ROS - Practical Session 4

ROS Workshop at the School of Engineering, UNAM

Viktor Seib

vseib@uni-koblenz.de

Institute for Computational Visualistics
University of Koblenz-Landau

August 13th, 2015

COMPUTERVISUALISTIK

UNIVERSITÄT KOBLENZ · LANDAU

AGAS

Task 1: Kinect to Laser

Task 2: Simple Leg Detection

Task 3: Turtle TF Frames ++

Task 1: Kinect to Laser

Task 2: Simple Leg Detection

Task 3: Turtle TF Frames ++

Kinect to Laser

Task:

- install kinect drivers ...
  ```
  $ sudo apt-get install ros-hydro-openni-camera
    ros-hydro-openni-launch
  ```
- ... or use the provided bagfile
  ```
  $ rosbag play kinect_data.bag -l
  ```
- create a package `kinect_to_laser_data` with the dependencies `roscpp`, `sensor_msgs`, `pcl_ros`
- install the Point Cloud Library (`$ sudo apt-get install libpcl-1.7-all libpcl-1.7-all-dev`)
- write a node `kinect_to_laser_data` that receives the message type `sensor_msgs/PointCloud2` from the kinect

Kinect to Laser

Task:

- the node should take the central horizontal line from the received 3D point cloud and convert it to a message of type `sensor_msgs/LaserScan`
- then send the LaserScan message out (topic name e.g. “/scan”)
- create a launchfile to start all necessary nodes and rviz to visualize the results

Hint:

- you can use the provided file `kinect_to_laser_data.cpp` and fill in all lines marked as “TODO”

Task 1: Kinect to Laser

Task 2: Simple Leg Detection

Task 3: Turtle TF Frames ++

Simple Leg Detection

Task:

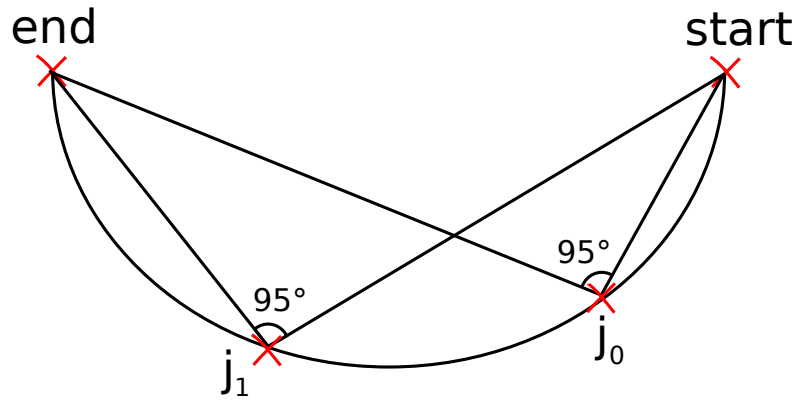
- create a new node `simple_leg_detection` and a package with the same name and the dependencies `roscpp`, `sensor_msgs`, and `visualization_msgs`
- the node should receive laser data on topic `\scan` from the provided bagfile, detect legs in the laser data and visualize them using `rviz`
- legs should be visualized in `rviz` using the message `visualization_msgs/Marker` of type `SPHERE_LIST`
- the marker should be send on topic `visualization_marker`
- create a launchfile to start your new node, the provided bagfile and `rviz` with the provided configuration file

Simple Leg Detection

Task:

- you can use e.g. the following simple strategy to detect legs
  1. find segments in the laser data that could represent different objects
  2. this is the case if two neigboring points are further away than a certain threshold
  3. only consider segments that are big/small enough to represent legs
  4. test whether a potential leg segment is shaped like a half-circle
  5. if this is the case, you have found a leg

Simple Leg Detection



Task:

- you can use e.g. the following simple strategy to test whether a segment is a half-circle
  1. find the beginning `start` and end `end` of a segment
  2. calculate the angle between the vector from `j` to `start` and the vector from `end` to `j`
  3. `j` is iterated over each point in the segment between `start` and `end`
  4. if the average angle is between 70 and 130 degrees, the segment is a half-circle
  5. the center of the half-circle can be approximated as the center between `start` and `end`

Simple Leg Detection

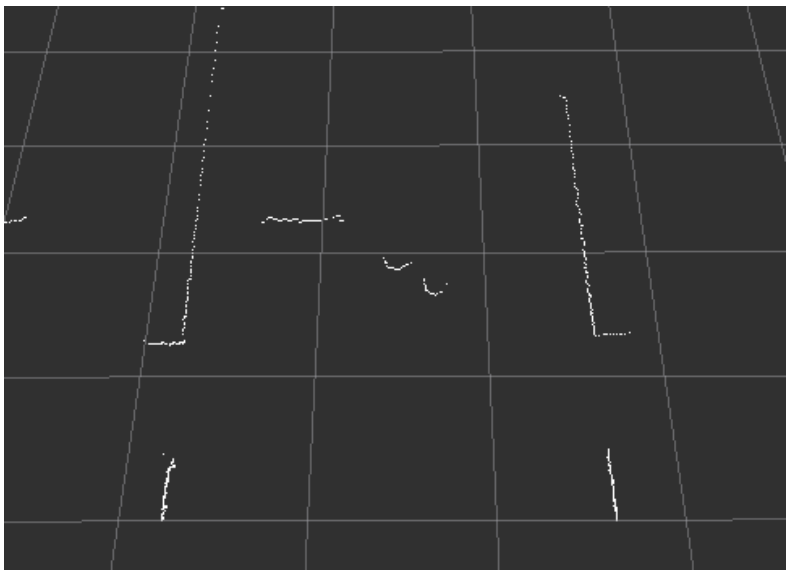
Hints:

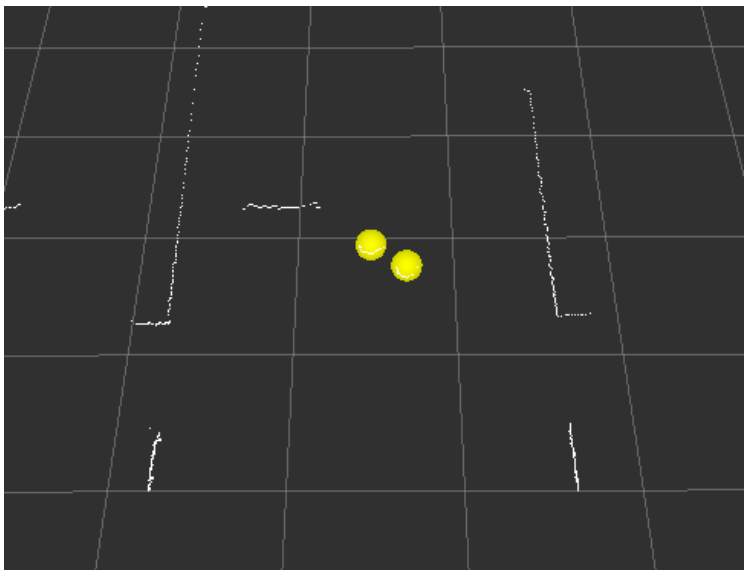
- you can use the provided file `simple_leg_detection.cpp` and fill in all methods marked as “TODO”
- if you use the provided file `simple_leg_detection.cpp`, also use the provided file `CMakeLists.txt`
- take a look at the documentation of the `sensor_msgs::LaserScan` message at `http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html`
- take a look at the description of the `visualization_msgs::Marker` message at `http://docs.ros.org/api/visualization_msgs/html/msg/Marker.html`

Simple Leg Detection

Goal:

left: laser data, right: detected legs





Task 1: Kinect to Laser

Task 2: Simple Leg Detection

Task 3: Turtle TF Frames ++

Simple Leg Detection

Task:

- combine the tasks
  - Session 3: Turtle TF Frames
  - Session 3: Turtle URDF
- extend the *Turtle TF Frames* for the turtle to have four legs
- connect the turtle description from *Turtle URDF* to the published TF frames
- while moving, the graphical turtle should move (like the frames in *Turtle TF Frames*)
- place your new node in the already existing package `turtle_tf_frames`

Simple Leg Detection

Task:

- instead of publishing the previous transforms for the legs, a *JointStatePublisher* needs to be used
- it has to publish a message of type `sensor_msgs::JointState` on the topic `/joint_states`
- the joint states to be published correspond to the joint names defined in the turtle URDF file
- note: in the URDF file you additionally need to specify a rotation axis for each joint (`<axis xyz="0 1 0"/>`)

Hint:

- you can use the provided file `turtle_tf_frames_extended.cpp` and fill in all lines marked as “TODO”

Simple Leg Detection

Goal: