Introduction to ROS

Lasse Einig, Dennis Krupke, Florens Wasserfall

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Technical Aspects of Multimodal Systems

April 6, 2015

Motivation

- Heterogeneity vs. Homogeneity
  - sensor types, actuators, ...
  - sensor model, kinematic chain, ...
- Abstraction
- Algorithm re-usability
  - 2D laser data mapping
  - object recognition
- Debugging
  - simulation, data visualization, ...

Idea

- Robot Operating System
- Meta operating system
- Open source
- Hardware abstraction
  - portability
  - simplification of sensors and actuators
- Recurring tasks already solved
  - Navigation, data filtering, object recognition ...

Current State

- Multiple versions actively used
  - may not be compatible to each other
  - may not provide same libraries
- Linux (Ubuntu!)
- Supports C/C++, Python, Java, Lisp, Octave ...
  - Python for high level code/fast implementation
  - C/C++ for algorithms/computation
- Functions and algorithms already available
  - May be difficult to find
  - Better than reimplementing

ROS System

- ROS nodes
  - sensors
  - actuators
  - logic
- ROS core
- Communication

ROS Node

- Discrete part of the system
- Specialized software/algorithm
- Many ROS nodes per system
- Example:
  - node gets image
  - runs edge detection algorithm on it
  - provides found edges

ROS Core

- Central unit, also called ROS master
  - nodes
  - sensors
  - communication
- Coordination of nodes
- Communication Management
- Exactly one per system
- Transparent to the user

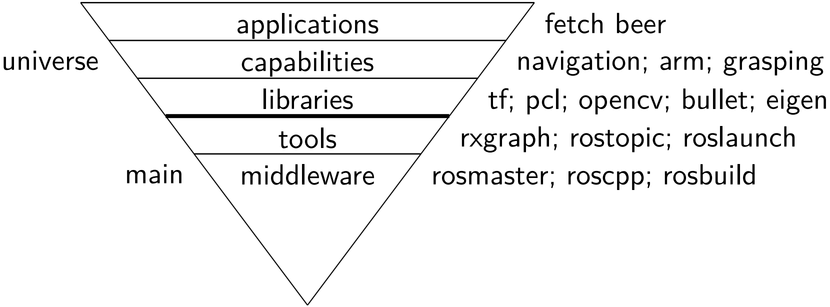
Communication

- Messages
  - standardized data types
- Topics
  - n:n communication
- Services and Actions
  - 1:1 communication

Sensors

- Exploration
- Localization
- Detection
- One node per sensor
  - provide data as topic
  - abstract from hardware

System structure



applications — fetch beer

universe: capabilities — navigation; arm; grasping

libraries — tf; pcl; opencv; bullet; eigen

tools — rxgraph; rostopic; roslaunch

main: middleware — rosmaster; roscpp; rosbuild

- universe → robot centric, developed by community
- main → general tools, maintained by Willow Garage

Messages

- Fundamental communication concept
- Description of data set
- Data types
  - ROS
  - general
- Header
  - time stamp
  - identifier

```
$ rosmsg show -r robot_msgs/Quaternion
# xyz - vector rotation axis, w - scalar term (cos(ang/2))
float64 x
float64 y
float64 z
float64 w
```

Messages

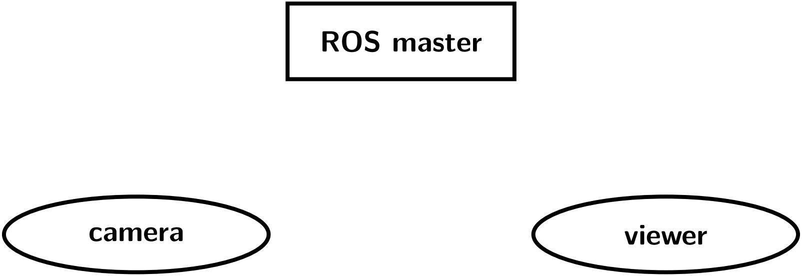
- Fundamental communication concept
- Description of data set
- Data types
  - ROS
  - general
- Header
  - time stamp
  - identifier

```
$ rosmsg show -r robot_msgs/Quaternion
# xyz - vector rotation axis, w - scalar term (cos(ang/2))
float64 x
float64 y
float64 z
float64 w
```

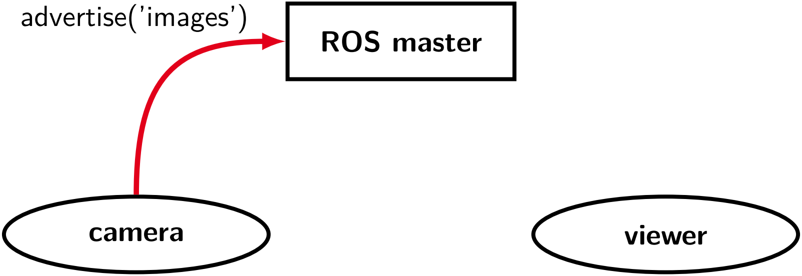
Topics

- Published by nodes
- Unique identifier
- Anonymity
- Open subscription
- Sensor data

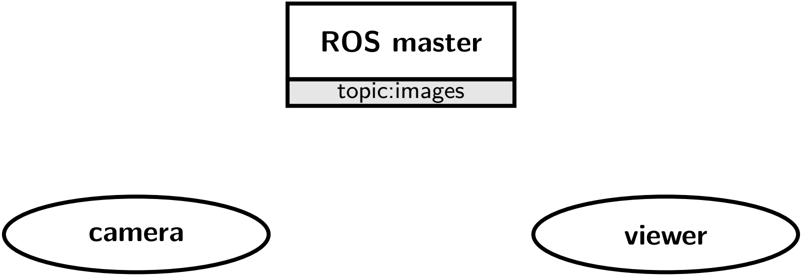
Communication - Example



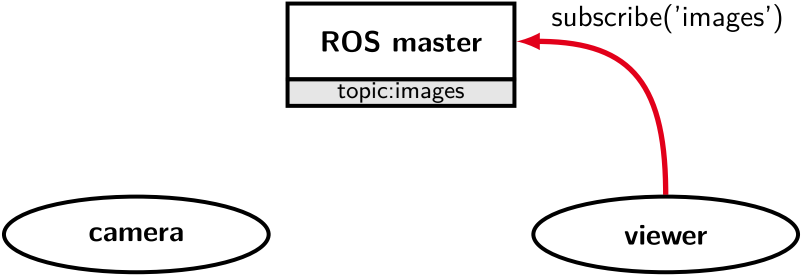
Communication - Example



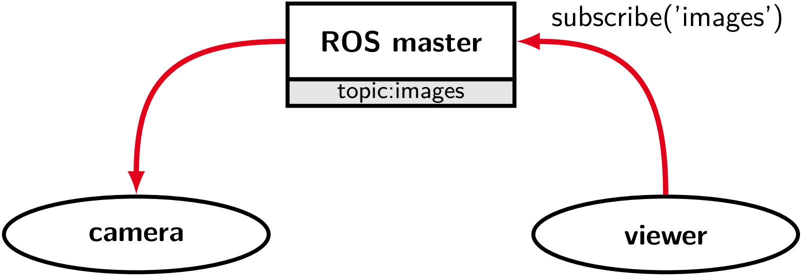
Communication - Example



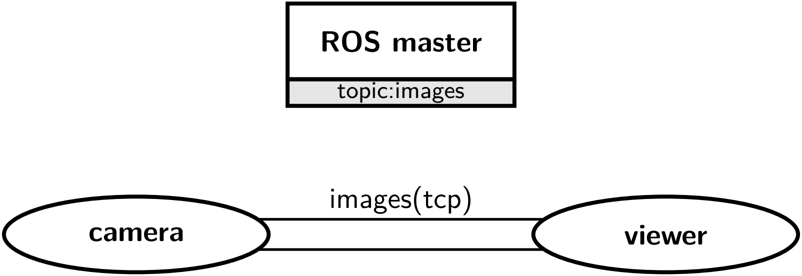
Communication - Example



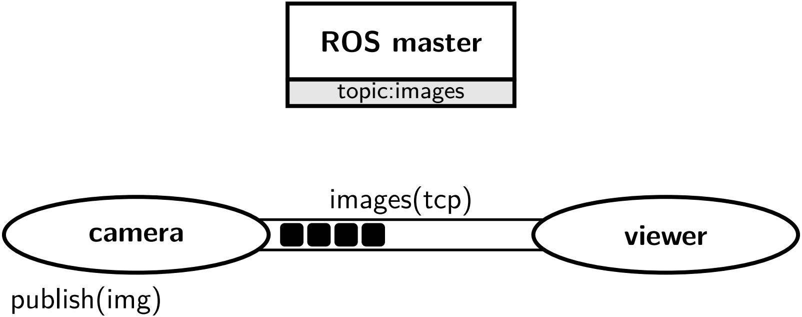
Communication - Example



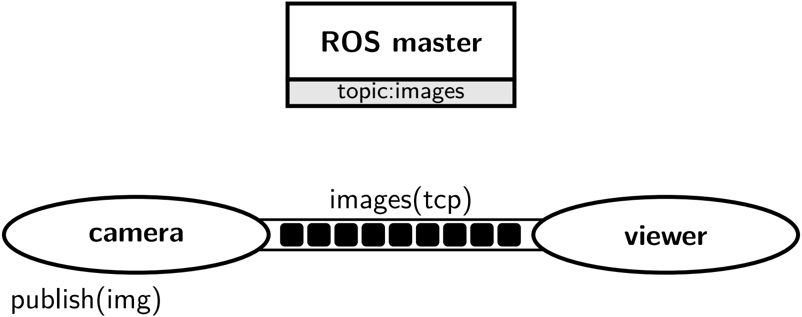
Communication - Example



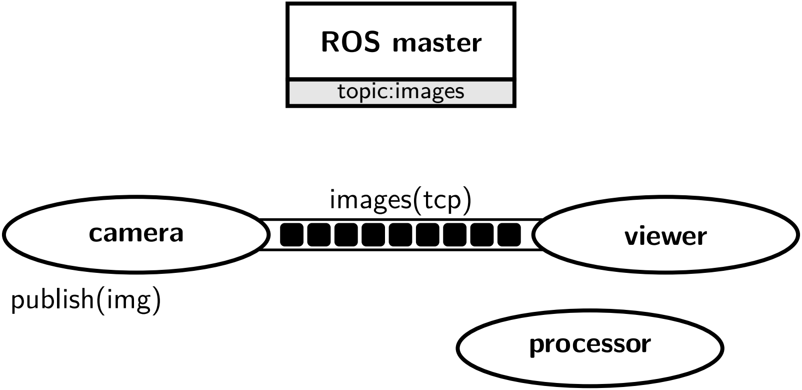
Communication - Example



Communication - Example



Communication - Example



ROS master

topic:images

images(tcp)

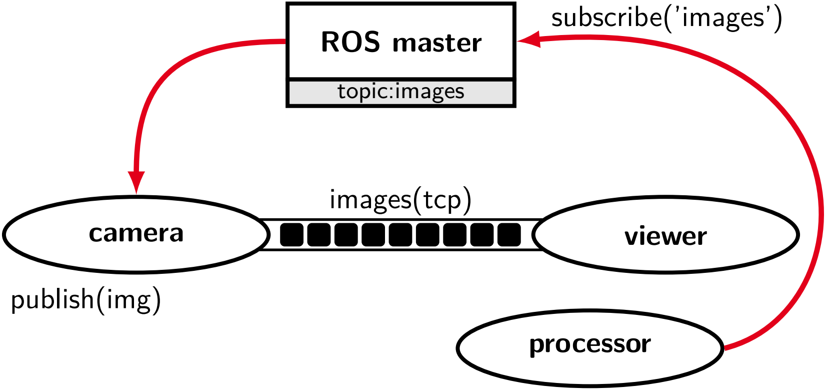
camera

viewer

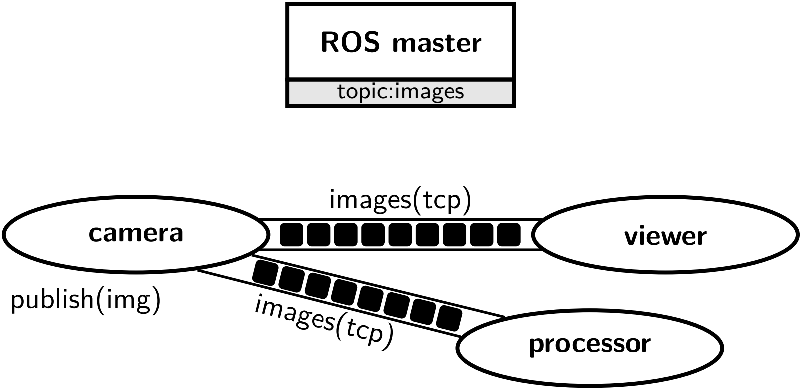
publish(img)

processor

Communication - Example



Communication - Example



Services

- 2 message types
  - request and response
- Synchronous protocol
  - client sends request
  - client waits for server
  - server replies

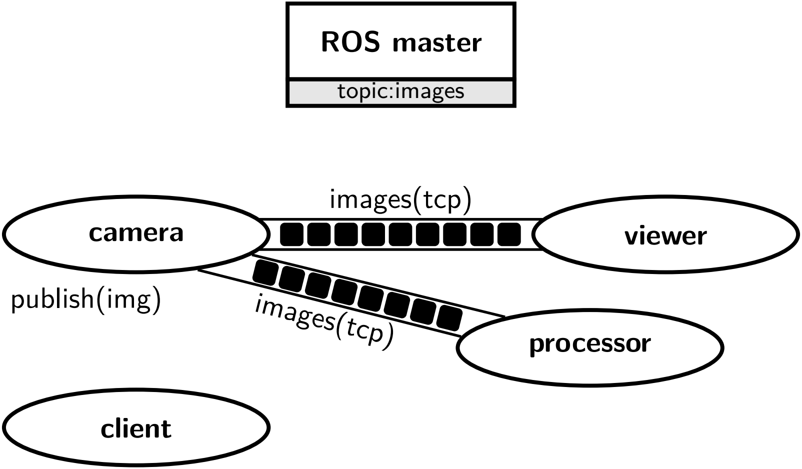
```
$ rosservice type add_two_ints | rossrv show
int64 a
int64 b
- - -
int64 sum
```

Services

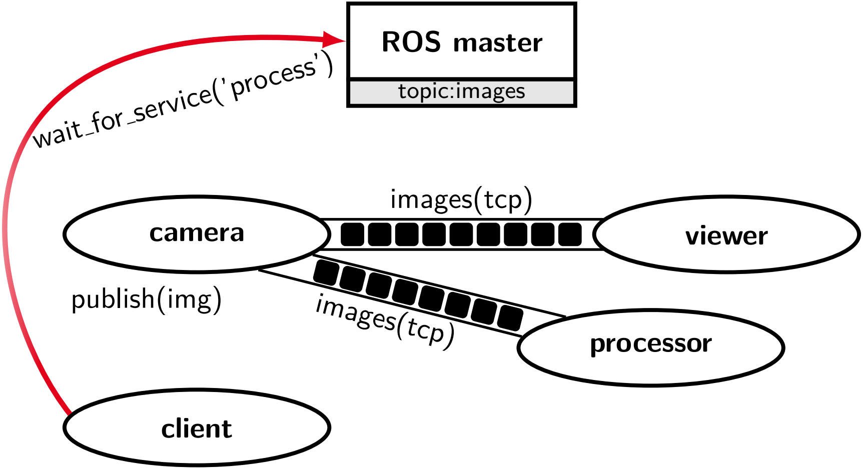
- 2 message types
  - request and response
- Synchronous protocol
  - client sends request
  - client waits for server
  - server replies

```
$ rosservice type add_two_ints | rossrv show
int64 a
int64 b
- - -
int64 sum
```

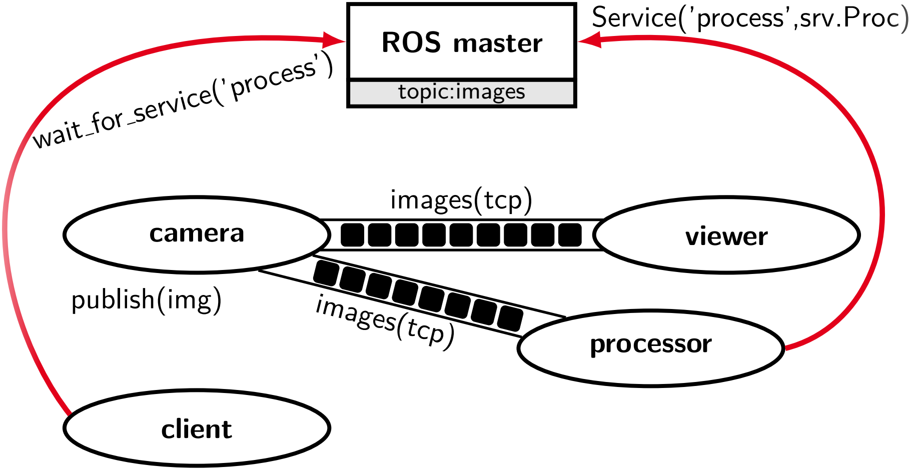
Communication - Example



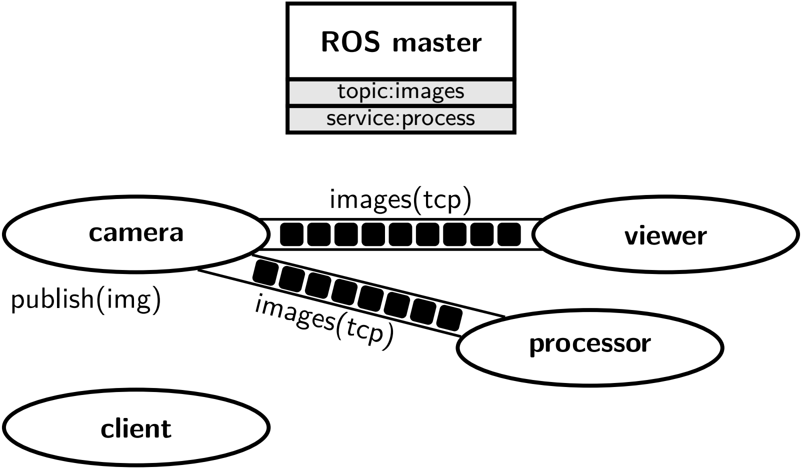
Communication - Example



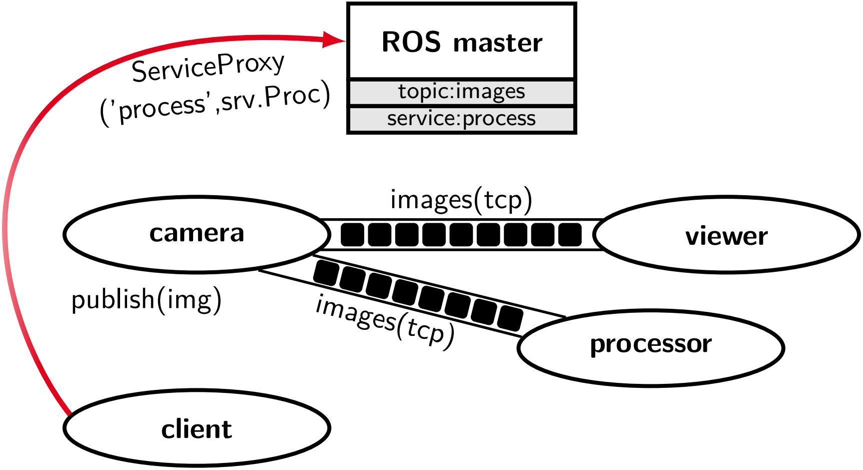
Communication - Example



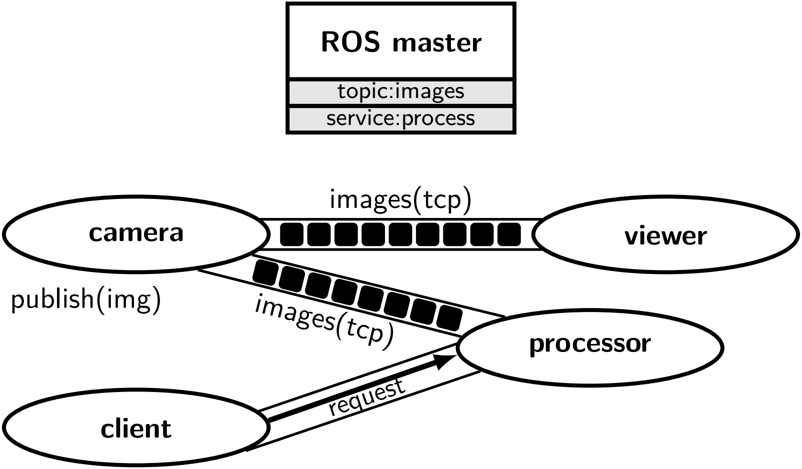
Communication - Example



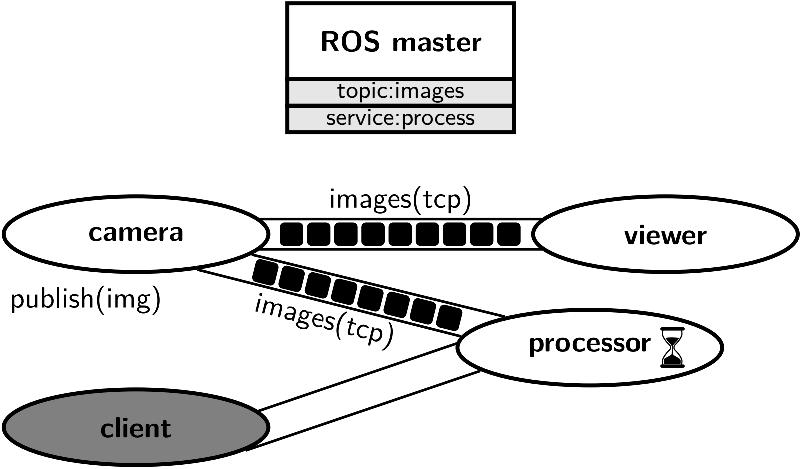
Communication - Example



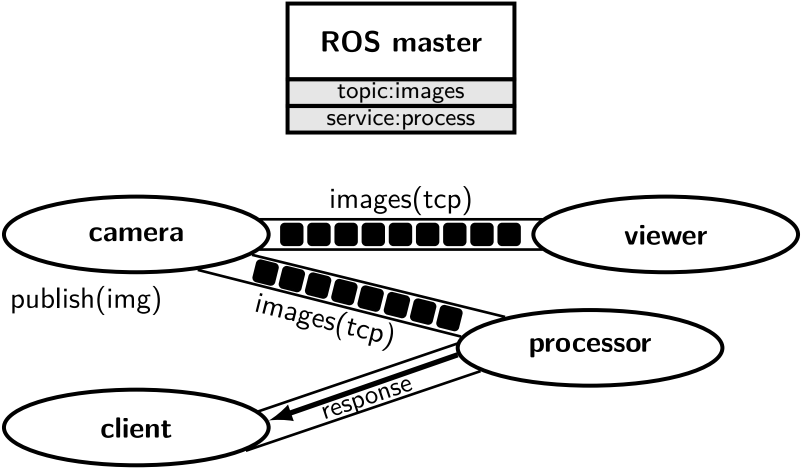
Communication - Example



Communication - Example



Communication - Example



Actions

- 3 message types
  - goal and result
  - optional feedback
- Asynchronous protocol
  - client sends goal
  - server may respond with feedback
  - server delivers result
- Interruptible

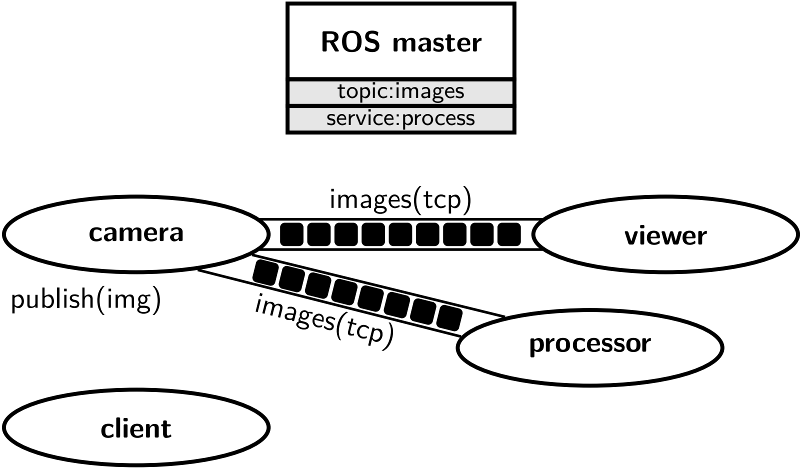
```
# Define the goal
uint32 dishwasher_id    # Specify which dishwasher we want to use
---
# Define the result
uint32 total_dishes_cleaned
---
# Define a feedback message
float32 percent_complete
```

Actions

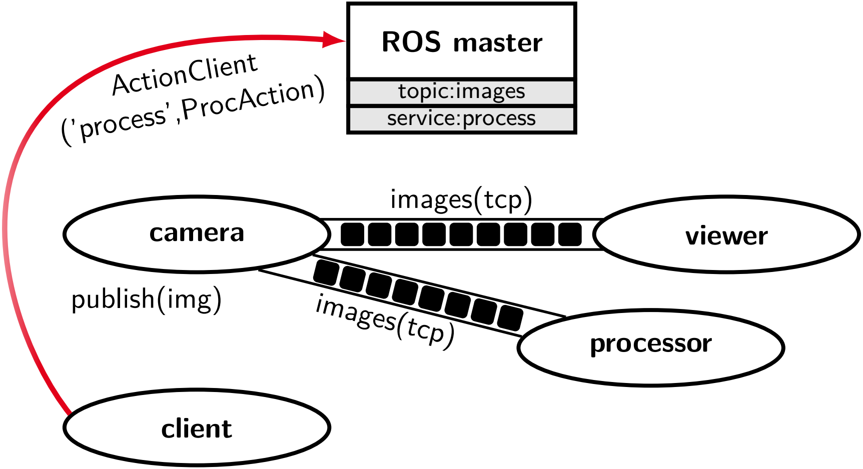
- 3 message types
  - goal and result
  - optional feedback
- Asynchronous protocol
  - client sends goal
  - server may respond with feedback
  - server delivers result
- Interruptible

```
# Define the goal
uint32 dishwasher_id    # Specify which dishwasher we want to use
- - -
# Define the result
uint32 total_dishes_cleaned
- - -
# Define a feedback message
float32 percent_complete
```

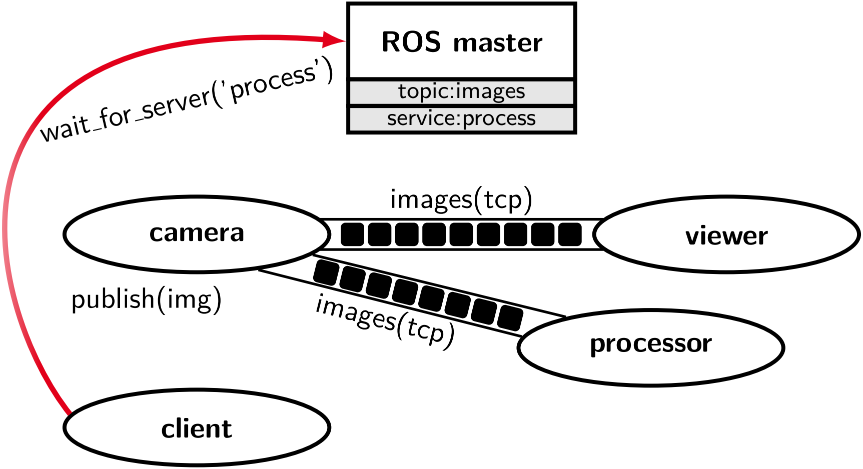
Communication - Example



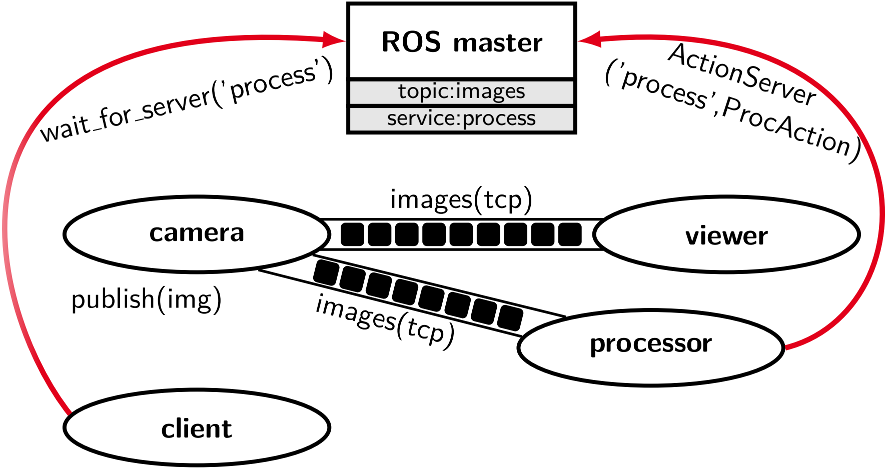
Communication - Example



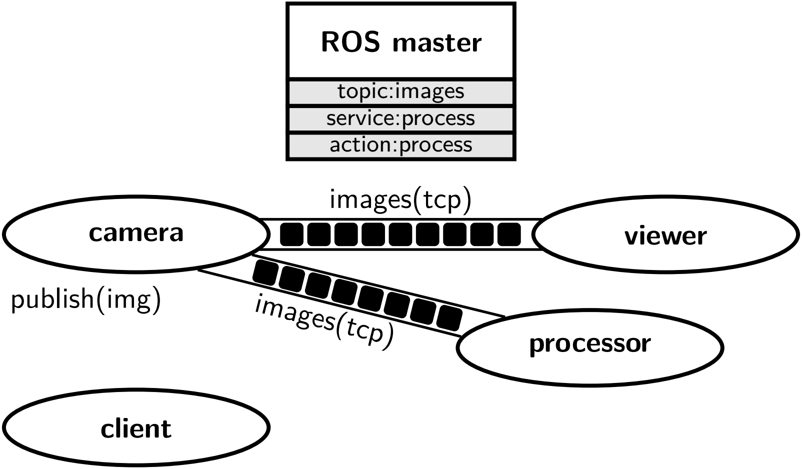
Communication - Example



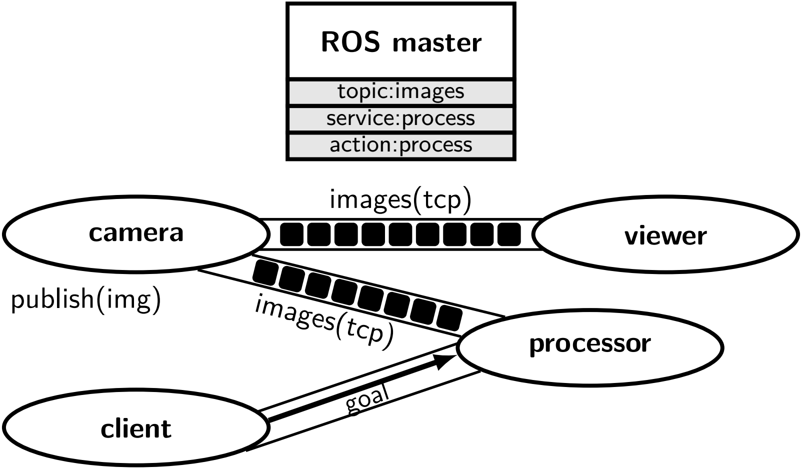
Communication - Example



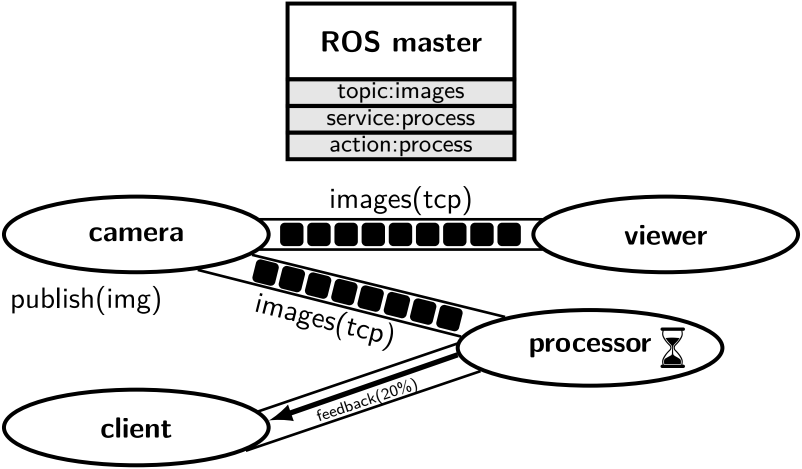
Communication - Example



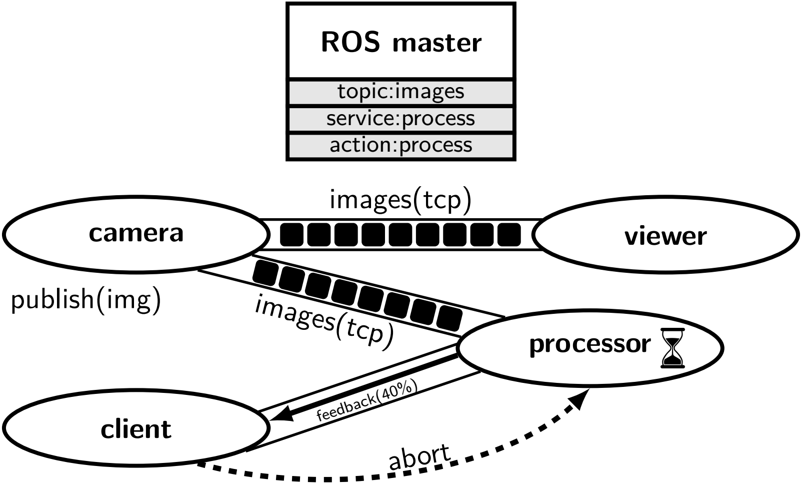
Communication - Example



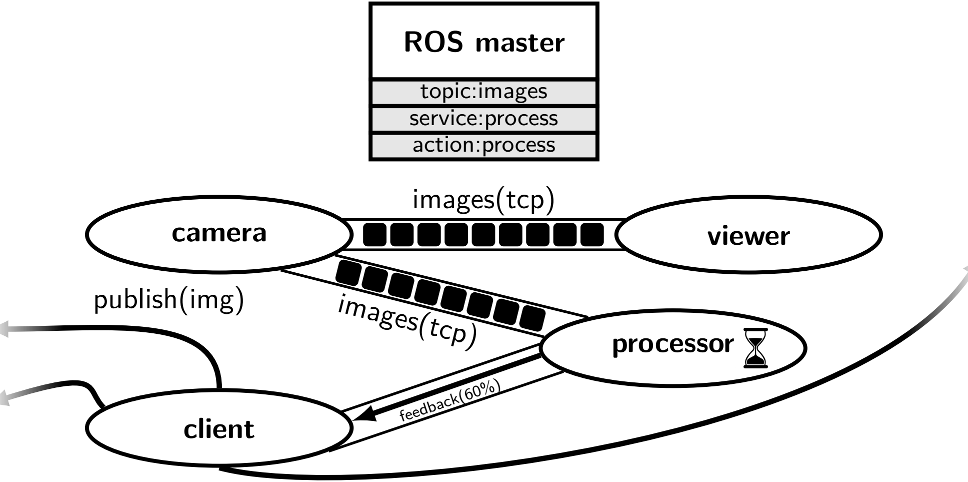
Communication - Example



Communication - Example



Communication - Example



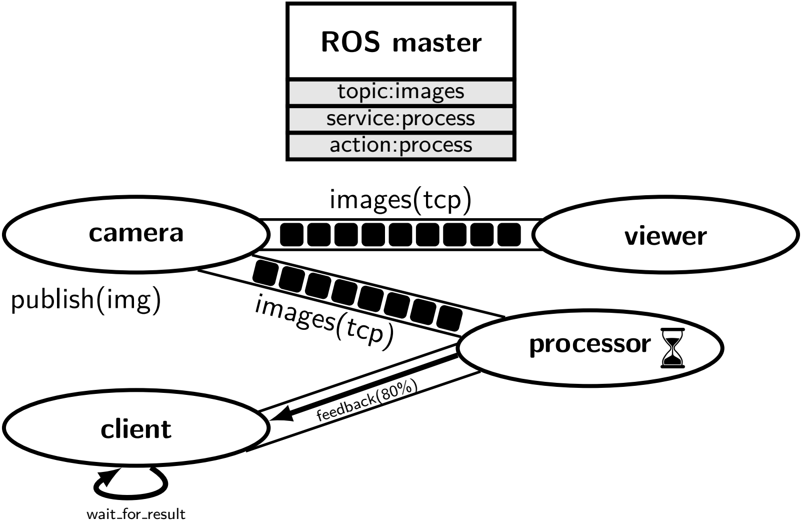
ROS master

topic:images

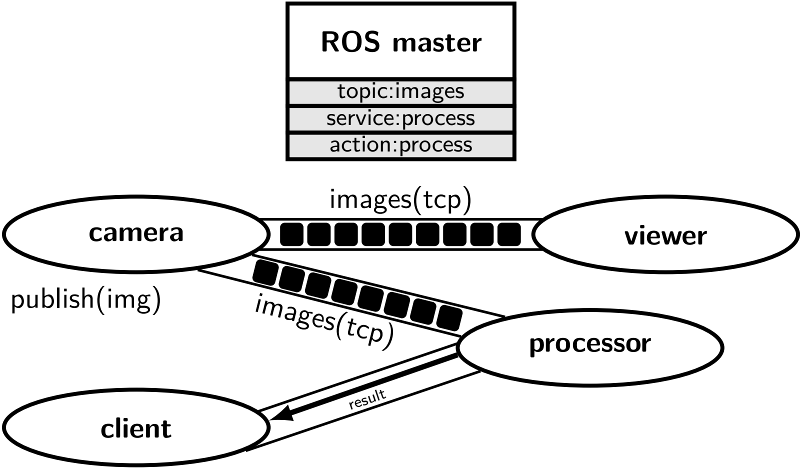
service:process

action:process

Communication - Example

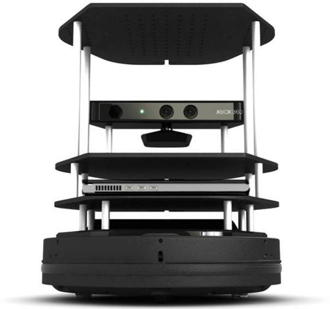


Communication - Example



Turtle-Bot

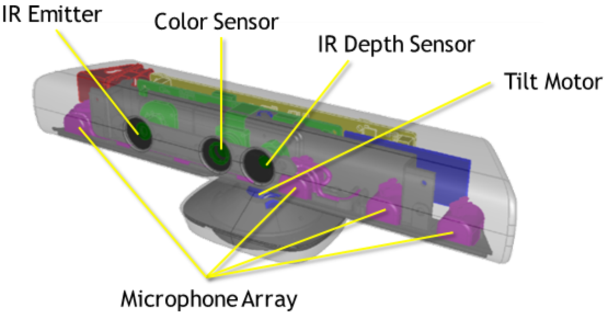
- Basic robot platform
- Capabilities
  - Kinect
  - navigation
  - transport
  - mapping
  - swarm tasks



Source: http://wiki.ros.org/Robots/TurtleBot

Microsoft Kinect

- Motion sensing device for the XBox 360 by Microsoft
- Range camera technology by PrimeSense
- 3D depth information from infrared structured light

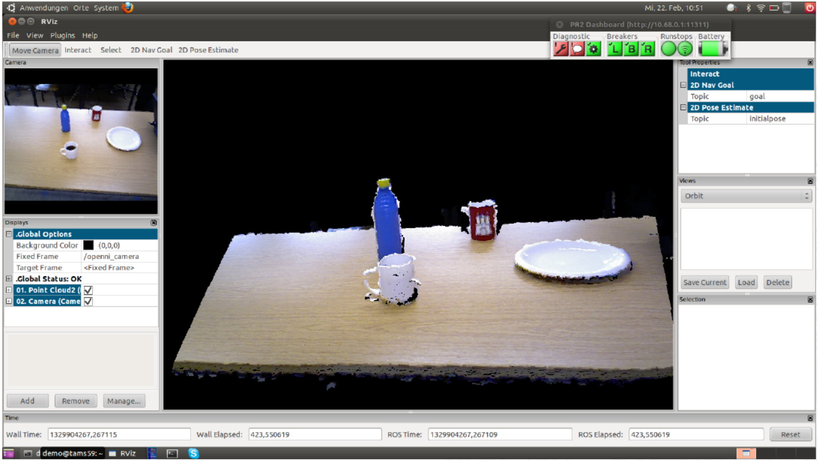


Source: https://msdn.microsoft.com

Kinect - technical details

- Resolution
  - 640 x 480 @ 30 Hz color
  - 320 x 240 @ 30 Hz depth
- FOV of 57° horizontally and 43° vertically
- Range ∼0.7 - 6 m
  - up to 3.5 m realistic
- Physical tilt range ±31°
- Microphone array with 16 bit @ 16 kHz
  - supports single speaker voice recognition
- OpenNI and Freenect drivers

Kinect - example



Interact

2D Nav Goal

Topic	goal

2D Pose Estimate

Topic	initialpose

Views: Orbit

Displays

.Global Options

Background Color	(0,0,0)
Fixed Frame	/openni_camera
Target Frame	<Fixed Frame>

.Global Status: OK

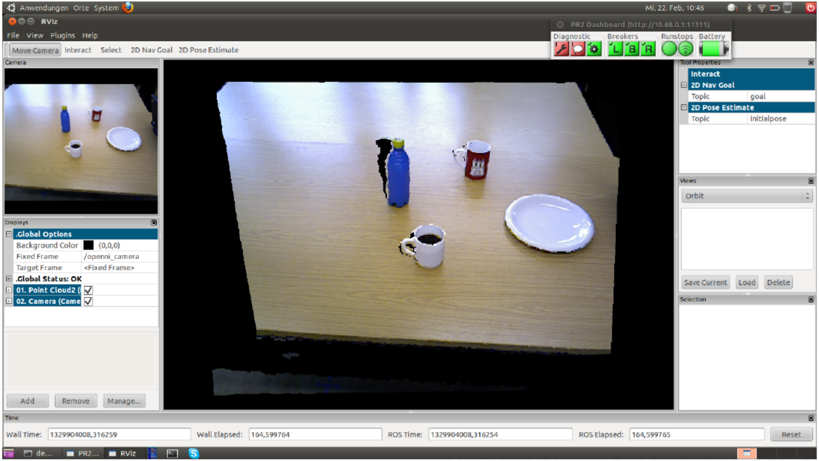
01. Point Cloud2 (☑

02. Camera (Came ☑

Wall Time: 1329904267,267115 Wall Elapsed: 423,550619 ROS Time: 1329904267,267109 ROS Elapsed: 423,550619

Source: TAMS, Uni Hamburg

Kinect - example



Displays:

.Global Options	
Background Color	■ (0,0,0)
Fixed Frame	/openni_camera
Target Frame	<Fixed Frame>
.Global Status: OK	
01. Point Cloud2 (	☑
02. Camera (Came	☑

Tool Properties:

Interact	
2D Nav Goal	
Topic	goal
2D Pose Estimate	
Topic	initialpose

Views: Orbit

Wall Time: 1329904008,316259 Wall Elapsed: 164,599764 ROS Time: 1329904008,316254 ROS Elapsed: 164,599765

Source: TAMS, Uni Hamburg

Simulations

- Important development tool
  - protects expensive hardware
  - develop and test without robot
  - high-level test
- Simulates sensor data
  - clean data
- Turtlesim
  - ROS learning tool
- Gazebo
  - ROS simulator
- RViz
  - ROS data visualization

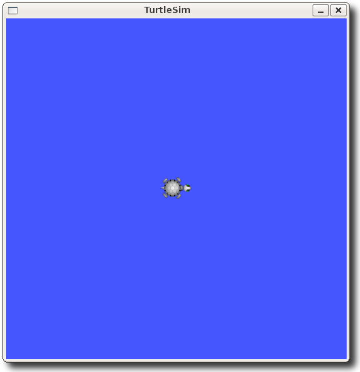
Simulations

- Important development tool
  - protects expensive hardware
  - develop and test without robot
  - high-level test
- Simulates sensor data
  - clean data

- Turtlesim
  - ROS learning tool
- Gazebo
  - ROS simulator
- RViz
  - ROS data visualization

Turtle Sim

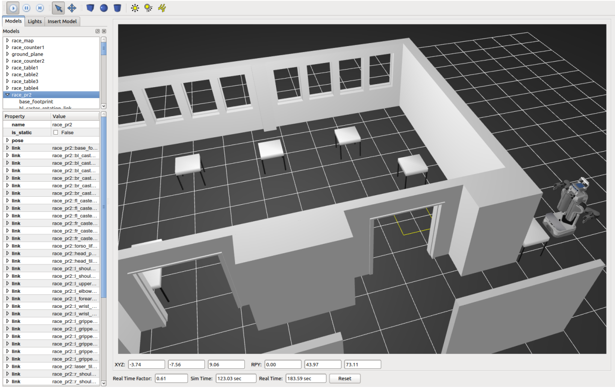
- Learning platform
- 2D turtle
  - move
  - turn
  - draw
- Communication
- ROS structure



Source: http://wiki.ros.org/turtlesim

Gazebo

- 3D rigid body simulator
- Simulates robots, environment and sensor data



Models

- race_map
- race_counter1
- ground_plane
- race_counter2
- race_table1
- race_table2
- race_table3
- race_table4
- race_pr2
  - base_footprint
  - bl_caster_rotation_link

Property	Value
name	race_pr2
is_static	□ False
pose	
link	race_pr2::base_fo...
link	race_pr2::bl_cast...
link	race_pr2::bl_cast...
link	race_pr2::bl_cast...
link	race_pr2::br_cast...
link	race_pr2::br_cast...
link	race_pr2::br_cast...
link	race_pr2::fl_caste...
link	race_pr2::fl_caste...
link	race_pr2::fl_caste...
link	race_pr2::fr_caste...
link	race_pr2::fr_caste...
link	race_pr2::fr_caste...
link	race_pr2::torso_lif...
link	race_pr2::head_p...
link	race_pr2::head_til...
link	race_pr2::l_shoul...
link	race_pr2::l_shoul...
link	race_pr2::l_upper...
link	race_pr2::l_elbow...
link	race_pr2::l_forear...
link	race_pr2::l_wrist_...
link	race_pr2::l_wrist_...
link	race_pr2::l_gripper...
link	race_pr2::l_gripper...
link	race_pr2::l_gripper...
link	race_pr2::l_gripper...
link	race_pr2::l_gripper...
link	race_pr2::l_gripper...
link	race_pr2::laser_til...
link	race_pr2::r_shoul...
link	race_pr2::r_shoul...

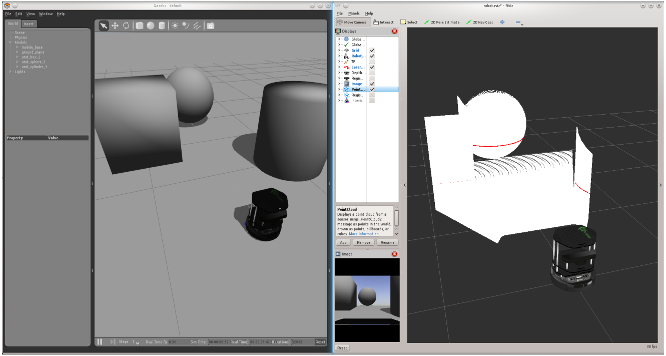
XYZ: -3.74 -7.56 9.06 RPY: 0.00 43.97 73.11

Real Time Factor: 0.61 Sim Time: 123.03 sec Real Time: 183.59 sec

Source: Lasse Einig

RViz

- 3D visualization environment
- Different data can be shown
  - Laser scan data, map, ...

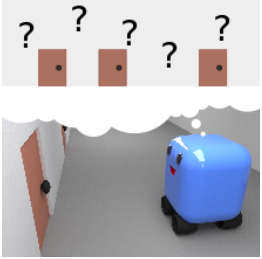


PointCloud
Displays a point cloud from a sensor_msgs::PointCloud2 message as points in the world, drawn as points, billboards, or cubes. More Information.

Source: http://wiki.ros.org/turtlebot_gazebo

Key questions

- Mapping
  - How to build a map?
- (Self-)Localization
  - How to find own position?
- Navigation
  - How to move from one position on the map to another?



Source: Wikipedia

Mapping

How to solve the
problem with the map

Mapping

Solution

- Gather 3D information
  - using available sensors
  - kinect, laser scanner
- Create a part of map
  - from observable area
- Fuse map parts
  - using additional information
  - e.g. odometry

Self-Localization

We already have a map

Self-Localization

Solution

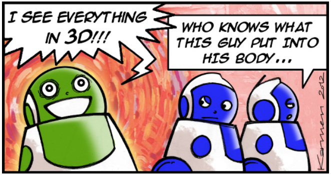
Estimate robot position from 3D information and odometry

Problems:

- odometry error
- occlusion
- sensor noise

Solutions:

- modeling the odometry error
- probabilistic algorithms
- preprocessing (e.g. smoothing)


I SEE EVERYTHING IN 3D!!!
WHO KNOWS WHAT THIS GUY PUT INTO HIS BODY...
Karmen 2012

Source: http://thecorpora.com/blog/wp-content/uploads/2012/03/Qbo-robot-Xtion-3D-sensor.jpg

Navigation

We know where we are

Navigation

Solution

Find path from current position to goal position

Prerequisites:

- precise map
- precise current position on the map
- appropriate way-points