Robot Vision Systems

Lecture 9: ROS History and Basics

Michael Felsberg

michael.felsberg@liu.se

Goals

- System design and programming in
  - –OpenCV 3.0 ~~rc1~~
  - –ROS (robot operating system) ~~Indigo (~~Jade ~~?)~~
- focus part 2: robot vision systems
  - –distributed computing with ROS
  - –efficient use of OpenCV in ROS
- access to robotic hardware is not part of the course – make use of available resources from your lab
- simulation as fallback

Organization

- lectures
  - –systems basics in ROS Jade
  - –Python introduction given by Hannes
- seminars
  - –participants who only participate in the ROS part (and did not read the OpenCV course): one seminar presentation required for credits
- exercises
  - –installation of ROS
  - –going through essential first steps
- project (example application)

Organization

- credits: 9hp if
  - –project work
  - –80% presence
  - –one seminar presentation
- without the project work: 6hp
- note: if you have participated in the course ‘visual computing with OpenCV’, you can only get 6hp (3hp without project)
- 'listen-only’: 0hp

What is ROS?

- Open Source Library for robotics middleware
- Standford AI now supported by Willow Garage / Open Source Robotics Foundation
- Free for use under the open source BSD license (most parts)
- Linux (Ubuntu)
  - Experimental support Win, Mac, other Linux
  - rosjava (platform independent) Android, Matlab

History of ROS

- 2007: “Switchyard” by the Stanford Artificial Intelligence Laboratory
- 2008-2013: development primarily at Willow Garage
- Since 2013: Open Source Robotics Foundation

Versions

- 2010: 1.0, Box Turtle, C Turtle
- 2011: Diamondback, Electric Emys
- 2012: Fuerte, Groovy Galapagos
- 2013: Hydro Medusa
- 2014: Indigo Igloo
- 2015: Jade Turtle

Note: recent versions often require also most recent Ubuntu

Middleware

- Goal (unreachable):
  - –Glue code
  - –Invisible
  - –No overhead / additional constraints
- Coupling of subsystems
  - –Communication
  - –Computation
  - –Configuration
  - –Coordination

ROS Properties

- Heterogeneous computer cluster MW
  - –Hardware abstraction
  - –Low-level device control
  - –Implementation of commonly used functionality
  - –Message-passing between processes
  - –Package management
- Graph architecture, processing in nodes
  - –Receive, post, and multiplex messages
  - –Sensor, control, state, planning, actuator, and other messages

ROS Ecosystem

- Tools for building and distributing ROS-based software
  - –Language- and platform-independent
  - –BSD License
- ROS client library
  - –such as roscpp, rospy, and roslisp
  - –BSD License
- Application packages
  - –Hardware drivers, robot models, datatypes, planning, perception, simultaneous localization and mapping, simulation tools, etc.
  - –Mostly open source licenses

Why Using ROS?

- De-facto standard
- Free to use
- Source code
- Quick bug-fixes
- Conceptually platform independent
- Rapid prototyping with Python
- OpenCV for visual perception
- Many areas and applications

ROS Areas

- Master coordination node
- Publishing or subscribing to data streams: images, stereo, laser, control, actuator, contact ...
- Multiplexing information
- Node creation and destruction
- Seamlessly distributed nodes: distributed operation over multi-core, multi-processor, GPUs, and clusters
- Logging
- Parameter server
- Test systems

ROS Applications

- Perception
- Object Identification
- Segmentation and recognition
- Face recognition
- Gesture recognition
- Motion tracking
- Egomotion
- Motion understanding
- Structure from motion (SFM)
- Stereo vision: depth perception via two cameras
- Motion
- Mobile robotics
- Control
- Planning
- Grasping

How does CVL use ROS?

- Collaboration with other labs
- For building distributed real-time systems
- Driver modularity (avoid own wrappers for hardware APIs, e.g. LadyBug3)
- Code modularity (avoid reinventing the wheel)
- CUAS
- CENTAURO

Installation OS

- ROS requires Ubuntu (Win: VirtualBox)
  - http://releases.ubuntu.com/14.04.2/ubuntu-14.04.2-desktop-amd64.iso
  - http://www.robotappstore.com/Knowledge-Base/ROS-Installation-for-Windows-Users/137.html
  - http://download.virtualbox.org/virtualbox/4.3.20/VirtualBox-4.3.20-96997-Win.exe
  - Install Ubuntu in a new virtual machine (min 16 GB)
  - Resolution will initially be poor; install Guest Additions by clicking 'devices' in the Virtual Machine

Installation Python / OpenCV

- Python installation:
  - –Ubuntu: via package tool
  - –Windows: WinPython 3.4.3.2 (OpenCV 3) http://www.sourceforge.net/projects/winpython/files/WinPython_3.4/3.4.3.2/
  - –3.5 requires alternative packager, e.g. Anaconda
- OpenCV installation
  - –note that Ceemple does not support Python
  - –Ubuntu: via package tool
  - –http://www.cvl.isy.liu.se/education/graduate/opencv/opencv-installation-windows (Win)

Install ROS

- All instructions for Ubuntu Trusty 14.04.2
- Setup sources.list: `$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" $ /etc/apt/sources.list.d/ros-latest.list`
- Setup keys:
  `$ sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net:80 --recv-key 0xB01FA116`
- Installation: `$ sudo apt-get update`
  `$ sudo apt-get install ros-jade-desktop-full`
- Initialize rosdep: `$ sudo rosdep init`
  `$ rosdep update`

Install ROS

- Environment setup: `$ echo "source /opt/ros/jade/setup.bash" >> ~/.bashrc`
  `$ source ~/.bashrc`
- Getting rosinstall:
  `$ sudo apt-get install python-rosinstall`
- Comments:
  - If there are dependency issues, follow the detailed instructions (section 1.4) on http://wiki.ros.org/jade/Installation/Ubuntu
  - Alternatives to Desktop-Full exist, see also 1.4
  - Available packages: `$ apt-cache search ros-jade`

First Steps

- http://wiki.ros.org/ROS/Tutorials
- Workspace
  - –All building happens here
  - –ROS build system is called catkin (old: rosbuild)
  - –`$ mkdir -p ~/catkin_ws/src`
  - –`$ cd ~/catkin_ws/src`
  - –`$ catkin_init_workspace`
- Build command
  - –`$ cd ~/catkin_ws/`
  - –`$ catkin_make`

First Steps

- Workspace contains: src, build, devel
  - To overlay workspace on top of environment:
    ```
    $ source devel/setup.bash
    ```
  - Verify correct overlay:
    ```
    $ echo $ROS_PACKAGE_PATH
    /home/micfe03/catkin_ws/src:/opt/ros/jade/share:/opt/ros/jade/stacks
    ```
  - Detailed information about catkin:
    http://wiki.ros.org/catkin
  - Install ros-tutorials
    ```
    $ sudo apt-get install ros-jade-ros-tutorials
    ```

Second Steps

- Filesystem concepts
  - Packages: software organization unit
  - Manifest: package.xml contains dependencies and meta information
  - Metapackages and VCS repositories replace stacks
- Filesystem tools (use <tab>!)
  - rospack: `# rospack find [package_name]`
    `$ rospack find roscpp`
  - roscd: `# roscd [locationname[/subdir]]`
    `$ roscd log`
  - rosls: `# rosls [locationname[/subdir]]`

Third Steps

- Create ROS Package: minimum two files `CMakeLists.txt`, `package.xml`
- Metapackages: boilerplate `CMakeLists.txt`
- One package per folder – no nesting / sharing of folders

```
workspace_folder/    – WORKSPACE
  src/               – SOURCE SPACE
CMakeLists.txt  – toplevel CMake file
package_1/
  CMakeLists.txt– CMakeLists.txt package_1
  package.xml   – Package manifest package_1
...
package_n/
  CMakeLists.txt– CMakeLists.txt package_n
  package.xml   – Package manifest package_n
```

Create catkin Package

- Within WS (`~/catkin_ws/src`)

```
# catkin_create_pkg
<package_name>
[depend1] [depend2] [depend3]
$ catkin create pkg
beginner_tutorials
std_msgs rospy roscpp
```

- Dependencies (1st order, from manifest)

```
$ rospack depends1 beginner_tutorials
```

- Dependencies (recursive)

```
$ rospack depends beginner_tutorials
```

Manifest package.xml

- Description
  `<description>The beginner tutorials package</description>`
- Maintainer
  `<maintainer email="you@yourdomain.tld">Your Name</maintainer>`
- License
  `<license>BSD</license>`
- Dependencies
  `build_depend, buildtool_depend, run_depend, test_depend`

Example package.xml

```
<?xml version="1.0"?>
<package>
  <name>beginner_tutorials</name>
  <version>1.0.0</version>
  <description>The beginner_tutorials package</description>

  <maintainer email=“michael.felsberg@liu.se">Michael Felsberg</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

</package>
```

Building Packages

- catkin_make:
  `$ catkin_make [make_targets] [-DCMAKE_VARIABLES=...]`
- Comparison CMake – catkin_make

```
# In CMake project
$ mkdir build
$ cd build
$ cmake ..
$ make
$ make install
```

```
# In catkin WS
$ catkin_make
$ catkin_make
install
```

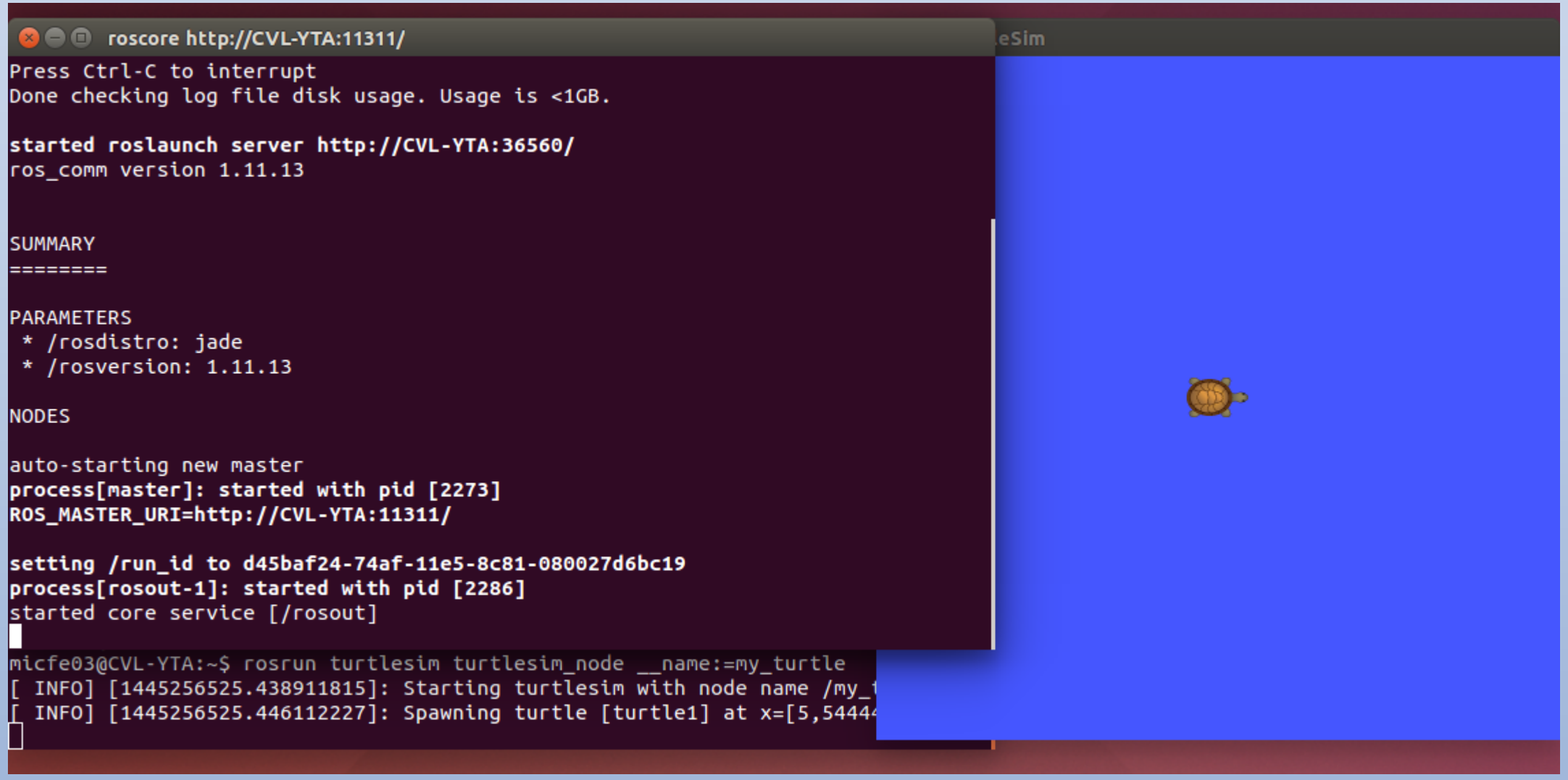
ROS Graph Concept

- Nodes: Executable that uses ROS to communicate with other nodes
- Messages: ROS data type used when subscribing or publishing to a topic
- Topics: Nodes can publish messages or subscribe (receive messages) to a topic
- Master: Name service (NS) for ROS
- rosout: ROS equivalent of stdout/stderr
- roscore: Master + rosout + parameter server

ROS Nodes

- Executable within package
  - Publish/subscribe to a Topic
  - Provide/use a Service
  - Use ROC client library (rospy, roscpp)
- Requires to run roscore: `$ roscore`
- Inspectiontool rosnode:
  - `$ rosnode list`
  - `$ rosnode info /rosout`
- Run nodes:
  - `$ rosrun [package_name] [node_name]`
  - `$ rosrun turtlesim turtlesim_node __name:=my_turtle`

ROS Nodes


roscore http://CVL-YTA:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://CVL-YTA:36560/
ros_comm version 1.11.13
SUMMARY
========
PARAMETERS
* /rosdistro: jade
* /rosversion: 1.11.13
NODES
auto-starting new master
process[master]: started with pid [2273]
ROS_MASTER_URI=http://CVL-YTA:11311/
setting /run_id to d45baf24-74af-11e5-8c81-080027d6bc19
process[rosout-1]: started with pid [2286]
started core service [/rosout]
micfe03@CVL-YTA:~$ rosrun turtlesim turtlesim_node __name:=my_turtle
[INFO] [1445256525.438911815]: Starting turtlesim with node name /my_t
[INFO] [1445256525.446112227]: Spawning turtle [turtle1] at x=[5,5444
eSim

ROS Topics

- Running >=2 nodes: communication

```
$ rosrun turtlesim turtle_teleop_key
```

ROS Topics

- turtlesim_node subscribes to the same topic that turtle_teleop_key publishes to
- Visualization:

```
$ rosrun rqt_graph rqt_graph
```