Using the ZED Camera with ROS

Objectives

In this laboratory exercise, you learn how to use the ZED camera with ROS. You will create a simple publisher and subscriber to access the ZED camera photos in real-time and publish them without modification.

Deliverables

- ❏ Use the ZED camera to capture video in the ROS environment.
- ❏ Write echo.py - a program that subscribes and immediately publishes the image it receives

Challenge Problems

- ❏ Create shape_echo.py - Directly modify the pixels in the image message to draw a static shape overlay before publishing the image
- ❏ create flip_echo.py - flip the image before publishing it. Create a rosparamter to dynamically switch between flip vertically or horizontally.
- ❏ Create instagram_echo.py - Increase the red hue of every pixel to give it a rosy appearance

Using the ZED Camera

Launch the zed camera:

```
roslaunch zed_wrapper zed.launch
```

Verify you can see the image from the zed camera. Using rqt image view and select the following topic from the drop down menu:

```
"/camera/rgb/image_rect_color"
```

Gather Test data

Launch the teleoperation and the zed_ros_wrapper. Use rosbag to record all topics and drive the car around the different markers we have provided.

```
rosbag record -O '/data/racecar/<INSERT_FILENAME_NAME_HERE>.bag' /tf
/odom /scan /camera/rgb/image_rect_color /camera/rgb/camera_info
```

If you're having trouble, you can use the instructor provided rosbag.

```
# NOTE: This code must be run on your LOCAL VM.
# This code will save the provided rosbag in /data/racecar on your
local VM.
sudo mkdir -p /data/racecar
sudo chown racecar /data/racecar
cd /data/racecar
wget
http://dl.dropboxusercontent.com/u/380036122/BWSI/ROS_BAGS/moving_blo
b_test.bag
```

Use rosbag to playback the data you collected. In the following steps, use the playback to assist with writing `echo.py`

Write echo.py -- subscriber

Write a ros subscriber for the zed camera topic:

```
/camera/rgb/image_rect_color
```

The zed camera publishes message of type Image from sensor msgs. You can learn about this message type by opening the file Image.msg located in

```
roscd sensor_msgs/msg
```

Image.msg has the information about how to access the pixels of the image.

Write echo.py -- publisher

Create a rospy publisher for the topic "`image_echo`" of message type `sensor_msgs Image.`

Publish the image message you immediately receive in the callback function for the Rospy Subscriber you wrote in the previous step.

Test echo.py

Launch teleoperation, zed_ros_wrapper, and echo.py. Use rviz or rosbag to view/record the data. Verify echo.py works.

Use OpenCV Bridge

Looking ahead, we are going to use OpenCV to allow to use many sophisticated vision algorithms. The first step is converting an image from a rostopic for use with OpenCV. ROS has a package, called cv_bridge, that provides this exact functionality.

Review the cv_bridge python tutorial here : http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

Then, modify echo.py to convert the sensor_msgs/Image to an OpenCV image. Then, convert it back and publish it. Use rqt_image_view to verify that this program works the same as before.