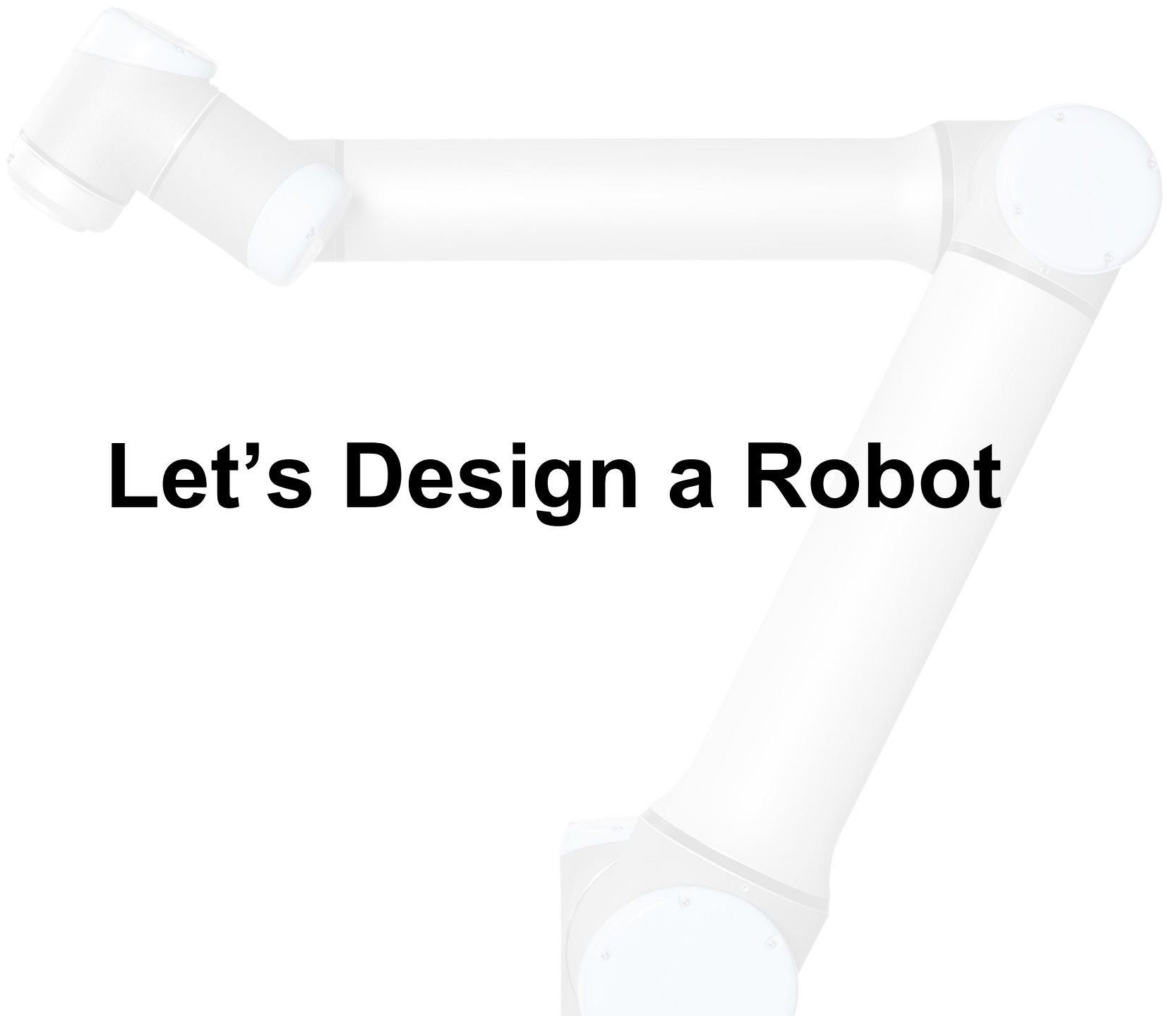




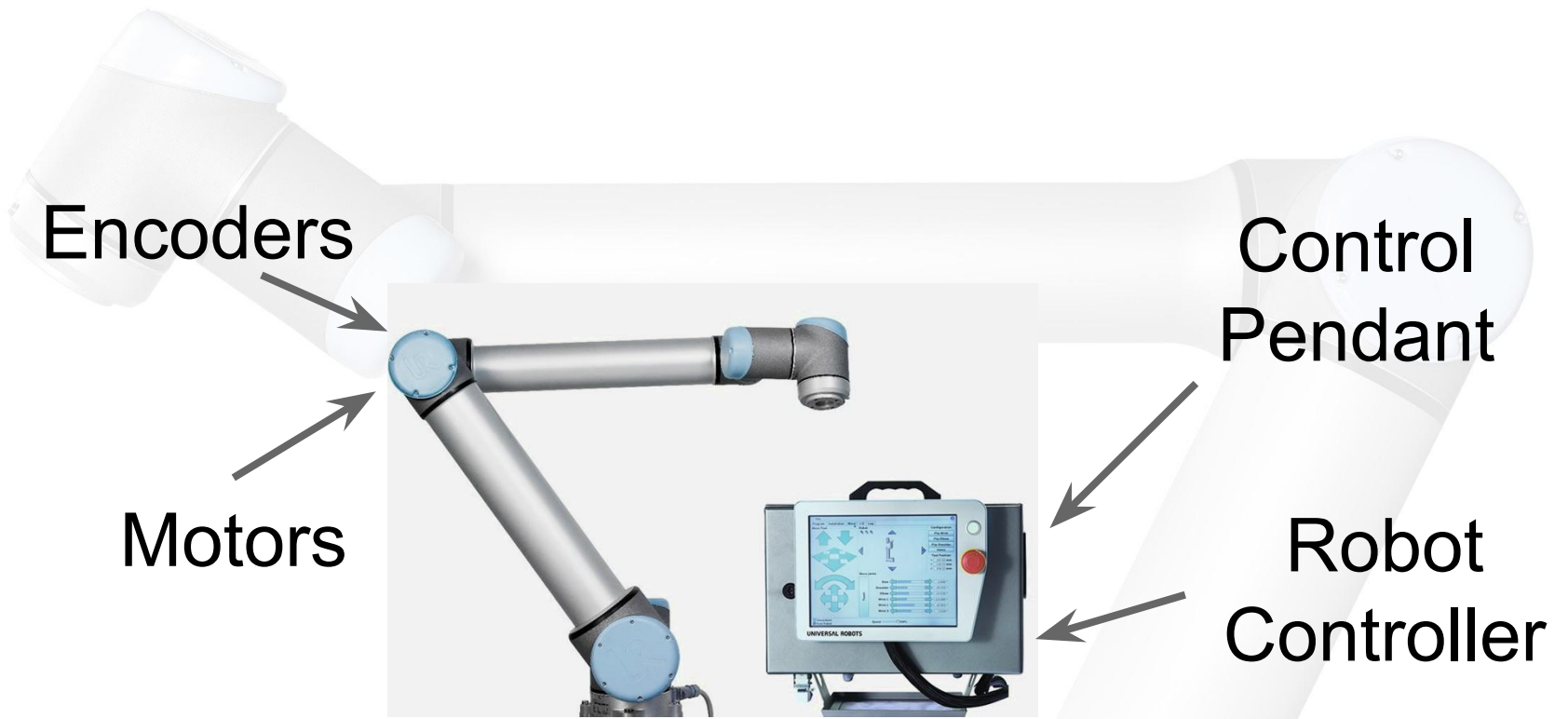
INF3480 - Introduction to Robot Operating System

February 26, 2016

Justinas Mišeikis



Let's Design a Robot



Motor
controllers

Inverse
Kinematics

GUI

Collision
Detection

Trajectory
Planner

PID
Controller

Error
Monitoring

Teaching
Mode



What a mess!

How can we deal with it?

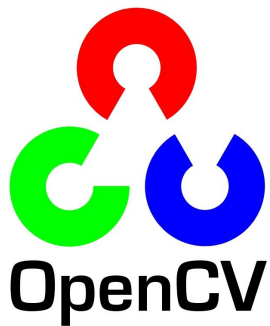


ROS is an open-source, meta-operating system



ROS

ROS is an open-source, meta-operating system



pointcloudlibrary

What's so special about ROS?



- Reusable robotics components!
- 95 Robotic platforms officially support ROS (+33 from this time last year) <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

Current Robotics Job Ads

A relevant degree is required, for instance in Computer Science or Engineering. A background in Robotics/Computer Vision is desirable, **while knowledge of the Robot Operating System (ROS)**, the Point Cloud Library (PCL), or the Open Source Computer Vision Library (OpenCV) is a big plus.

Goal of this PhD is to study, **design and build novel industry-level software based on ROS or ROS-Industry** which is modular, reconfigurable, adaptive, easy to use to integrate and control various robotic systems.

The candidate should be **knowledgeable about C/C++, ROS and matlab/simulink**. Scientific curiosity, large autonomy and ability to work independently are also expected.

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.

Nice To Haves:

- **Experience with Robot Operating System (ROS)**.
- Experience with OpenCV and/or PCL.
- Strong background in machine learning.

Required Skills

- * MSc in Engineering / Computer Science or equivalent.
- * Experience with Robotics
- * **Knowledge about ROS (Robot Operating System) and CV.**
- * Advanced experience with C++ and soft real-time programming.
- * Team spirit and ability to work independently.
- * Excellent communication skills, flexibility and creativity.

ROS

Plumbing

Tools

Capabilities

Ecosystem



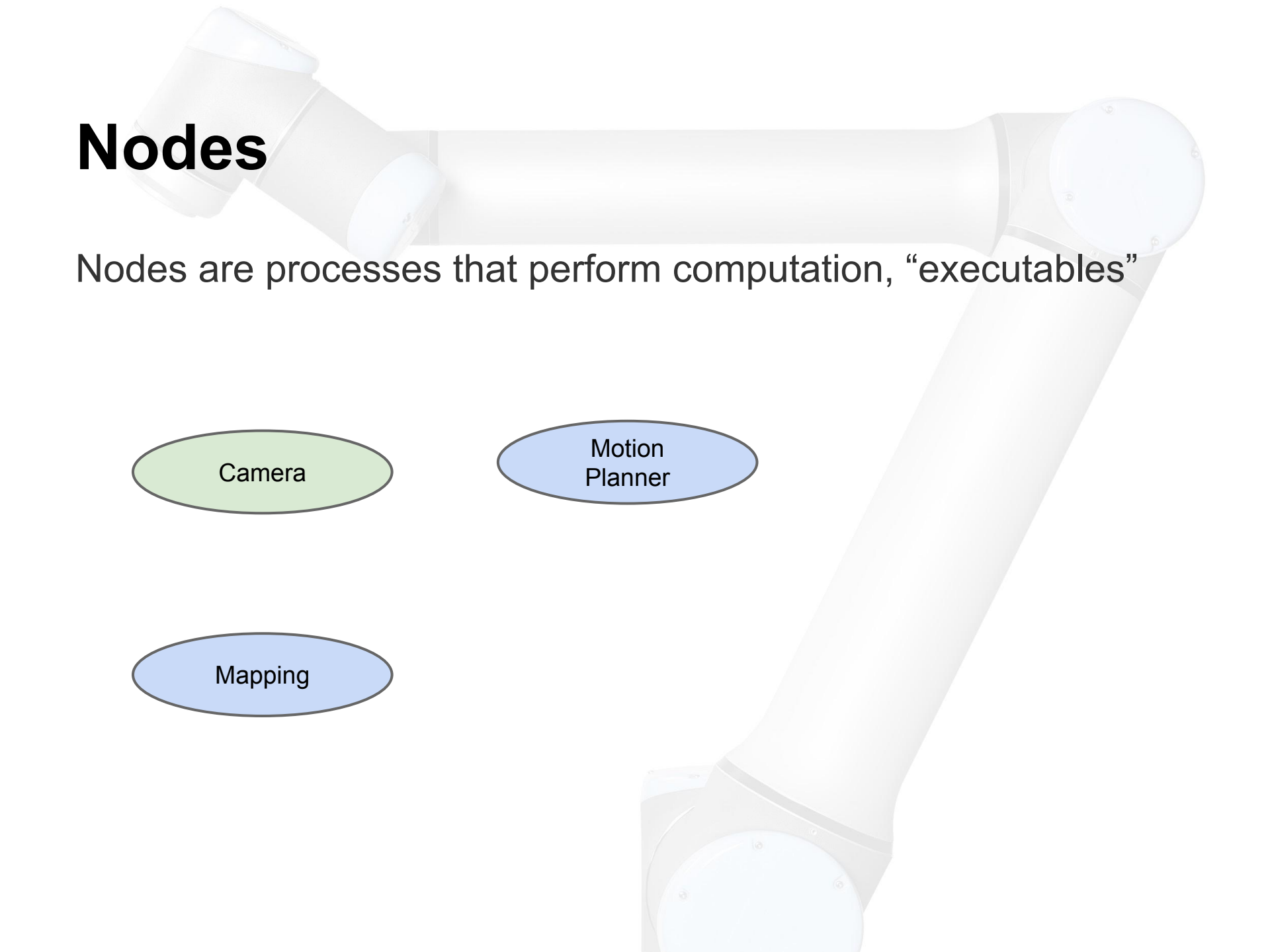
Let's see how it works!

“Plumbing”



Nodes

Nodes are processes that perform computation, “executables”



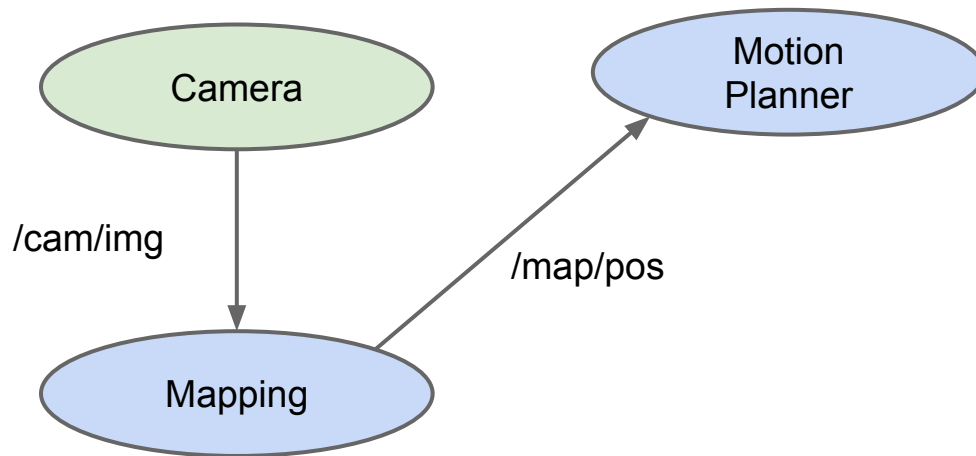
Camera

Motion
Planner

Mapping

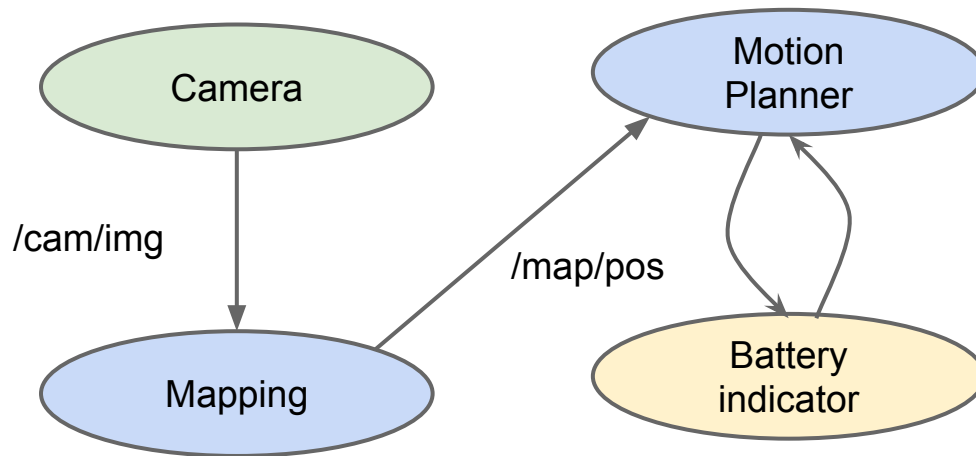
Topics

Topics are streams of data with publish / subscribe semantics.
They are uniquely identifiable by its name



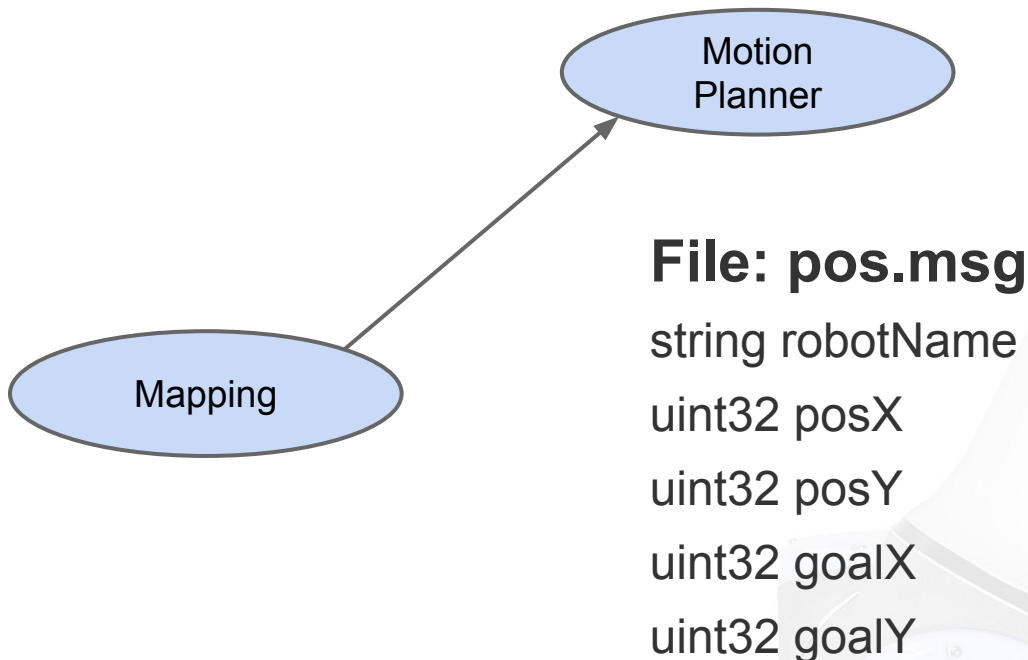
Services

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



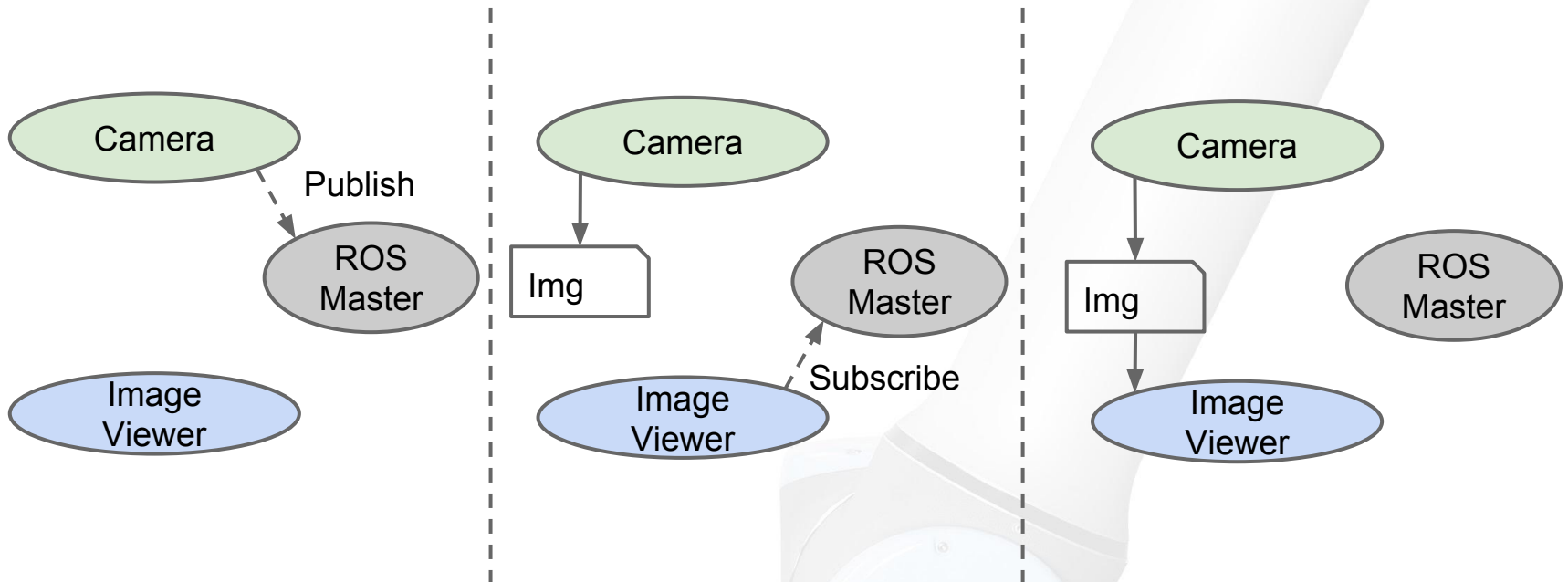
Messages

A message is simply a data structure, comprising typed fields.
Language agnostic data representation. C++ can talk to Python.



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.

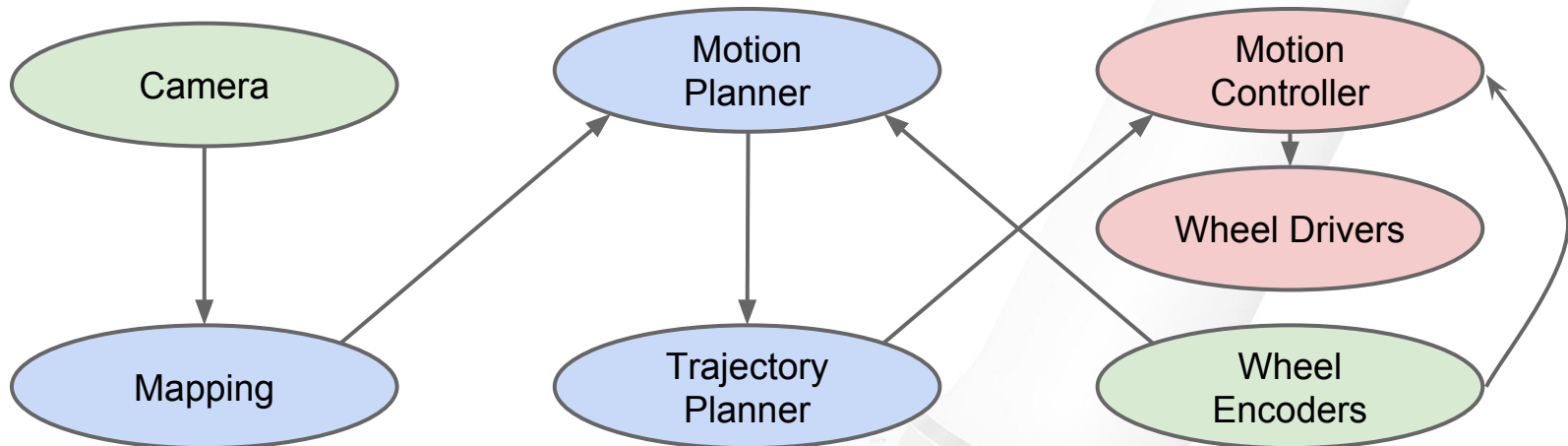


Example System - Mobile Robot

Green - Sensors

Blue - Planning algorithms

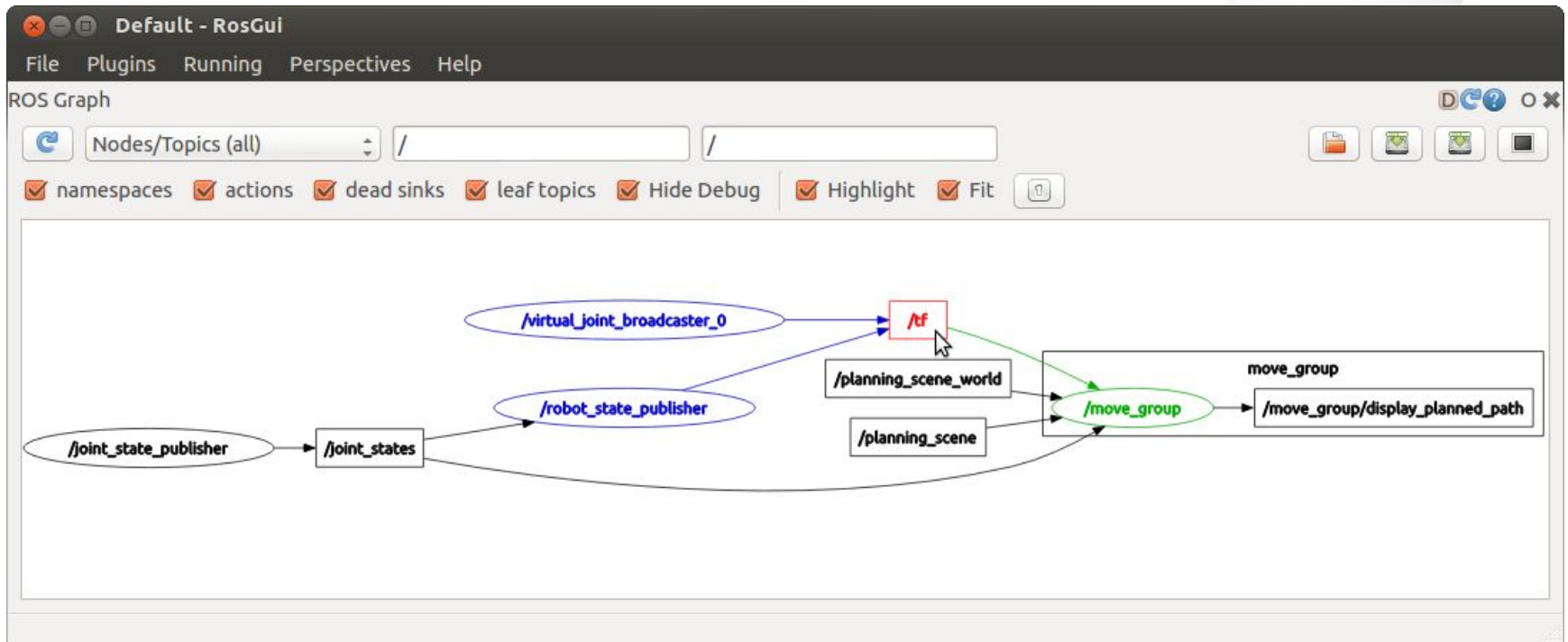
Red - Hardware integration



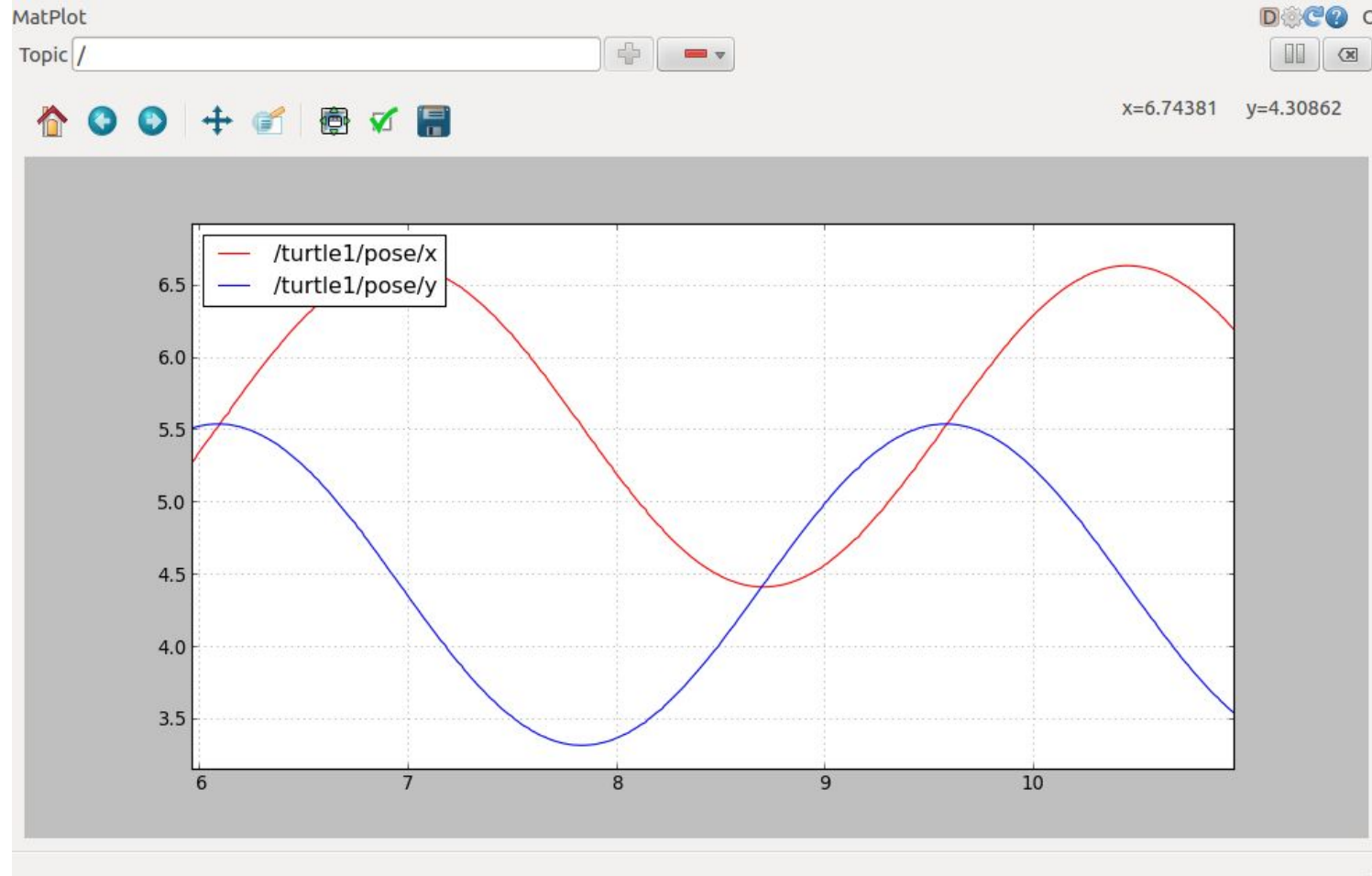


“Tools”

System Visualisation: rqt_graph



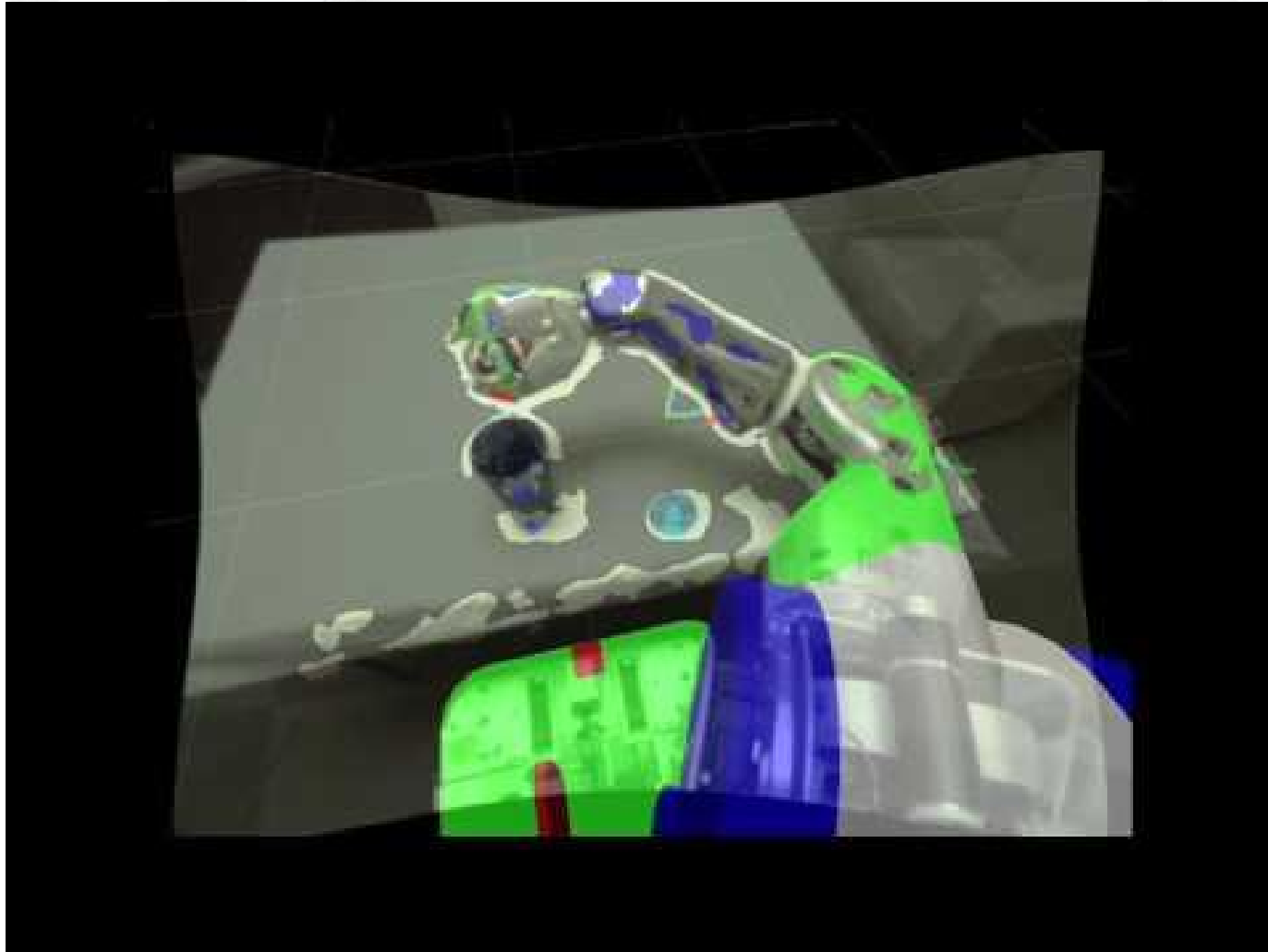
Live Plotting: rqt_plot



Logging and Visualization Sensor Data: rosbag and rqt_bag

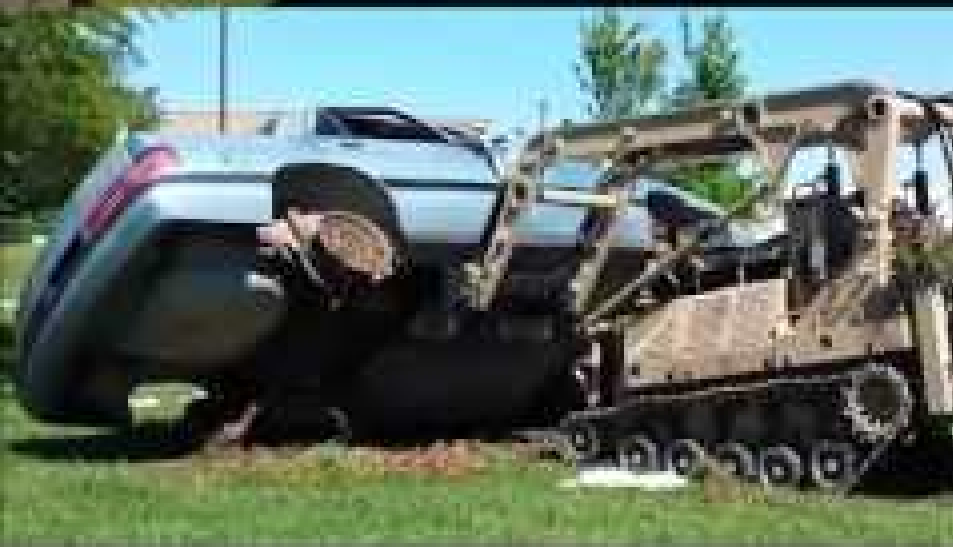


3D Visualisation: RVIZ





“Capabilities”



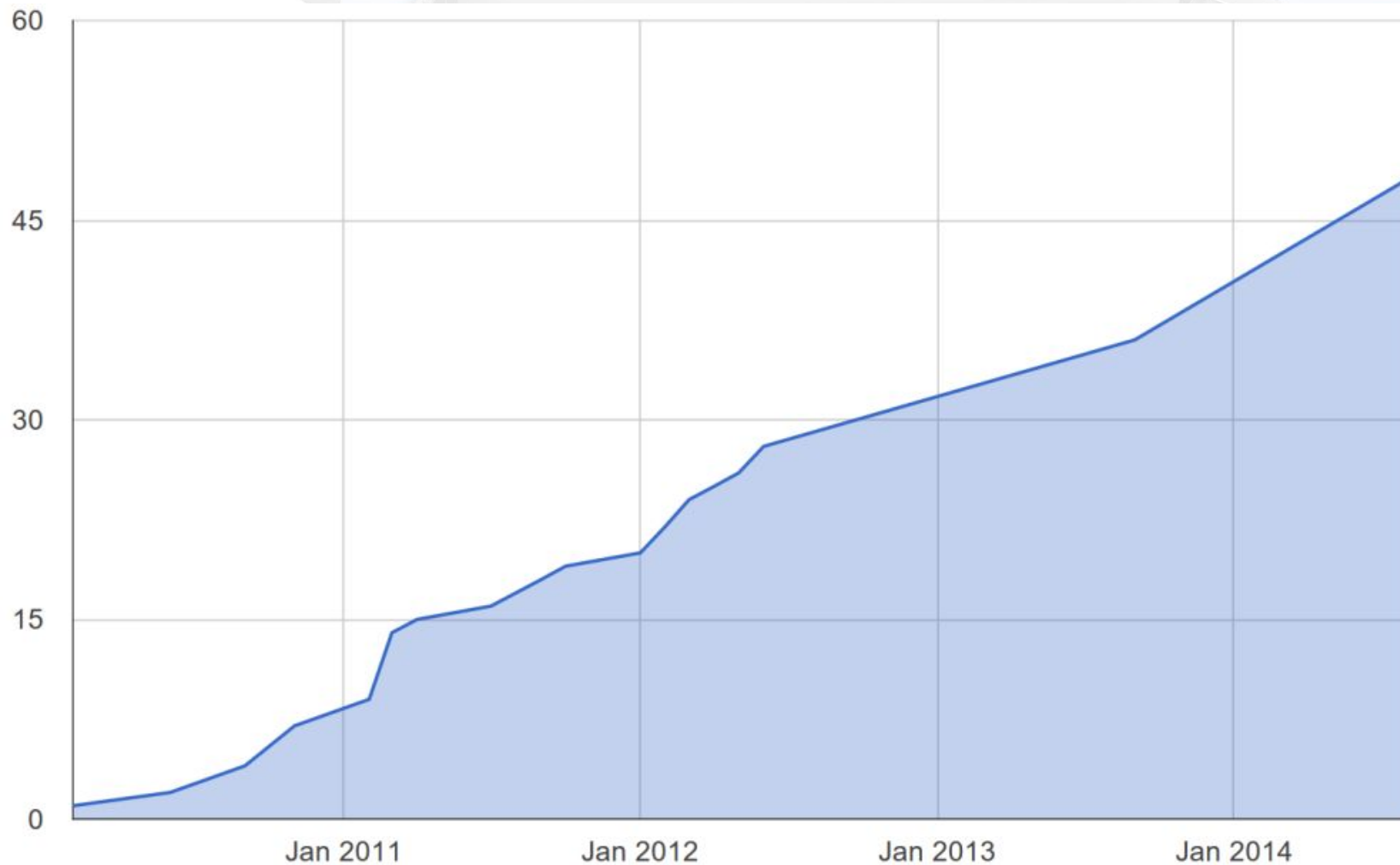


“Ecosystem”

Worldwide User Base



Number of Robots Supporting ROS





 **MoveIt!**

The MoveIt! logo is positioned in the lower half of the image. It consists of a blue icon on the left, which is a stylized representation of a robotic gripper or a similar mechanical component. To the right of the icon, the word "MoveIt!" is written in a bold, sans-serif font. The "Move" part is in a dark grey color, and the "It!" part is in a bright blue color.

Technical Capabilities



- Motion Planning
 - Fast and good quality paths
 - Kinematic Constraints
- Fast and flexible collision checking
- Integrated Kinematics
- Integrated Perception for Environment Representation
- Standardised Interfaces to Controllers
- Execution and Monitoring
- Kinematic Analysis
- Simulated Robots



Motion Planning

MoveIt! includes a variety of motion planners:

- Sampling-based motion planners
(implementations from [OMPL])
- Search-based motion planners
(implementations from SBPL)
- Optimization-based motion planners (CHOMP)

Motion Planning - 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.



Collision Detection

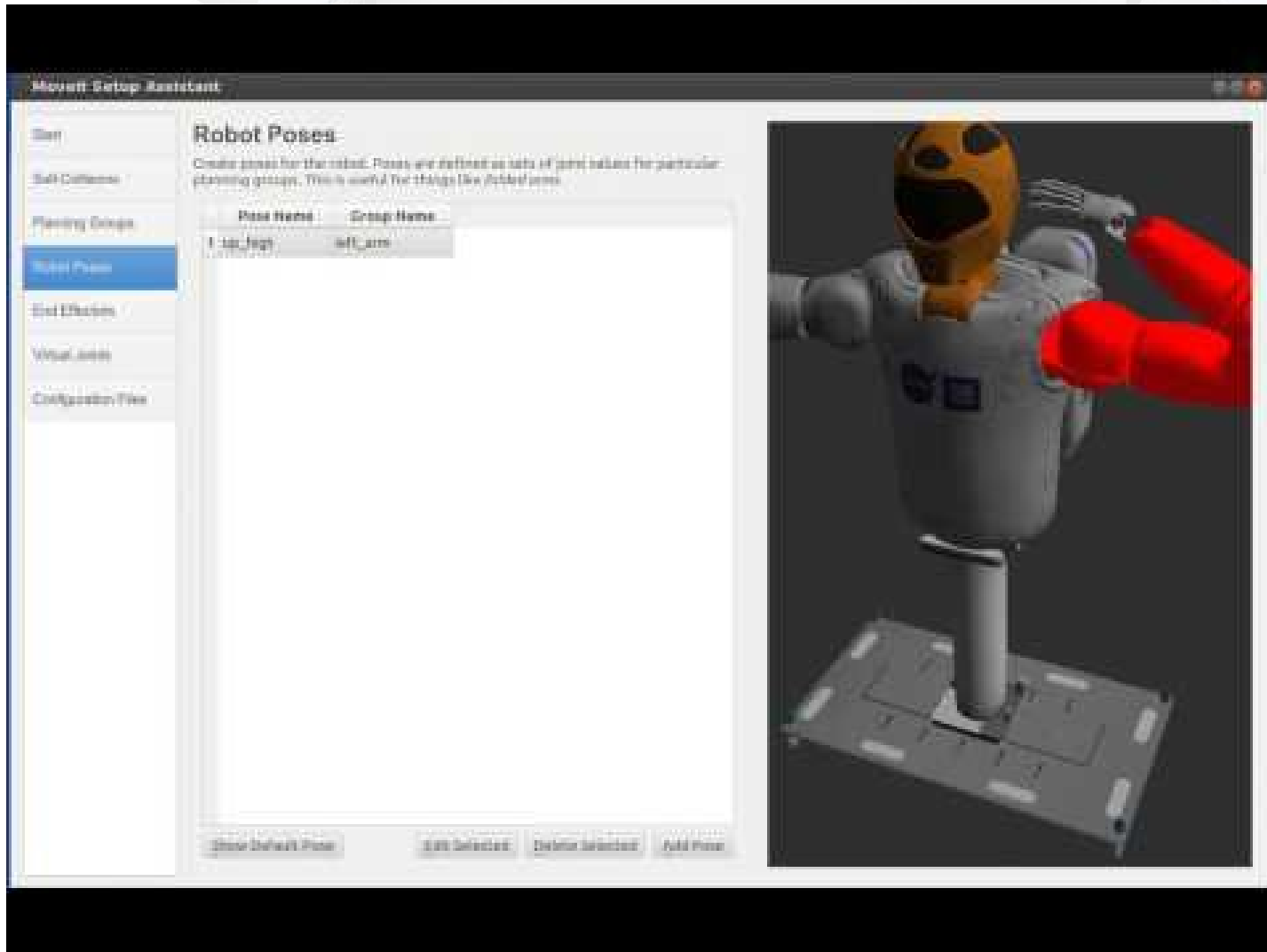
Flexible Collision Library (FCL) is used.

Types of objects supported:

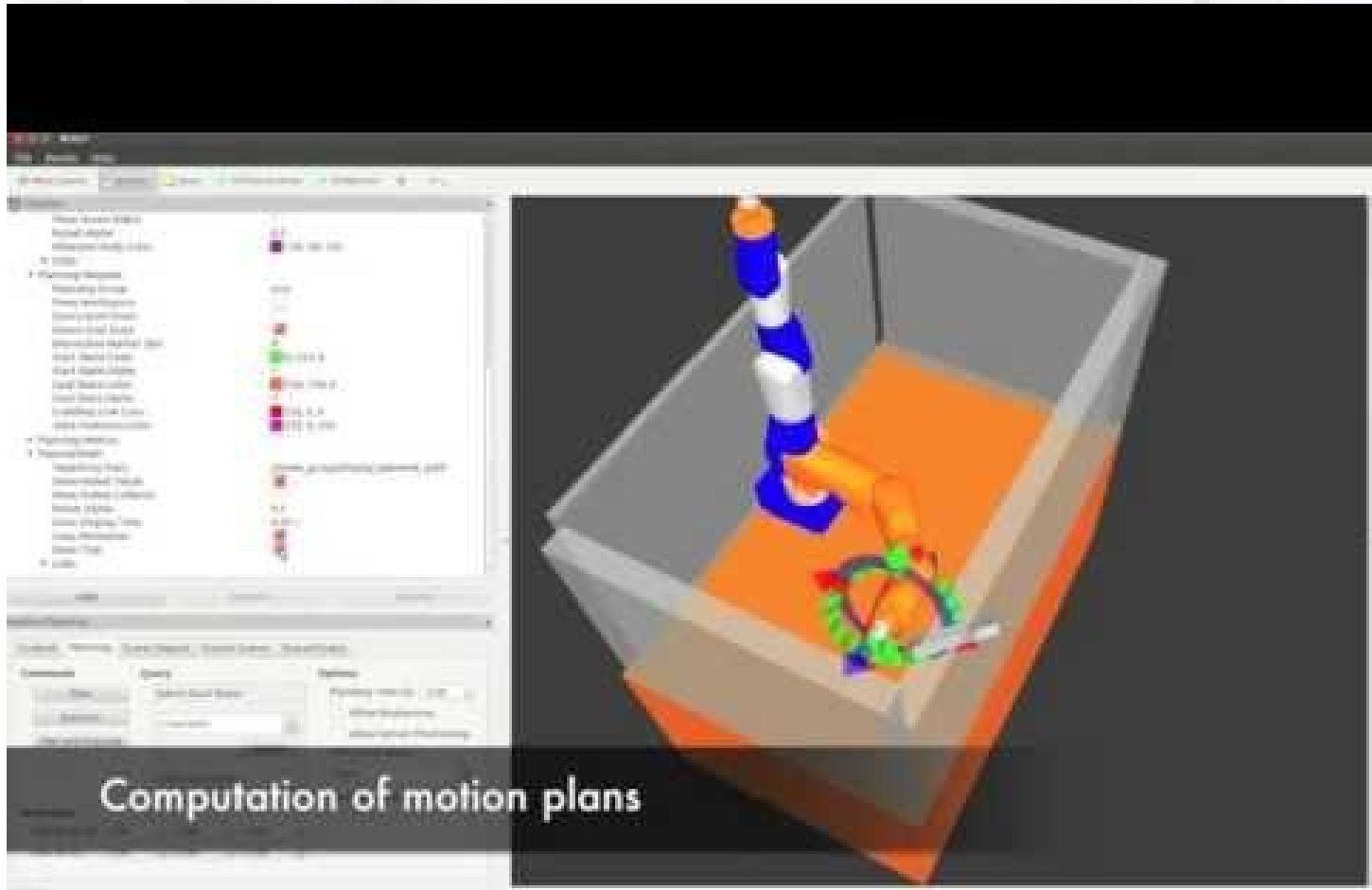
- Meshes
- Primitive shapes (boxes, cylinders, cones)
- Octomap

40,000 to 80,000 collision checks per second!

Movelt - Robot Setup Assistant



Movelt Capabilities



How to use it?



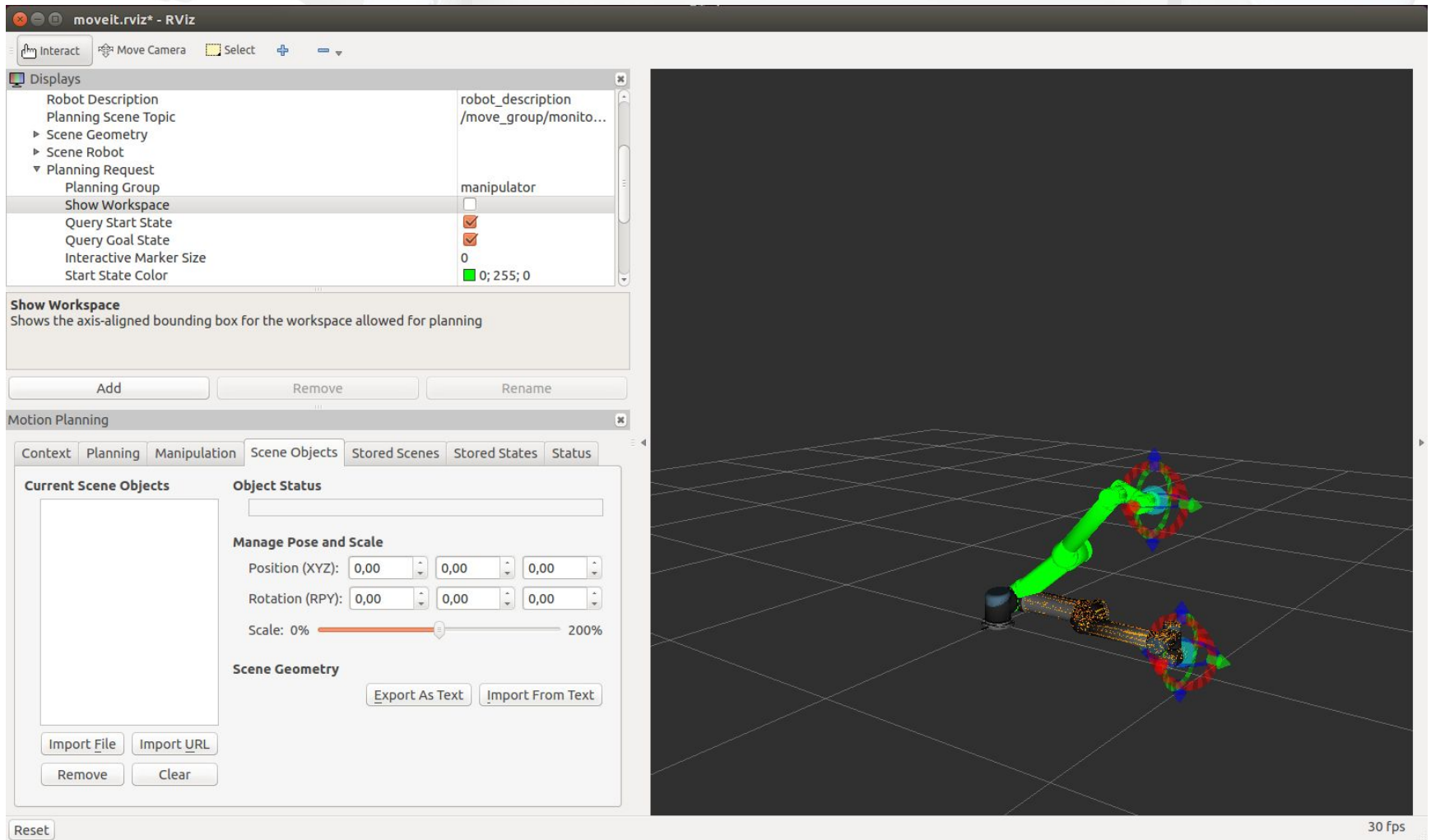
To simulate and play around with Universal Robot UR5

- 1) Have ROS installed (current version: Indigo)
- 2) Install MoveIt for UR5:
 - “sudo apt-get install ros-indigo-ur5-moveit-config”
- 3) Launch UR5 RViz simulator:
 - “roslaunch ur5_moveit_config demo.launch”

That's it!

http://wiki.ros.org/universal_robot/Tutorials/Getting%20Started%20with%20a%20Universal%20Robot%20and%20ROS-Industrial

How to use it?



The screenshot displays the MoveIt! RViz interface. The main window shows a 3D visualization of a robot arm in a workspace. The interface is divided into several panels:

- Displays:** A tree view on the left showing the hierarchy of visual elements. The right pane shows the configuration for the selected 'manipulator' display, including 'robot_description' and '/move_group/monito...'. Checkmarks are visible for 'Query Start State' and 'Query Goal State'. The 'Start State Color' is set to green (0; 255; 0).
- Show Workspace:** A panel below the displays that shows the axis-aligned bounding box for the workspace allowed for planning.
- Motion Planning:** A panel with tabs for Context, Planning, Manipulation, Scene Objects, Stored Scenes, Stored States, and Status. The 'Scene Objects' tab is active, showing controls for 'Current Scene Objects', 'Object Status', 'Manage Pose and Scale' (with Position (XYZ) and Rotation (RPY) fields and a Scale slider), and 'Scene Geometry' (with 'Export As Text' and 'Import From Text' buttons).
- Buttons:** 'Add', 'Remove', and 'Rename' buttons are located below the 'Show Workspace' panel. 'Import File', 'Import URL', 'Remove', and 'Clear' buttons are located below the 'Scene Objects' panel. A 'Reset' button is at the bottom left.

The 3D view shows a red robot arm with a blue base, positioned on a grid floor. The robot's end effector is highlighted with a red and blue bounding box.



Future?

ROS 2

coming soon...



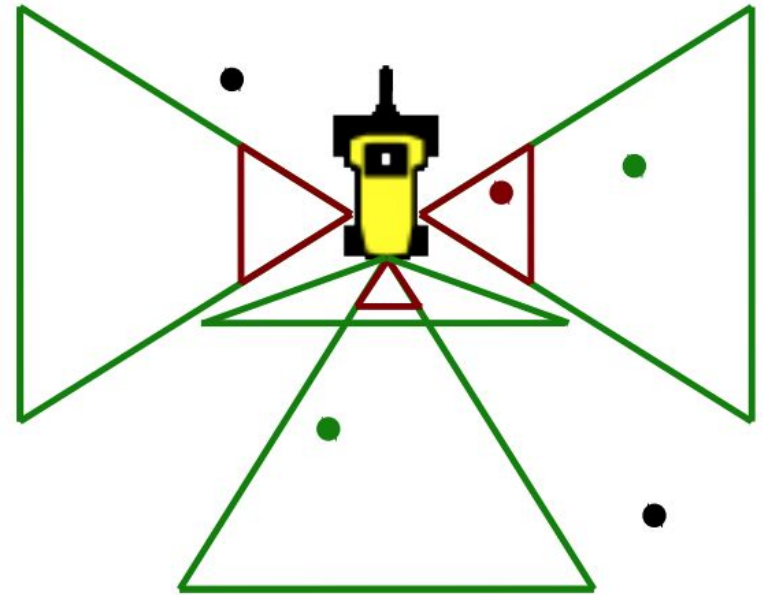
Actual Project:

People detection for HMC

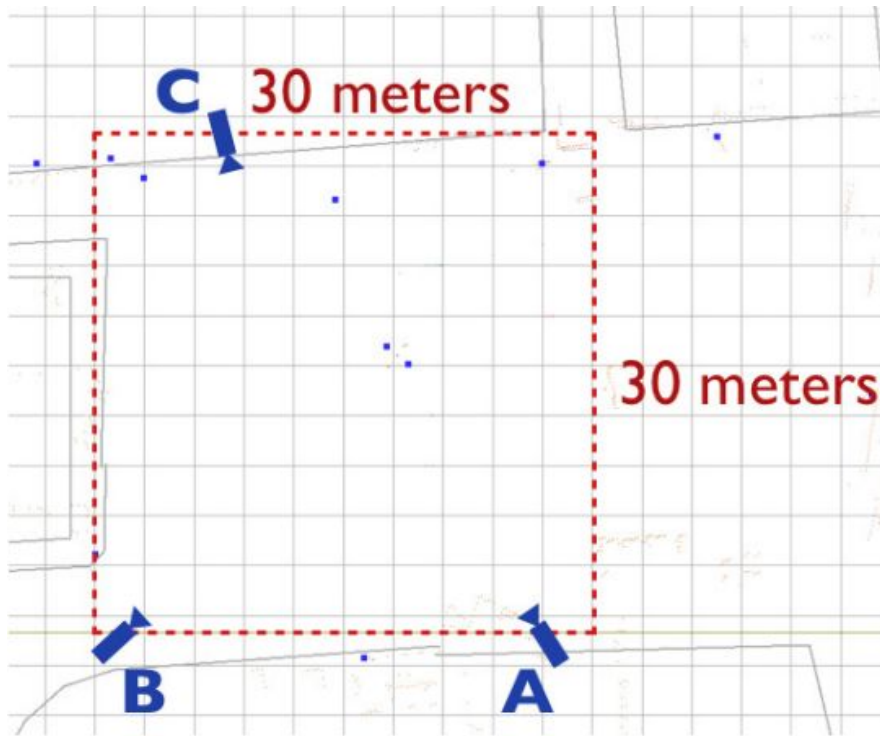
Joint Human Detection From Static and Mobile Cameras

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6894232&tag=1

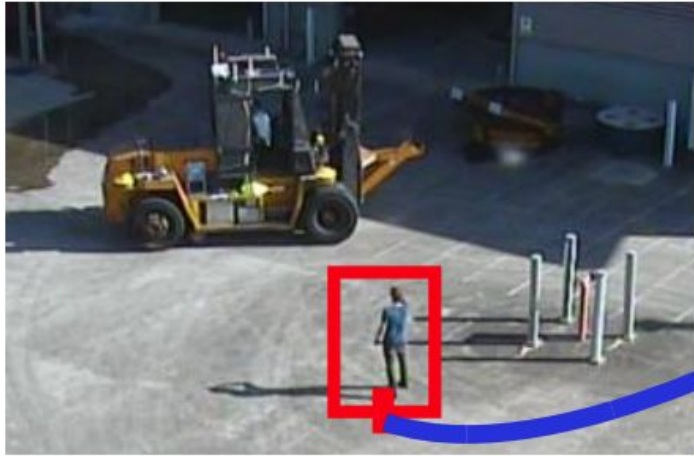
Autonomous Hot Metal Carrier (HMC)



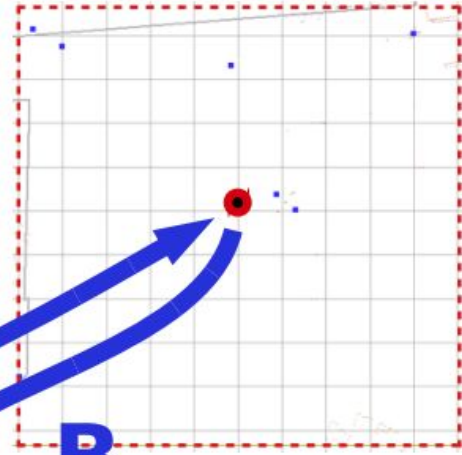
Workspace



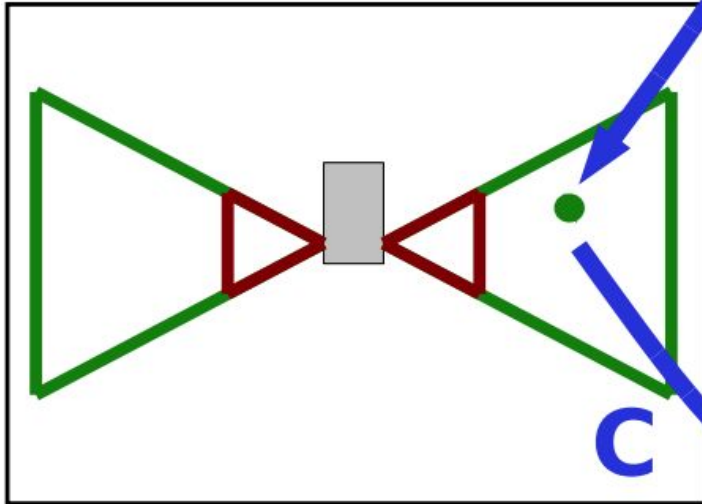
Detection Process



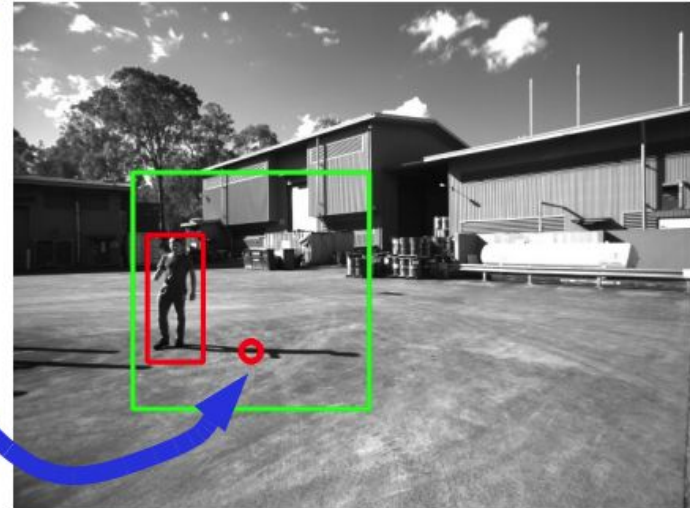
A



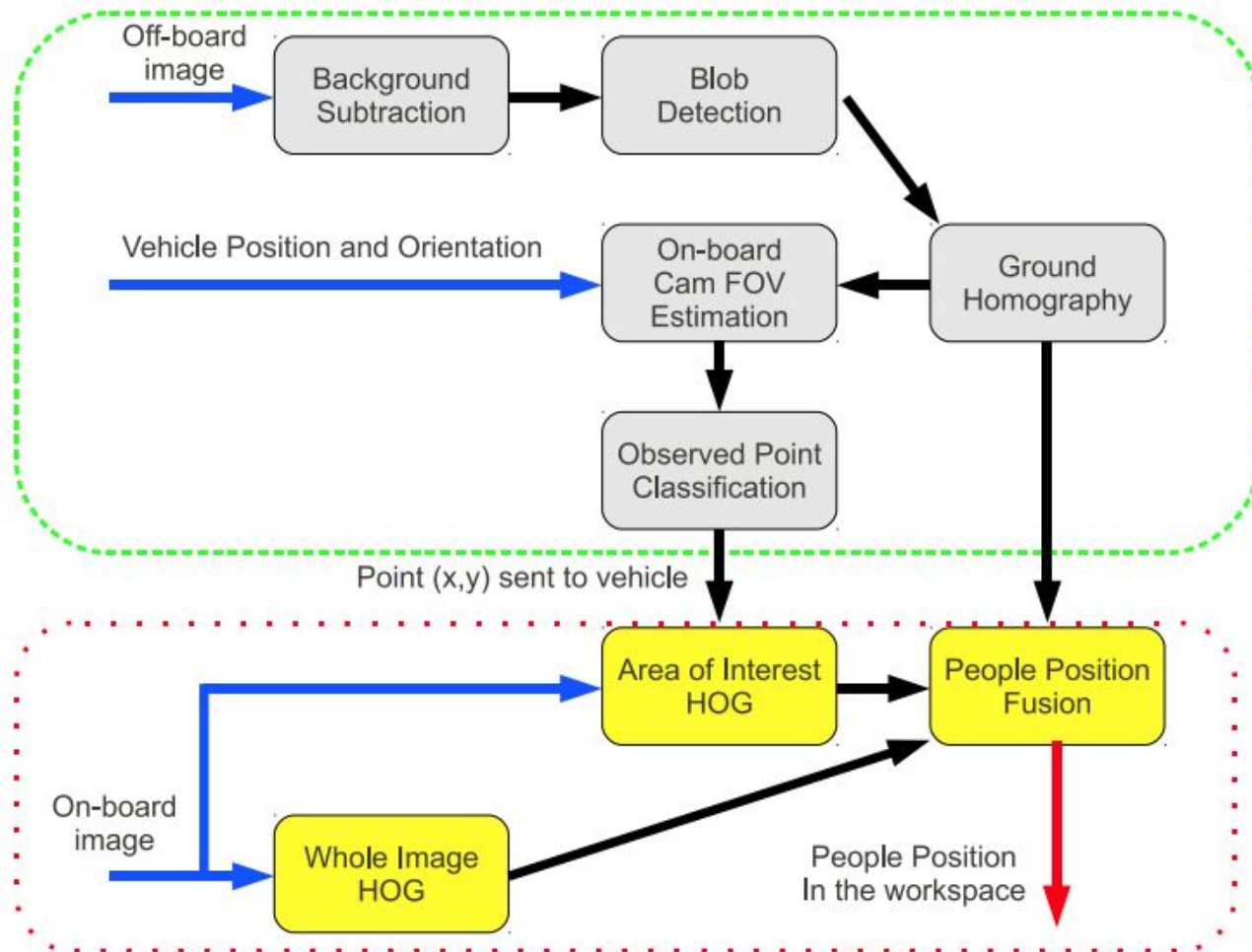
B



C



Algorithm Diagram



Summary



- ROS is a meta-operating system for robotics
- Provides basic (and many!) algorithms for robotics
- Modular approach allows easy adaptation to hardware changes and both hw and sw updates
- Effective visualisation and simulation tools
- World-wide spread in research and commercial use
- BSD license - open source, free to use!
- Over 90 robot platforms support ROS, and growing!
- Easy to start
- Linux based, best works on Ubuntu
- Easy to parallelise, nodes based approach communicate over TCP and can be synchronised using timestamps for messages

Useful URLs



- <http://www.ros.org/> - ROS homepage
- <http://www.ros.org/is-ros-for-me/> - Is ROS for me?
- <http://wiki.ros.org/ROS/Installation/TwoLineInstall>
- <http://moveit.ros.org/> - MoveIt
- <http://wiki.ros.org/rviz> - RViz
- <http://nootrix.com/downloads/> - ROS virtual machine
- <http://opencv.org/> - OpenCV
- <http://pointclouds.org/> - Point Cloud Library