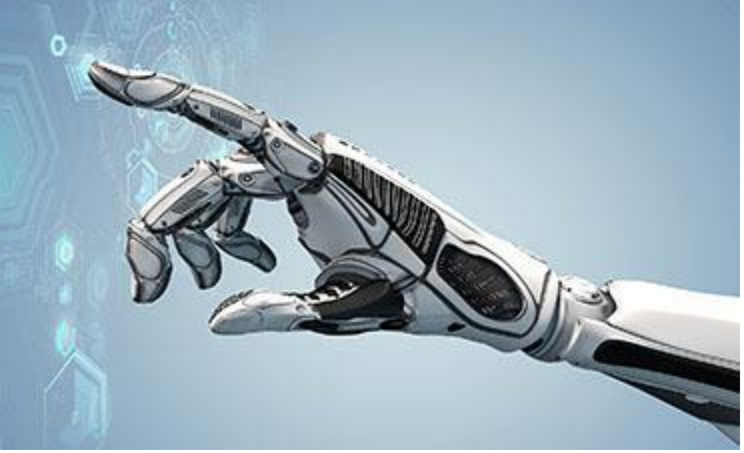


CSE444: Introduction to Robotics

Lesson 6: Programming and Control

ALL Follows

Summer 2019

Discussion Points

- Introduction
- Basic Workflow
- Robot Research Software
- Functional Control Architecture
- Robot Programming using ROS

Introduction

- real-time operating system
- sensory data reading
- motion control execution
- world modeling
- physical/cognitive interaction with the robot
- fault detection
- error recovery to correct operative conditions
- programming language (data structure + instruction set)

programming environments will depend also on the level at which an operator has access to the functional architecture of the robot

Basic Workflow

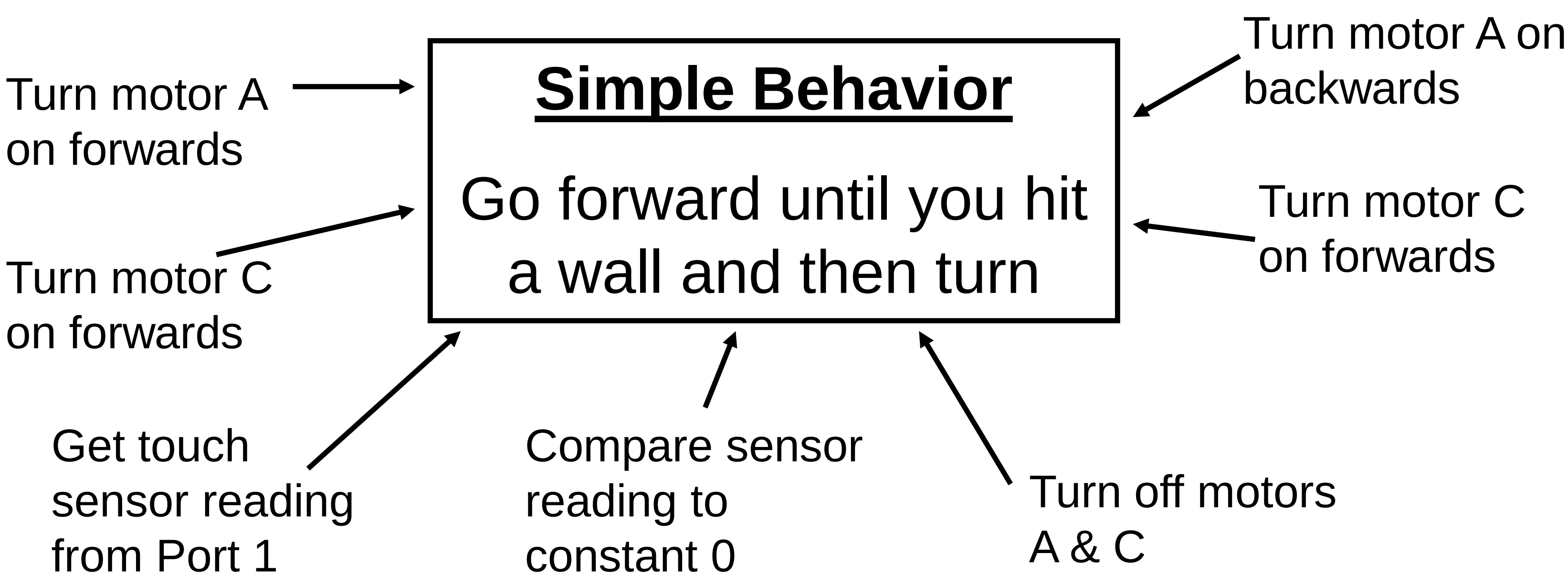
- Programming Behavior
  - Behaviors describe the actions and decisions of your robot
- Individual, bite-size functions that your robot performs directly

<u>Basic Behavior</u>

Turn motor A on forwards

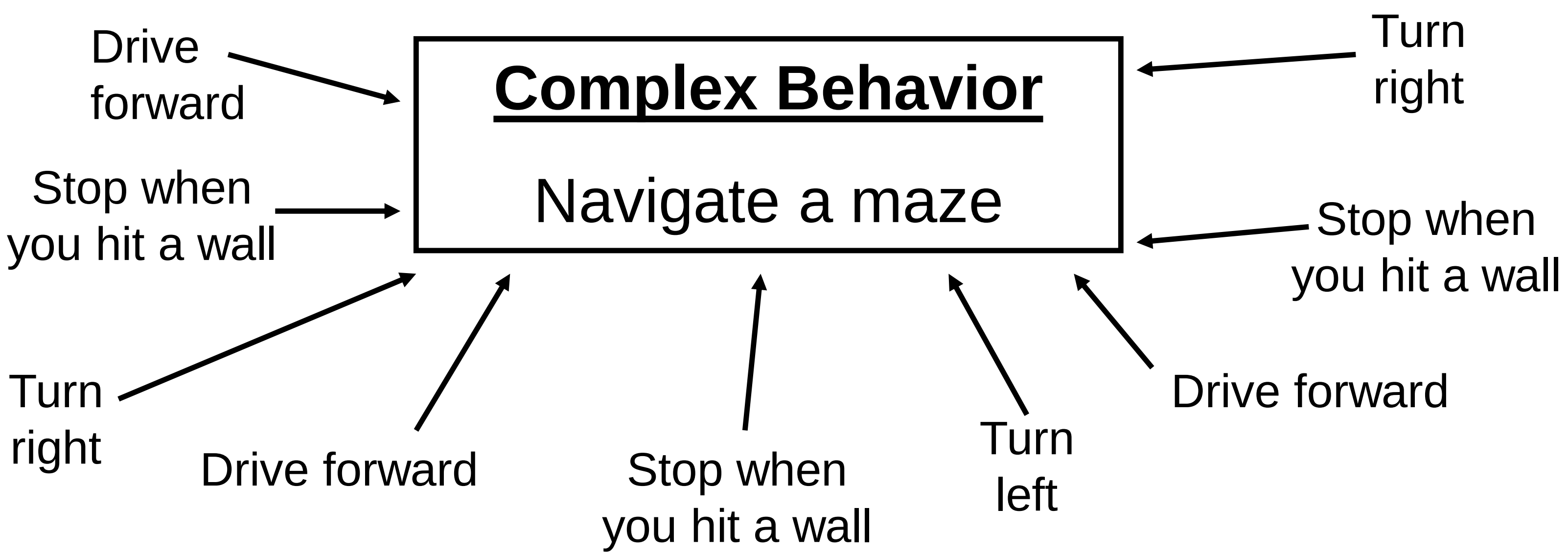
Simple Behaviors

- Built of several basic behaviors
- Let you describe a full action of the robot



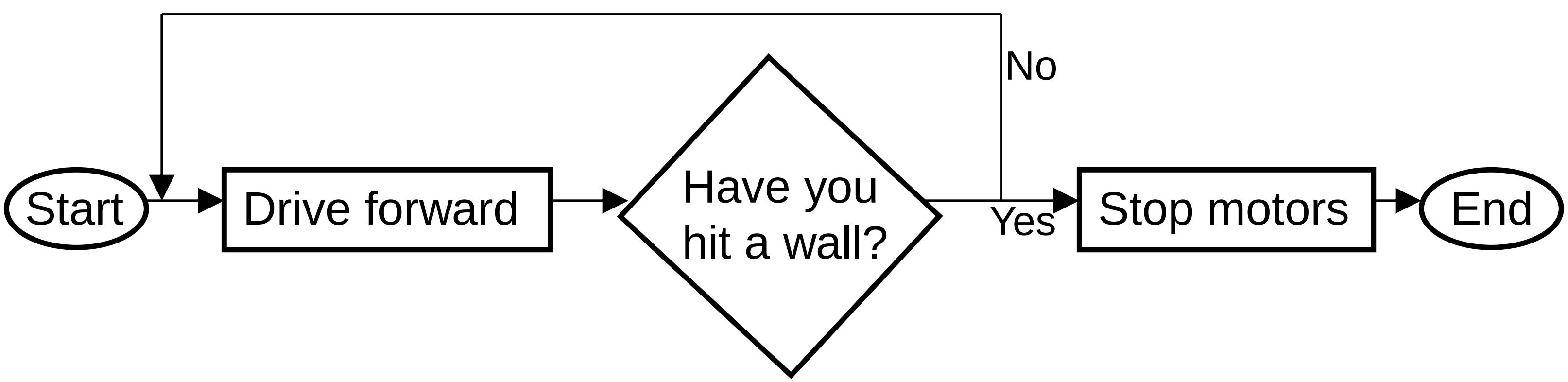
Complex Behaviors

- Describe the full scope of what the robot can do
- Always composed of smaller behaviors, so you can break them down



Flowcharts

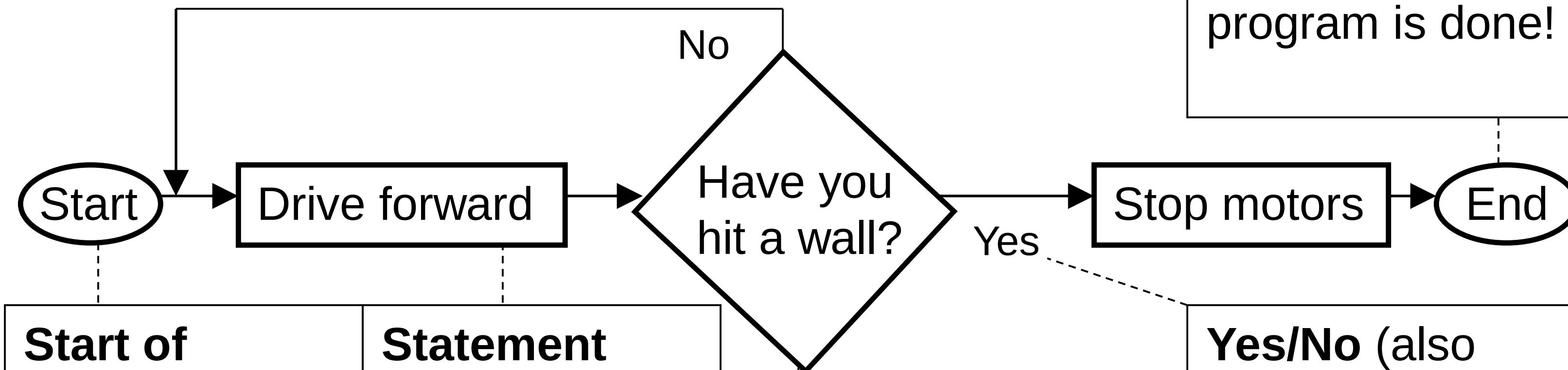
- Visually organizes steps in different shaped bubbles
- Good way to work out steps before you translate them into code



Flowcharts

- Parts of a Flowchart

End of Program - Marks the end of the program. If you reach this point, the program is done!



Start of Program — Marks the beginning of the program. Begin here. Follow the line to get to the next block.

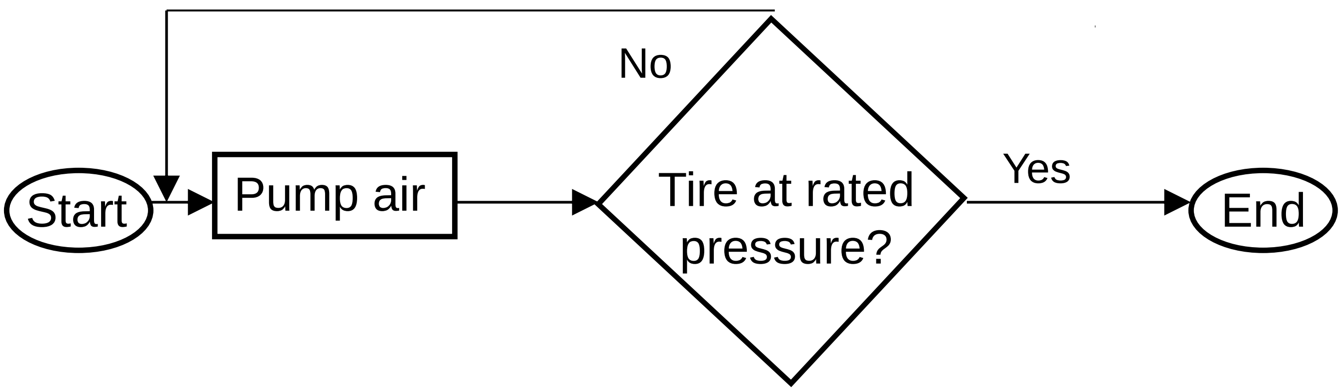
Statement Block — A statement to execute,or a behavior to perform.

Decision Block — A decision point in your program. Ask a simple question, and do different things depending on the answer.

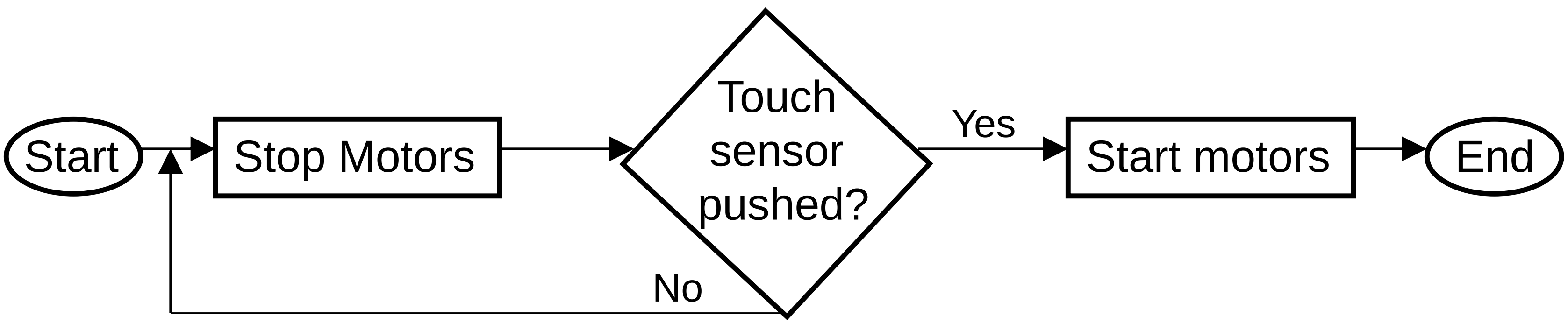
Yes/No (also True/False, etc.) — Answers to the question posed in the decision block. Follow the line labeled with the appropriate answer.

Flowcharts

What does this flowchart describe? *Filling a tire*



What about this one? *Go when touch sensor is pushed*



Robot Research Software

- a (partial) list of open source robot software
  - for simulation and/or real-time control
  - for interfacing with devices and sensors
  - research oriented

Player/Stage playerstage.sourceforge.net

- networked robotics server (running on Linux, Mac OS X) as an abstraction layer supporting a variety of hardware + 2D robot simulation environment
- Gazebo: 3D robot simulator (with ODE physics engine and OpenGL rendering), now an independent project

VREP (edu version) www.coppeliarobotics.com

- each object/model controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution
- controllers written in C/C++, Python, Java, Matlab, ...

Robot Research Software

Robotics Toolbox (free addition to Matlab) www.petercorke.com

- study and simulation of kinematics, dynamics, and trajectory generation for serial-link manipulators

OpenRDK openrdk.sourceforge.net

- "agents": modular processes dynamically activated, with blackboard-type communication (repository)

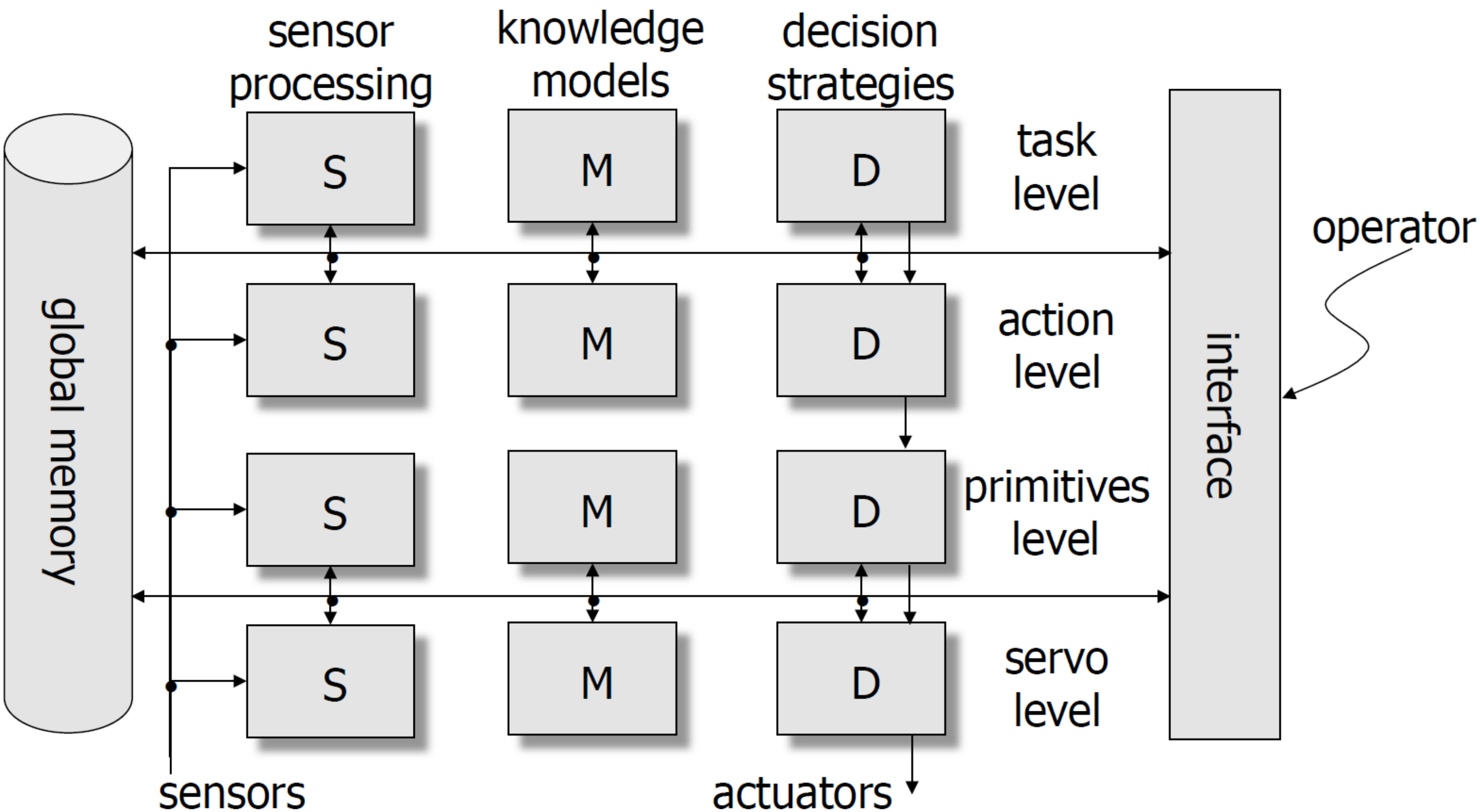
ROS (Robot Operating System) www.ros.org/wiki

- middleware with: hardware abstraction, device drivers, libraries, visualizers, message-passing, package management
- "nodes": executable code (in Python, C++) running with a publish/subscribe communication style

Pyro (Python Robotics) pyrorobotics.org

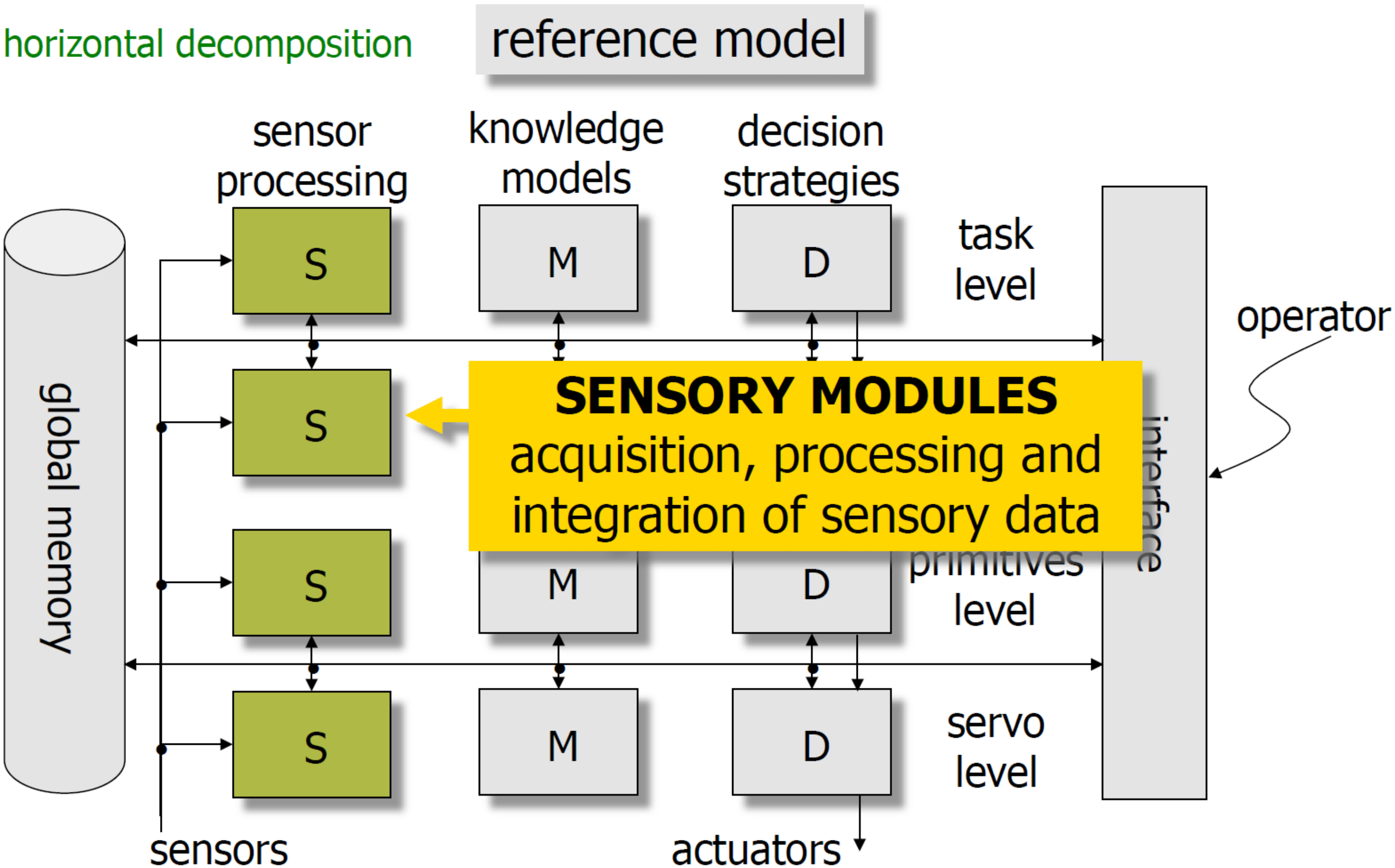
Functional Control Architecture

reference model



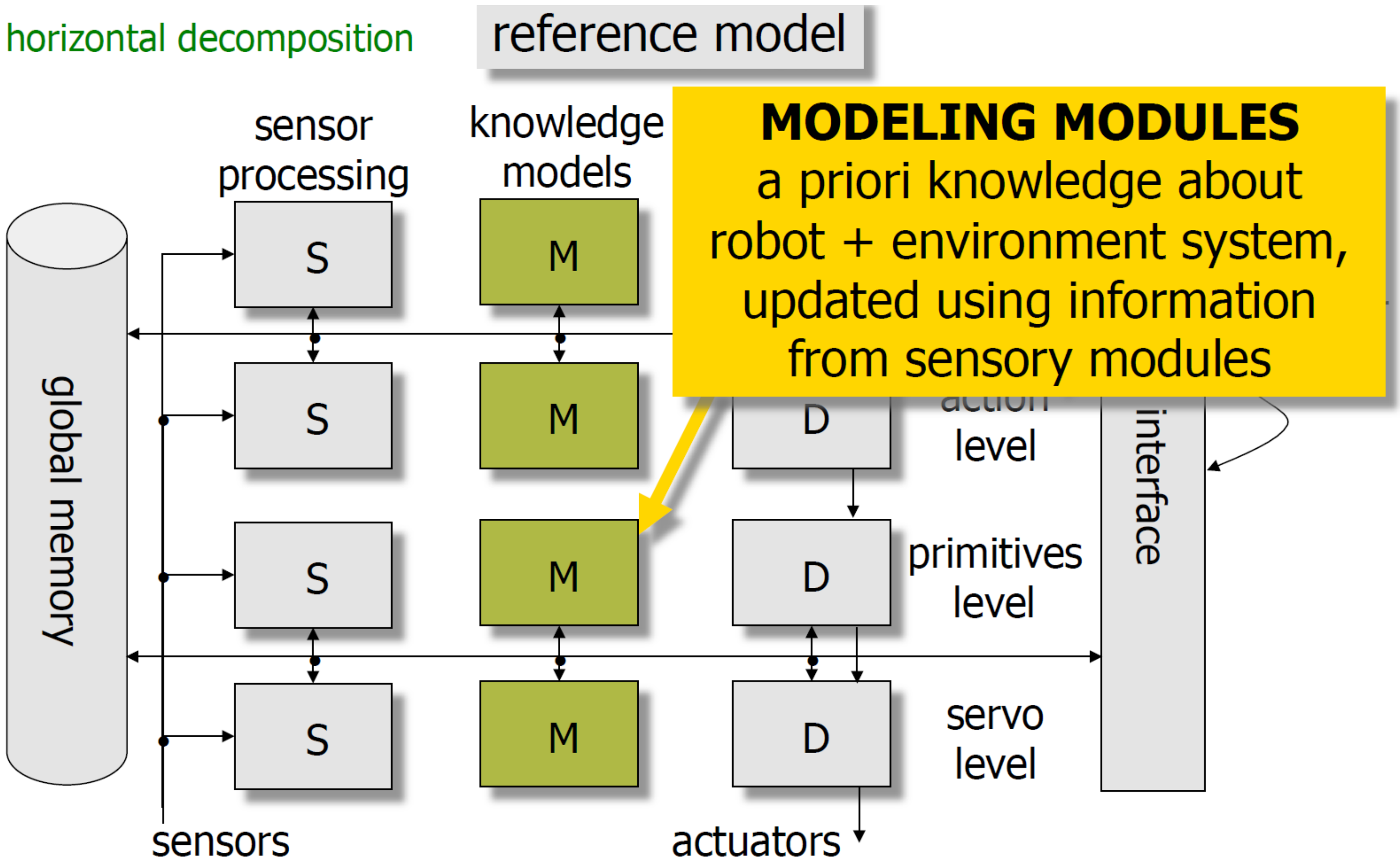
Functional Control Architecture

horizontal decomposition



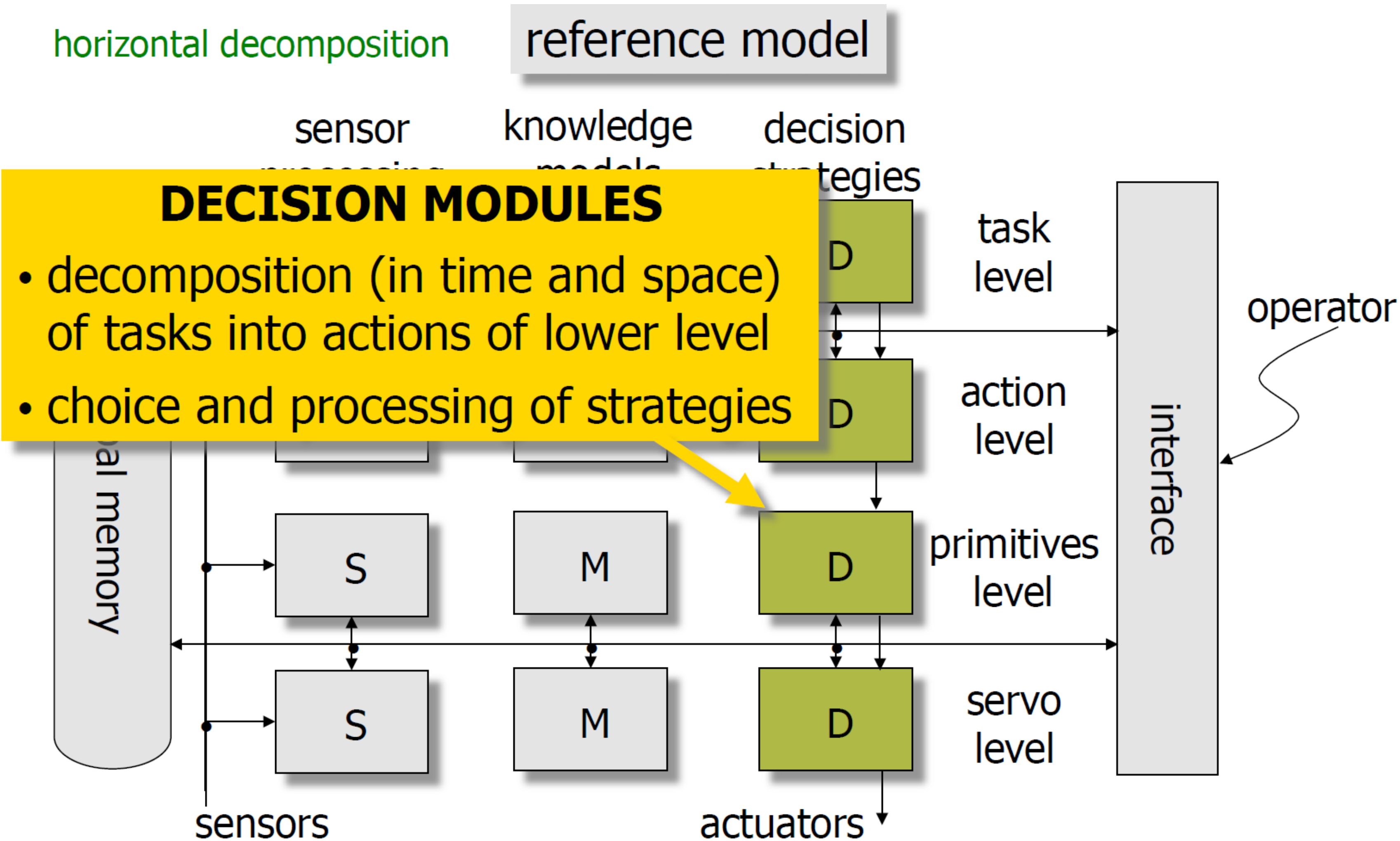
Functional Control Architecture

horizontal decomposition

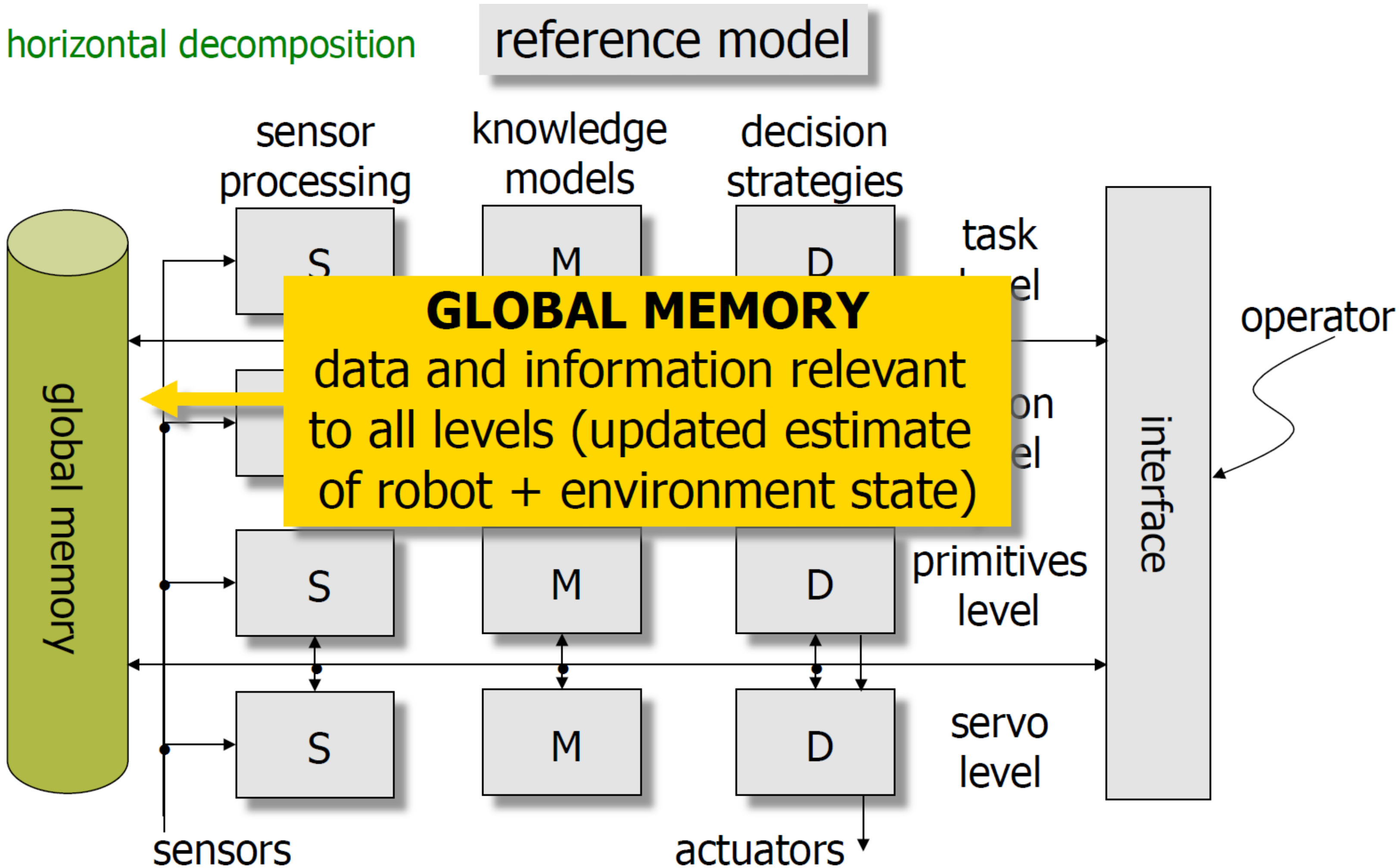


Functional Control Architecture

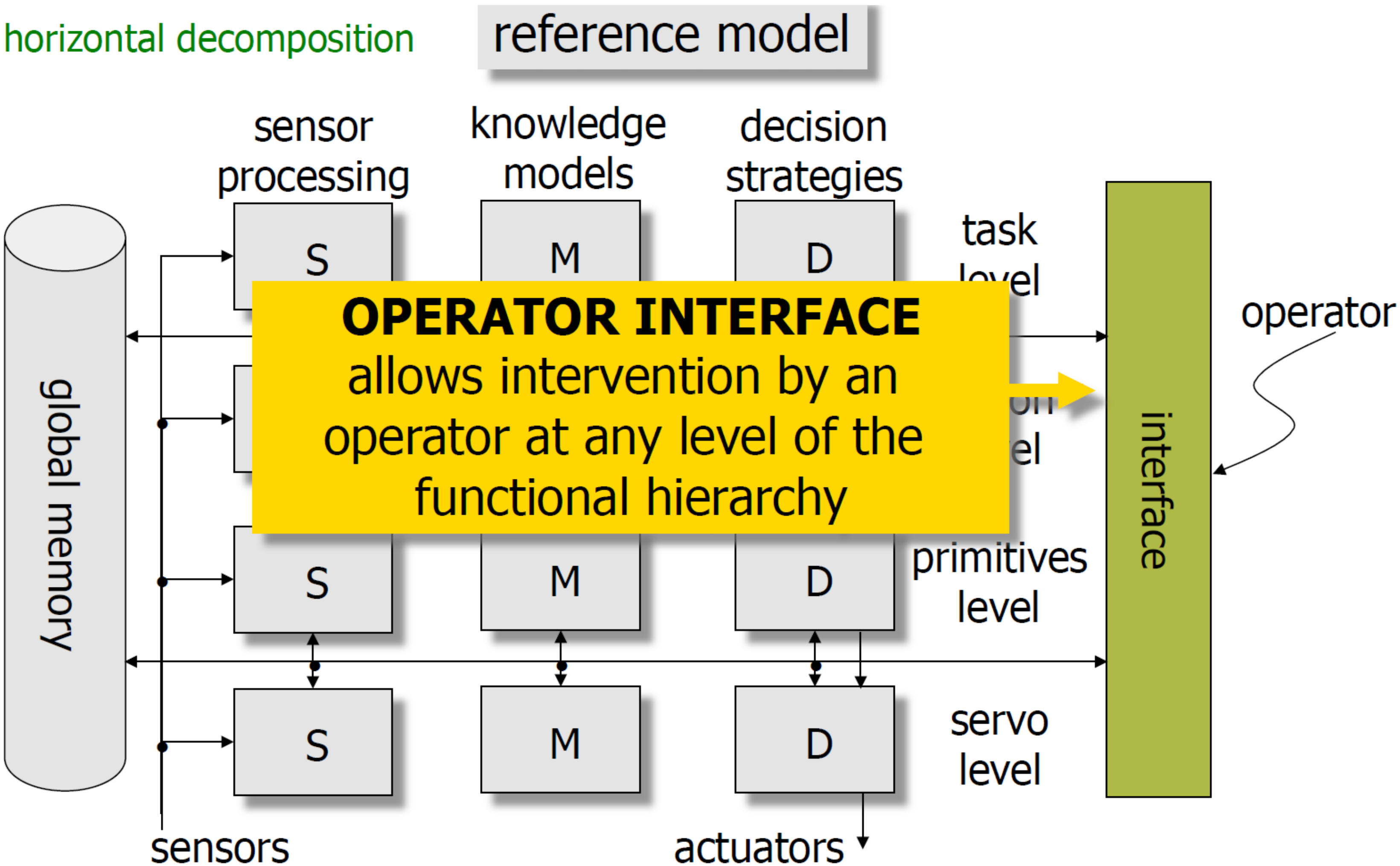
horizontal decomposition



Functional Control Architecture



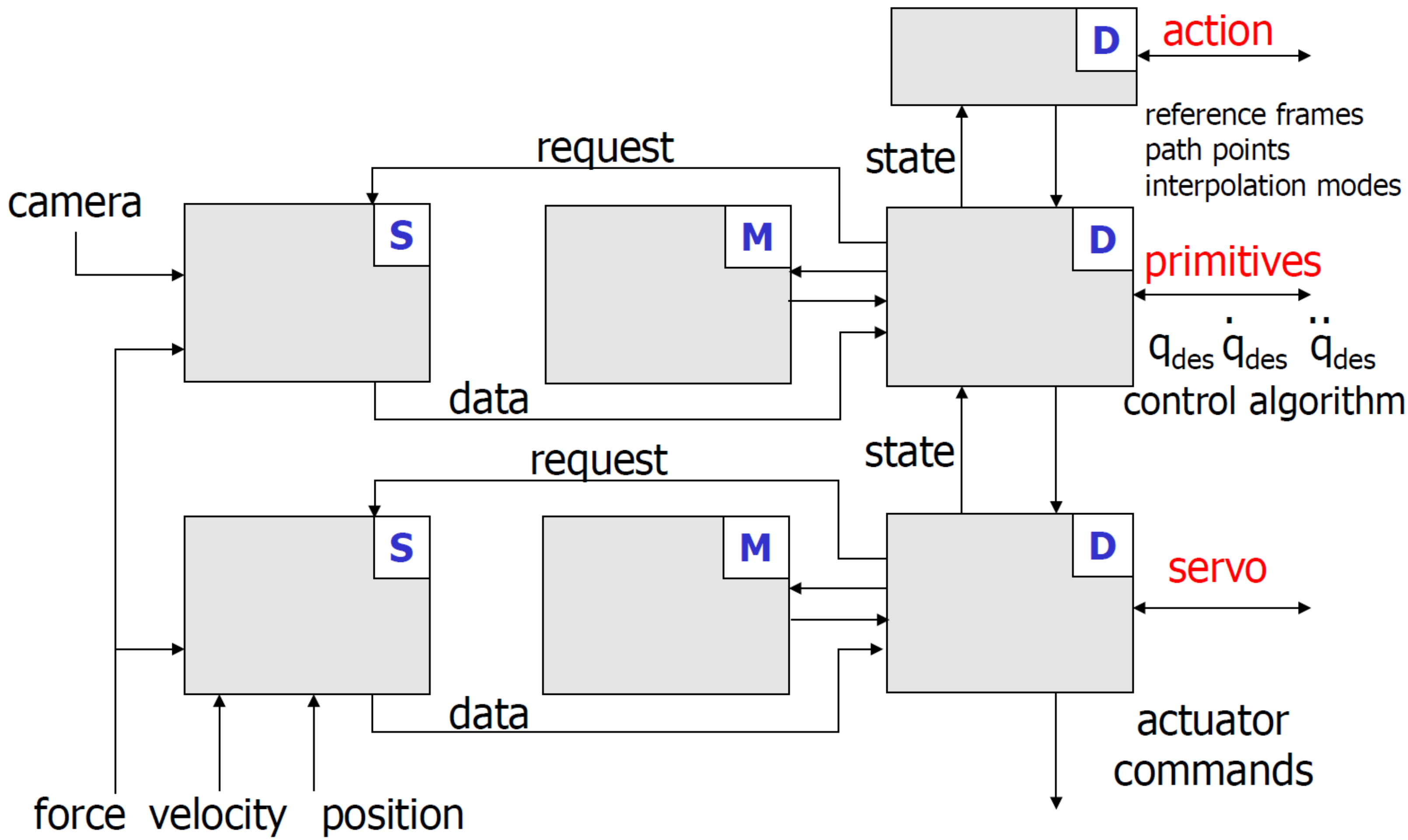
Functional Control Architecture



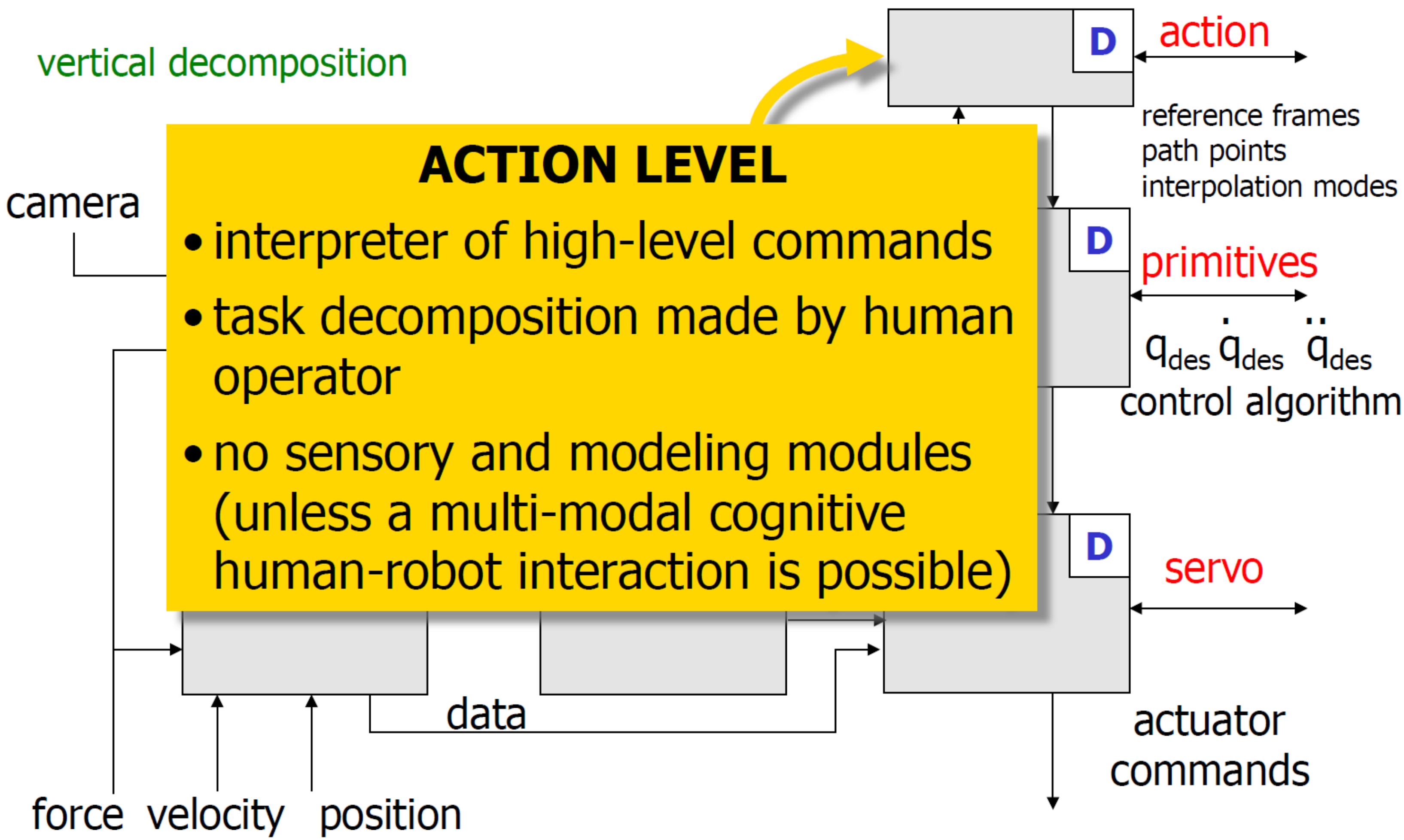
Levels for Reference Model

- task level: objective of the task (as specified by the user) analyzed and decomposed into actions (based on knowledge models about the robot and the environment systems)
- action level: symbolic commands converted into sequences of intermediate configurations
- primitives level: reference trajectories generation for the servo level, choice of a control strategy
- servo level: implementation of control algorithms, real-time computation of driving commands for the actuating servomotors

Industrial Robot: Functional Architecture



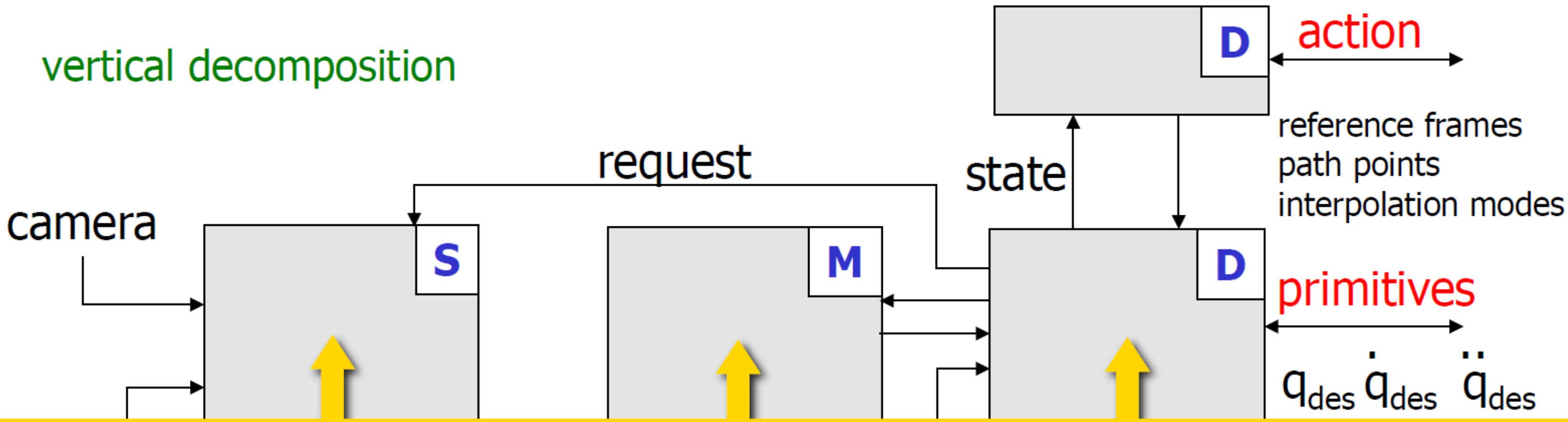
Industrial Robot: Functional Architecture



ACTION LEVEL

- interpreter of high-level commands
- task decomposition made by human operator
- no sensory and modeling modules (unless a multi-modal cognitive human-robot interaction is possible)

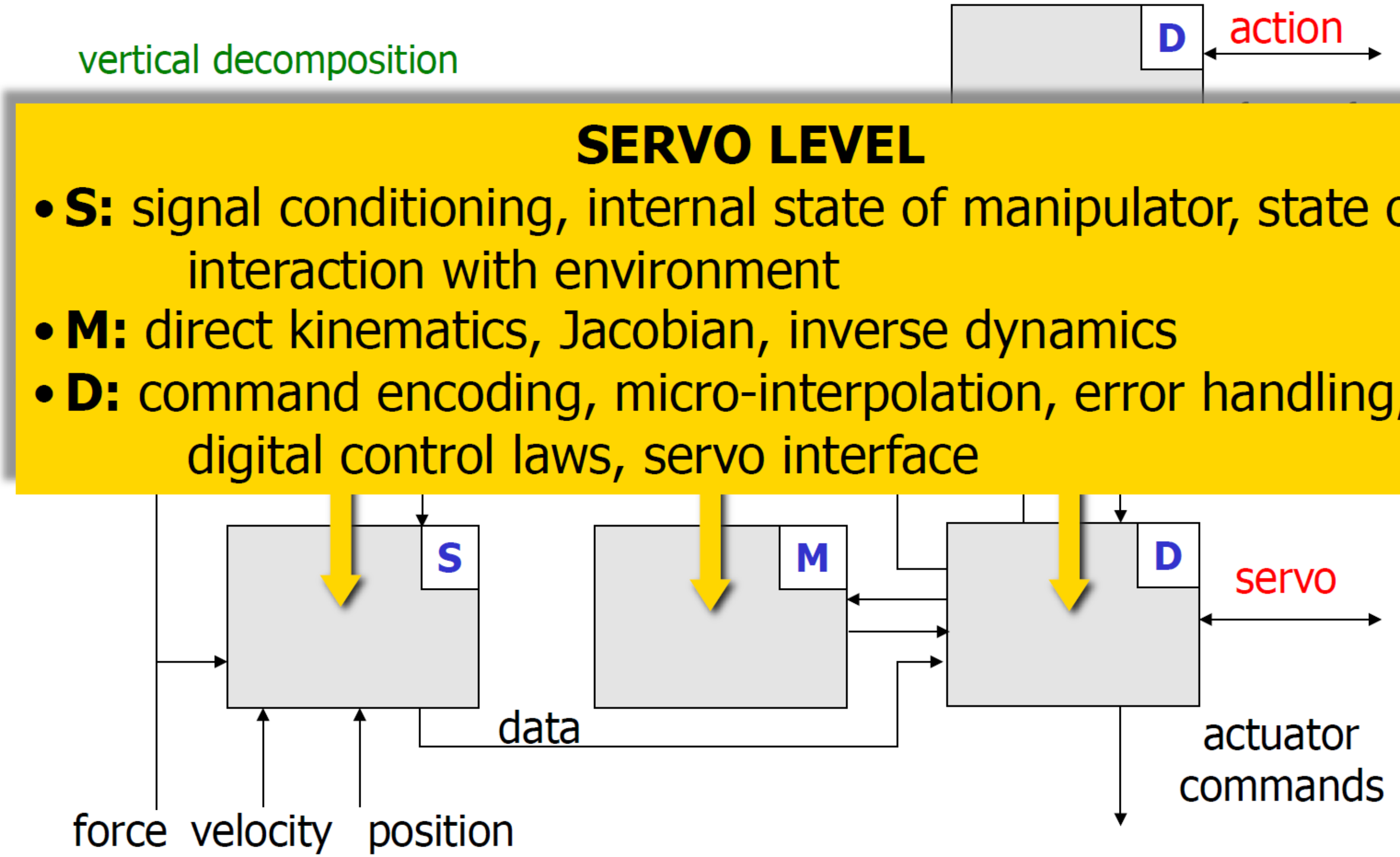
Industrial Robot: Functional Architecture



PRIMITIVES LEVEL

- **S:** (only for an active interaction with the environment) world geometry, interaction state
- **M:** direct and inverse kinematics, dynamic models
- **D:** command encoding, path generation, trajectory interpolation, kinematic inversion, analysis of servo state, emergency handling

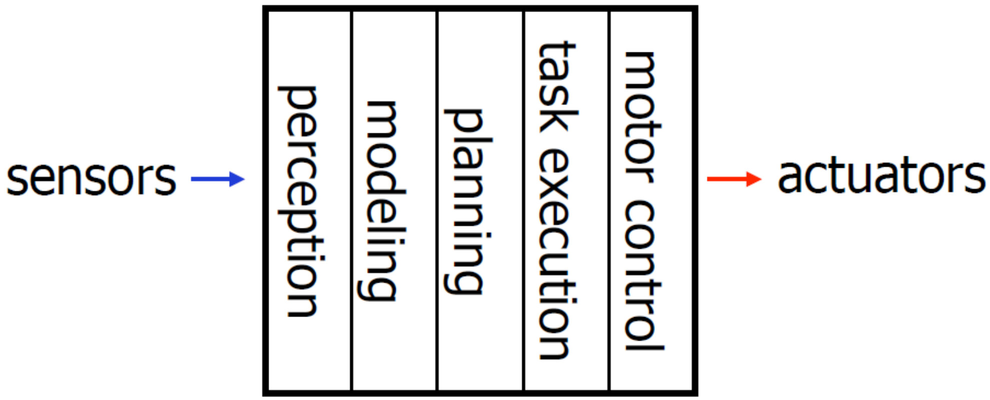
Industrial Robot: Functional Architecture



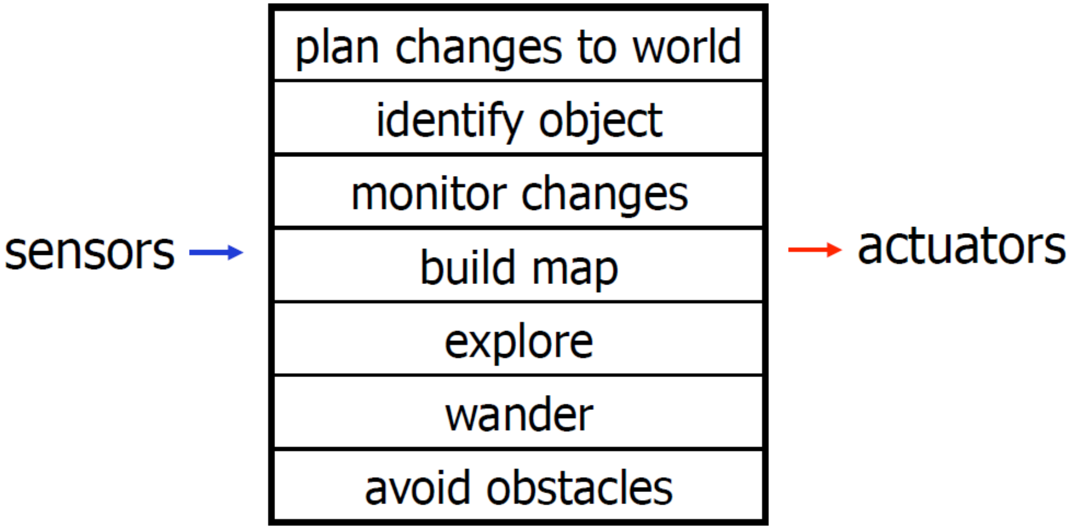
SERVO LEVEL

- **S:** signal conditioning, internal state of manipulator, state of interaction with environment
- **M:** direct kinematics, Jacobian, inverse dynamics
- **D:** command encoding, micro-interpolation, error handling, digital control laws, servo interface

Interactions: Modules



horizontal
activation
(sequential)



vertical
activation on demand
(subsumption)

MoveIt!

ROS industrial

ROS

ROS is an open-source, meta-operating system

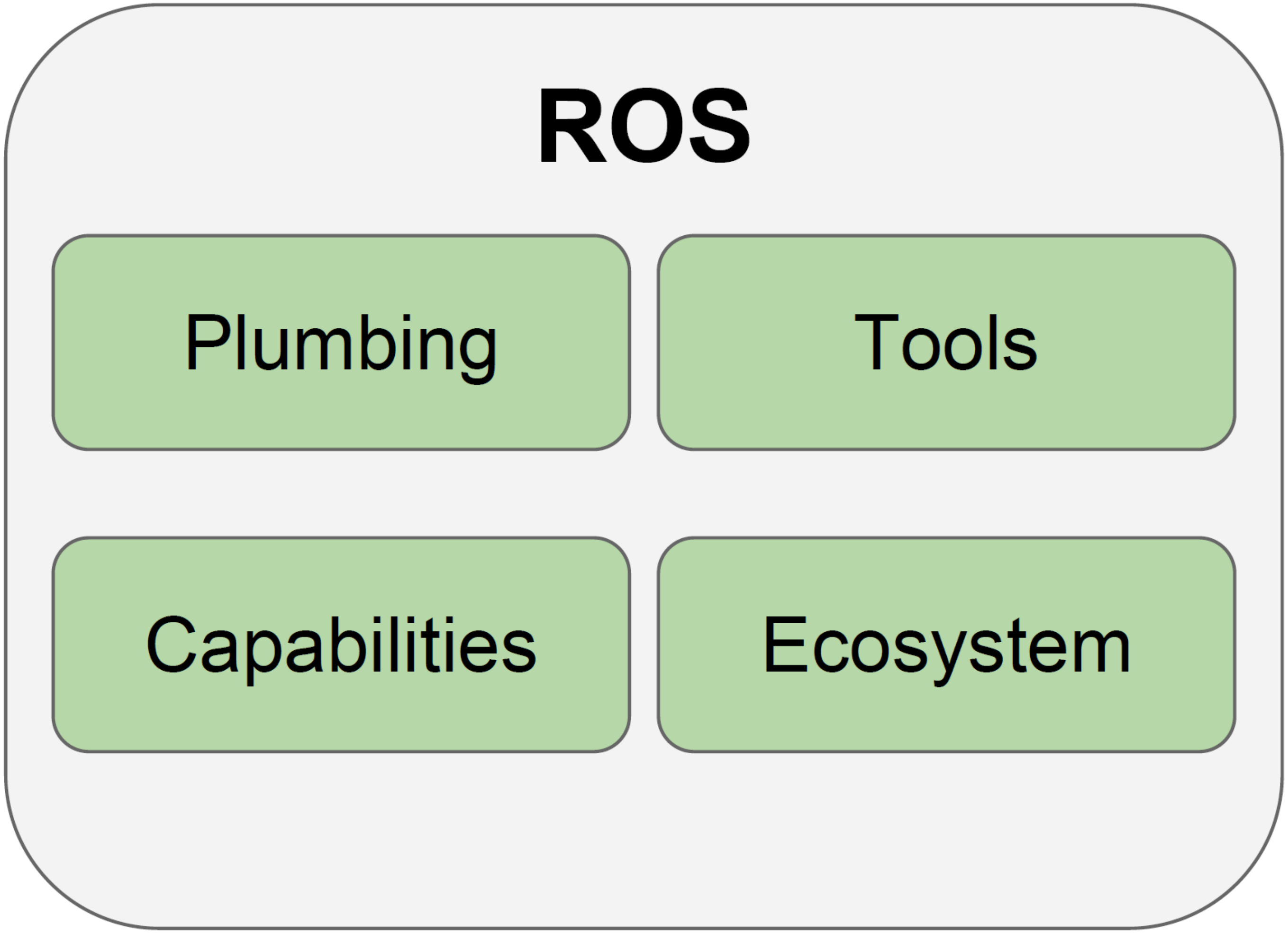
OpenCV

pointcloudlibrary

What is special in ROS?

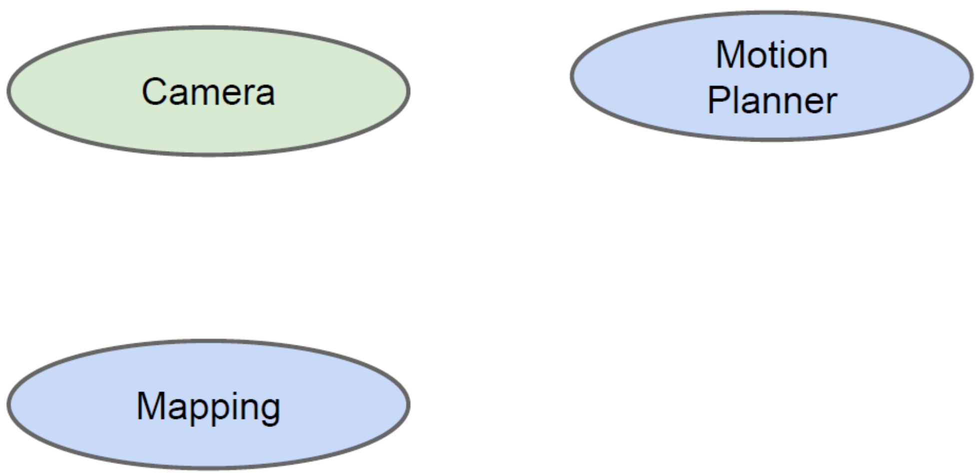
- <u>Reusable robotics components!</u>
- 62 Robotic platforms officially support ROS
  http://wiki.ros.org/Robots
- Modular design
- Hundreds of ready to use algorithms
- Efficient, so it can be used for actual products, not just prototyping
- Runs on Ubuntu, also ARM Processors
- Parallelisation and networking made easy, can use multiple machines simultaneously

ROS Components



Nodes

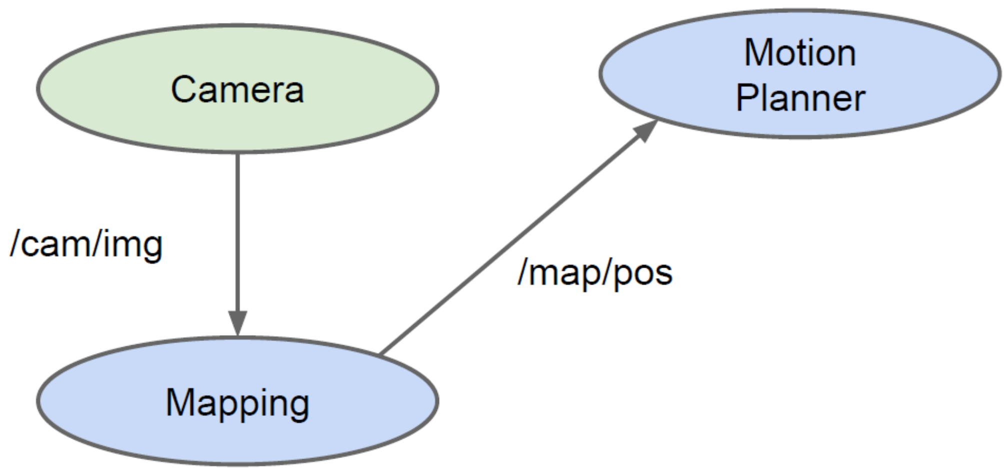
Nodes are processes that perform computation, “executables”



Topics

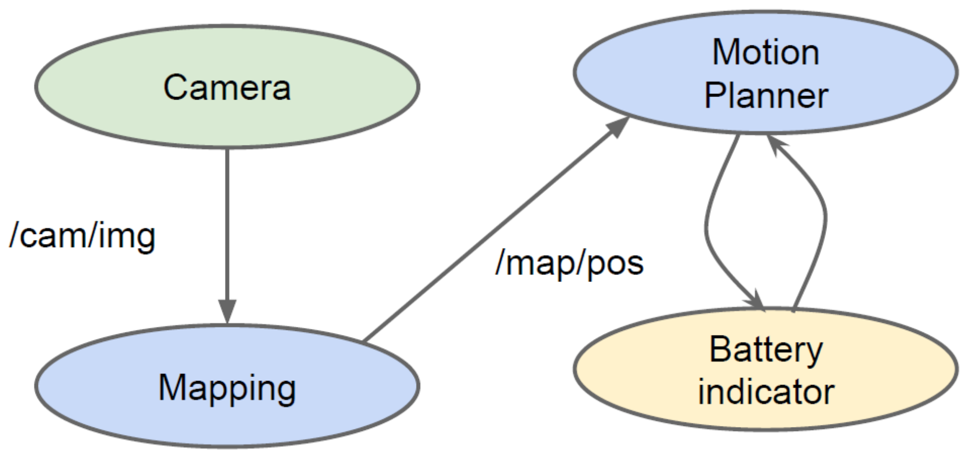
Topics are streams of data with publish / subscribe semantics.

They are uniquely identifiable by its name



Services

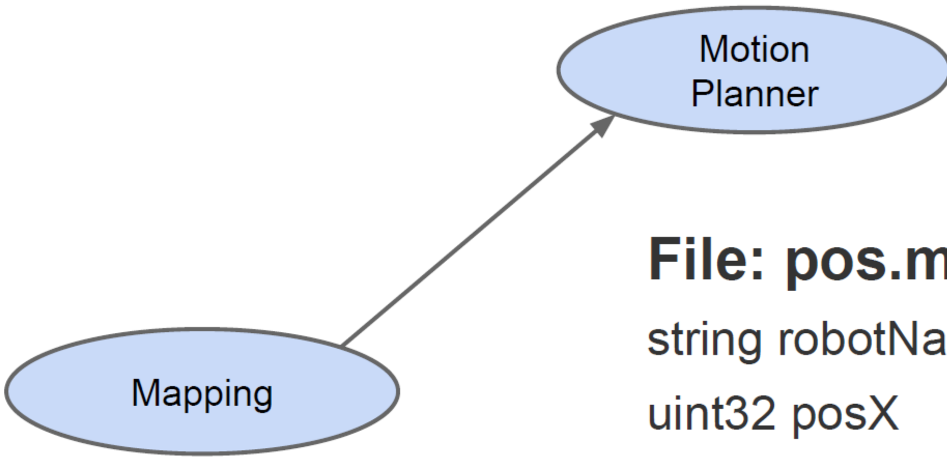
Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply.



Messages

A message is simply a data structure, comprising typed fields.

Language agnostic data representation. C++ can talk to Python.



File: pos.msg

string robotName

uint32 posX

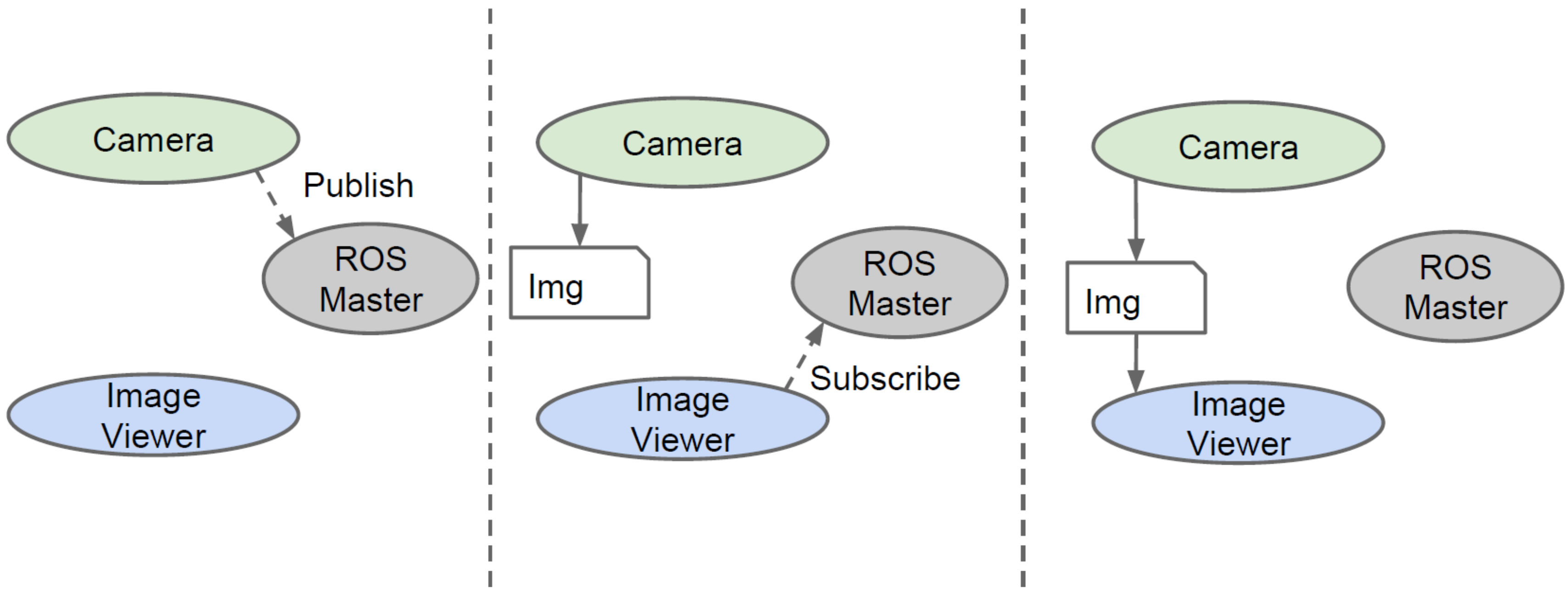
uint32 posY

uint32 goalX

uint32 goalY

ROS Master

The ROS Master provides name registration and lookup to nodes. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

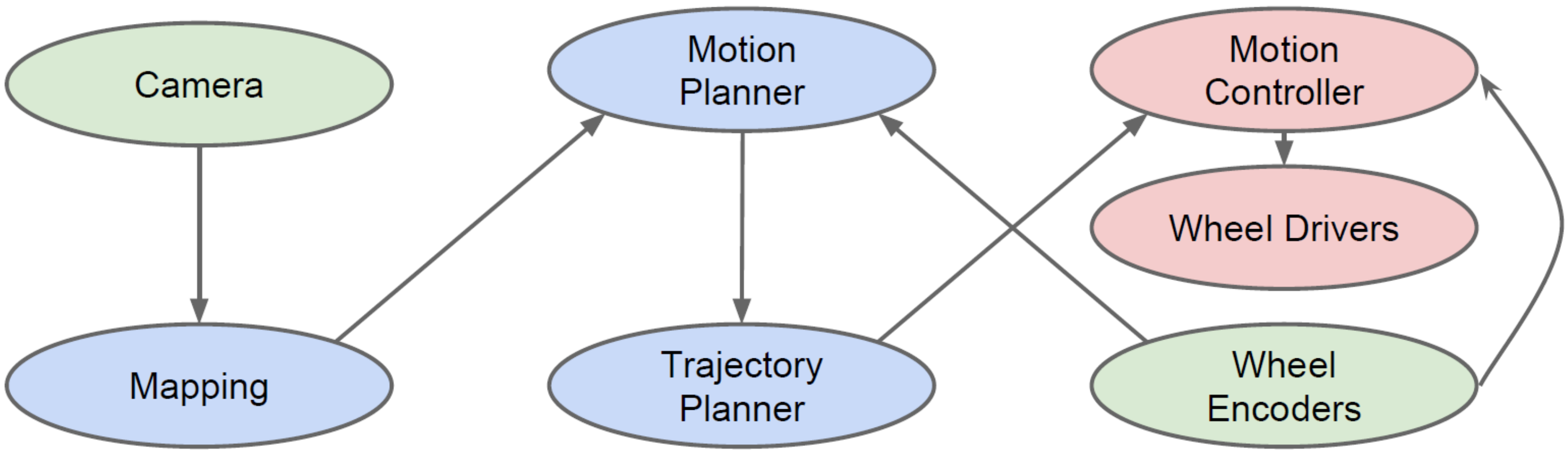


Example: Mobile Robot

Green - Sensors

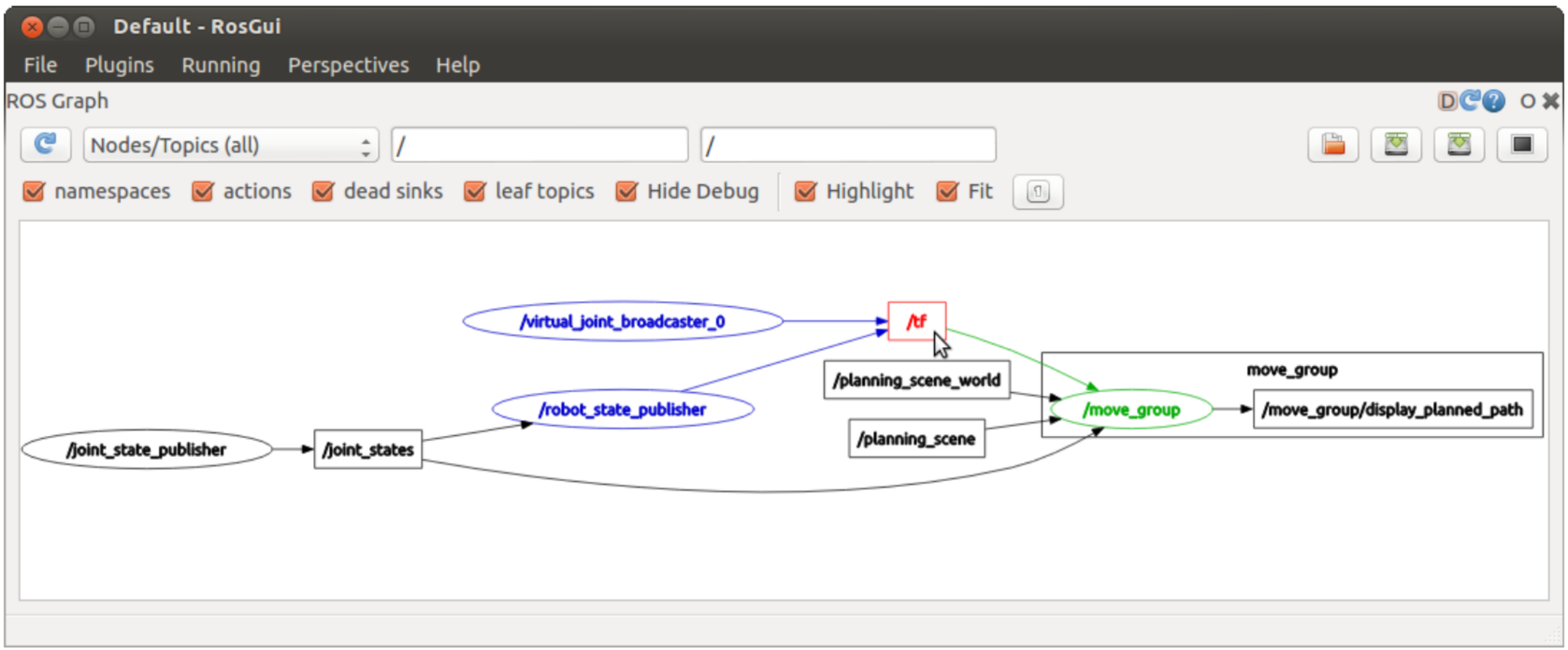
Blue - Planning algorithms

Red - Hardware integration



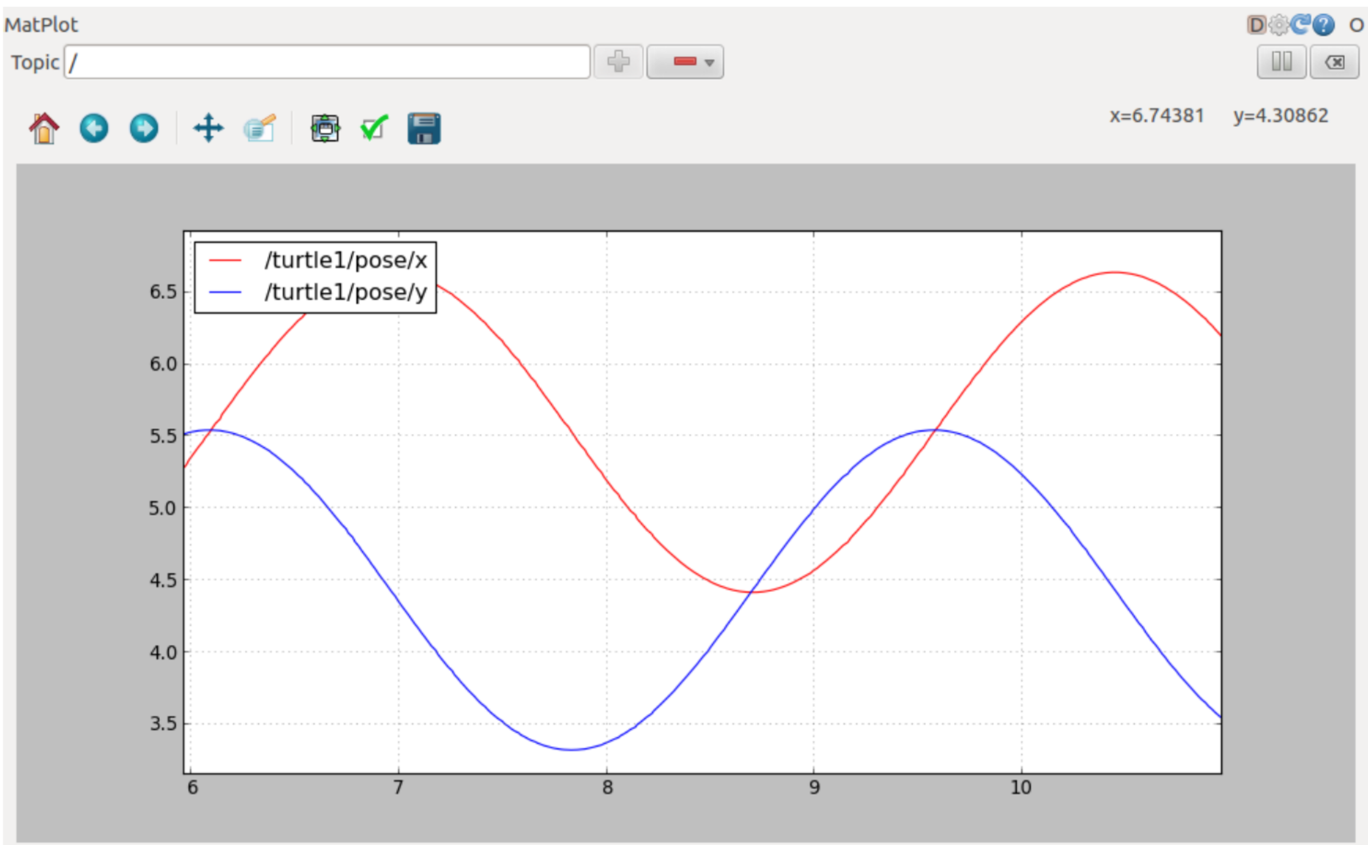
ROS Tools

System Visualisation: rqt_graph



ROS Tools

Live Plotting: rqt_plot



ROS Tools

Logging and Visualization Sensor Data: rosbag and rqt_bag

ROS Tools

3D Visualisation: RVIZ