

Programming Familiarization Part 3

ROS Packages

The main goal of this assignment is to familiarize you with various ROS packages like tf2, apriltag_ros, hector_trajectory_server, and a tiny bit of OpenCV. It is essential that you form an understanding of how to look up documentation and integrate ROS packages all on your own.

In this assignment you will be making use of the apriltag_ros package to detect the pose of AprilTags in a video stream (contained in a bag file) and publishing those poses to the tf2 server (see motion_decoder.cpp). You will also be taking those images, drawing circles on them and republishing them (see ImageConverter.hpp). In order to tie the whole thing together, you will then write another two launch files for your hector trajectory server; one will draw the trajectory of the april tag frame moving in space relative to the camera frame. The other will draw the trajectory of the camera moving in space relative to the april tag frame. You will then modify the existing motion decoder launch file to launch (one of) your new hector trajectory server launch file(s) and have it draw the motion of the april tag frame relative to the camera frame. Then clone that one and modify it to draw the motion of the camera frame relative to the april tag frame.

Deliverables:

1. All source code; that is, your motion_decoder node, ImageConverter header, and 4 launch files. (zip up the *src* folder in your workspace that contains the relevant packages but DO NOT include the bagfiles)
 - a. Hector_trajectory_camera_moving.launch
 - b. Hector_trajectory_apriltag_moving.launch
 - c. Motion_decoder_camera_moving.launch
 - d. Motion_decoder_apriltag_moving.launch
2. Screenshots of images with the green circles on them (3 of them at different intervals), and screenshots of RVIZ showing the trajectory the april tag took and the trajectory the camera took (3 of them at the end of the run but at different angles, also show the TF frame names). Add these to your report and label them appropriately.
3. Short video clips of the end result. (Trajectories shown in RVIZ and the green circles drawn on the video feed). Send us a google drive share link or an unlisted youtube link. Ensure that the link is sharable
4. Answers to the short questions at the end of the assignment in a pdf file with the name Team[Letter]_AndrewID_TaskN.Part3.pdf where N is the task number on Canvas.
5. Zip all of the above into a file named Team[Letter]_AndrewID_TaskN.Part3.zip where N is the task number on Canvas (e.g., TeamC_jdolan_Task7.Part3.zip).

General Setup

1. Attempt to install `apriltag_ros`, `apriltags` and the `hector_trajectory_server` package using `apt-get`. Run the command:

```
sudo apt-get install ros-your distro-apriltag-ros  
sudo apt-get install ros-your distro-apriltag  
sudo apt-get install ros-your distro-hector-trajectory-server
```

Note: This is the step where different distros might cause some issues. It is up to you to find the correct packages that provide either the same functionalities or similar functionalities.

Ever since ROS Melodic, the name of the package has changed from `apriltag_ros` to `apriltag_ros`, either package is fine, but indicate in your report. If need be, include the package with source code in your submission so that it can be `catkin_make` together with your `motion_decoder`.

`Apriltags` and `apriltag_ros/apriltags_ros` may not be available for all distributions of ROS. If you are not able to install these packages via `apt-get`, you can install them from source. Go to your project `src` folder, once there you can use `git` to clone the packages into the directory by running:

```
$ git clone https://github.com/RIVeR-Lab/apriltags_ros.git
```

This repository contains both `apriltags` and `aptriltags_ros`.

2. Unzip the assignment package. You should see a package called `motion_decoder`. In the package, you should have the starter code and the folder for the necessary bag files.
Reminder: DO NOT upload the bag file in your submission zip folder.
 - a. Download the bag files that are provided separately, unzip them and put them in the bags folder.
 - i. <https://drive.google.com/file/d/16Dv--OQoUM9skOV385t5GRvZHowx21jO/view?usp=sharing>
 - ii. <https://drive.google.com/file/d/1aMpW5myRE7IFNXi04vJ0IFGH2JZkgXM7/view?usp=sharing>

Part 1:

1. The first step is to make the `motion_decoder` node subscribe to the Apriltag Detection messages. In order to complete this, you will need to find the message structure of Apriltags detection. (Hint: `rosmmsg`). You will also need to figure out the topic name for the detection messages generated by the `apriltags_ros` node. You will not need to manually start the `apriltags_ros` node. It is automatically started via the launch file provided.
2. The next step is to extract the pose information from the Apriltag detection message and make it into a Transform.
3. Once you have the transform, then you need to broadcast the transform with the correct frame id and child frame id.
4. The camera frame must be named "camera" and the apriltag frame must be named "april_tf".

Part 2:

1. For this part, you will need to manipulate some images. Once you have the detections from the `apriltags_ros` node, you need to overlay a shape over the locations where the Apriltag was found in the image. There is no need for exact locations. A simple way would be to draw small circles based on the path taken by the Apriltag. This should be done for every frame where it detects the Apriltag.
2. Once you are done manipulating the images, you will need to figure out the code to convert from OpenCV image format to ROS `sensor_msgs Image` format and publish the modified image as a `sensor_msgs Image`. (Hint: it's just one or two lines of code and can be easily found in the tutorials.)

Part 3:

1. Generate two different hector server launch files. One launch file draws the trajectory taken by the Apriltag and the other launch file draws the trajectory taken by the camera. (Hint: This should be straightforward. The only change that you will need to make between the two launch files is to change the frames.)
2. Include one of each of the two hector server launch files into the appropriate `motion_decoder` launch files. I.e. when I launch `motion_decoder`, the correct hector launch file is automatically launched together.

Part 4:

1. Create the `rosgaph` for this exercise.
2. Hypothetically, how could you use these packages in your project?
3. What did you think of this assignment? Was it helpful to give you a little taste of ROS packages? Was the difficulty manageable?