ROS: Robotic Operation System

Libor Wagner

Centre for Machine Perception
Czech Technical University
wagnelib@cmp.felk.cvut.cz

April 2, 2013

Outline

1. ROS Introduction
   - Design goals
   - Basic concepts
   - Software development
   - Community and resources
2. ROS and CloPeMa
   - Useful packages and libraries
   - ROS Release

ROS Introduction

- Open Source framework (middleware) for robot software development.
- Started at Stanford Artificial Intelligence Lab, further developed at Willow Garage.
- Strong emphasis on distributed computation and development.
- Active community, widespread use.

Design goals

Peer-to-peer ROS components, potentially on different hosts, are connected in peer-to-peer topology.

Tool-based Microkernel design, with large number of small tools, used to build, run and analyse ROS components.

Multi-lingual ROS components can be written in various languages.

Thin Drivers and algorithms are encouraged to be written in separated libraries.

Open-Source ROS is distributed under terms of the BSD license.

Basic concepts

Node A single computation unit (component).

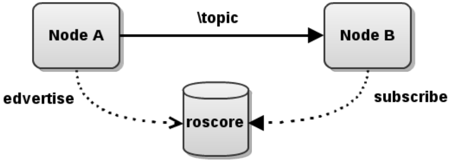
Message Data structure used by nodes to communicate.

Topic Broadcast communication between nodes.

Service Synchronous communication between nodes.

Node

- Single process that performs particular computation.
- ROS system is composed from large number of nodes.
- Communicate with each other by passing **messages**, through **topic** or **service**.
- Connection between two nodes is accomplished through **roscore**, which acts as a name server.



Message

- Strictly typed data structure.
- Support standard primitive types (integer, float, boolean, etc.) and arrays.
- Messages can be composed of other messages and arrays of messages.

```
# Header message
uint32 seq
time stamp
string frame_id


# Composite message
Header header
int32 x
int32 y
```

Topic and service

- Topic
  - A named broadcast stream of messages.
  - Generally there can be more publishers of the same topic.
  - Publishers are aware if someone is subscribed.
  - Topic is defined by name and message type.
- Service
  - A named synchronous communication.
  - There can be just one node providing a service of some name.
  - Calling service is generally blocking.
  - Service is defined by name and two message types – request and reply.

Programing languages and platform

- Each ROS node can be written in different language.
- Message type is defined in plain text using **Message Description Language** and code is generated for each supported language.
- ROS currently support **C++**, **Python** and **Lisp**.
- Other languages are supported unofficially: Java, Haskell ...
- **Ubuntu** linux is the only supported platform.
- Support for other platforms, including Windows, is experimental.

Supporting tools and packages

rviz Visualisation tool.

rosbag Allows to record all communication between nodes and then play it back.

rosdep Tracks external dependencies.

rxgraph Visualise graph of ROS system.

rosparam Store and manipulate data on the ROS parameter server.

...

Community and resources

- ROS is supported by active community.
- ROS documentation wiki (`www.ros.org/wiki/`)
- ROS user forum (`answers.ros.org`).
- There is already around 600 packages in ROS distribution.
- More packages can be found on ROS page (`www.ros.org/browse`).

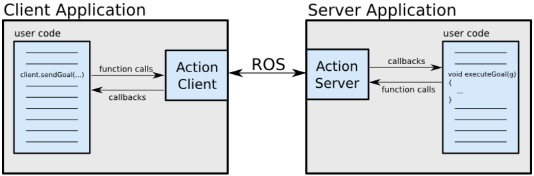
Image Transport

- Provide support for image transport in arbitrary representation.
- The complexity is abstracted from the developer, which only sees standard image message.
- Particular transport representations are provided by plugins.
- Currently supported representations are raw, JPEG/PNG compression, and Theora for streaming video.

`http://www.ros.org/wiki/image_transport`

ActionLib

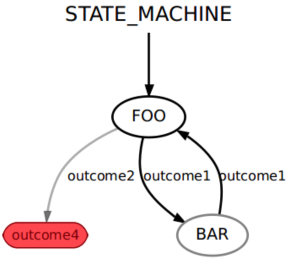
- Provide preemptible task execution.
- Communication build on top of ROS messages.
- Action is specified by three messages: goal, feedback and result.



http://www.ros.org/wiki/actionlib

SMACH

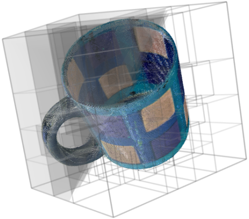
- Stand-alone Python library for structured plan execution.
- Based on hierarchical state machines.
- State is defined by set of possible outcomes.
- Simple states are encapsulated in containers, that can be used as states.



http://www.ros.org/wiki/smach

PCL: Point Cloud Library

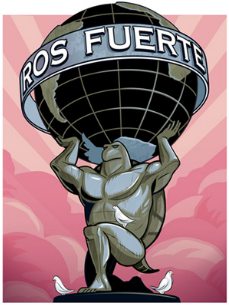
- Stand-alone C++ library for 3D point cloud processing.
  - Filtering
  - Registration
  - Segmentation
  - Feature extraction
  - Keypoints detection



http://www.ros.org/wiki/pcl

ROS Release

- ROS uses six month release cycle similar to Ubuntu.
- Current ROS release is **Fuerte Turtle**.
- Compatibility between releases is not guaranteed.



ROS: Example

Libor Wagner

Centre for Machine Perception
Czech Technical University in Prague
wagnelib@cmp.felk.cvut.cz

April 2, 2013

Outline

1 Model problem description

2 Components
- Perception in ROS
- Planning in ROS
- Rocot control in ROS

Model problem description

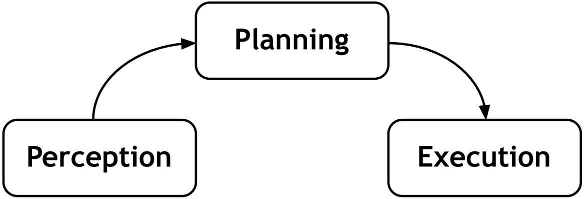
- Pick-and-place task.
- Unknown position of the objects.
- Possibility of collisions.

Components

Perception Detect objects in the image captured by camera and provides their position.

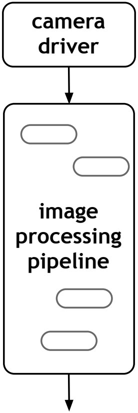
Planning Plan a collision free trajectory to pick and place detected object.

Execution Execute the trajectory on the robot.



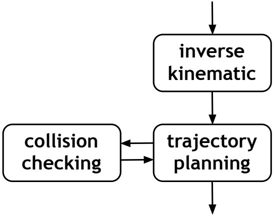
Perception in ROS

- Camera driver
  - output: Image
  - package: camera_drivers, camera1394
- Image processing pipeline
  - input: Image
  - output: ObjectPosition
  - package: image_pipeline, pcl



Planning in ROS

- Inverse kinematics
  - input: Pose
  - output: RobotConfiguration
  - reference: OpenRave, OMPL, CTU
- Trajectory planning
  - input: RobotConfiguration
  - output: Trajectory
  - reference: arm_navigation
- Collision checking
  - input: RobotConfiguration, CollsionModel
  - output: OK/NotOK
  - references: arm_navigation



Robot control in ROS

- Research robots
  - supported: PR2, Nao, TurtleBot, ...
  - reference: Willow Garage
- Industrial robots
  - supported: ABB, Adept, Fanuc, Motoman, Universal
  - promised: Comau, Kuka
  - reference: ROS Industrial