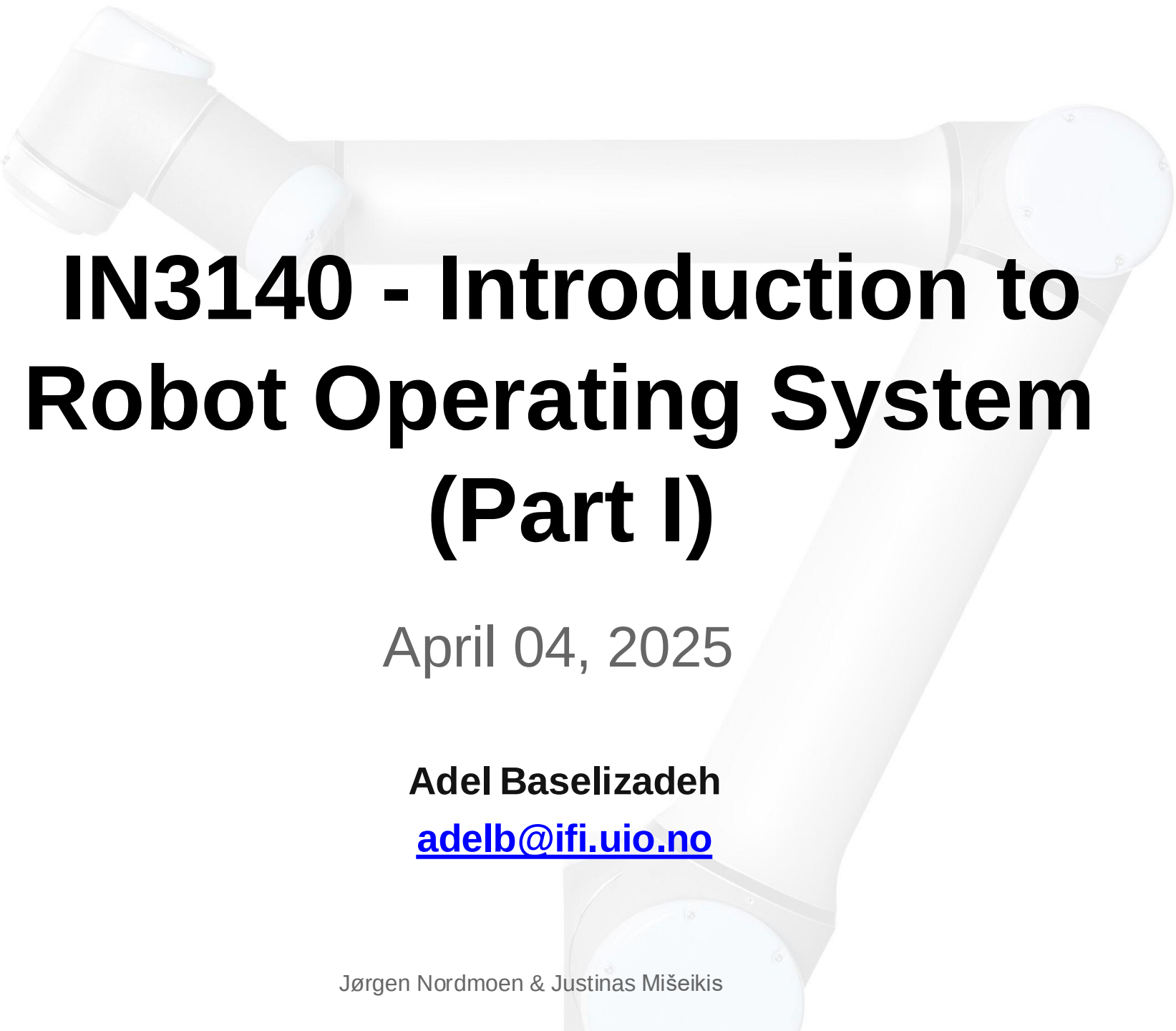


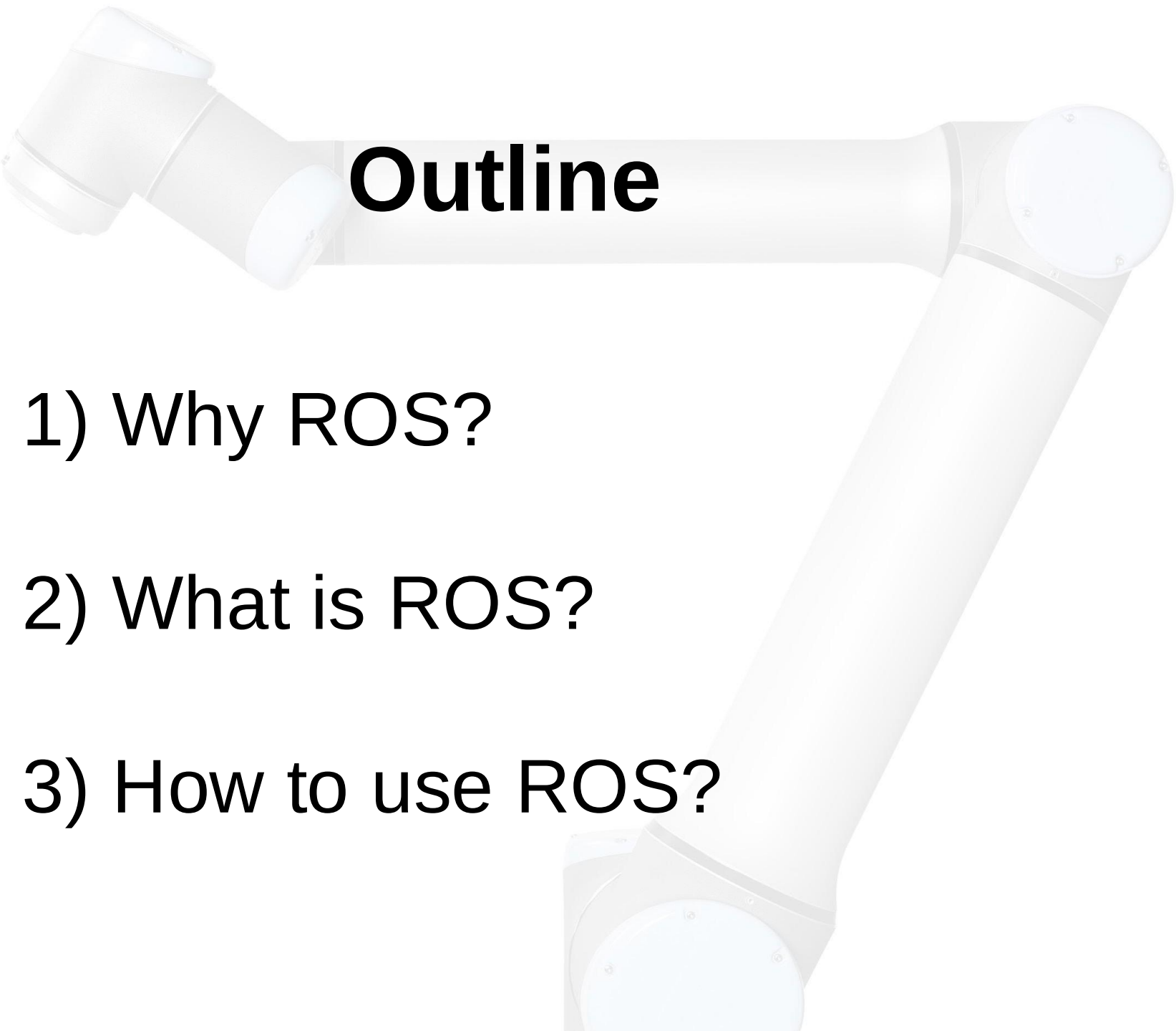


IN3140 - Introduction to Robot Operating System (Part I)

April 04, 2025

Adel Baselizadeh

adelb@ifi.uio.no

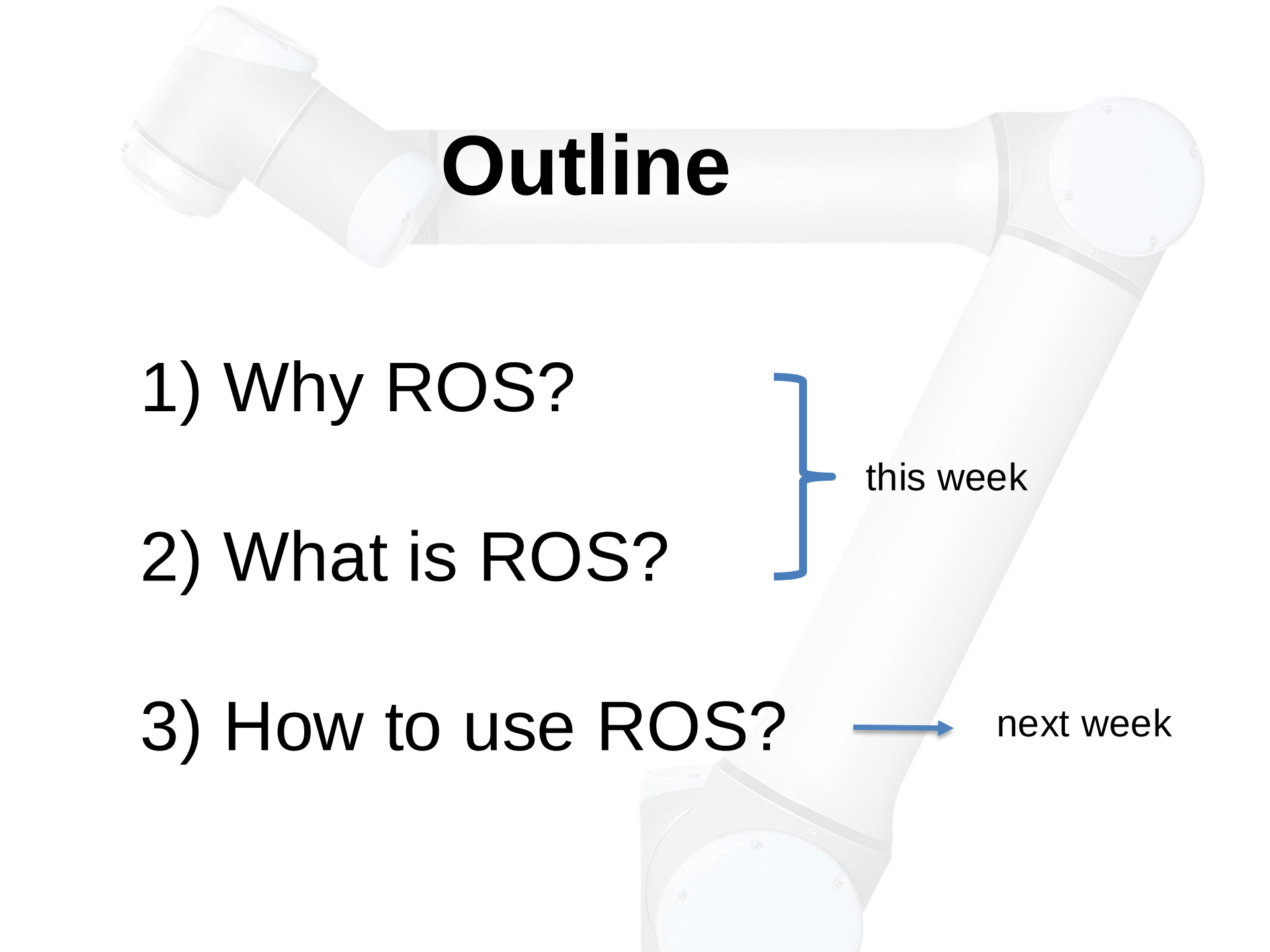


Outline

1) Why ROS?

2) What is ROS?

3) How to use ROS?



Outline

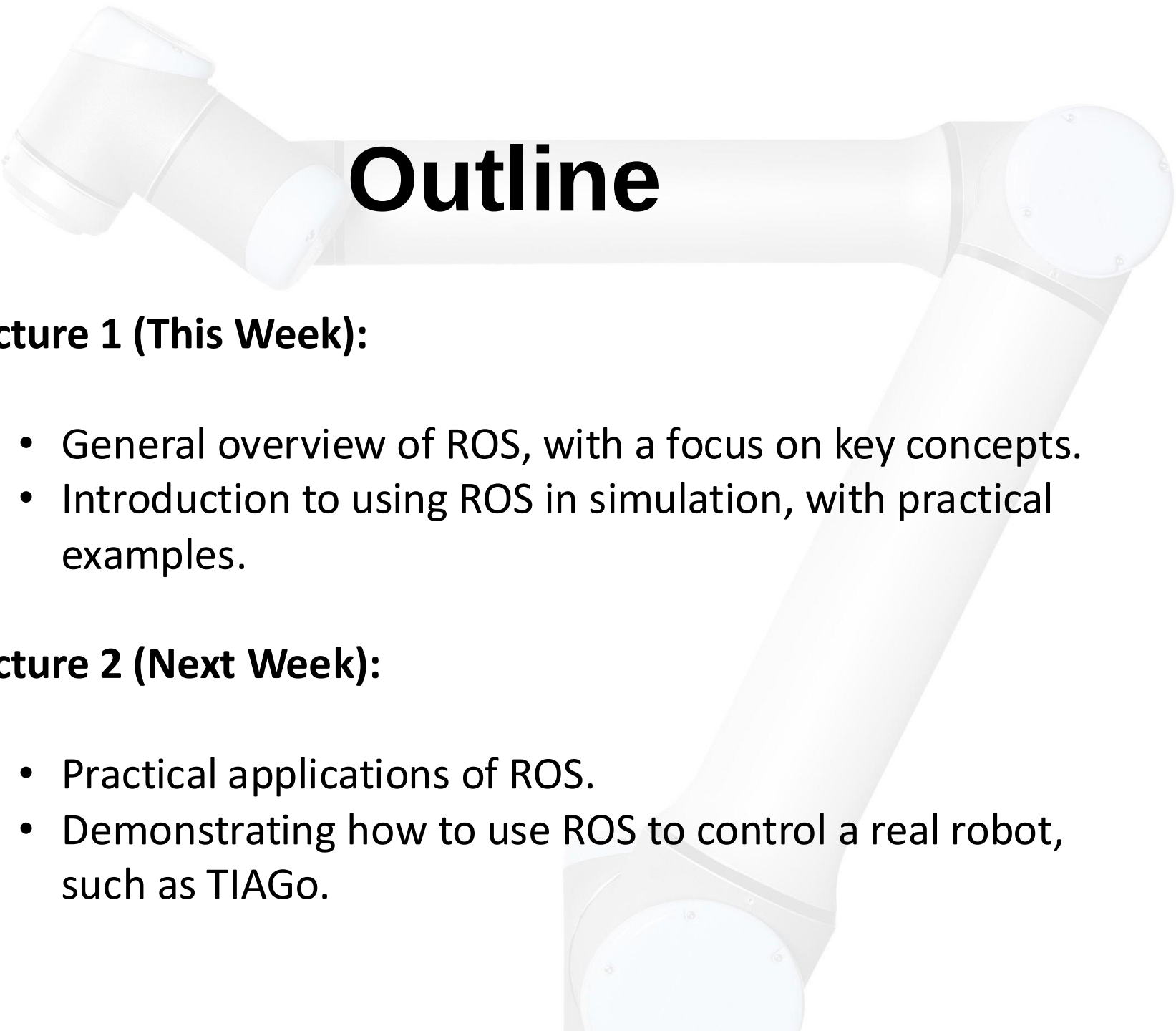
1) Why ROS?

2) What is ROS?

} this week

3) How to use ROS?

→ next week



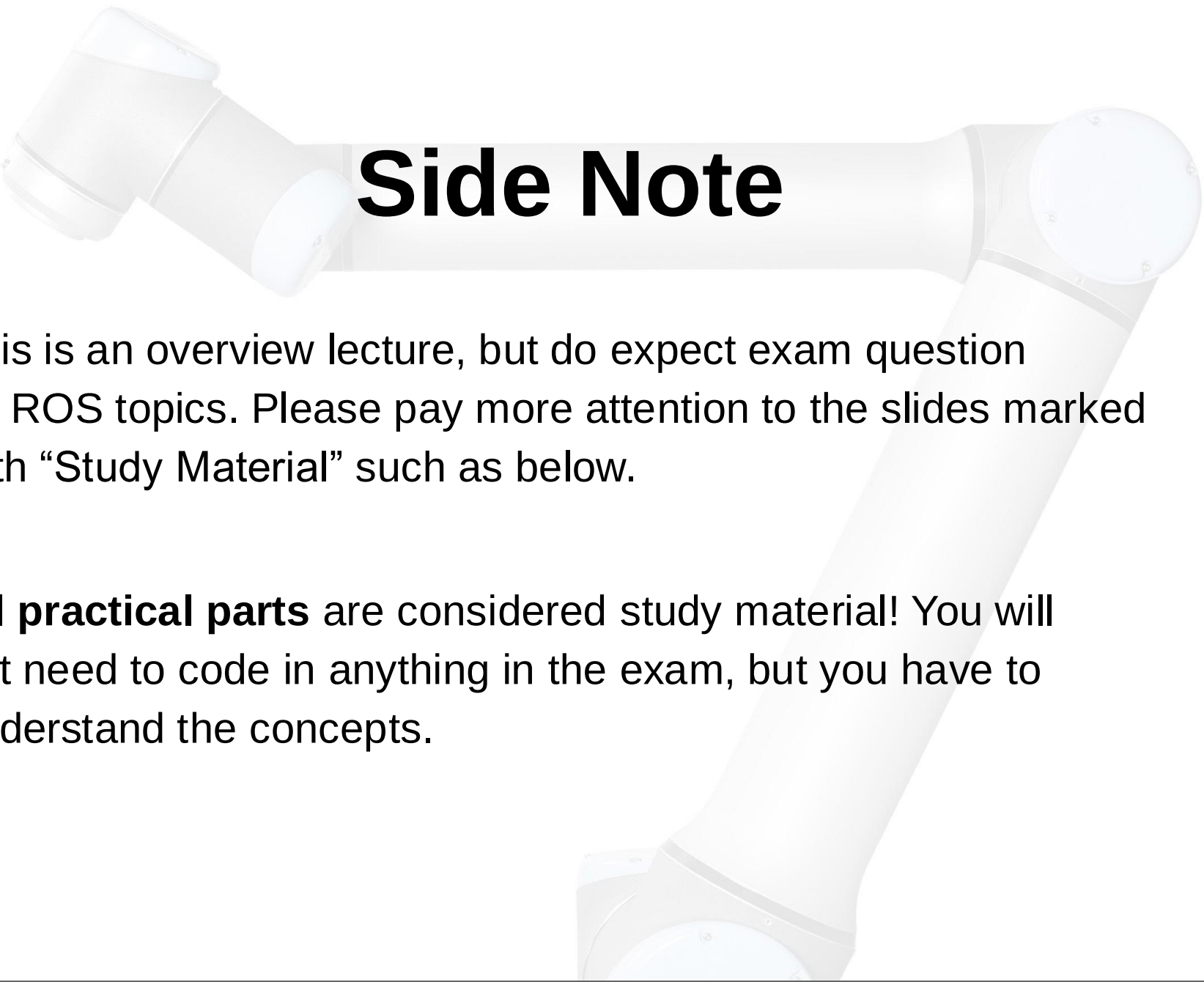
Outline

Lecture 1 (This Week):

- General overview of ROS, with a focus on key concepts.
- Introduction to using ROS in simulation, with practical examples.

Lecture 2 (Next Week):

- Practical applications of ROS.
- Demonstrating how to use ROS to control a real robot, such as TIAGo.



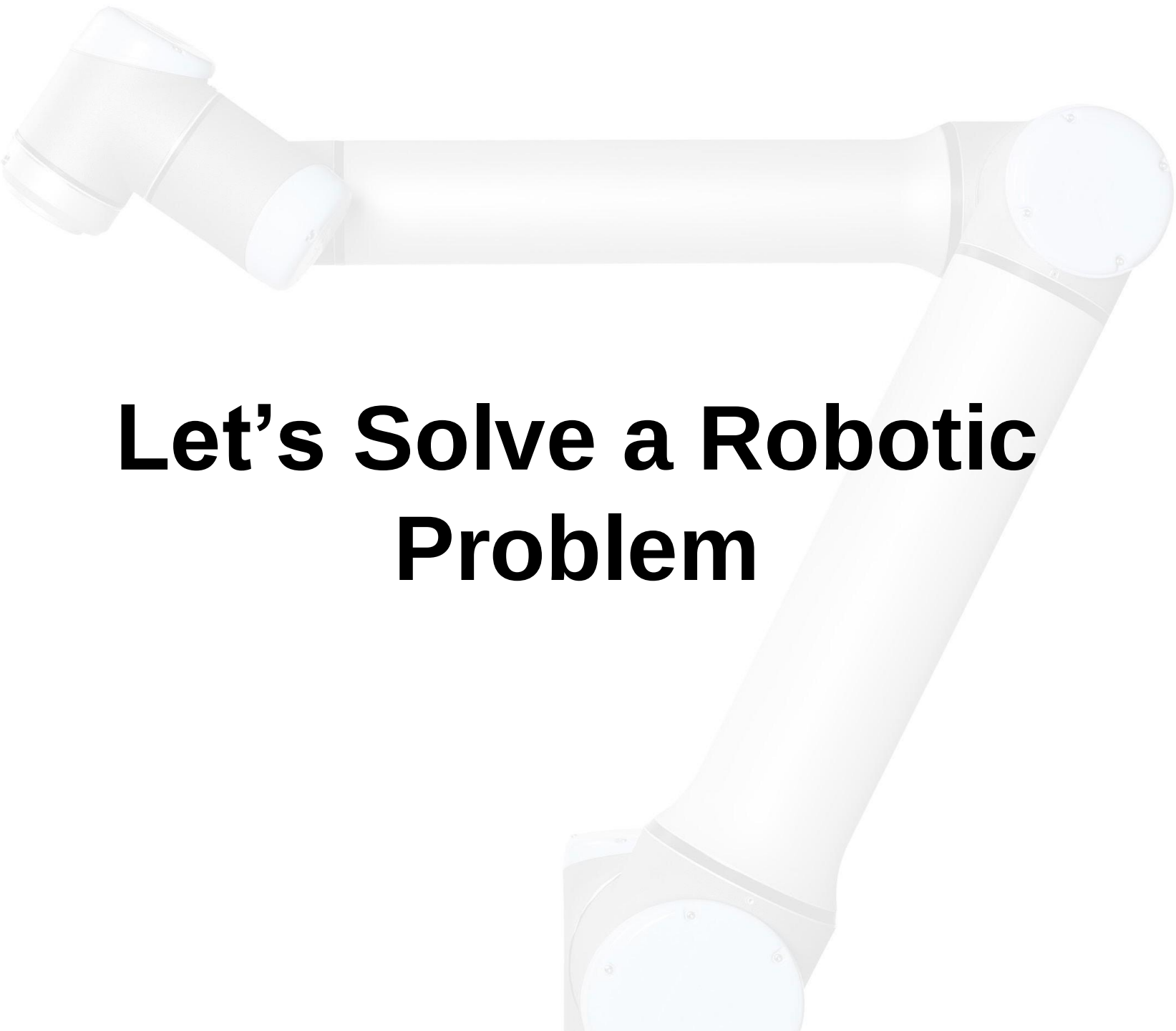
Side Note

This is an overview lecture, but do expect exam question on ROS topics. Please pay more attention to the slides marked with “Study Material” such as below.

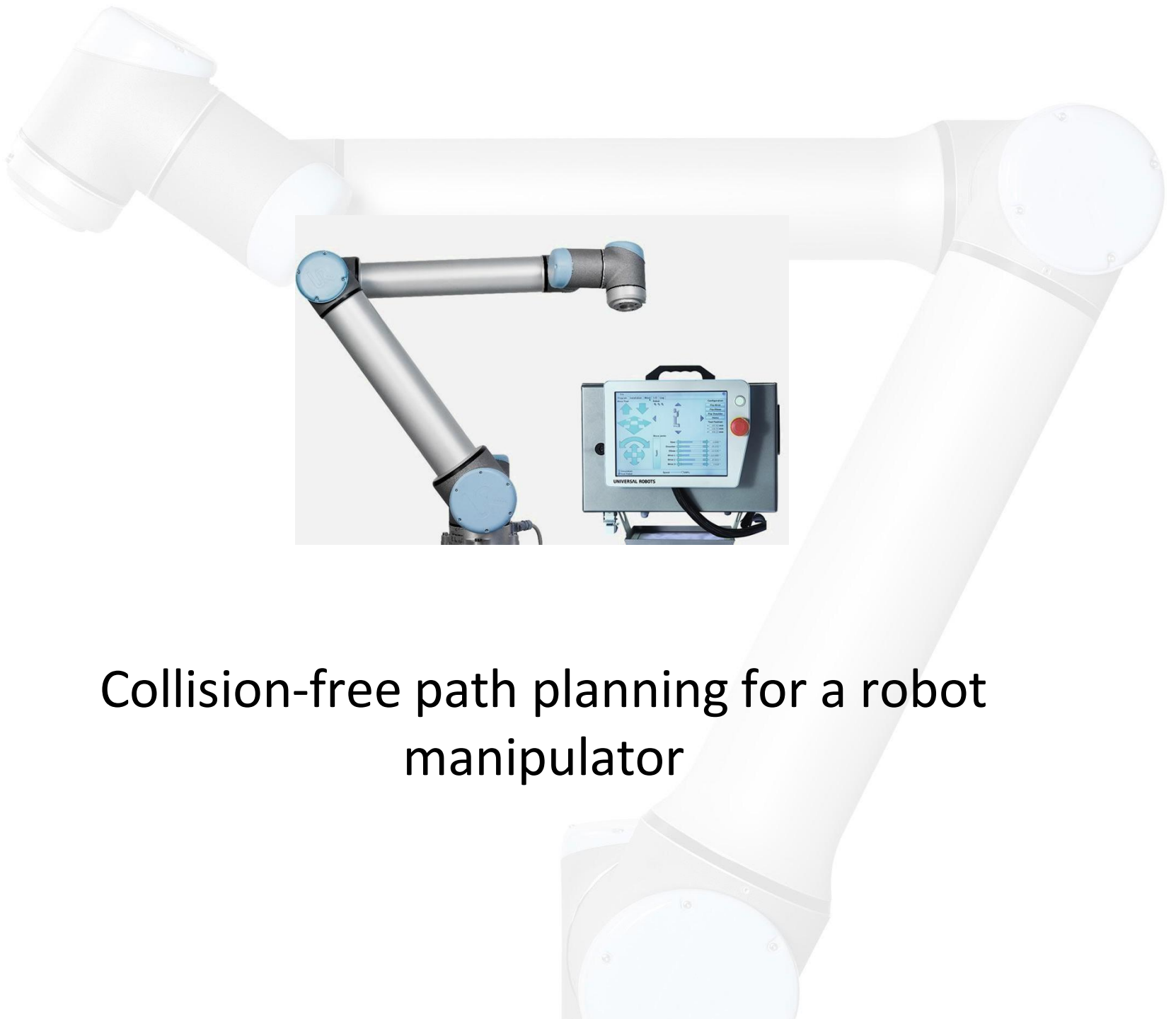
All practical parts are considered study material! You will not need to code in anything in the exam, but you have to understand the concepts.



Why ROS?
What is ROS?
How to use ROS?



**Let's Solve a Robotic
Problem**



Collision-free path planning for a robot
manipulator

Encoders

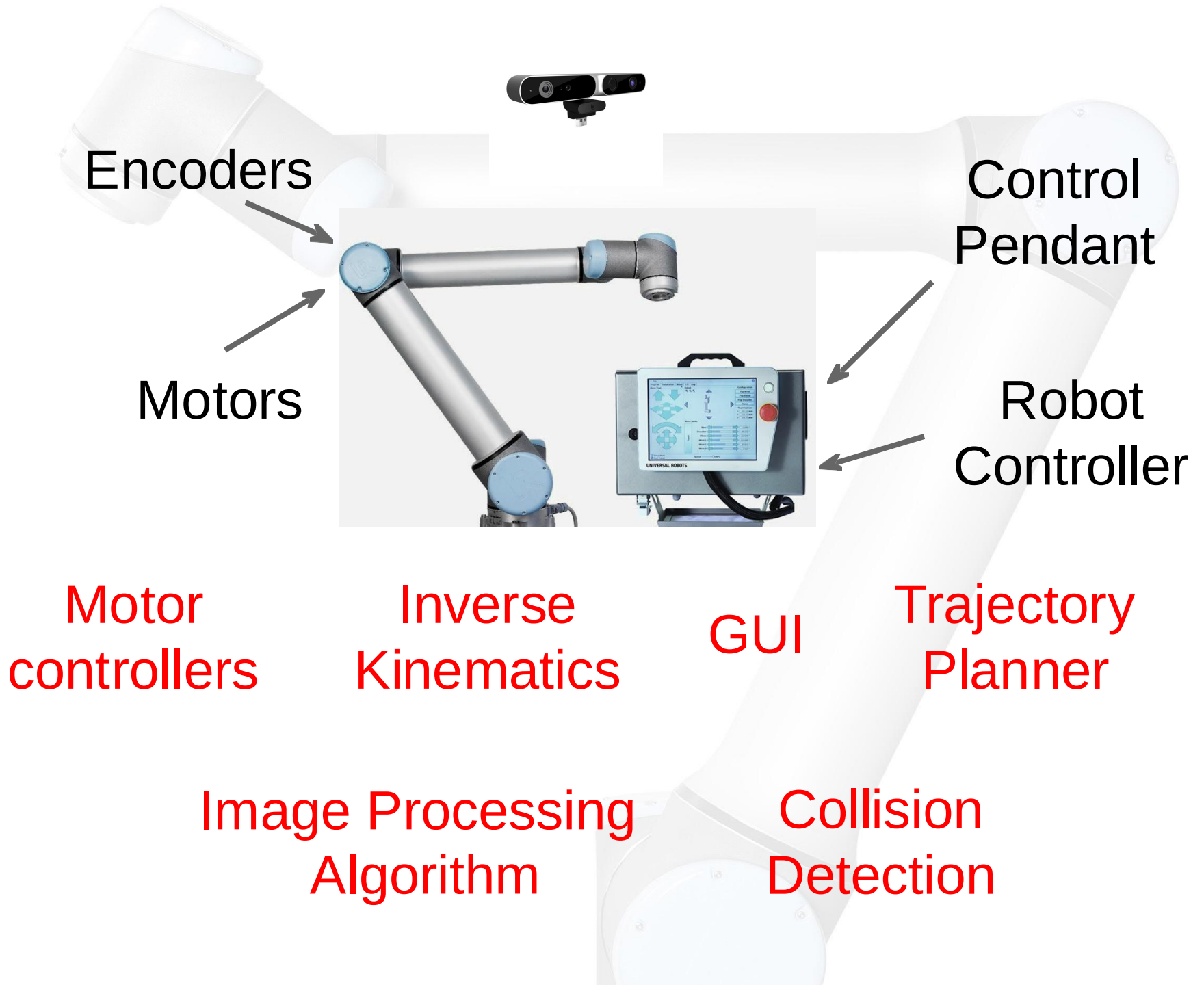
Motors

Control
Pendant

Processor



RGB-D Sensor





What a mess!

How can we deal with it?



ROS is an open-source, meta-operating system

Built on top of the operating system and helps us to manage different processes in a robot in real-time.



ROS

ROS is an open-source, meta-operating system



pointcloudlibrary

What's so special about ROS?

- 1250+ Robotic platforms officially support ROS (ROS metrics report 2024) <http://wiki.ros.org/Robots>
 - 2023 --> 910 2022 --> 740
- Thousands of ready to use algorithms
- Efficient, so it can be used for actual products, not just prototyping
- Runs on Ubuntu, also ARM Processors
 - Could be run on Windows and OS X through containers like Docker and Singularity.
- Parallelisation and networking made easy, can use multiple machines simultaneously

Current Robotics Job Ads

Experience in developing robotics software, e.g., kinematics/dynamics, control of actuators/sensors, distributed systems. Provable proficiency in C, C++ and experience in at least another programming language (eg. python, java). **Hands-on experience in robotics middlewares, e.g. ROS, YARP, Orocus**

Must haves: Excellent general structured problem-solving and software architecture skills. Demonstrated strong software engineering and design fundamentals Fluency in C/C++. Experience with path planning and navigation. **Experience in ROS and simulation environments.** Experience developing in a Linux environment.

Robotics Specialist. Core tasks are the development of algorithms for grasp calculation and the improvements of existing solutions. Skills: 3+ years C++ development, Machine Learning, **ROS, Ubuntu/Linux, PCL**

The candidate must be a **proficient user of C/C++ and ROS and any relevant computer vision library (e.g., ViSP, OpenCV, PCL)**. Scientific curiosity, large autonomy and ability to work independently are also expected.

The technical Requirements: C++ or Python programming knowledge in Linux, **Knowledge of ROS. You have to be able to write ROS programs, debug and find your way Knowledge of Gazebo.**

Robotician: Path-planning Specialist

- Own the navigation costmaps area and implement various data processing algorithms
- Have experience and knowledge on 2D data processing for motion planning, e.g. Fast Marching Methods
- Have experience with state-of-the-art path-planning approaches, e.g. RRT*
- **Very good C++ skills (ROS, OpenCV, Linux)**

One of many sources: <http://www.theconstructsim.com/ros-jobs/>

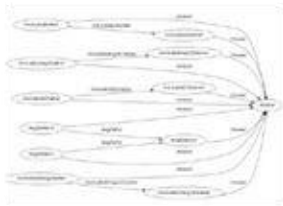


Why ROS?
What is ROS?
How to use ROS?

What is ROS?

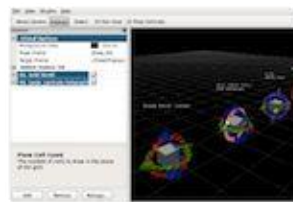


=



Plumbing

+



Tools

+



Capabilities

+



Ecosystem

<http://www.ros.org/about-ros/>

ROS

Plumbing

Tools

Capabilities

Ecosystem



Let's see how it works!

Plumbing

ROS provides publish-subscribe messaging infrastructure designed to support the quick and easy construction of distributed computing systems.

Nodes

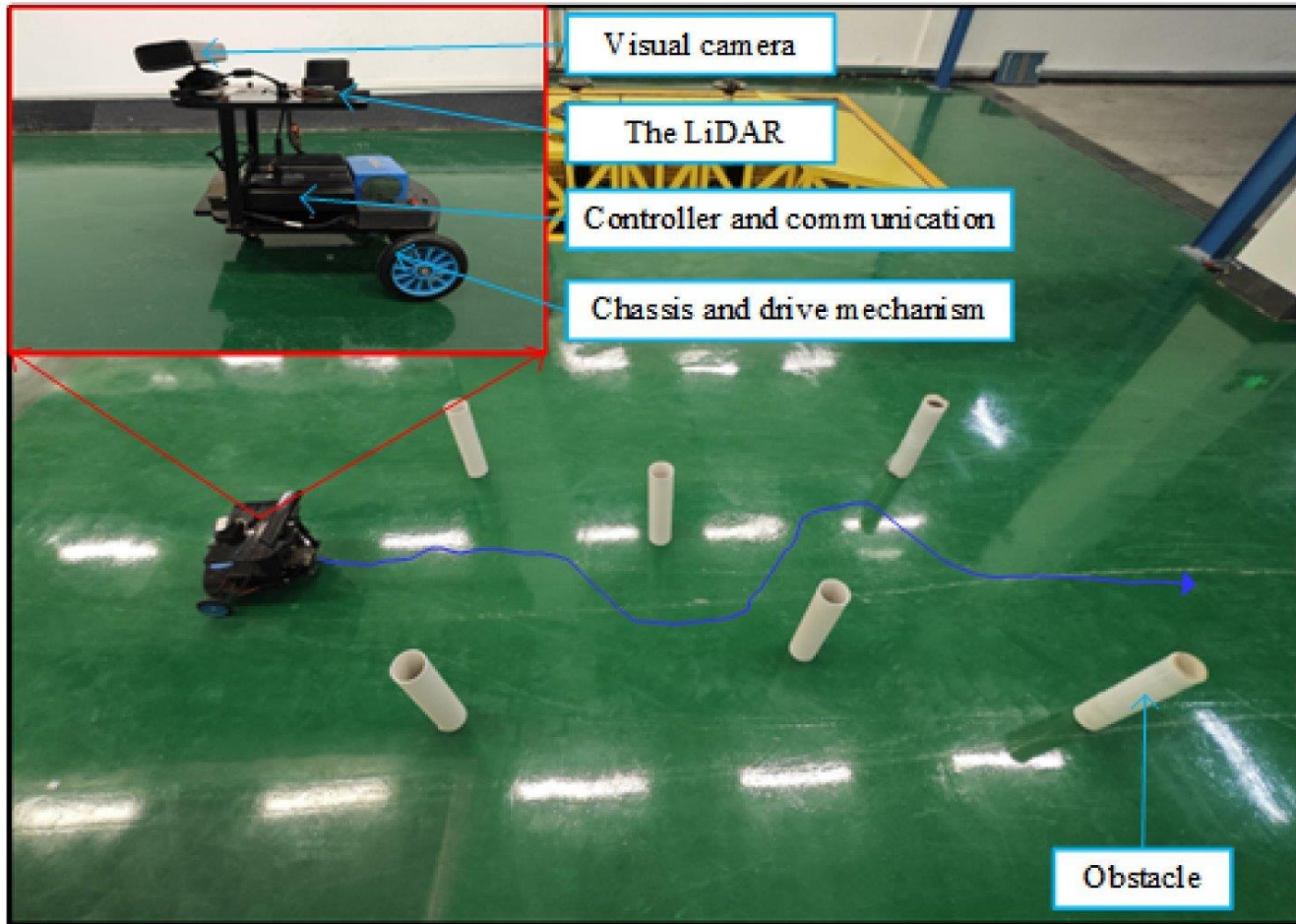
<http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes>

Nodes are processes that perform computation, “executables”. Each node performs a specific processing part, usually a part of the algorithm.

Nodes

<http://wiki.ros.org/ROS/Tools/ROS/Tutorials/Using%20ROS%20Nodes>

Nodes
Each node
the algorithm

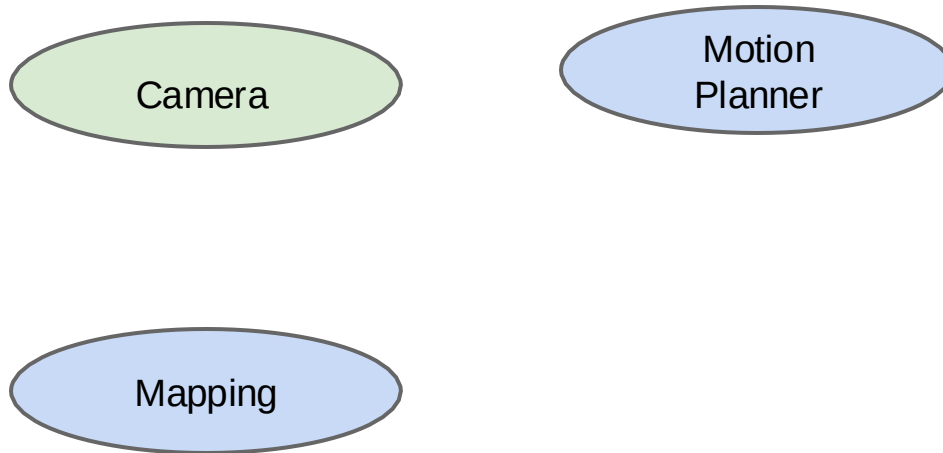


les".
part of

Nodes

<http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes>

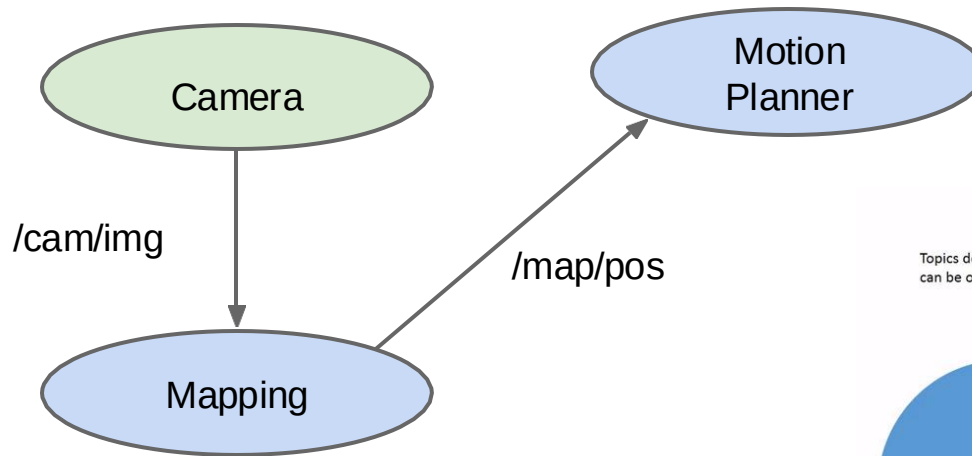
Nodes are processes that perform computation, “executables”. Each node performs a specific processing part, usually a part of the algorithm.



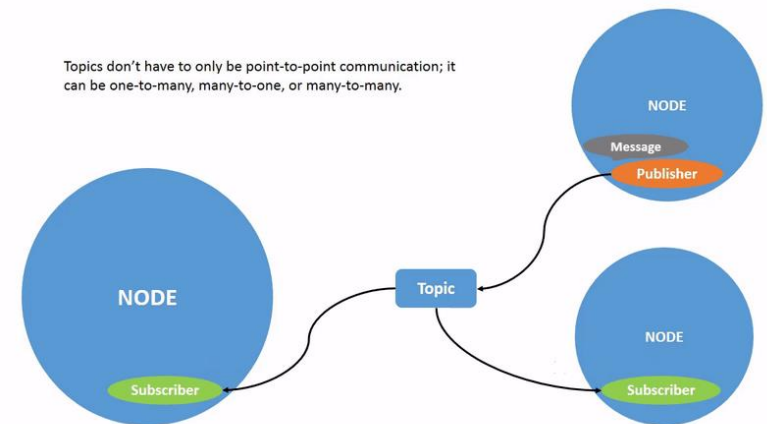
Topics

<http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics>

Topics are data streams with a publish/subscribe model. Each topic is uniquely identifiable by its name. Nodes can publish to or subscribe to topics to exchange data.



Topics don't have to only be point-to-point communication; it can be one-to-many, many-to-one, or many-to-many.

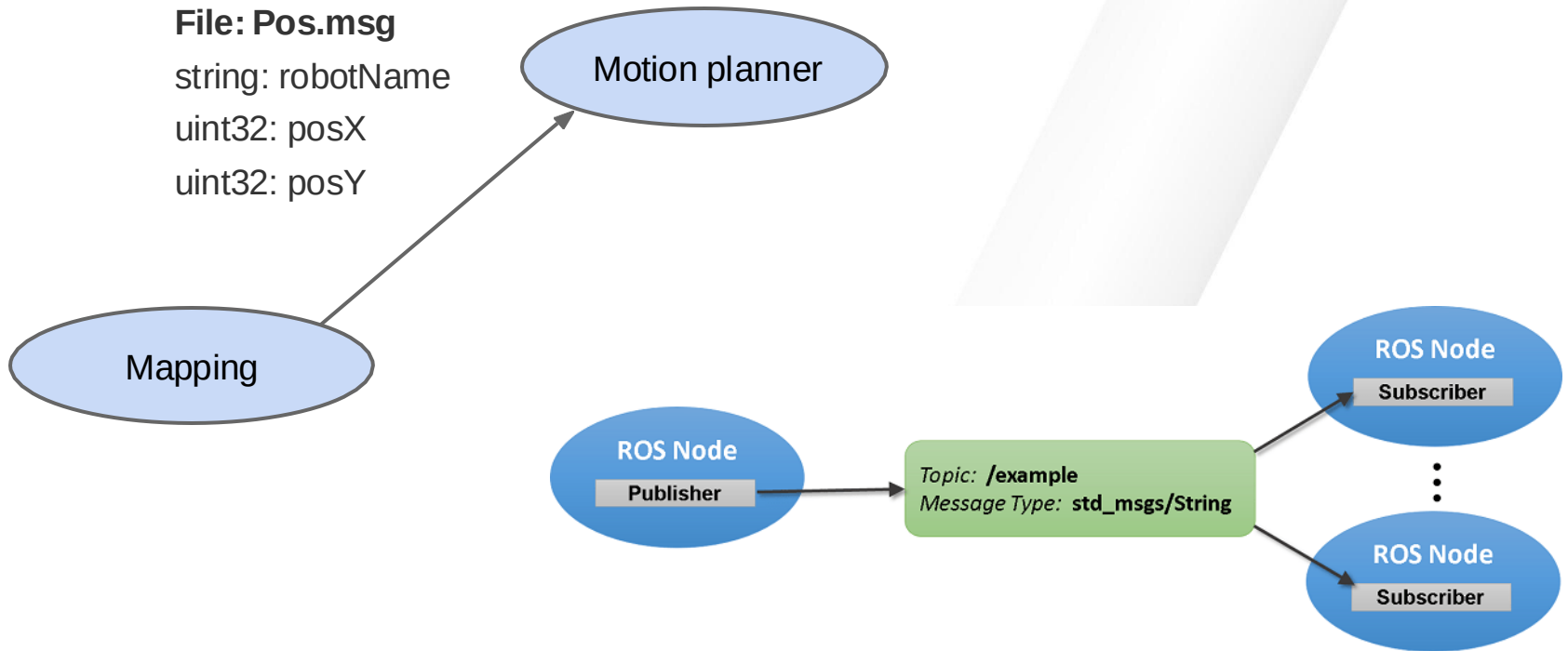


Study Material - highly recommend to check the URL

Messages

<http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv>

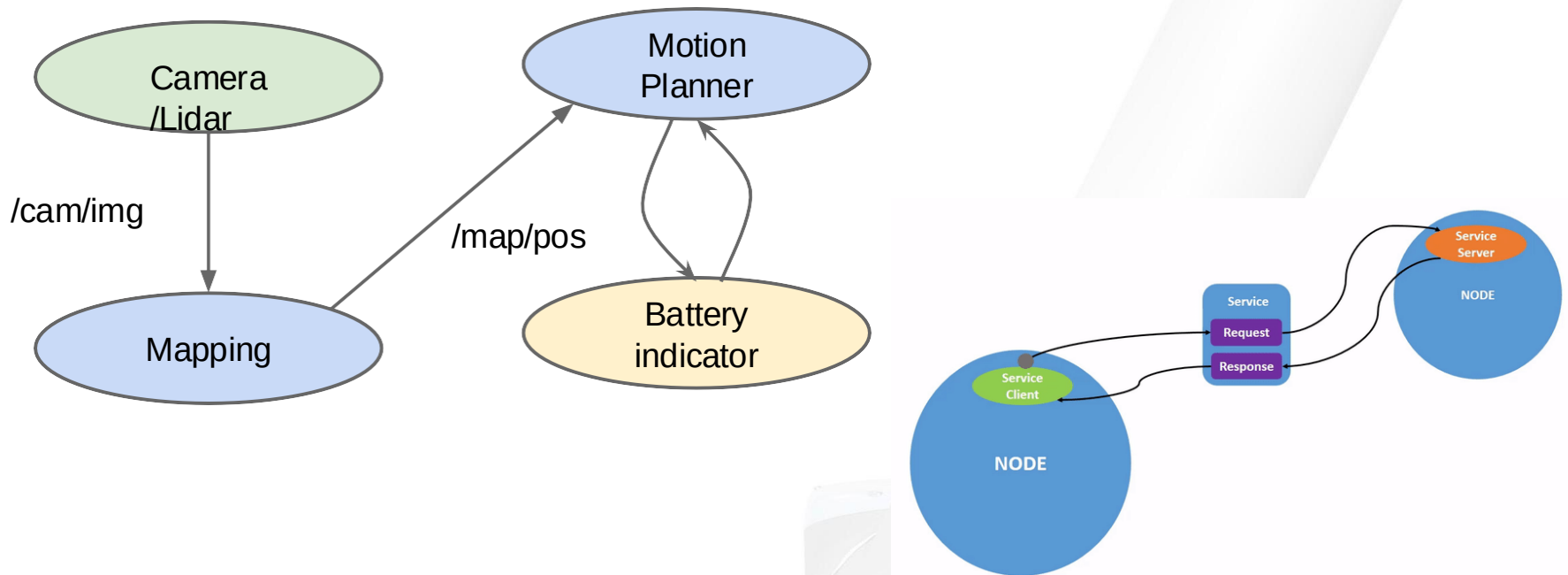
A **message** in ROS is a data structure, used for exchanging information between nodes. Messages are sent over predefined topics to share data like sensor readings or control commands.



Services

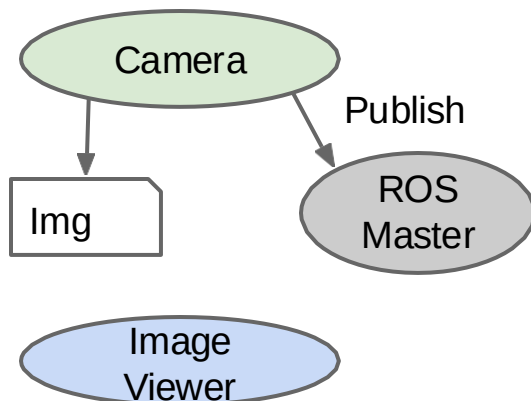
<http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams>

In ROS, **services** provide a request-response communication mechanism between nodes. A service consists of two parts: a **service request** and a **service response**. One node sends a request, and another node processes it and sends a response.



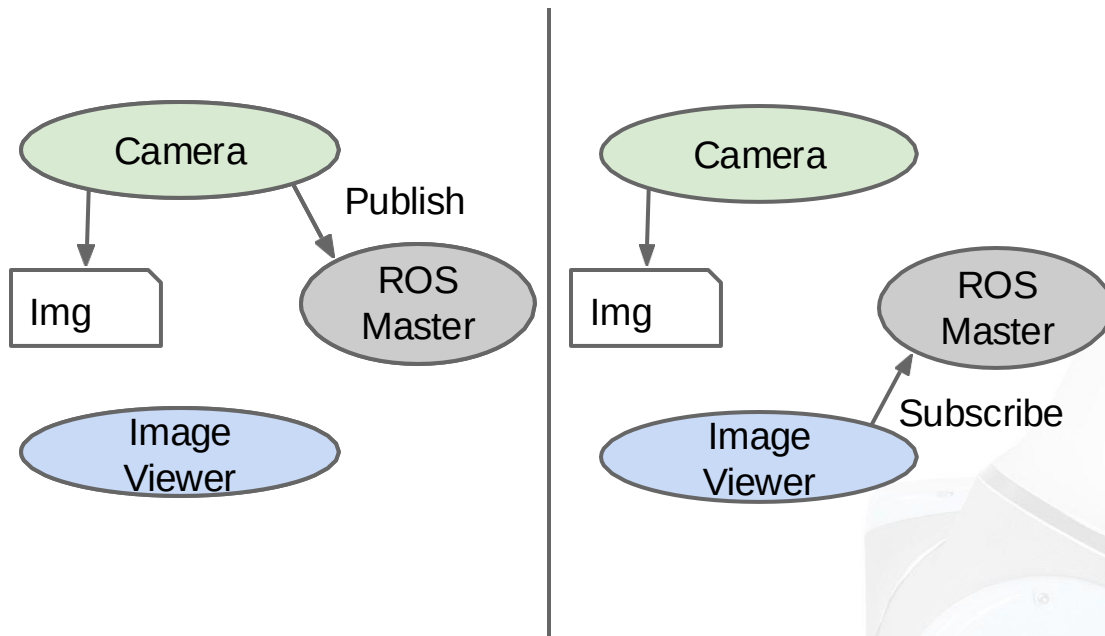
ROS Master

The **ROS Master** is a central component in ROS that manages communication between nodes. It keeps track of all active nodes, topics, and services, allowing them to discover each other and exchange messages. The Master facilitates node registration, name resolution, and overall coordination in a ROS system.



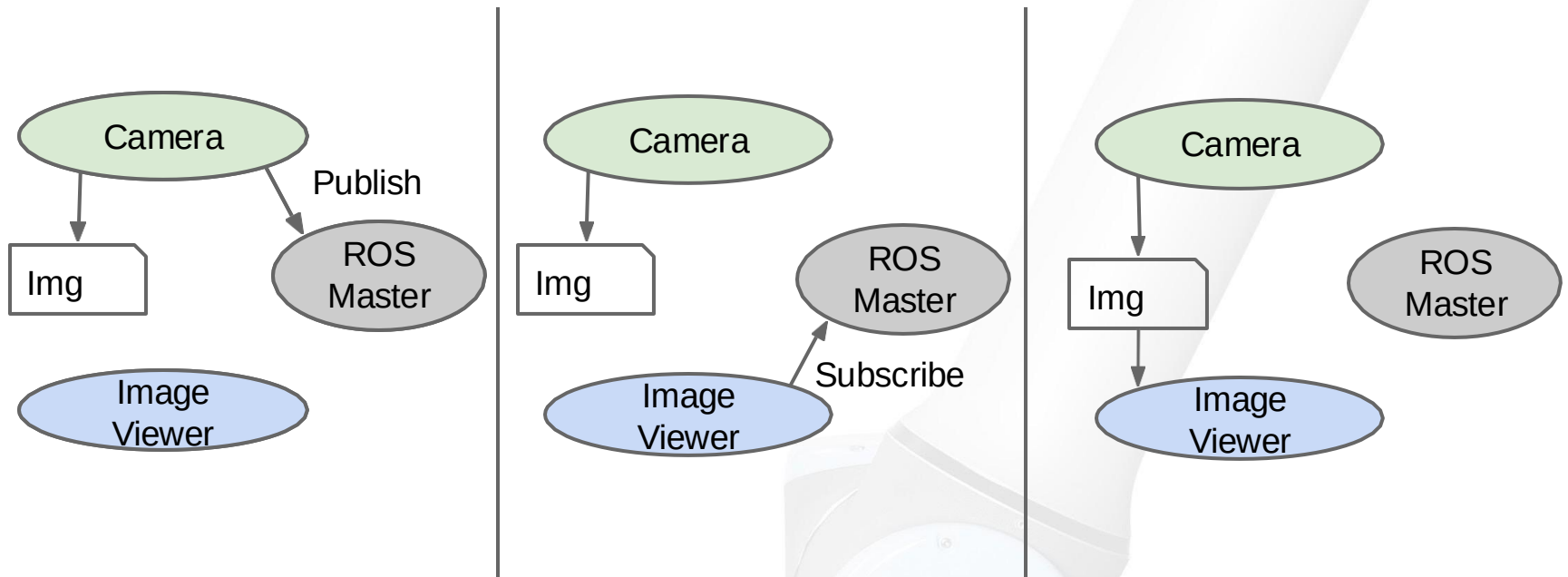
ROS Master

Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.



ROS Master

Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

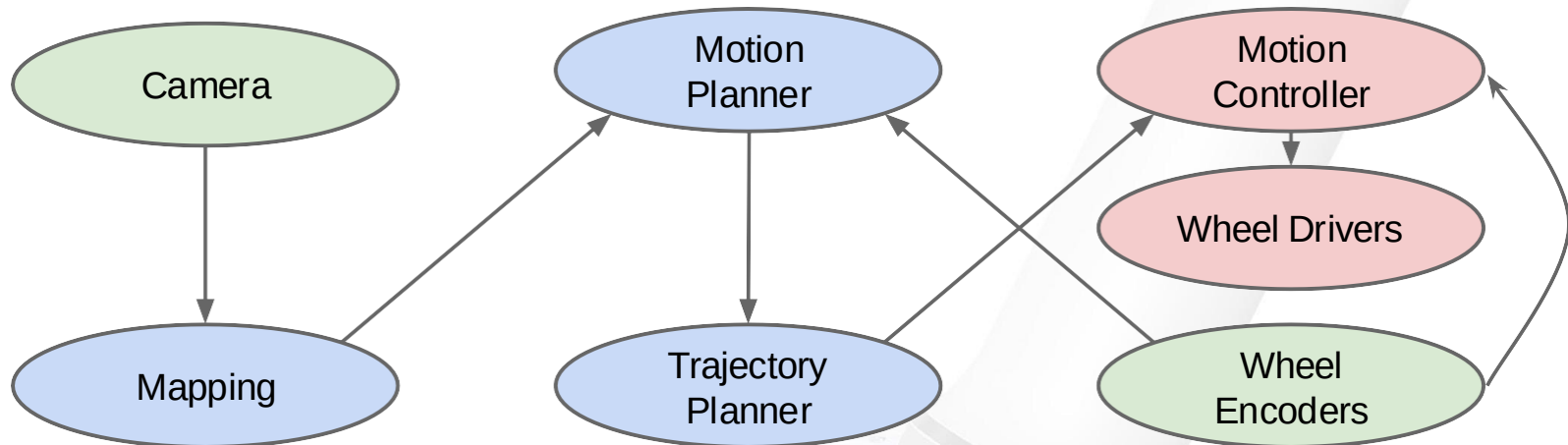


Example System - Mobile Robot

Green - Sensors

Blue - Planning algorithms

Red - Hardware integration





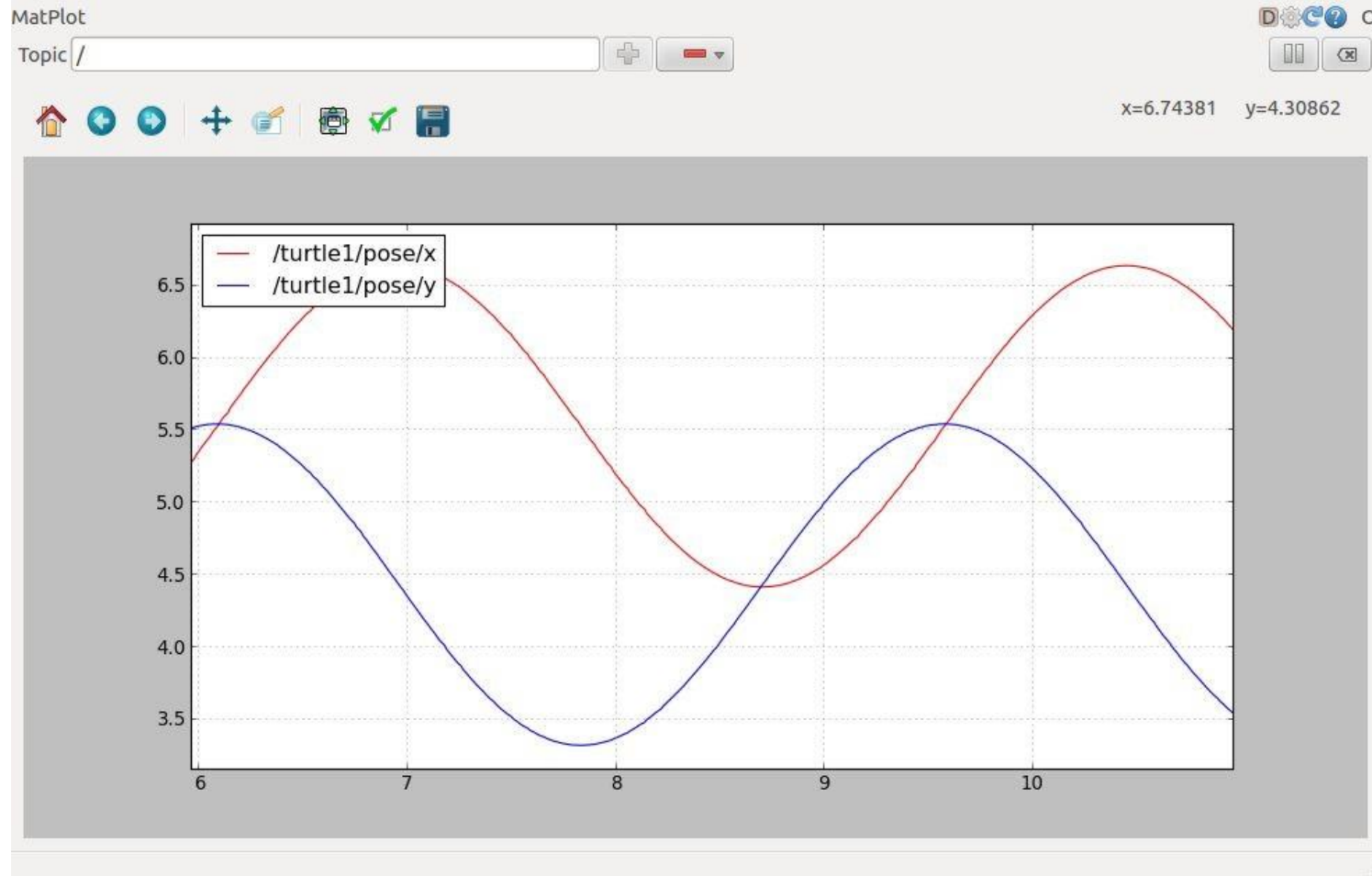
Tools

www.wiki.ros.org/Tools

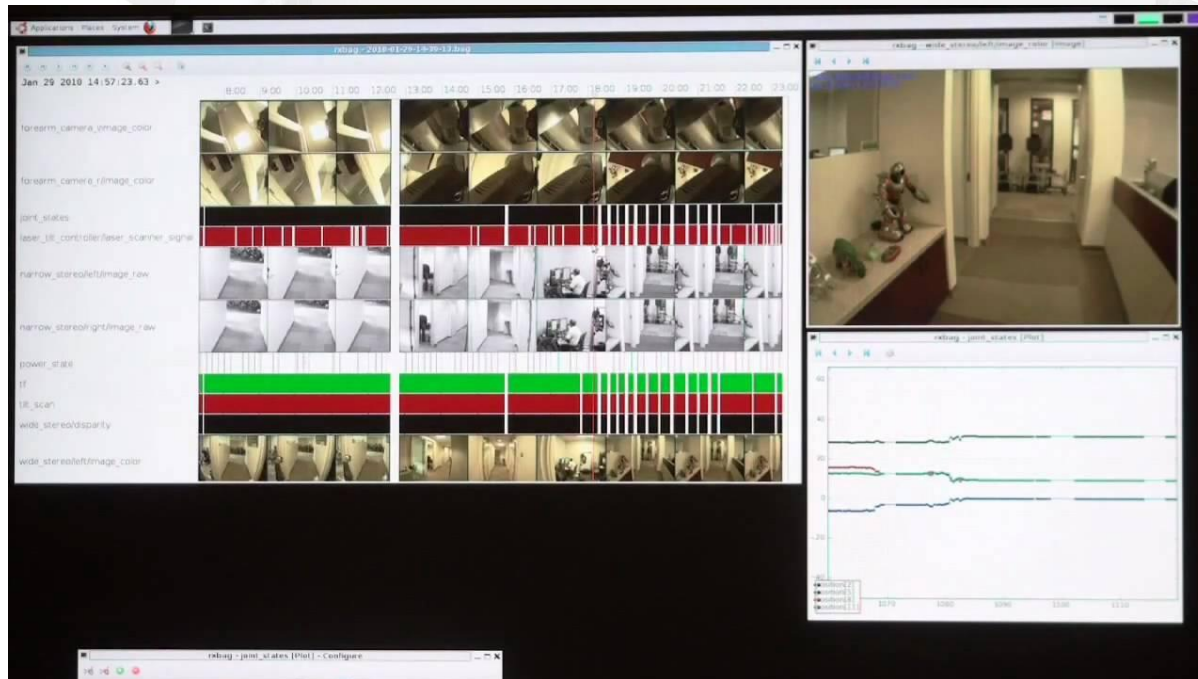
System Visualisation: rqt_graph



Live Plotting: rqt_plot



Logging and Visualization Sensor Data: rosbag and rqt_bag

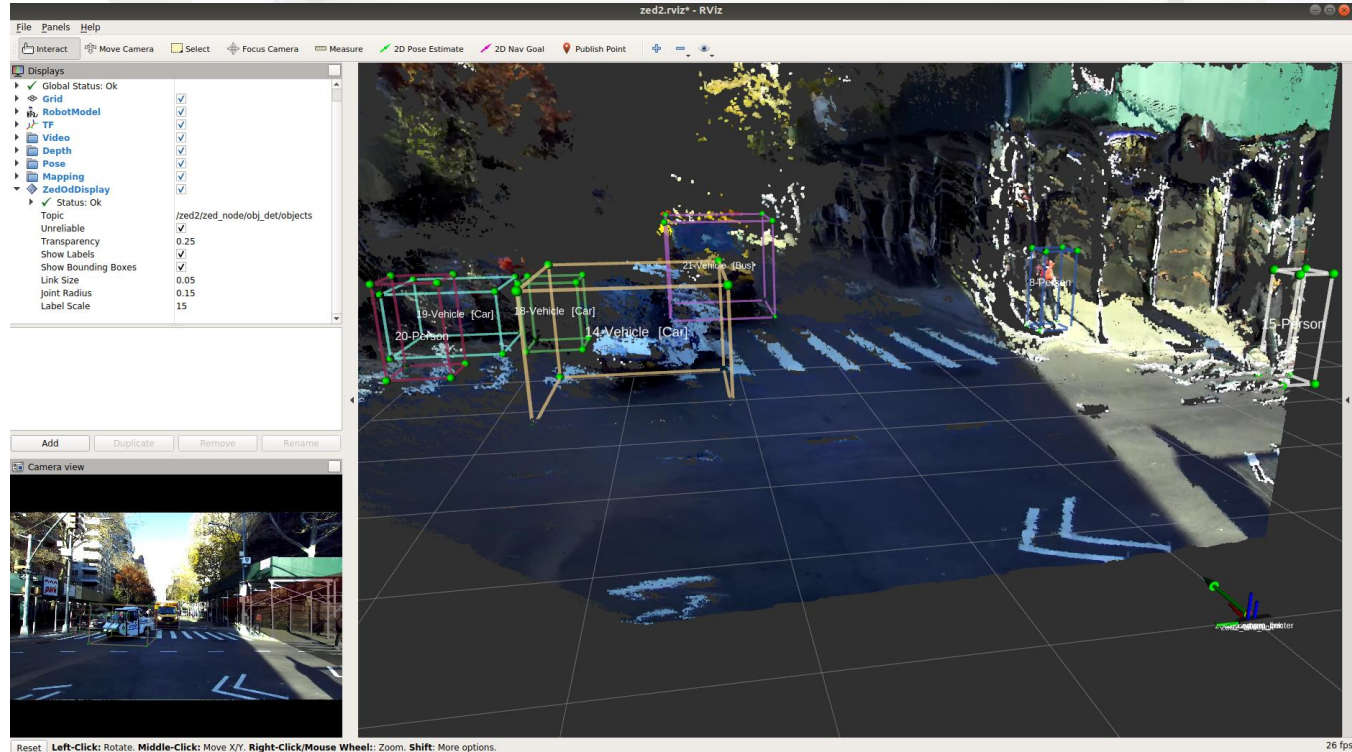


This is a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoids deserialization and reserialization of the messages.

https://www.youtube.com/watch?v=pwlbArh_neU

<https://drive.google.com/file/d/1NmHkk8ne6ASiYjCgu0yKc7rNQ7VuuVPu/view>

3D Visualisation: RVIZ



RVIZ is a ROS graphical interface that allows you to visualize a lot of information, using plugins for many kinds of available topics.

<https://www.youtube.com/watch?v=i--Sd4xH9ZE>



Capabilities

<https://www.youtube.com/watch?v=mDwZ21Zia8s>



Review of Technical Capabilities

<http://gazebosim.org/>



GAZEBO

Gazebo basics, Gazebo files

- **Gazebo** is an open-source ROS-based simulator used for simulating robots, environments, and sensors. It provides a 3D visualization of the robot and environment, allowing you to observe robot motions, sensor data, and interactions with the simulated world.
- **Physics simulation:** Gazebo provides realistic physics simulation, including gravity, friction, collisions, etc.
- **Sensor simulation:** It can simulate various sensors such as cameras, lidars, IMUs, and more.
- **Robot modeling:** You can design and test robot models in Gazebo using URDF (Unified Robot Description Format) or SDF (Simulation Description Format). It supports different types of robots, from simple mobile platforms to complex humanoids and drones.



GAZEBO

Gazebo basics, Gazebo files

To run a Gazebo simulation you need:

- **A world file:** A file with extension `.world` that contains all the elements in a simulation, including robots, lights, sensors, and static objects, formatted using the Simulation Description Format (SDF).



GAZEBO rrbot example

RRBot, or "Revolute-Revolute Manipulator Robot", is a simple 3-linkage, 2-joint arm.

```
cd ~/catkin_ws/src/
```

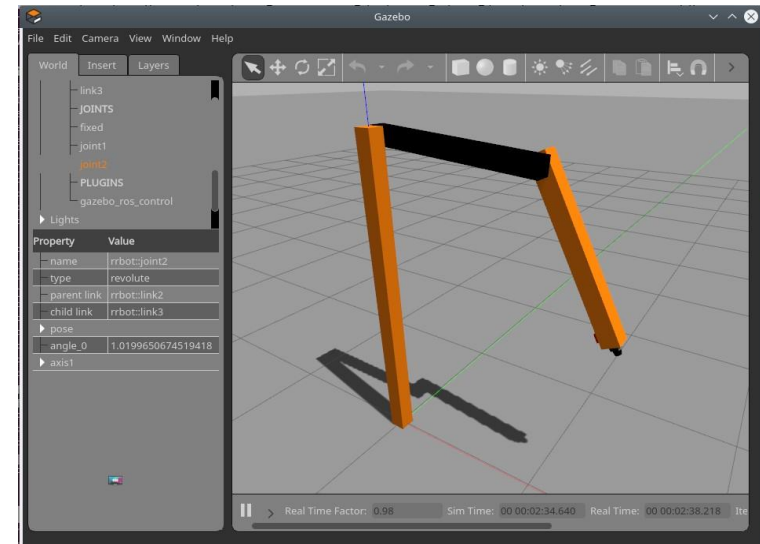
```
git clone https://github.com/ros-simulation/gazebo_ros_demos.git
```

```
cd ..
```

```
catkin_make
```

```
roscd rrbot_description rrbot.xacro
```

```
roslaunch rrbot_gazebo rrbot_world.launch
```





Review of Technical Capabilities

<https://moveit.ros.org>

<https://www.youtube.com/watch?v=vAeEEoxVhAo>

MoveIt Motion Planning

MoveIt! includes a variety of robust and state-of-the-art motion planners:

- Sampling-based motion planning algorithms (OMPL)
- Covariant Hamiltonian optimization for motion planning (CHOMP)
- Stochastic Trajectory Optimization for Motion Planning (STOMP)
- TrajOpt is a sequential convex optimization algorithm

MoveIt Constraints

You can specify the following kinematic constraints:

- **Position constraints** – restrict the position of a link to lie within a region of space
- **Orientation constraints** – restrict the orientation of a link to lie within specified roll, pitch or yaw limits
- **Visibility constraints** – restrict a point on a link to lie within the visibility cone for a particular sensor
- **Joint constraints** – restrict a joint to lie between two values
- **User-specified constraints** – you can also specify your own constraints with a user-defined callback.



Scene Collision Objects

You can specify the following kinematic constraints:

- static objects (objects rigidly fixed on the robot workspace)
- dynamic objects (objects with which the robot can interact, i.g. pick, place, push ...etc)



Ecosystem

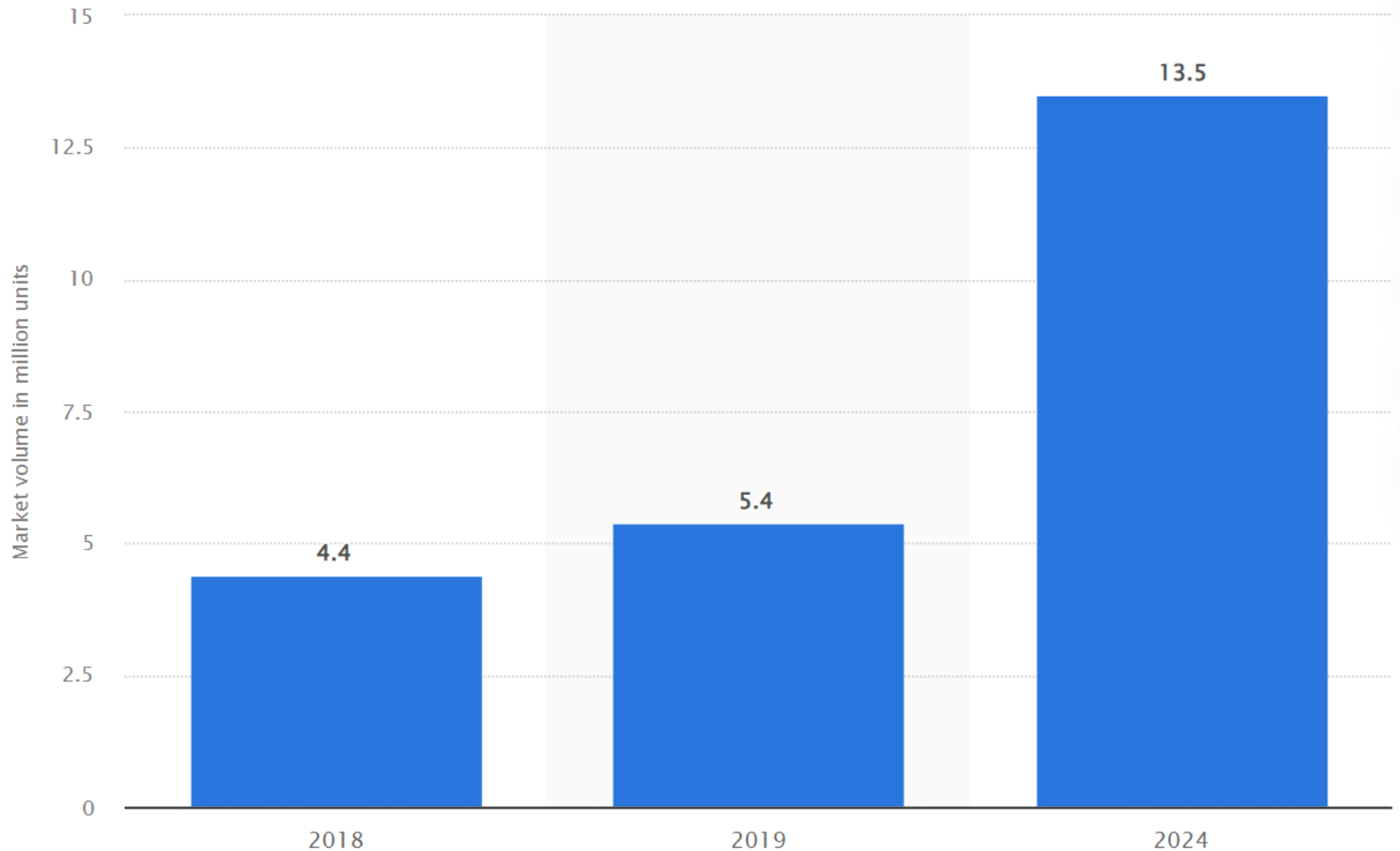
<https://docs.ros.org/>

<https://wiki.ros.org/>

Worldwide User Base



ROS-based robot market volume worldwide between 2018 and 2024 (www.statista.com)





Why ROS?
What is ROS?
How to use ROS?



Thank You!

Any Questions?