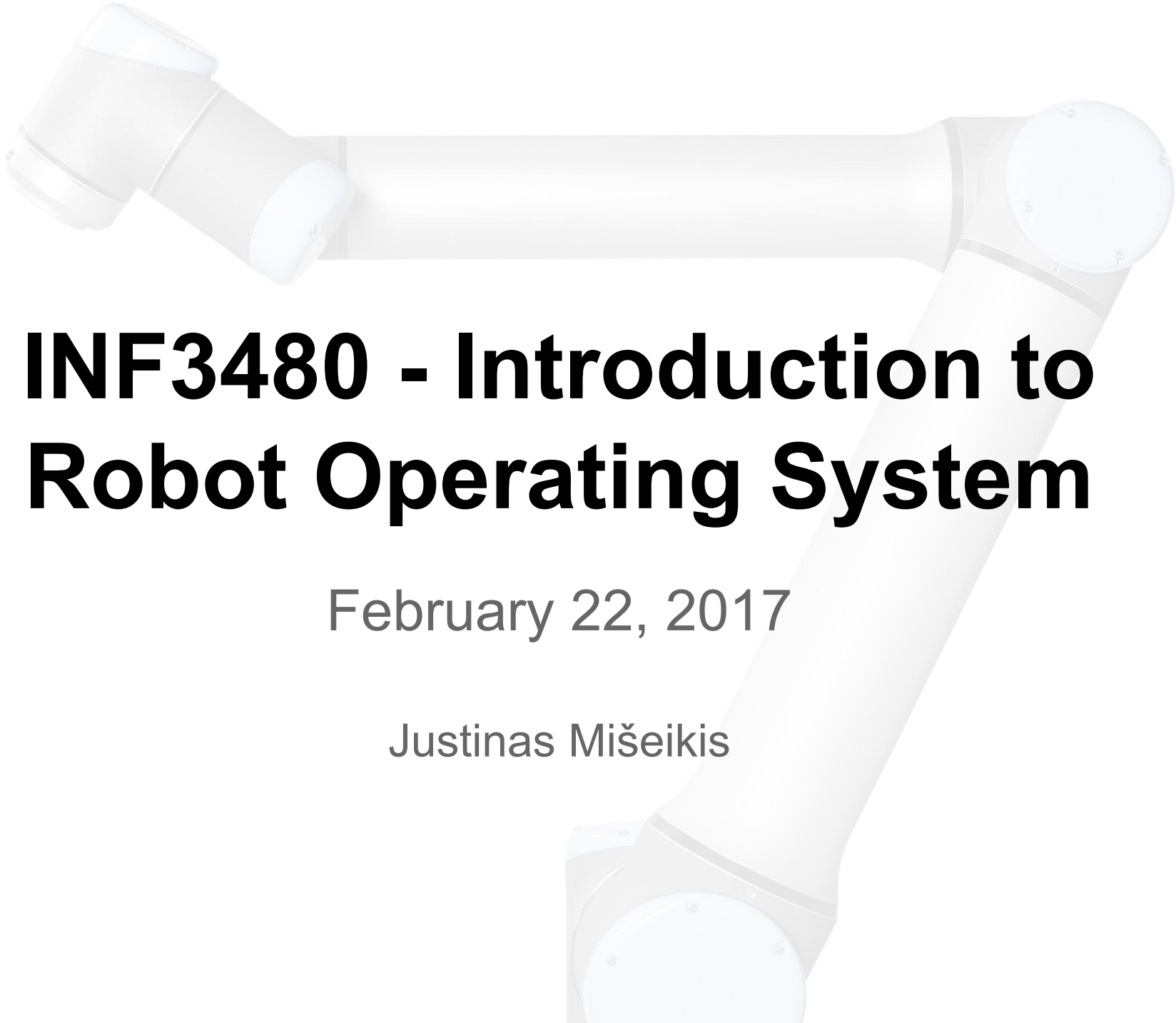


INF3480 - Introduction to Robot Operating System

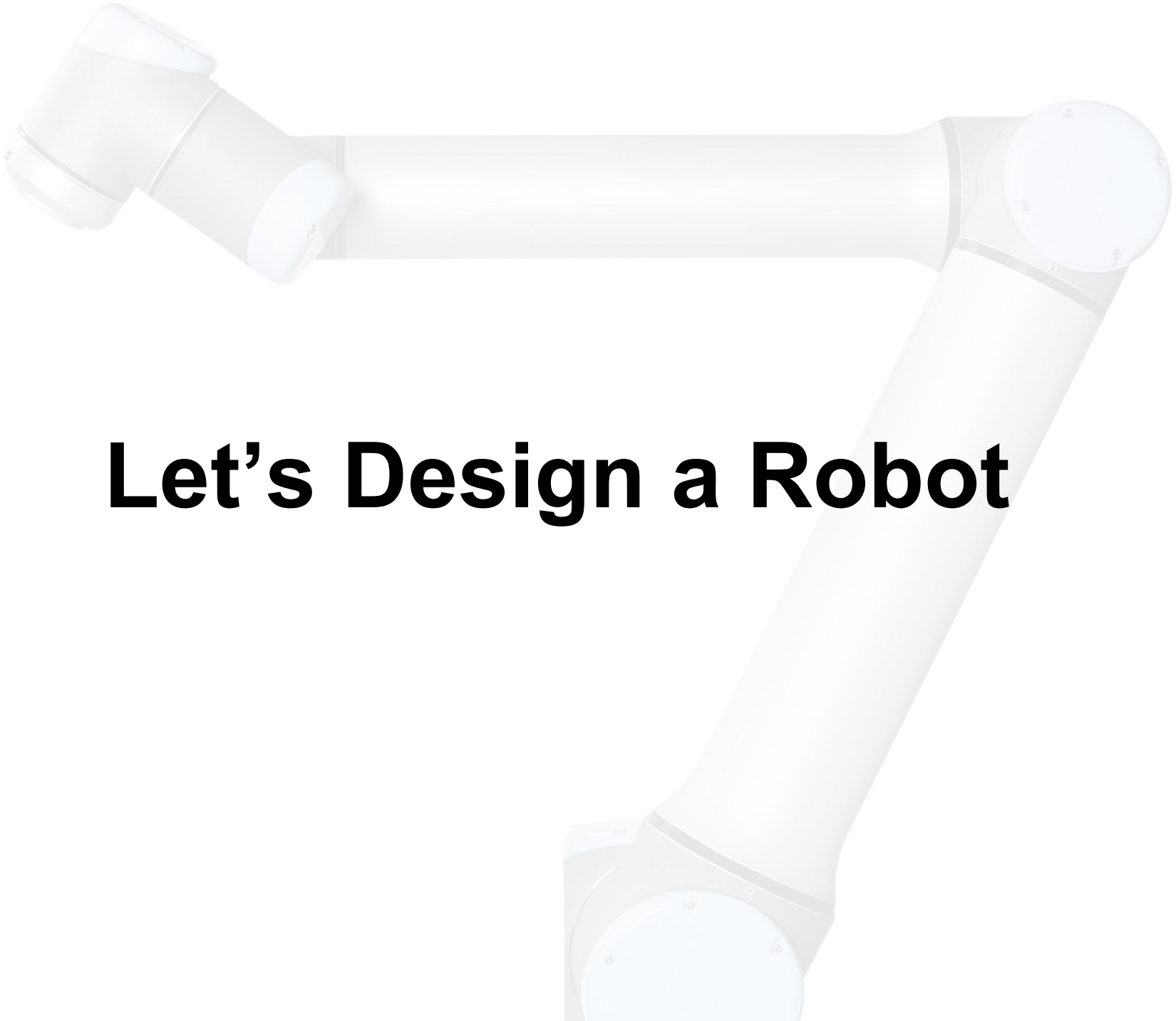
February 22, 2017

Justinas Mišeikis

Side Note

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

Study Material



Let’s Design a Robot

Encoders

Motors

Control Pendant

Robot Controller

Motor controllers

Inverse Kinematics

GUI

Collision Detection

Trajectory Planner

PID Controller

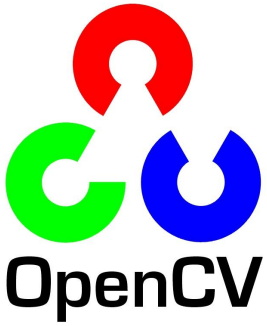
Error Monitoring

Teaching Mode

What a mess!

How can we deal with it?

ROS is an open-source, meta-operating system



pointcloudlibrary

What's so special about ROS?

- Reusable robotics components!
- 120+ Robotic platforms officially support ROS (+25 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
  - Experimental versions for OS X, Android, Arch Linux, Debian, OpenEmbedded/Yocto
- 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.

** Job Requirements **
- Experience with software development and integration preferably in a Linux environment
- Proficient in C, C++, Python and Matlab
- Those with past experience using ROS will be preferred

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.

** Required Qualifications **
- A university degree (Master or Diplom-Ingenieur)
- Proven programming skills in Matlab, C / C ++, ROS
- Good knowledge of cooperative software development with GIT or SVN

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

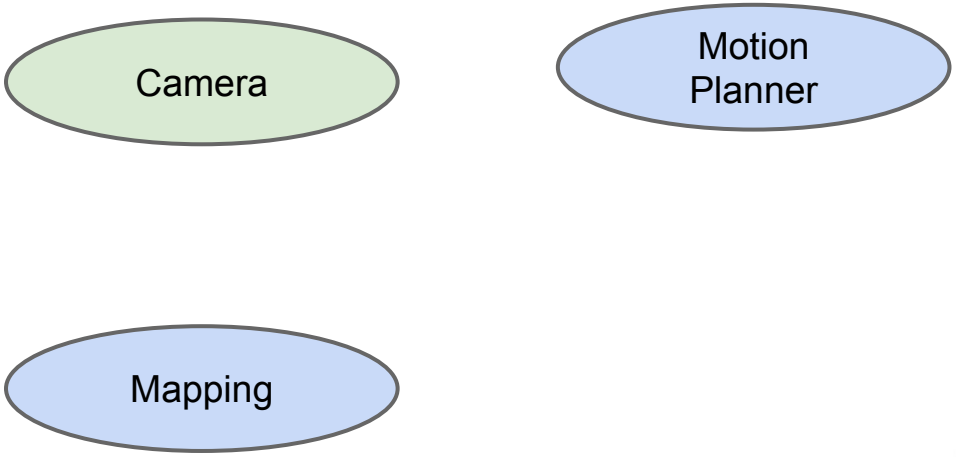
Let’s see how it works!

“Plumbing”

Nodes

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.



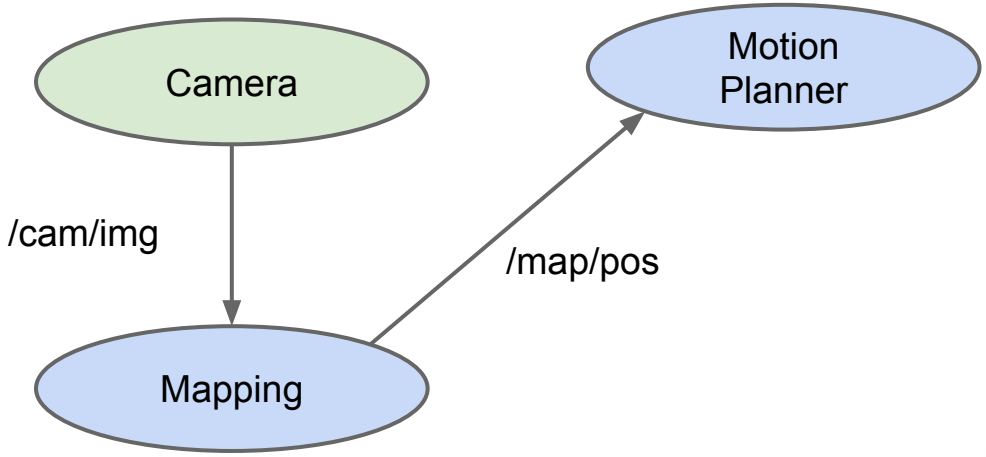
Study Material - highly recommend to check the URL

Topics

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

Topics are streams of data with publish / subscribe semantics.

They are uniquely identifiable by its name. Nodes can publish and subscribe to topic in order to transfer data.

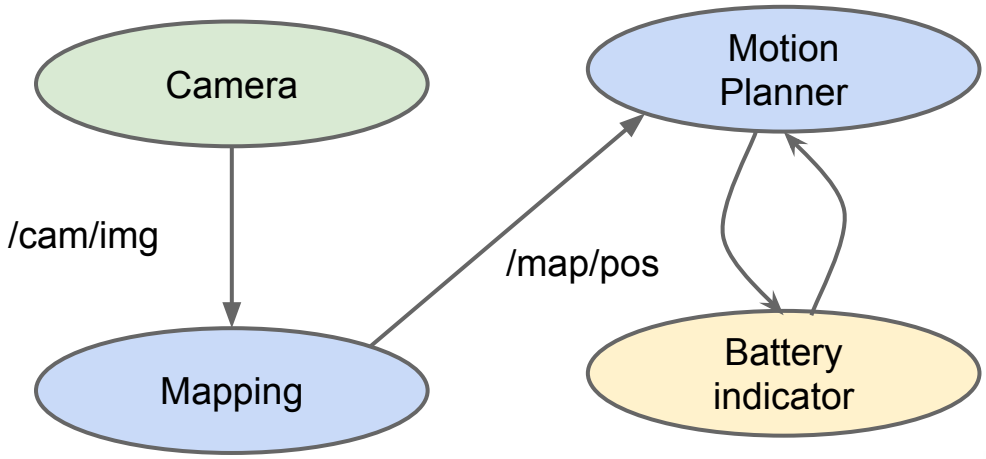


Study Material - highly recommend to check the URL

Services

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

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



Study Material - highly recommend to check the URL

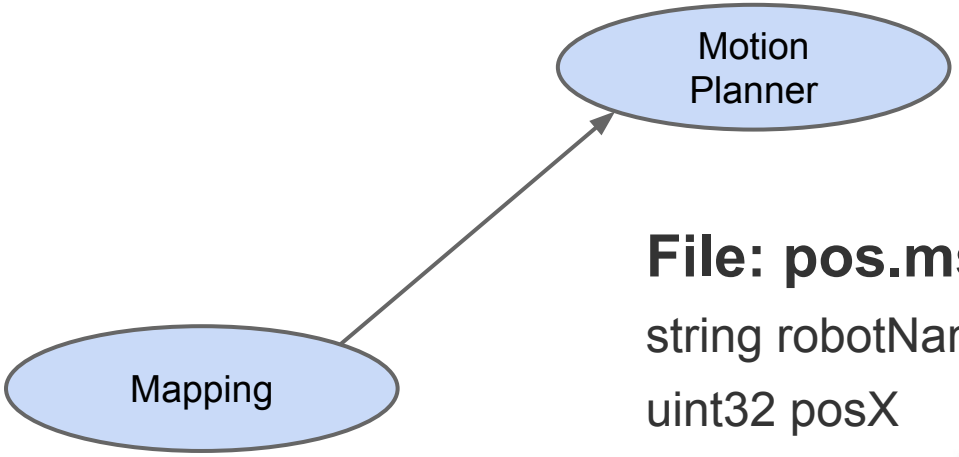
Messages

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

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

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

Messages are sent on defined topics.

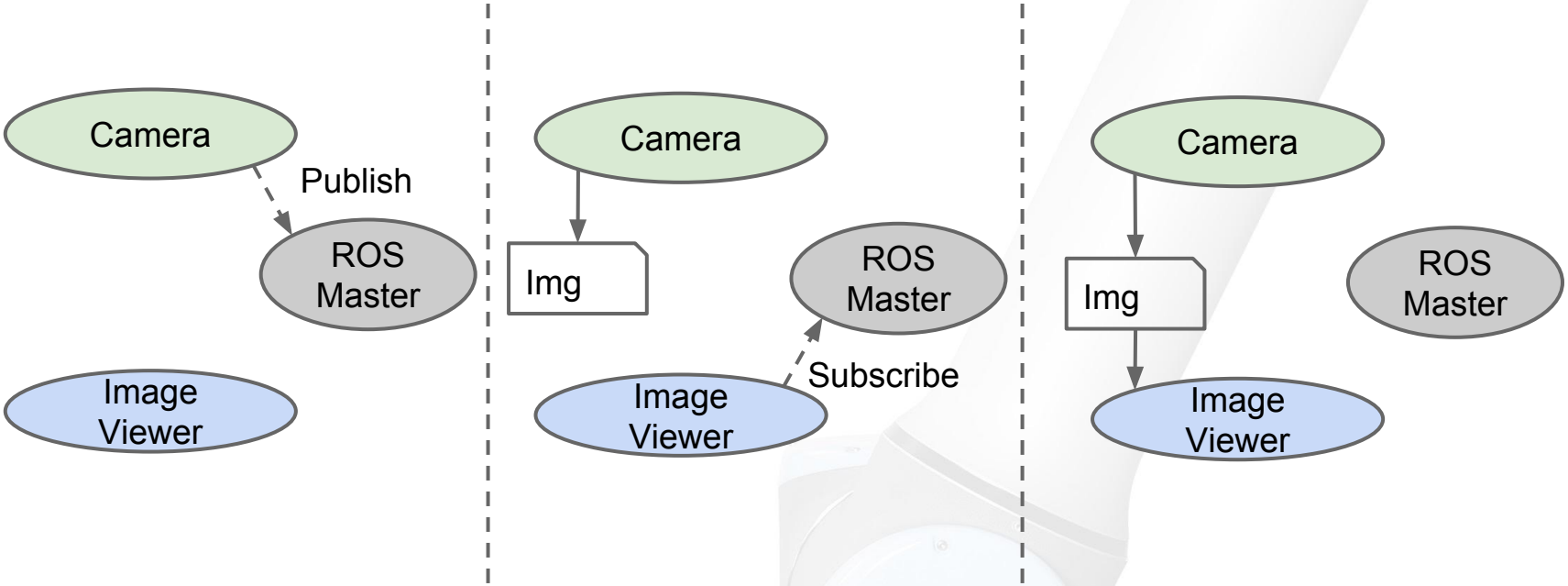


File: pos.msg

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.

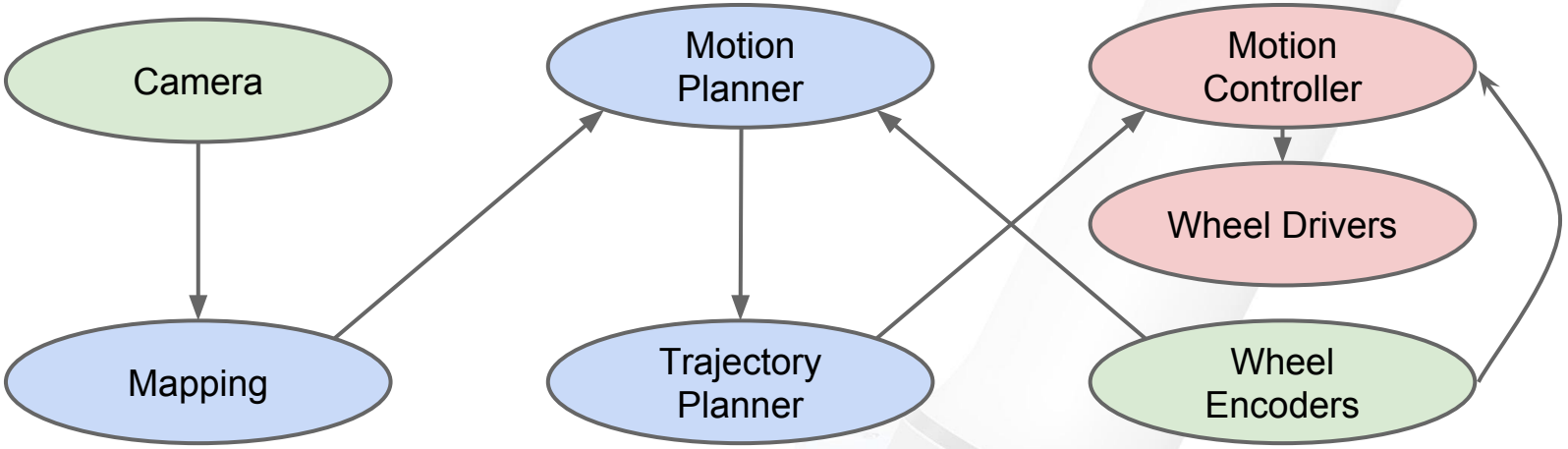


Example System - Mobile Robot

Green - Sensors

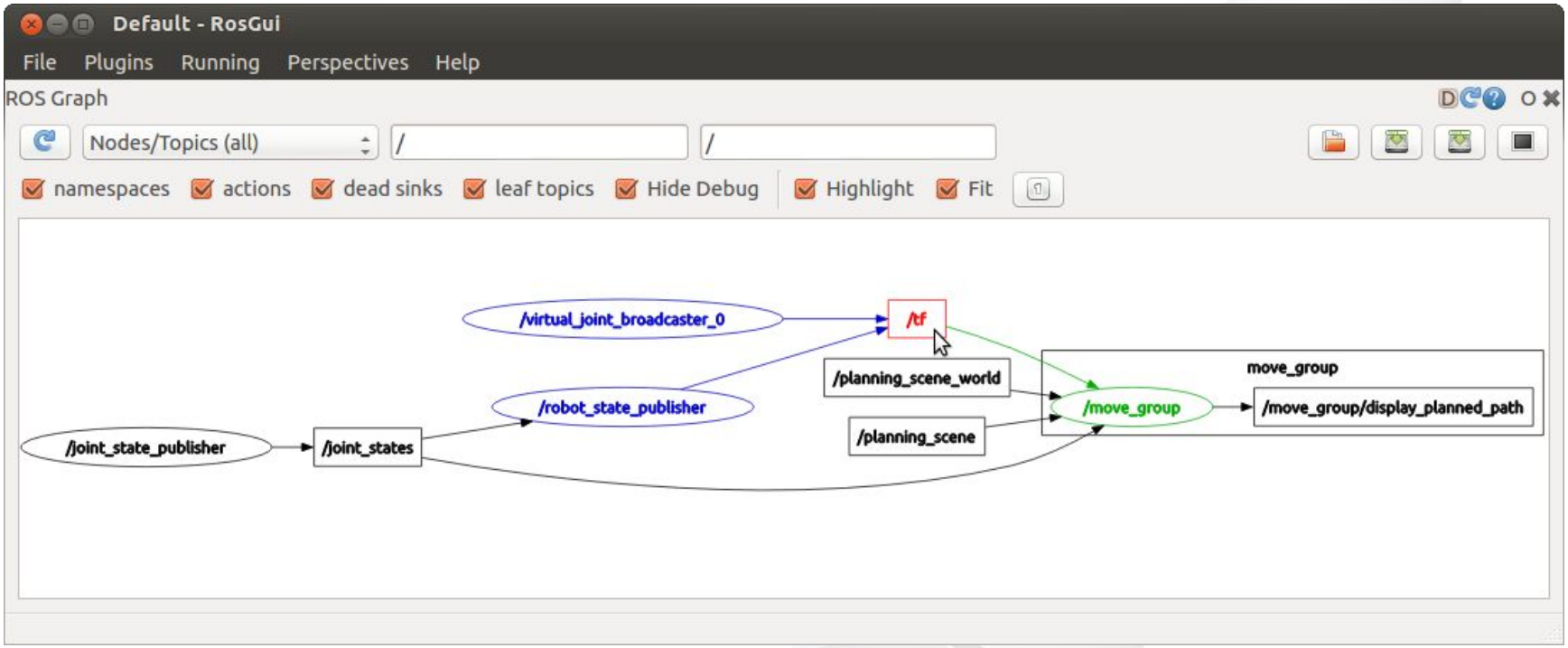
Blue - Planning algorithms

Red - Hardware integration

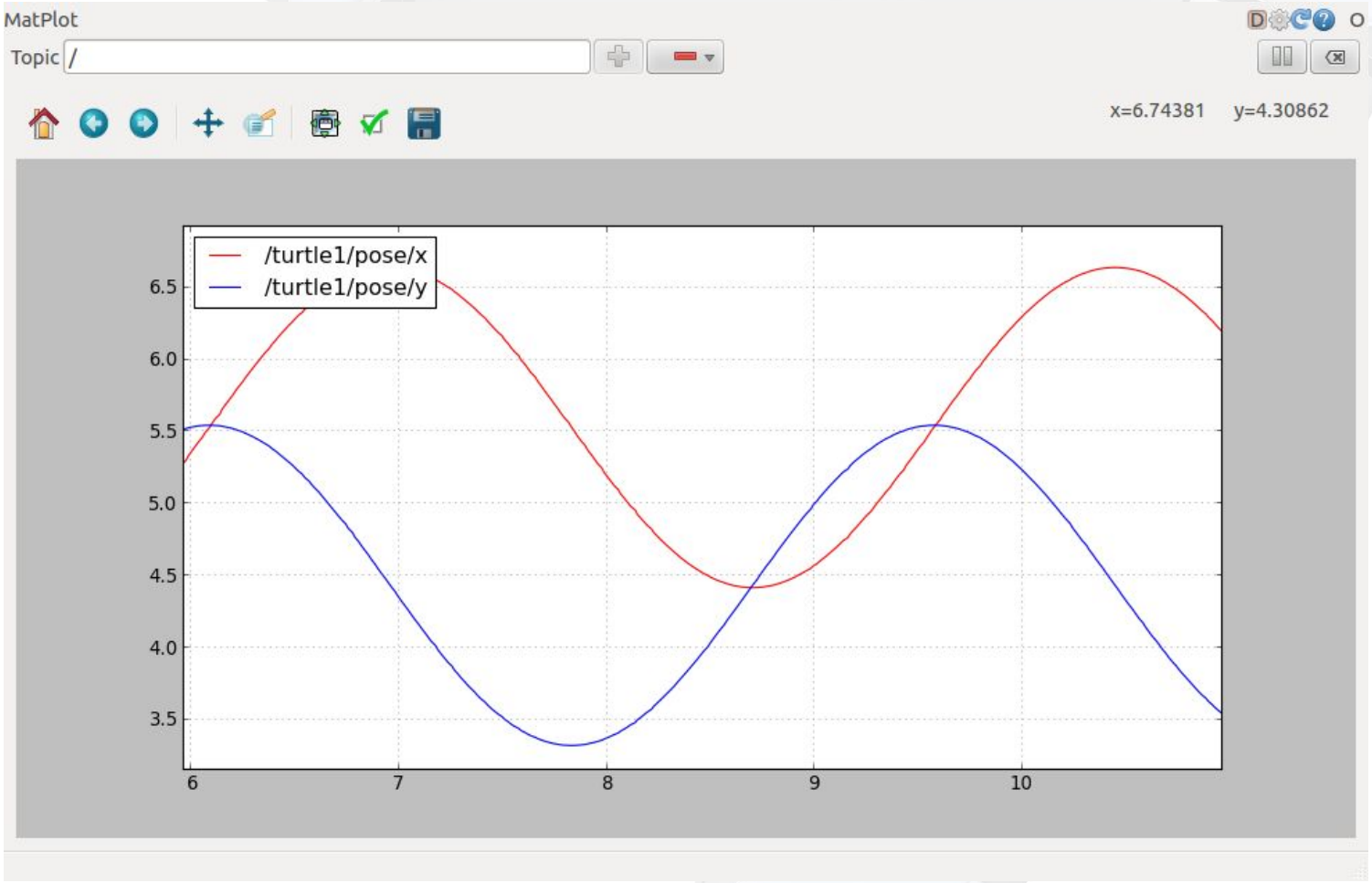


“Tools”

System Visualisation: rqt_graph

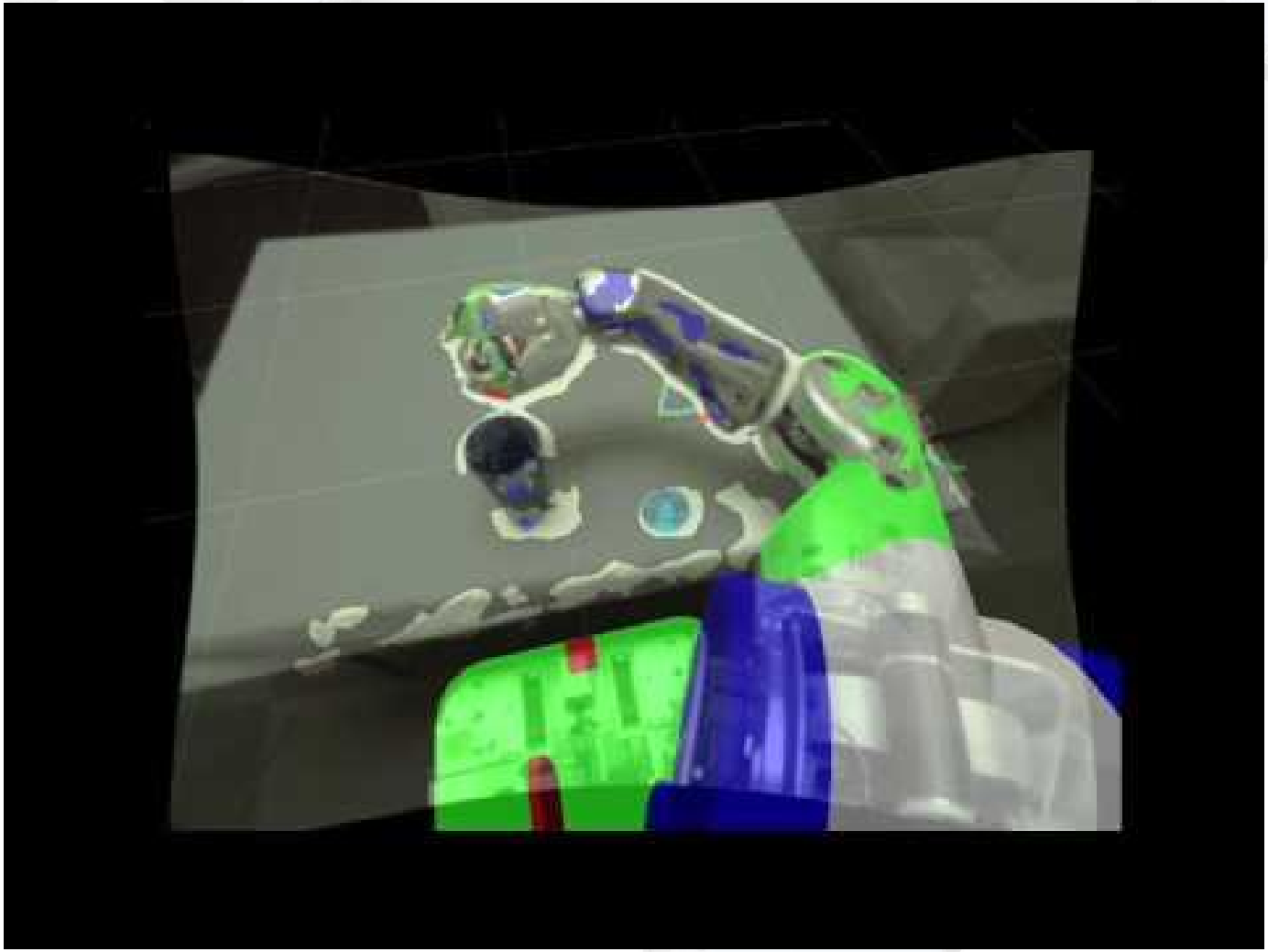


Live Plotting: rqt_plot



Logging and Visualization Sensor Data: rosbag and rqt_bag

3D Visualisation: RVIZ



“Capabilities”


PAL
SINTEF

“Ecosystem”

ROS Statistics

July 2015 - July 2016

Total unique ROS Downloads (.deb files):

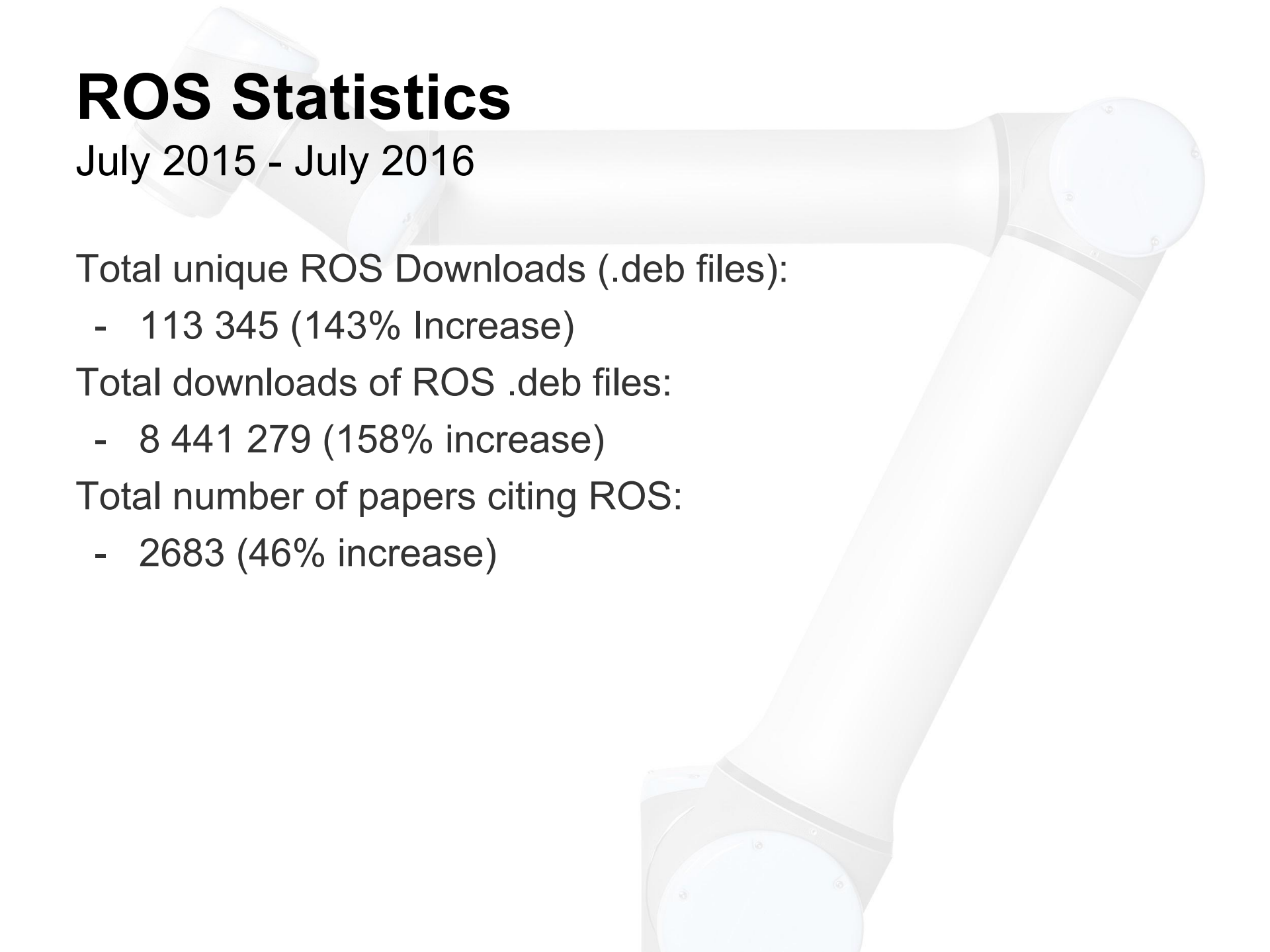
- 113 345 (143% Increase)

Total downloads of ROS .deb files:

- 8 441 279 (158% increase)

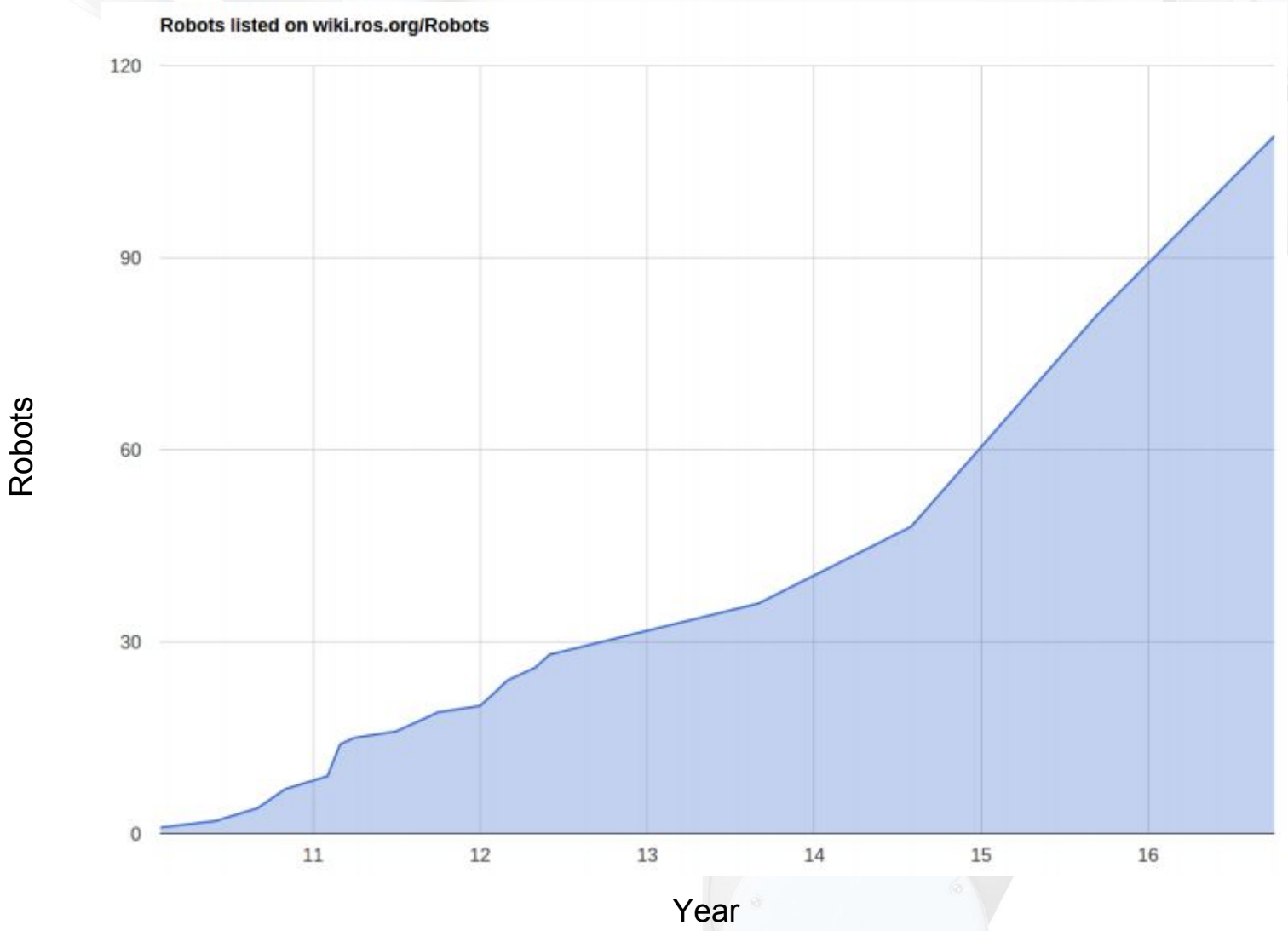
Total number of papers citing ROS:

- 2683 (46% increase)



Worldwide User Base

Number of Robots Supporting ROS




ROS
industrial


MoveIt!

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

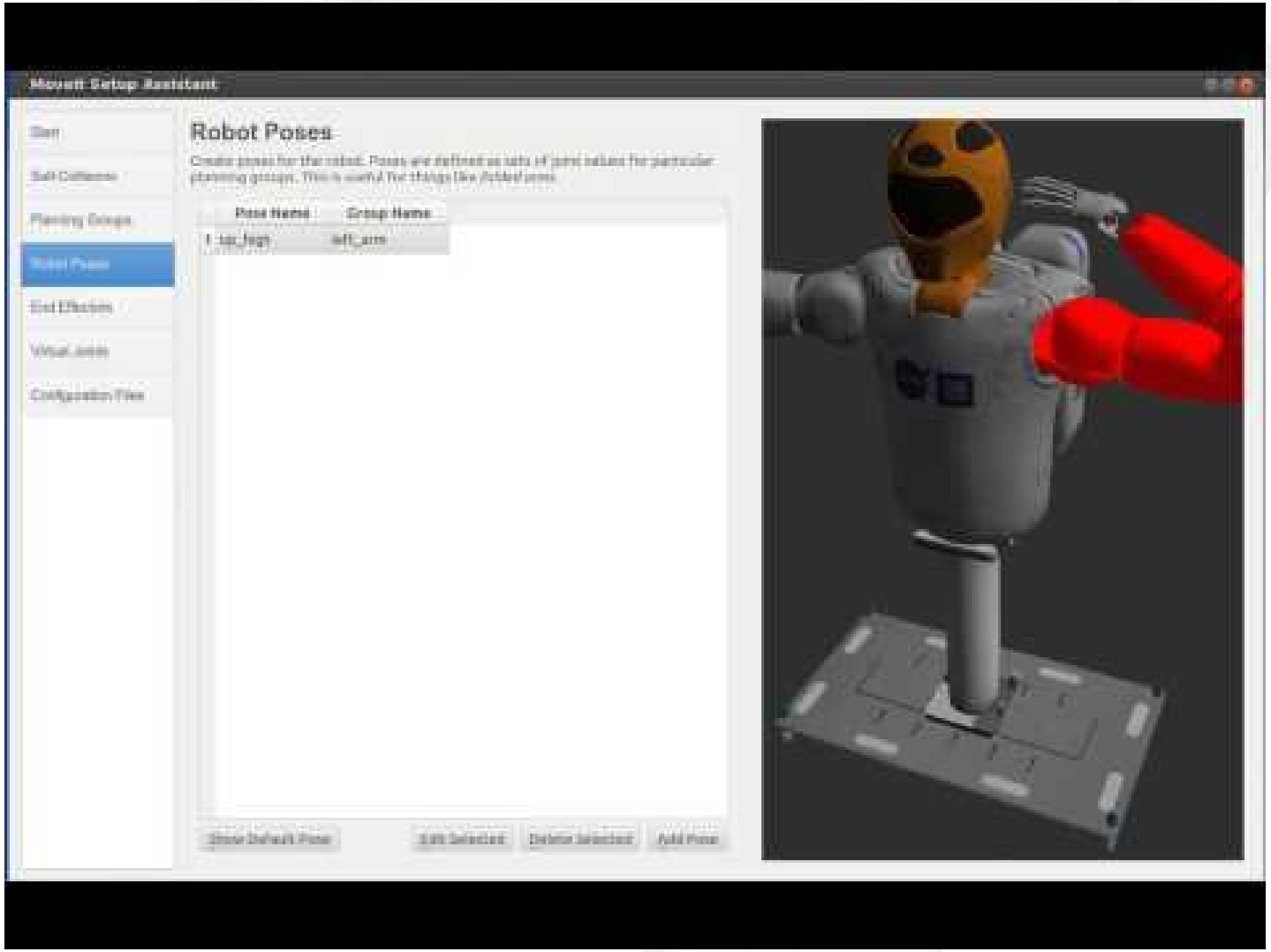
Flexible Collision Library (FCL) is used.

Types of objects supported:

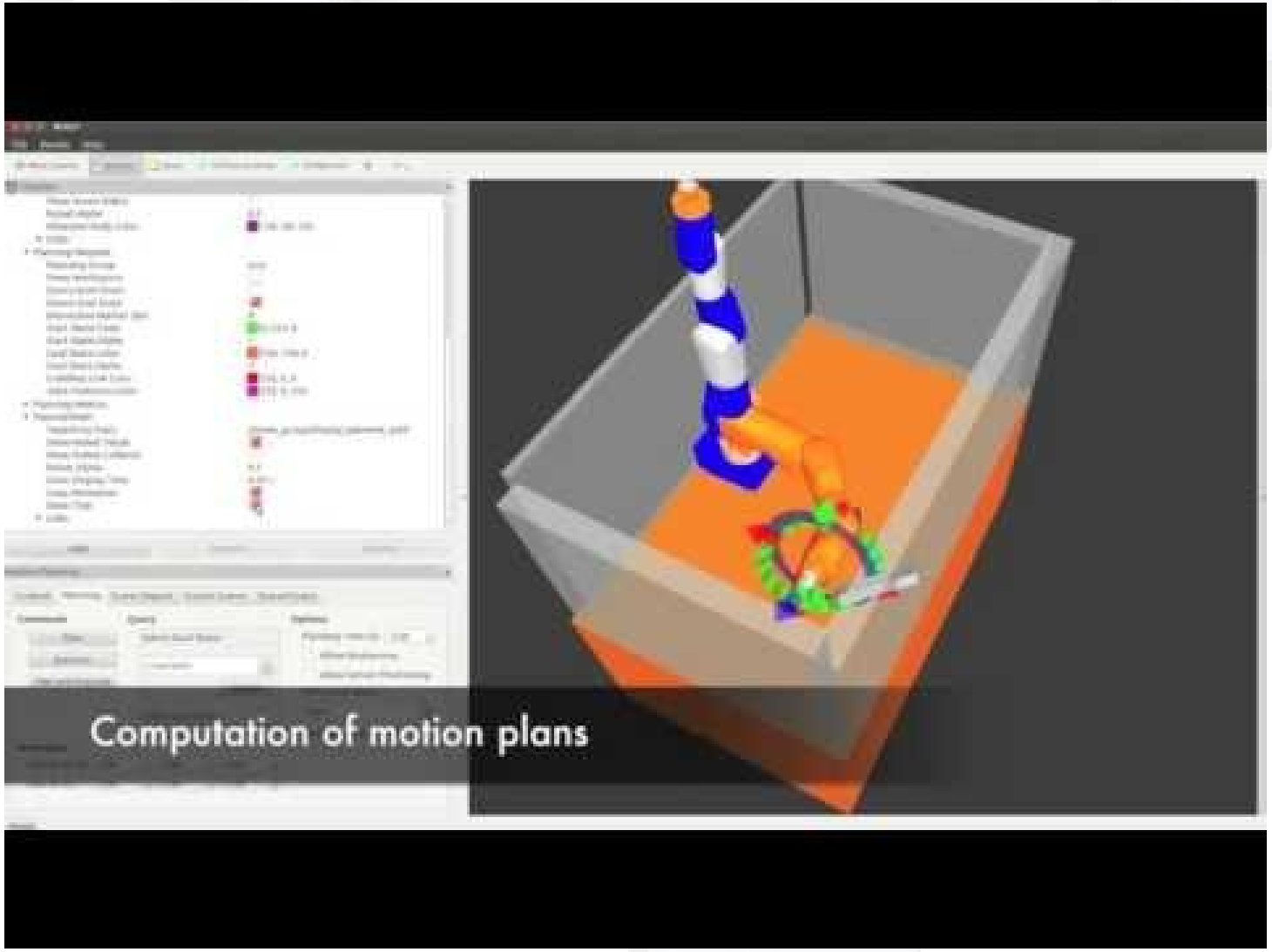
- Meshes
- Primitive shapes (boxes, cylinders, cones)
- Octomap

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

MoveIt - Robot Setup Assistant



MoveIt Capabilities



How to use it?

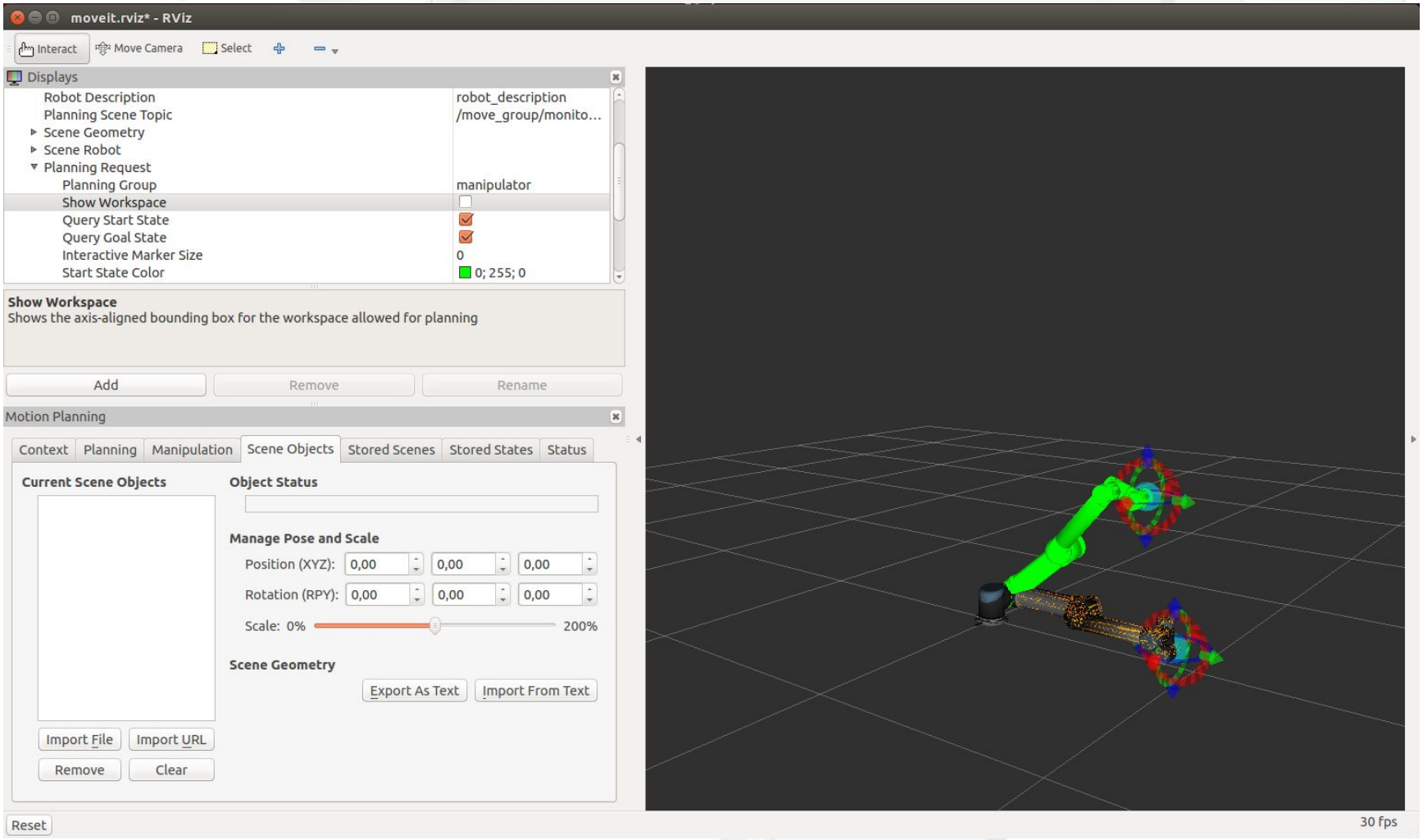
To simulate and play around with Universal Robot UR5

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?


moveit.rviz* - RViz
Interact
Move Camera
Select
Displays
Robot Description
robot_description
Planning Scene Topic
/move_group/monito…
Scene Geometry
Scene Robot
Planning Request
Planning Group
manipulator
Show Workspace
Query Start State
Query Goal State
Interactive Marker Size
0
Start State Color
0; 255; 0
Show Workspace
Shows the axis-aligned bounding box for the workspace allowed for planning
Add
Remove
Rename
Motion Planning
Context
Planning
Manipulation
Scene Objects
Stored Scenes
Stored States
Status
Current Scene Objects
Object Status
Manage Pose and Scale
Position (XYZ):
0,00
0,00
0,00
Rotation (RPY):
0,00
0,00
0,00
Scale: 0%
200%
Scene Geometry
Export As Text
Import From Text
Import File
Import URL
Remove
Clear
Reset
30 fps

Future?

ROS 2

coming soon...

ETH Zürich

Autonomous Systems Lab

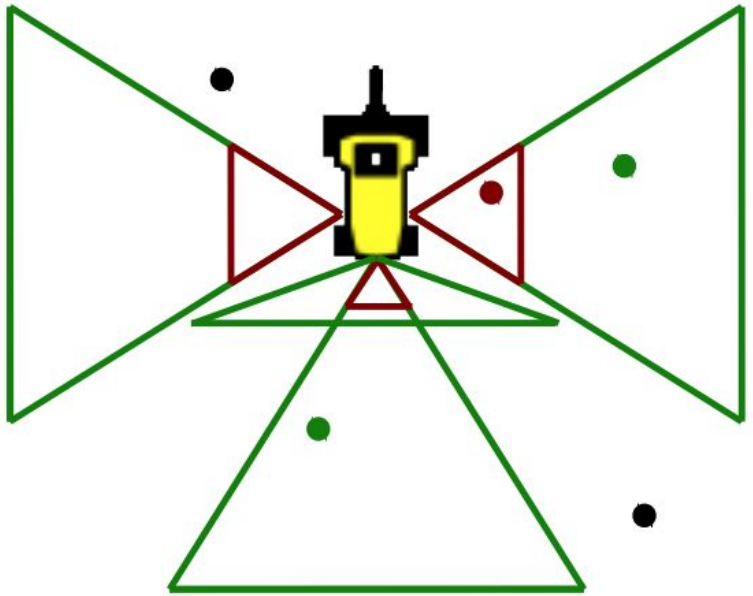
CSIRO

Actual Projects:

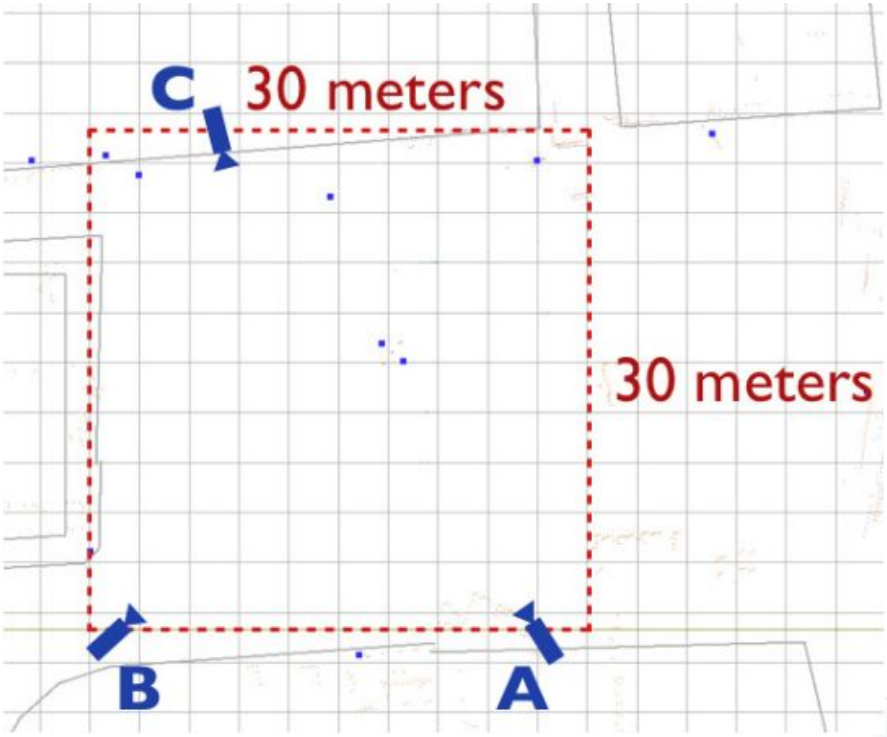
People detection for HMC

Joint Human Detection From Static and Mobile Cameras
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6894232&tag=1

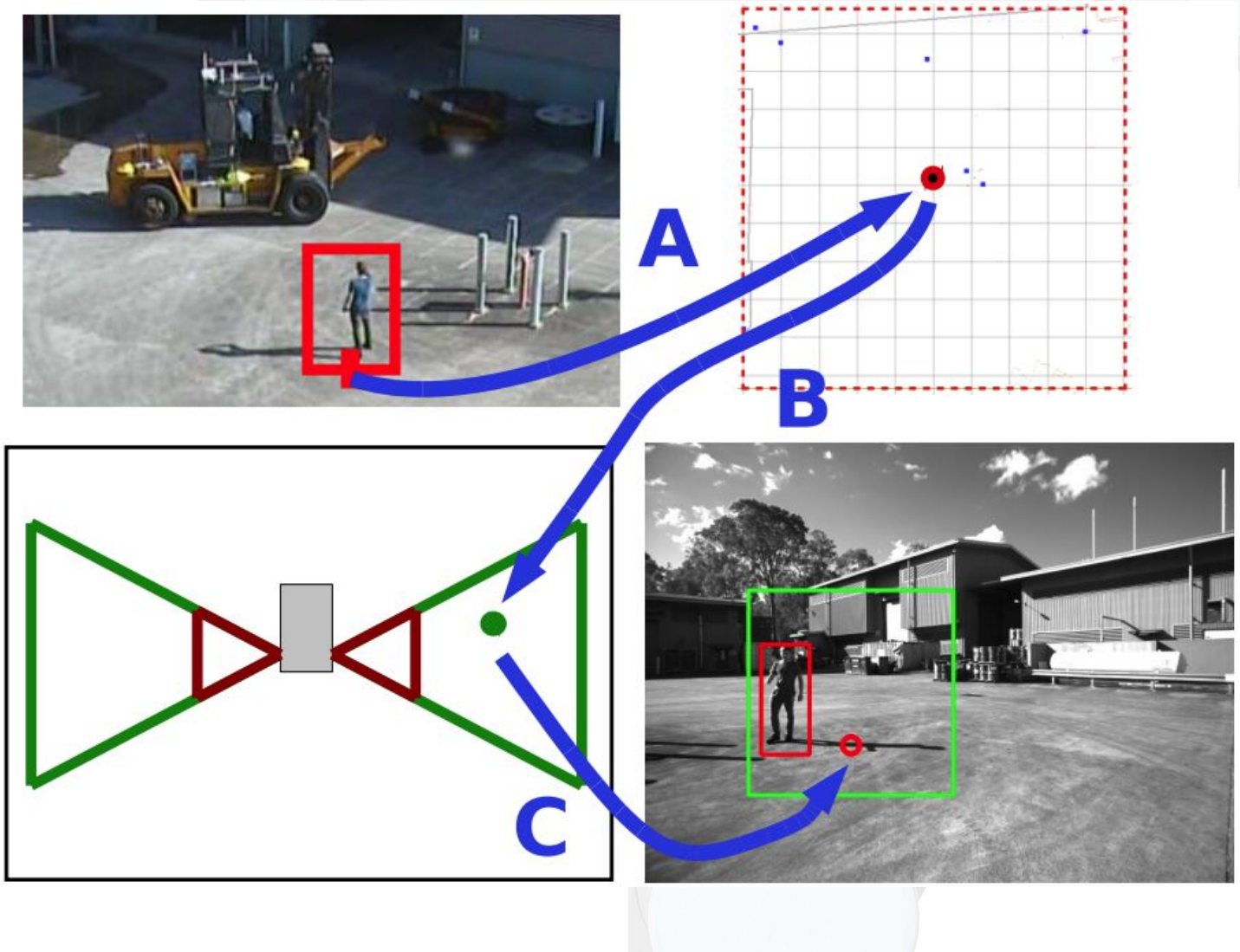
Autonomous Hot Metal Carrier (HMC)



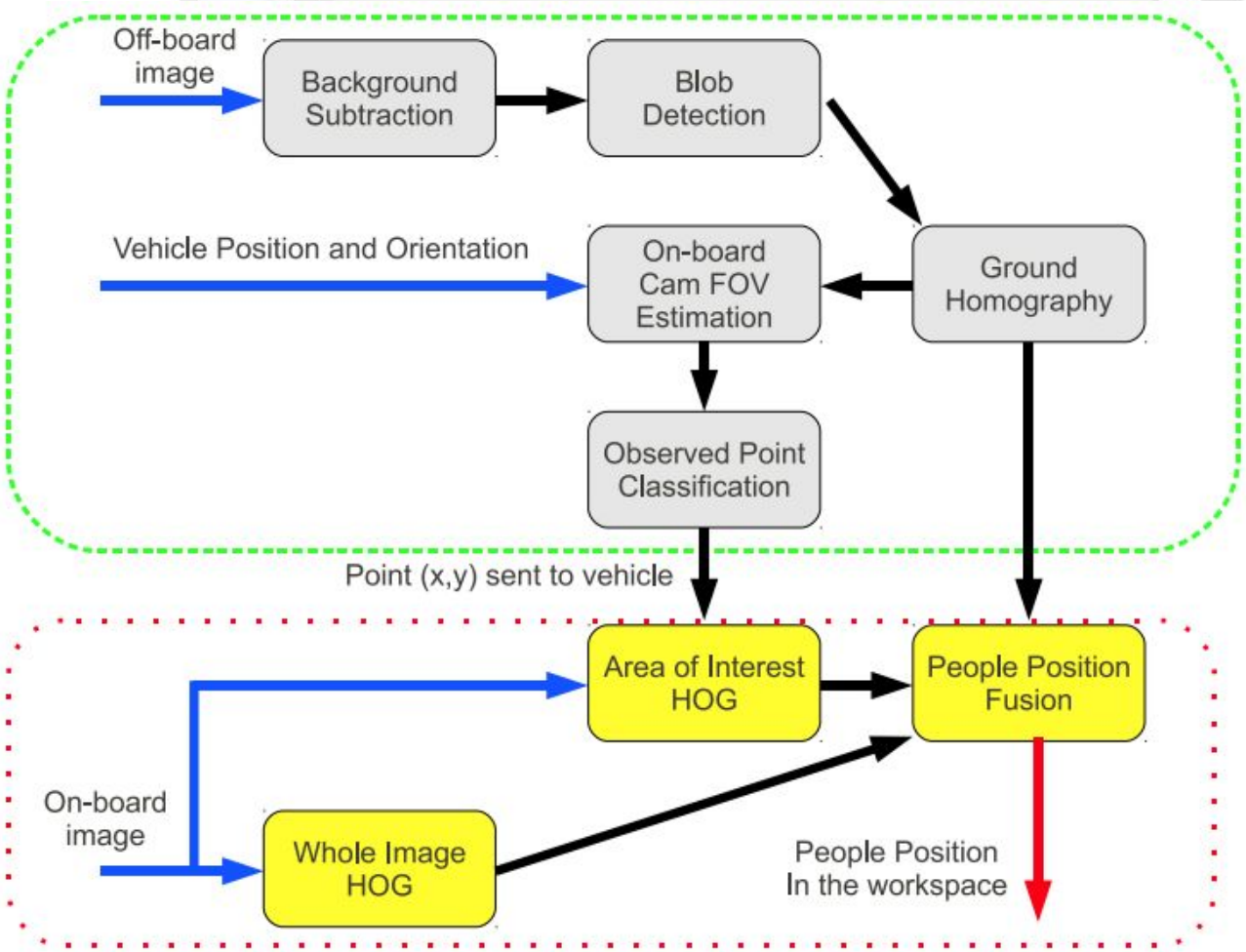
Workspace



Detection Process



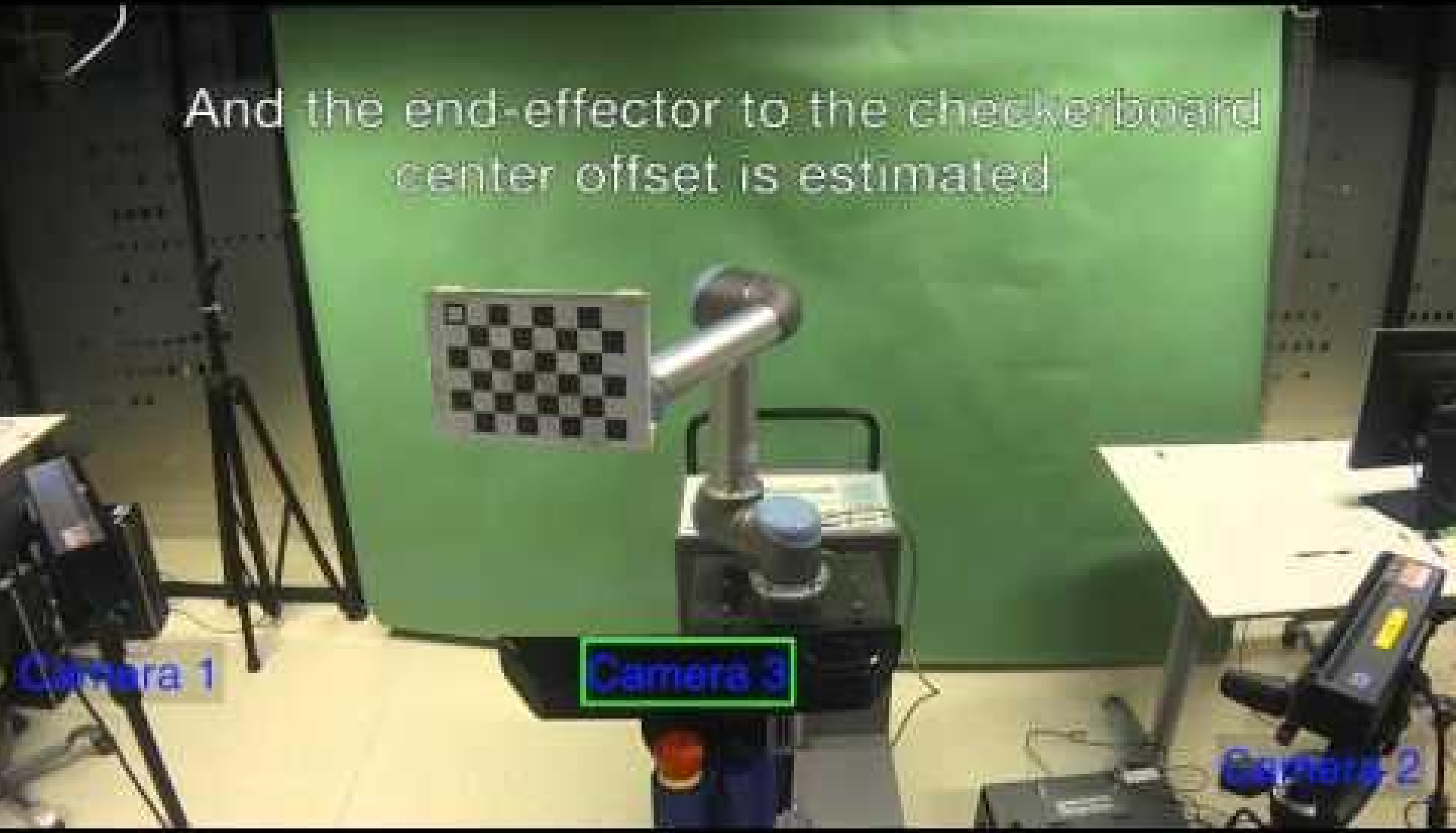
Algorithm Diagram



UiO : University of Oslo

Automatic Calibration of Kinect - Robot System

Automatic Calibration of a Robot Manipulator and Multi 3D Camera System
https://arxiv.org/abs/1601.01566

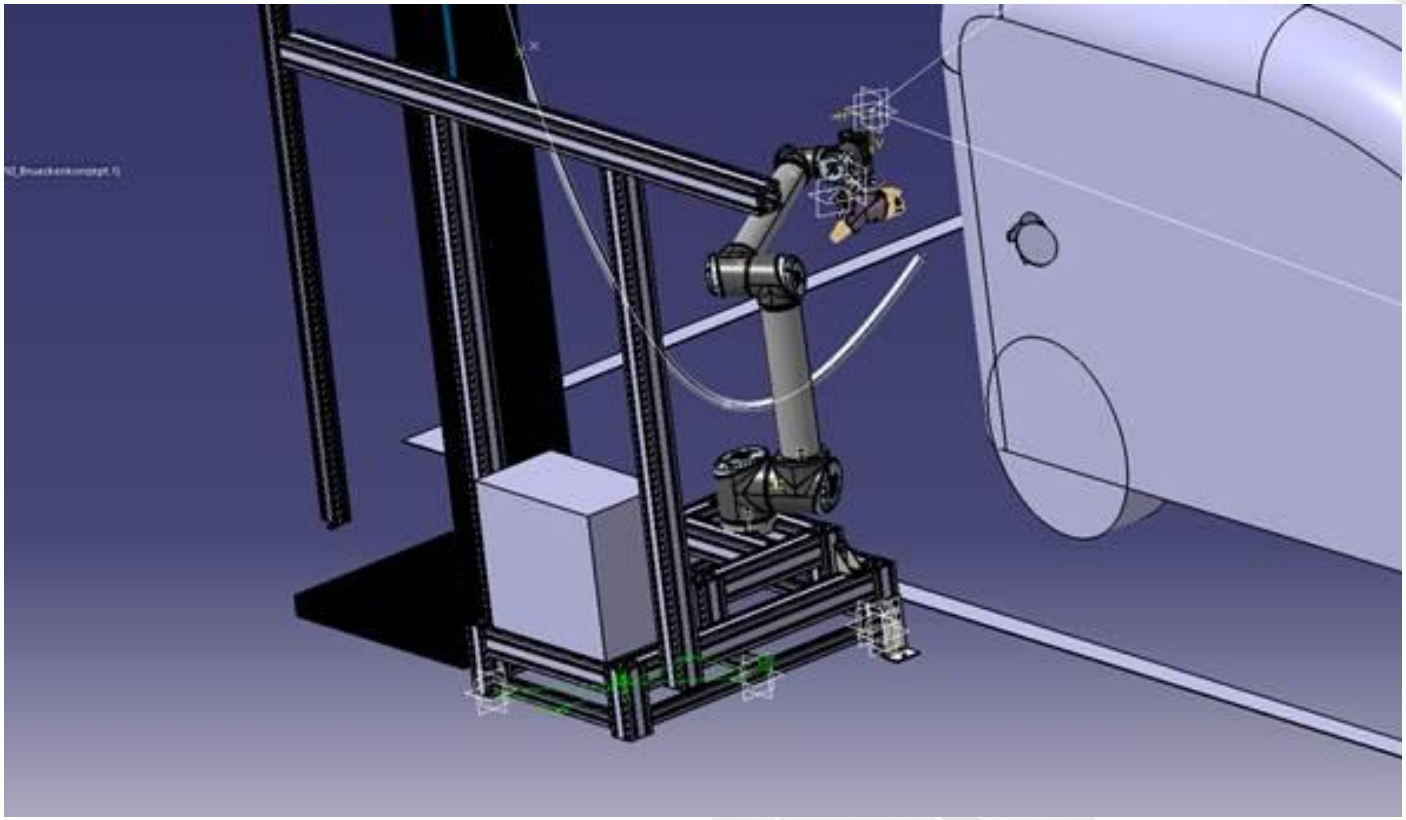

And the end-effector to the checkerboard
center offset is estimated
Camera 3

UiO : University of Oslo

TU Graz

Robot-Based Electric Vehicle Charging Station

Work in progress...



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 120 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

Thank You!

Any Questions?