MIDWA Lab 4
Working with images in ROS.
Programming an image republisher

Goal

In this lab you will practice image subscription and publishing in ROS. You will program a ros node that subscribes to one of the camera images of the robot and after overlaying your names onto the image it will be re-published with diferent topic name so it can be visualized in the screen of the robot.

Deliverables

- Sheet of the lab, filled with the answers to the questions
- ROS package **baxter_img**

Send them, zipped together, as an attachment to the following address:

`salvador_8fb7@sendtodropbox.com`

Create a single zip file with all deliverables inside. Naming convention will be **`Name1_Name2_code.zip`** where code is a random string of four characters to make sure your submission cannot be overwritten, e.g. « Smith_Baker_xyrt.zip ».

No need to include any message body. Your file will be automatically date-stamped and added to a specific folder of my DropBox account. Beware!

Existing material

During the labs we will use all the time the documentation provided by the maintainers of ROS officially through the ROS wiki : http://wiki.ros.org and apply it to our robot Baxter.

First task: familiarizing with Baxter's camera images

- Visualize camera images of the right and left arms using **rviz**. Which are the names and type of those topics? Explain how you found them.
- Look for the name of the topic of the screen's image, (hint: use search pattern « **screen** ». Which are its name and type?
- Which images are being published in the robot?

Second task: Programming an image republisher

In this task you will create a ros node that listens to the image topic « **/image_in** », overlays a text in the center of the screen given as parameter and publishes the result with image topic name « **/image_out** ».

Write a launcher that :

- remaps the topic « **/image_in** » into the topic name of one of the camera images of the robot

- reads the text to show as a parameter
- remaps the topic « **/image_out** » into the topic name of the screen's image.

Instructions

- Create the ros package for this lab:

  ```
  >> cd ~/catkin_ws/src

  >>catkin_create_pkg baxter_img
  ```

- Create a src folder inside the package and create an empty file inside called **baxter_img_node.cpp**

  ```
  >>roscd baxter_img

  >>mkdir ./src

  >>gedit baxter_img_node.cpp
  ```

- To know how to write an image subscriber and publisher take a look at the examples in the ROS wiki. To access these examples, simply type « *image subscriber* » or « *image publisher* » respectively in the search box. Notice that instead of showing the image on your computer you will publish it with a different name. You have to register the publisher with the topic name "/image_out". Notice also that, in order to be accesible from inside the image callback, the publisher should be declared as a global variable.

- To overlay some text on the image you can use the Opencv function:

  ```
  cv::putText(const_cast<cv::Mat&>(img), text, cv::Point2f(100,100), CV_FONT_HERSHEY_SIMPLEX,
  1.0, cv::Scalar(255, 255, 255), 2, CV_AA);
  ```

  where img is the opencv image and text is the string variable read as parameter

- Modify the files package.xml and CMakeList.txt conveniently to specify the dependencies and the necessary compilation directives to create the executable. Compile it.

  ```
  >>cd ~/catkin_ws

  >>catkin_make
  ```

- Create the launcher to remap the input and output images. Execute the launcher and check if the image is shown on the screen of the robot with the text. Be sure that no other student is sending images on it.

- When finished, show it to the teacher for validation.

At the end of the lab session: Zip your package folders and save them to your pen drive or by mailing it to yourself. Anything that remains on the PC will be deleted and replaced by the initial version without code for the next group.