ELEC 5660: Introduction to Aerial Robotics
Project 2: Phase 1

Assigned: Mar. 16th, 2021 Due: 11:59 PM Mar. 23rd, 2021

1 Project Work

In Project 2 Phase 1, you need to implement the 3D-2D pose estimation algorithm learned in the lecture to estimate the camera's poses with images. This is an individual project, which means you must complete it by yourself.

1.1 Project Assignments

Three assignments are required for this project including:

1. Calculating the camera's pose corresponding to every image.
2. Publishing camera pose information in the form of `nav_msgs/Odometry`.
3. Plotting these poses with **rqt_rviz**.
4. Comparing your result with the reference.

Note: You are **NOT** allowed to use any OpenCV functions in your implementation.

1.2 Project Details

You will be provided with a ROS package named `tag_detector`, where a serial of points and their positions will be calculated with images. You need to implement this project based on the point and position array. You can follow the below procedures to prepare for your coding.

1. put `aruco-1.2.4` and `tag_detector` in your workspace (`catkin_ws/src/`)
2. install aruco (following `aruco-1.2.4/README`)
3. Setup your ROS environment and compile the `tag_detector` package.
4. Find `bag_tag.launch` in `tag_detector/launch`, and `images.bag` in `tag_detector/bag`.
5. Use `bag_tag.launch` and `images.bag` to run this package (roslaunch `bag_tag.launch`).
6. Read the comments in `tag_detector/src/tag_detector_node.cpp` **carefully**.

7. Add your code into `tag_detector/src/tag_detector_node.cpp`.

8. Note that the pose you calculated is $(\mathbf{t_{cw}}, \mathbf{R_{cw}})$, which represents the pose of world frame respecting to the camera frame.

2 Submission

When you complete the tasks you could submit your code and documents to canvas before **Mar. 23rd, 2021 23:59:00**. The title of your submission should be "`YOUR-NAME_YOUR-STUDENT-ID`".

Your submission should contain:

1. A **maximum 2-page** document including:
   (a) Figures plotted by **rviz**.
   (b) Statistics about your result. (For example, RMS error between the poses you calculate with the reference ones)
   (c) Descriptions about your implementation.
   (d) Any other things we should be aware of.
2. Files `tag_detector_node.cpp`, as well as any other c++ files you need to run your code.

3 Basic use of RVIZ

1. open one terminal, input:roscore; open another termianl, input: rosrun rviz rviz.
2. click add button, add your topic.
3. change the frame to "world".
4. change the color of your odometry and reference odometry.

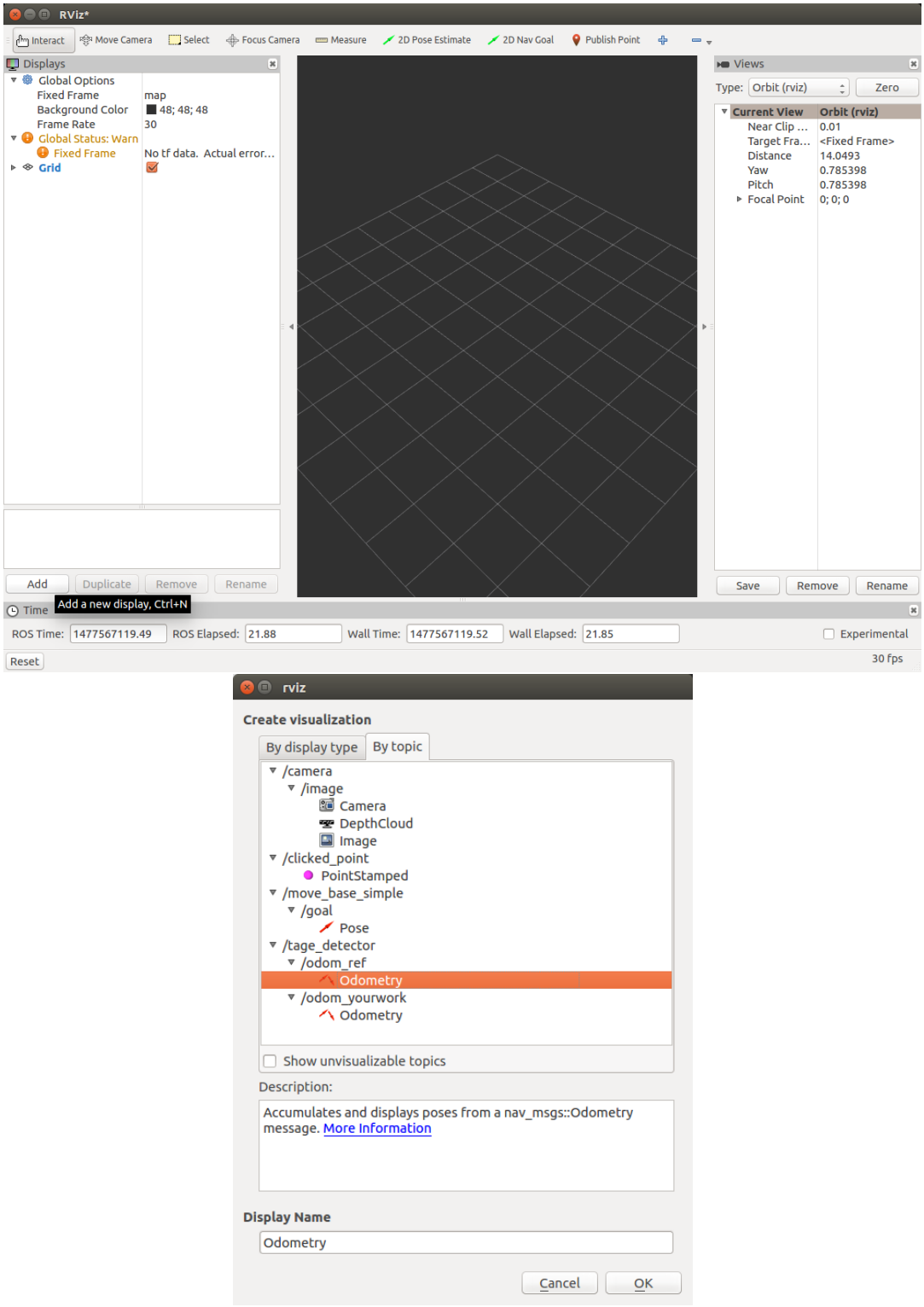

Figure 1: step 2

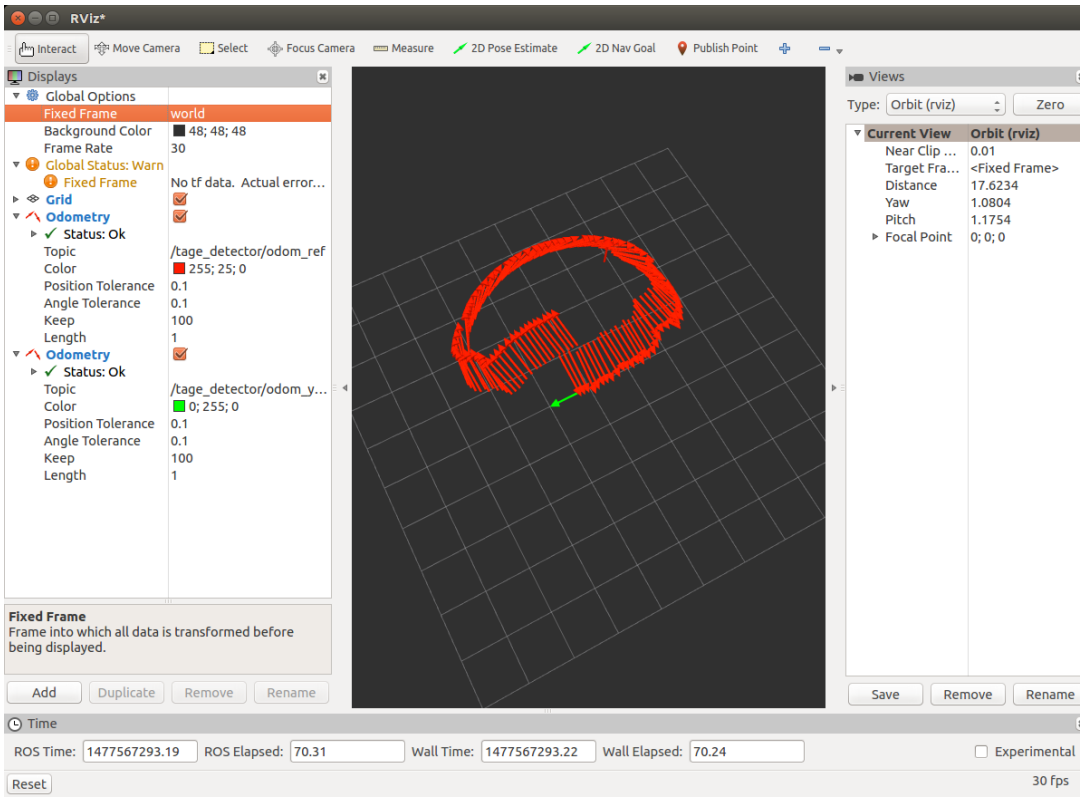

Figure 2: step 3