UNIVERSITY OF GOTHENBURG

Hands-on
Robot Operating System (ROS)
programming robots and language technology

Aram Karimi

LT2318 H21 Artificial Intelligence: Cognitive Systems

November 22th, 2020

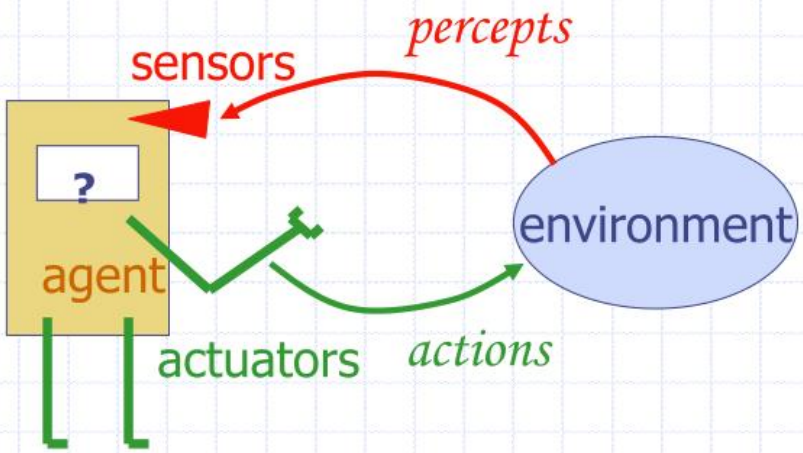
CLASP centre for linguistic theory and studies in probability

UNIVERSITY OF GOTHENBURG

Outline

- Intelligent Agent
- Agents and environments
- Robot Operating System(ROS)
  - A “Meta” Operating System.
  - Nodes
  - Message passing
  - Multiple language support
- Hands-on Tutorial
  - Ros Installation
  - Connect to kinect
  - Keras in ROS

Intelligent Agent



- Definition: An **intelligent agent** perceives its **environment** via **sensors** and acts rationally upon that environment with its **actuators**.

Characteristics of intelligent agents:

- They have some level of autonomy that allows them to perform certain tasks on their own.
- They have a learning ability that enables them to learn even as tasks are carried out.
- They can interact with other entities such as agents, humans, and systems.
- New rules can be accommodated by intelligent agents incrementally.
- They exhibit goal-oriented habits.
- They are knowledge-based. They use knowledge regarding communications, processes, and entities.

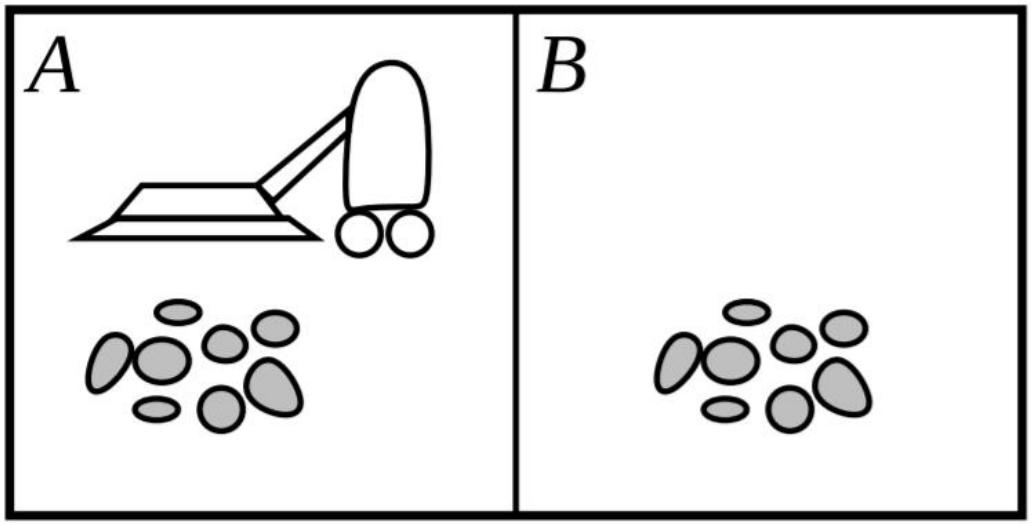
Agents include humans, robots, softbots, thermostats, etc.
The agent function maps from percept histories to actions:

$$f : P^* \rightarrow A. \tag{1}$$

The agent program runs on the physical architecture to produce f
How do we know if this is a good agent function?

- What is the best function?
- Is there one?
- Who decides this?

Vacuum-cleaner world



Percepts: location and contents, e.g., [*A*, *Dirty*]
Actions: *Left*, *Right*, *Suck*, *NoOp*
A simple agent function is:
If the current square is dirty, then suck;
otherwise, move to the other square.

Rationality: **A rational agent chooses any action that**

- maximizes the expected value of the performance measure
- given the percept sequence to date

Rational $\neq$ *omniscient*

- percepts may not supply all relevant information

Rational $\neq$ *clairvoyant*

- action outcomes may not be as expected

Rational $\neq$ *successful*

- *Rational* $\Longrightarrow$ *exploration*, *learning*, *autonomy*

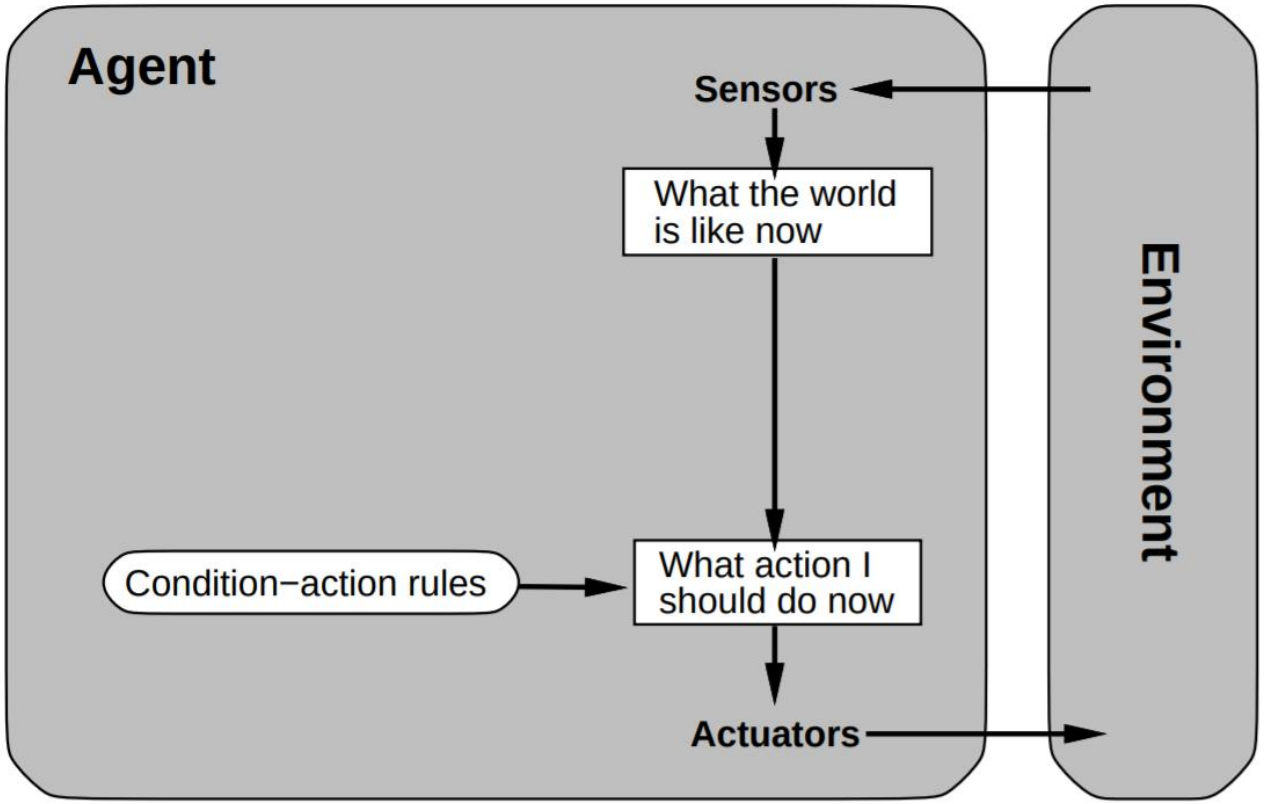
Agent types

Four basic types in order of increasing generality:

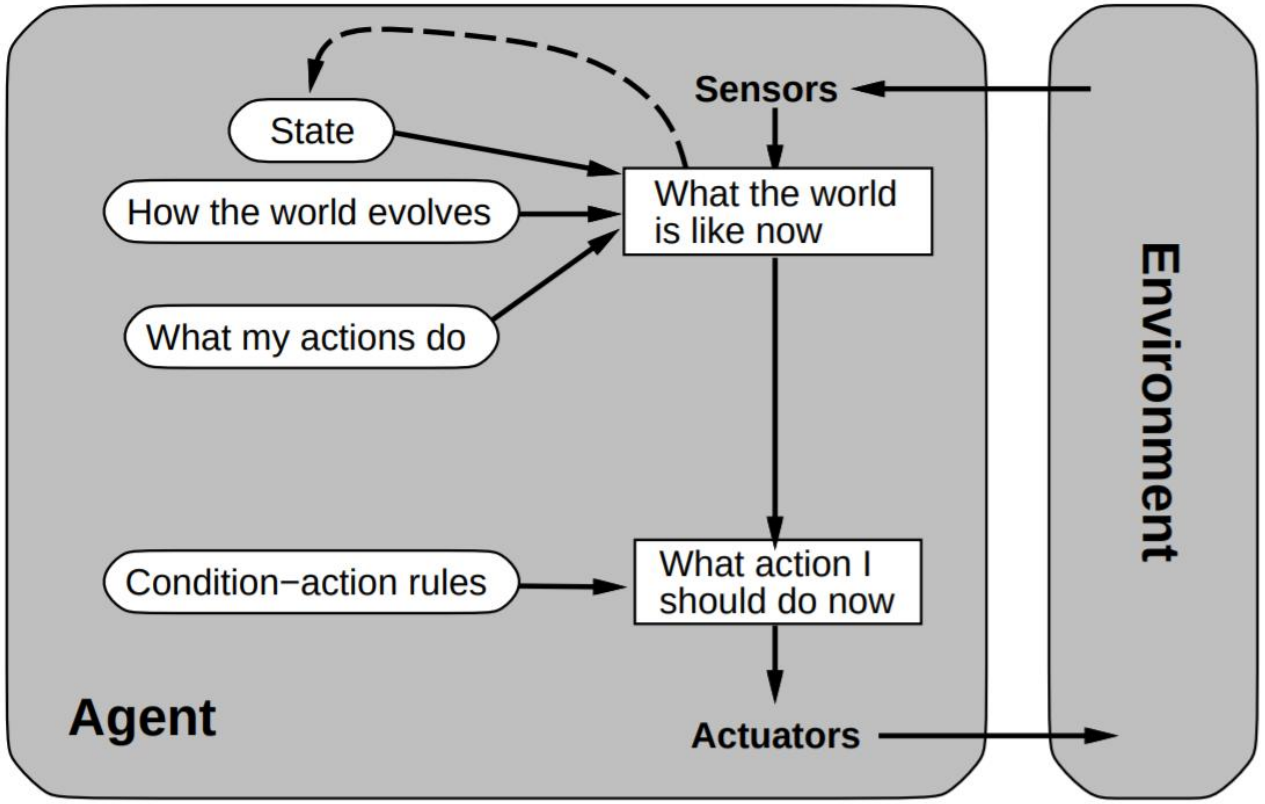
- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents

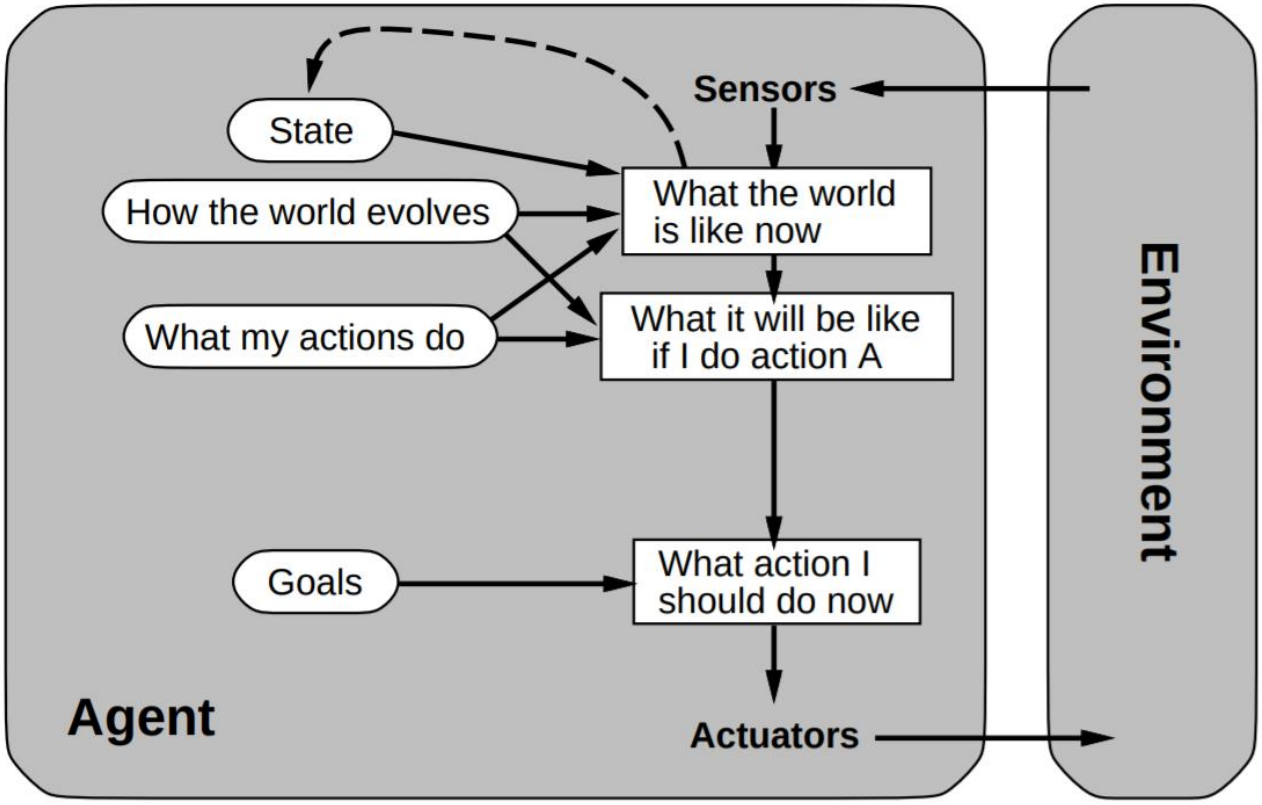
Simple reflex agents



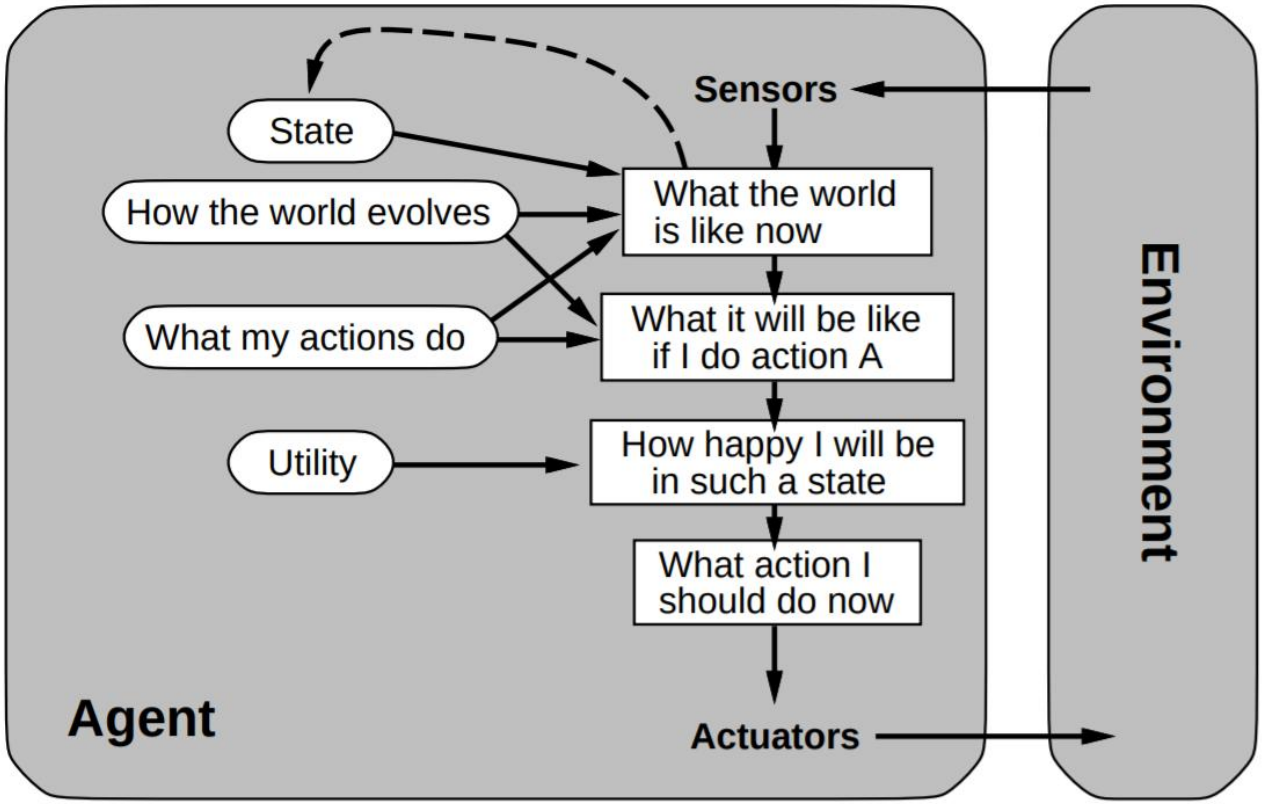
Reflex agents with state



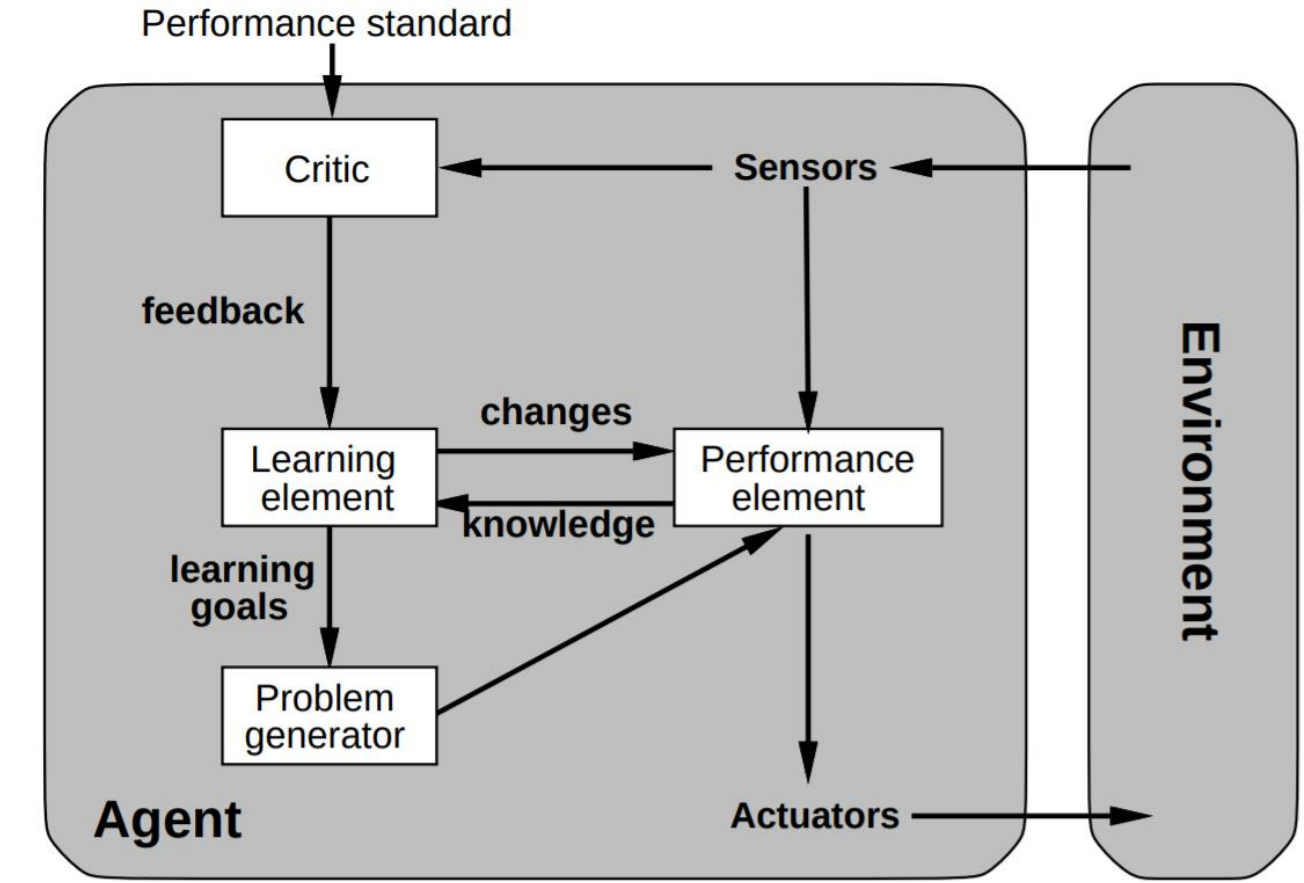
Goal-based agents

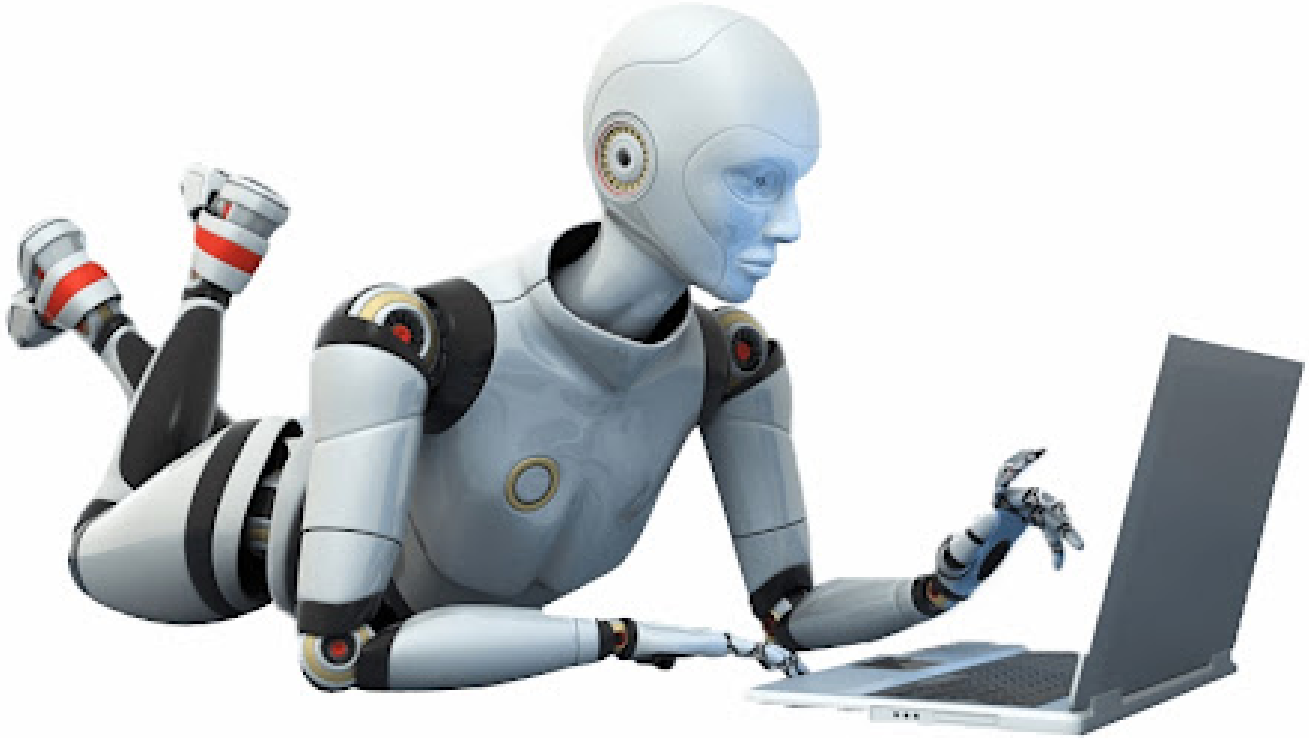


Utility-based agents



Learning agents





What is ROS(Robotic Operating System)?

- It is not a Operating System (OS)
- It is not an Application Programming Interface (API)
- It is not a simple framework

ROS is a middleware for robotic programming, specifically designed for complex applications

- ROS is a platform for robot software.
- Hardware abstraction
- Message passing between nodes
- Sophisticated build environment
- Advance open-source robotics
- Debugging and Visualization Tools

UNIVERSITY OF GOTHENBURG

ROS as a framework

The ROS framework is component oriented.
Each component is called a node. Nodes communicate using topics or services.

- topics represents data-flow. For instance: camera images, robot configuration or position can be model as topics. Topics values are often published regularly to keep the whole system up-to-date.
- services represents queries which are sent asynchroneously, and usually at a low frame-rate. Slow components such as motion planning nodes for instance usually provide services.

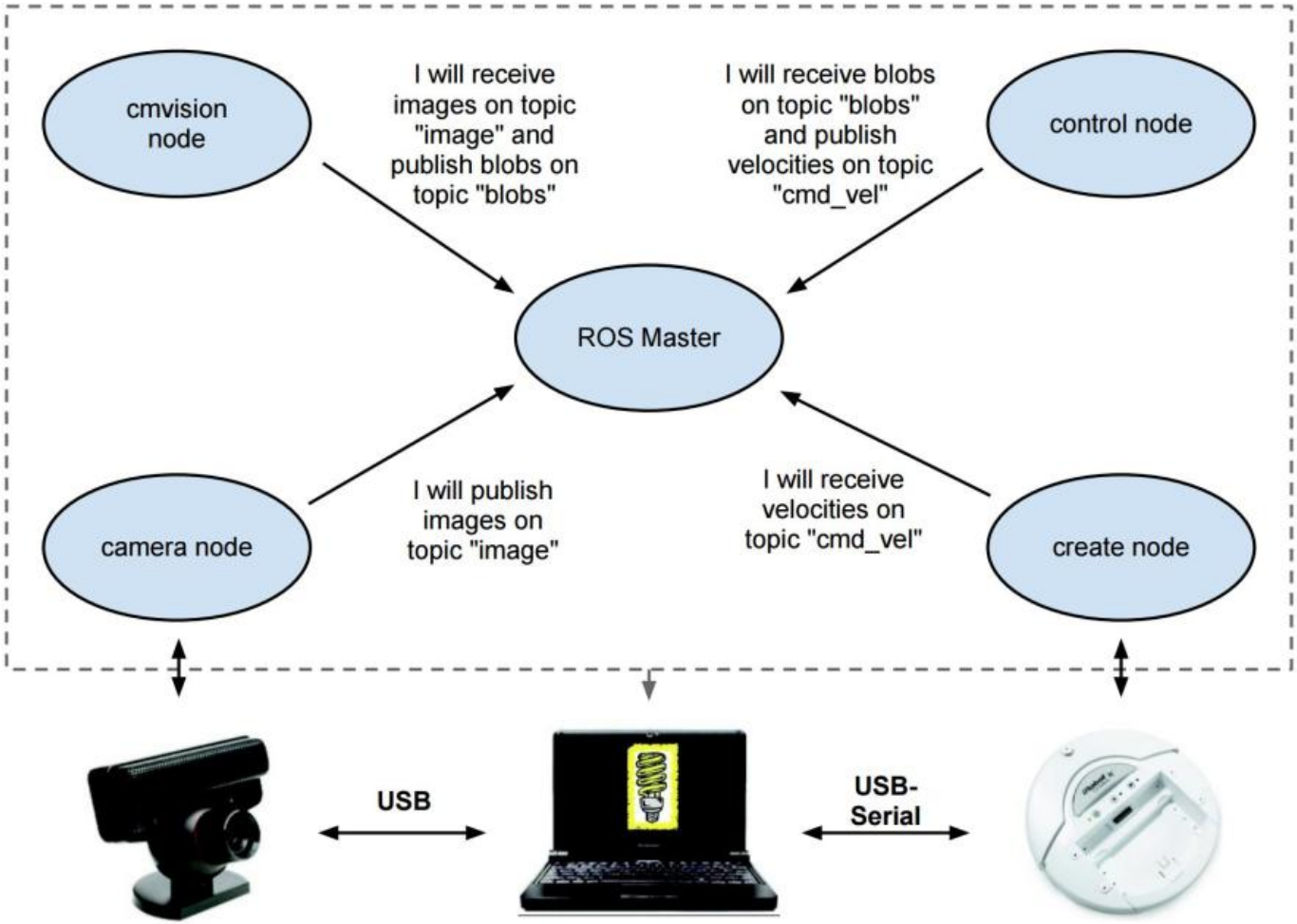
The ROS framework is a graph

- Each node can listen or publish on a topic
- Each node can call or provide one or more services.
- Each node can also set or get one or more parameters. Some are public and can be modified by other nodes. The other are private and are defined at startup only.

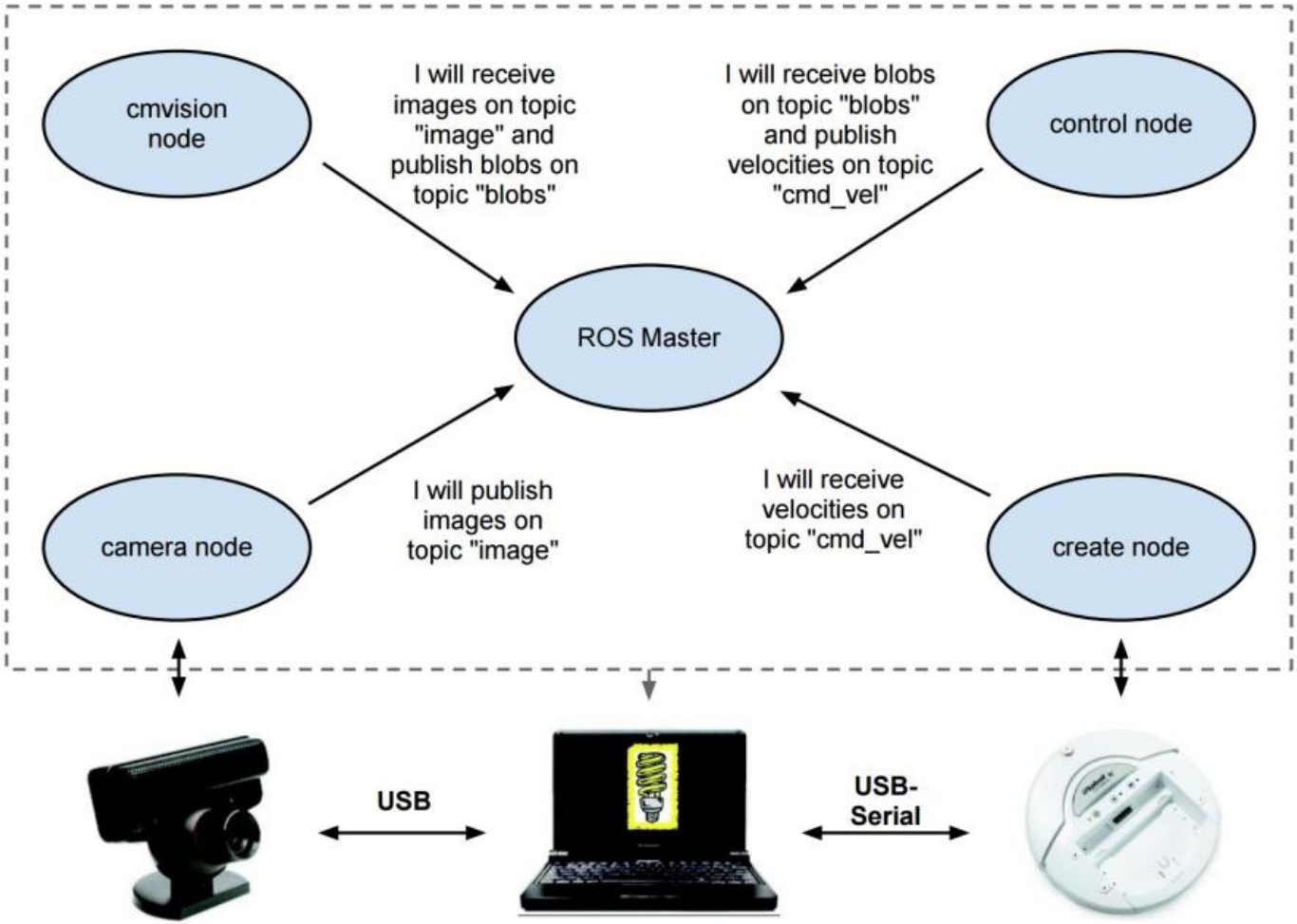
Packages

- ROS software is organized into packages
- Each package contains some combination of code, data, and documentation

ROS Architecture: Basics



ROS Architecture: Basics



UNIVERSITY OF GOTHENBURG

Why ROS?

- Avoid complexity of a big software.
- Abstraction for specific robot hardware.
- Sequential programming on asynchronous environment.
- Setting up the Robot takes too much time
- Programs can be run on multiple computers and communicate over the network.
- Multi-language support
- Free and open-source

UNIVERSITY OF GOTHENBURG

Configuring the ROS environment

- You need Ubuntu (Install Virtualbox if you don't have Ubuntu)
- Programming knowledge (Python, C++)
- Robot simulation / any robot compatible with ROS:
  - http://wiki.ros.org/Robots
- Robotics by nature is multi-disciplinary: Some knowledge from other fields is required.
- Use open-source projects

Live Coding!

UNIVERSITY OF GOTHENBURG

References