”

Practicalities

- Thinking about autonomous cars (e.g. Google’s “Waymo” car project)
- Discuss in the discussion board:
  - Which aspects would be appropriate for model based control?
  - Which aspects would require reactive behaviours?
  - What type of sensors would be required to avoid pedestrians or cyclists?

Summary

- Control Models:
  - The sense-think-act control cycle
  - Model-based controllers
- Reactive Robotics
  - Behaviour-based controllers
- Other Approaches
  - Other reactive controllers, hybrid controllers
  - Learning robots
- Practicalities

Questions?