Cognitive Systems

Introduction to Practical Session

Prof. Dr.-Ing. habil. Alois Knoll
Reinhard Lafrenz, Florian Röhrbein, Sascha Griffiths
Robotics and Embedded Systems (Informatik VI, Prof. Knoll)
TUM

Outline

- Demonstrator overview
- Preparation for practical session
- Basic introduction to ROS
- ROS components for demonstrator
- Live Demo
- ROS tutorial

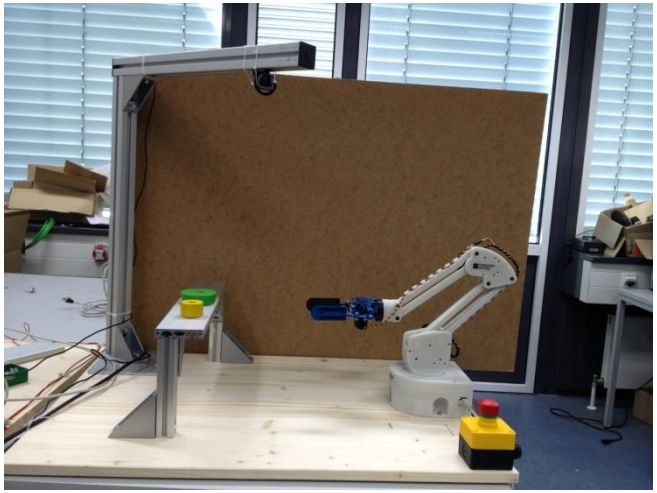
Demonstrator Overview

Gripper Bot

- Pick and Place Task
- Object Detection
- Robot Motion
- Grasping

Resources

- 3DOF Robot
- Calibrated Monocular Camera
- Table with work pieces



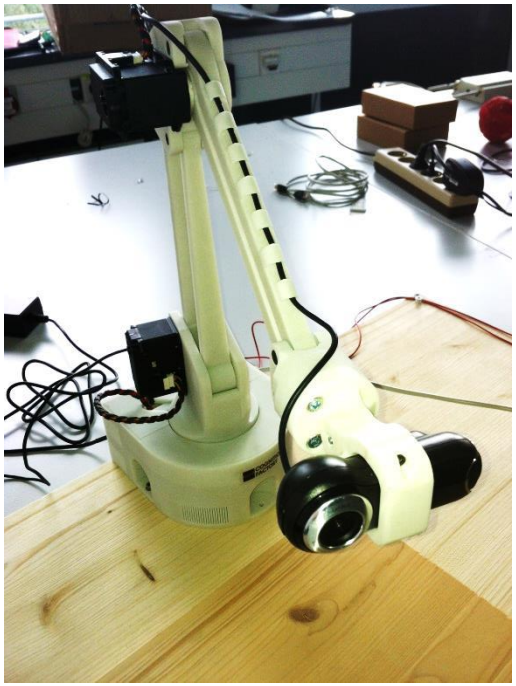
Demonstrator Overview

Cam Bot

- Visual Servoing Task
- Object Tracking
- Robot Motion

Resources

- 3DOF Robot
- Monocular Camera
- Object



Preparation for Practical Session

Requirements

- Groups of ~5
- Min 1 laptop per group
- Min 1 C++ GURU per group ☺ (optional: JAVA)
- Virtual box installation (https://www.virtualbox.org/)
- Setup OS image from memory sticks provided
- Boot the image and check installation

Basic Introduction to ROS

- Meta operating system serving as a middleware
- Linux OS, Limited support for Windows
- Open Source under BSD License (core parts)
- Developed by Willowgarage
- Provides communication infrastructure for information exchange
  - Synchronous (Client/Server)
  - Asynchronous (Publish/Subscribe)
- Custom text based Message and Service description language
- Central resource registry called roscore
- C++/JAVA/Python/Prolog/Lisp/JavaScript

ROS Packages

- Container for software components
- Cmake based build system
- Dependencies on other ROS packages through Manifest (xml file) and system dependencies rosdep definitions (yaml file)
- Command line tool rosmake builds ROS packages
- Command line tool rosrun to run ROS packages
- Command line tool roslaunch to startup multiple nodes ROS packages which are defined in a xml based launch file

ROS Messages

- Messages are exchanged through ROS topics
- ROS topic is identified by a topic name
- Publish and Subscribe mechanisms for exchanging data on message topics
- ROS topics is connected to one specific message type
- Text based custom message definitions
- Availability of basic data types (int, float, bool etc) which can be aggregated to custom structures

ROS Service

- Hosted service is identified by a service name
- ROS service is connected to one specific service type
- Text based custom service definition consisting of request and response definitions
- Service implementation through callback functions
- ROS Client is blocked until response is generated

ROS Utilities

- roscd <package_name>: change to directory of ros package
- rosmake <package_name>: make ros package
- rosservice
  - rosservice list : lists all registered services
  - rosservice call : manually issue a service call from a command line
  - ..
- rostopic
  - rostopic list : lists all registered topics
  - rostopic echo <topicname> : listen to a rostopic
  - ..

ROS Components for Demonstrator

ROS workspace location: ~/fuerte_workspace/cognitivesystems

Collection of ros packages serving as software building blocks for the demonstrators:

Cambot	Gripperbot
camera	camera
object_tracker	object_detector
robot_control	robot_control
ros_tum_msgs	gripper_control
ros_comm_lib	ros_tum_msgs
	ros_comm_lib

ROS Package

camera

- Function: Grabbing images from camera and publishing them
- Interface
  - ROS Topic: /camera/image
  - ROS Service: /ros_tum_msgs/CameraInfo
- Message:
  - sensor_msgs/Image
  - ros_tum_msgs/CameraCalibData

ROS Package

object_detector

- Function: Detects 3D position of workpieces on the table wrt reference frame
- Interface: ROS Topic /object_detector/objects_data
- Msg: /ros_tum_msgs/ActorVec
  - ActorVec: array of message ros_tum_msg/Actor
  - Actor:
    - geometry_msgs/Pose[] targetPoseVec
    - string targetType
    - string targetProperty
    - string timeStamp
    - int32 targetId

ROS Package

object_tracker

- Function: Tracks image space position of target within camera view
- Interface: ROS Topic /object_tracker/object_data
- Msg: /ros_tum_msgs/ActorVec
  - ActorVec: array of message ros_tum_msg/Actor
  - Actor:
    - geometry_msgs/Pose[] targetPoseVec
    - string targetType
    - string targetProperty
    - string timeStamp
    - int32 targetId

ROS Package

robot_control

- Function: Hosts services to move robot in Operational/Configuration space
- Interface:
  - ROS Service /cambot_control/move_to_os
  - ROS Service /gripperbot_control/move_to_os
- Service
  - MoveToOS
    - request
      - string effector (must be „gripper“ or „wrist“)
      - float32 x,y,z
    - response
      - bool success (if target position could be reached or not)

ROS Package

gripper_control

- Function: Hosts services to open/close gripper
- Interface:
  - ROS Service /gripper_control/open_gripper
  - ROS Service /gripper_control/close_gripper
- Service
  - OpenGripper (empty list)
  - CloseGripper (empty list)

ROS Package

ros_tum_msgs

- Custom message type and service definitions

ros_comm_lib

- Convenience wrappers and utility functions

LIVE DEMO

Hands on Part

ROS TUTORIAL

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

(Please use **fuerte** release ONLY!)