F1TENTH Autonomous Racing **(Due Date:)**

Lab 8: Perception and Vision

Instructor: INSTRUCTOR *Name:* STUDENT NAME, *StudentID:* ID

Course Policy: Read all the instructions below carefully before you start working on the assignment, and before you make a submission. All sources of material must be cited. The University Academic Code of Conduct will be strictly enforced.

THIS IS A GROUP ASSIGNMENT. Submit one from each team.

1 Learning outcomes

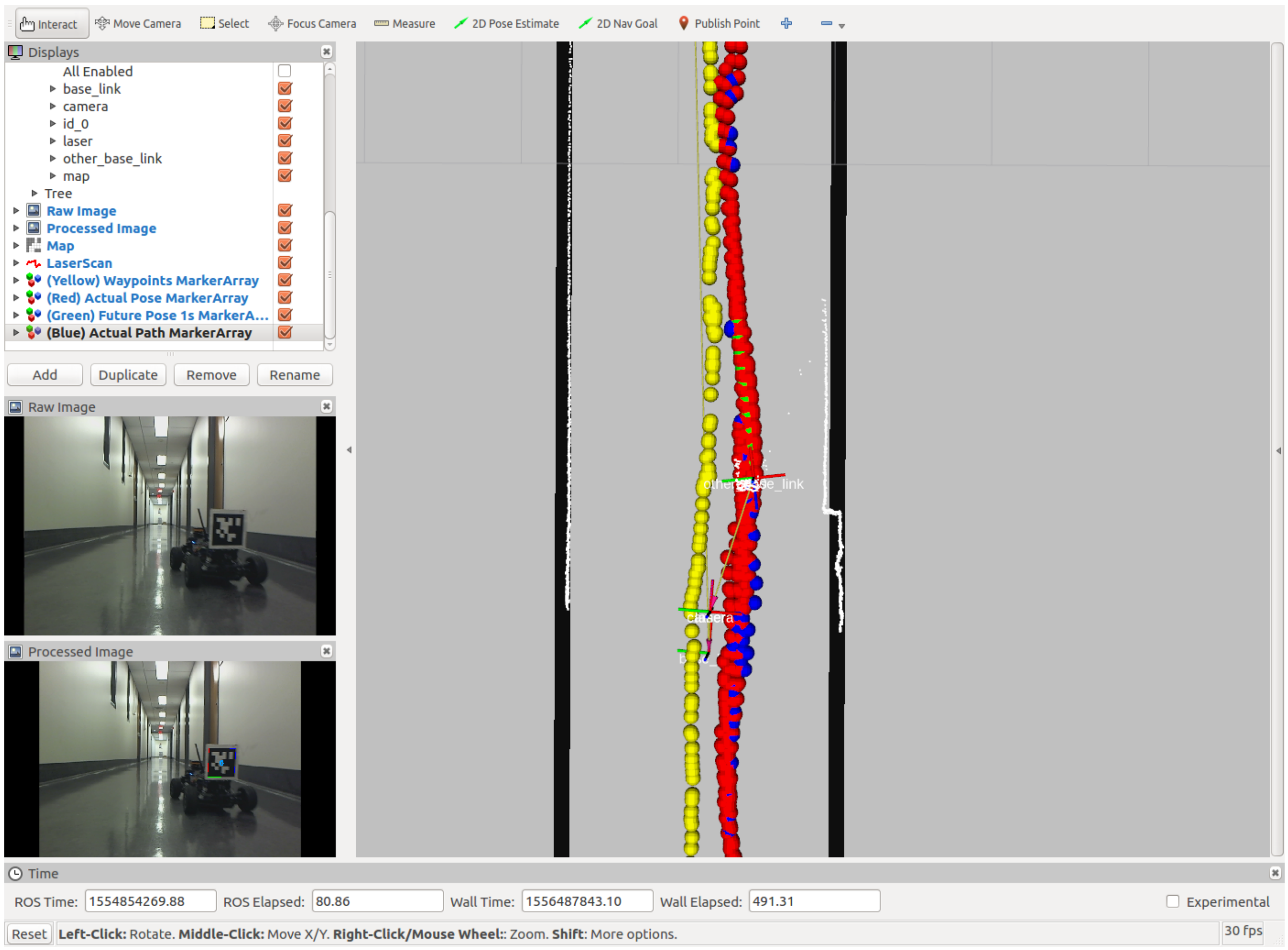
The following fundamentals should be understood by the students upon completion of this lab:

- *Camera model and parameters:* You should be able to understand how the real camera model and the pin hole camera model are related and what assumptions are made while going from the former to the latter.
- *Transformations:* You should be able to understand 3 dimensional transformations and know the representations of transformations.
- *Single View Geometry:* You should be able to understand and work with single view geometry and how world coordinate frame points translate to camera coordinate frame points.
- *Homography:* You should be able to work with homography and how it can be used to calculate pose of the camera given the world coordinate correspondences.
- Working with images on ROS
- Transformations using tf and implementation of AprilTags (and similar) librarie(s).
- Implementing nodelets in ROS and their

2 Overview

Most racing series feature multiple vehicles competing simultaneously on a single track. As such, safely overtaking and navigating in the presence of other vehicles is paramount to performing well.

Since the strategy and decisions of of other vehicles cannot be known ahead of time and may, in fact, be stochastic it is necessary for your car to react to new constraints imposed by other vehicles online. Thus, the goal of this lab is to track and predict the pose of an opponent vehicle (see Fig. 1)



ROS Time: 1554854269.88 ROS Elapsed: 80.86 Wall Time: 1556487843.10 Wall Elapsed: 491.31

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options.

Figure 1: Tracking and predicting the pose of an opponent

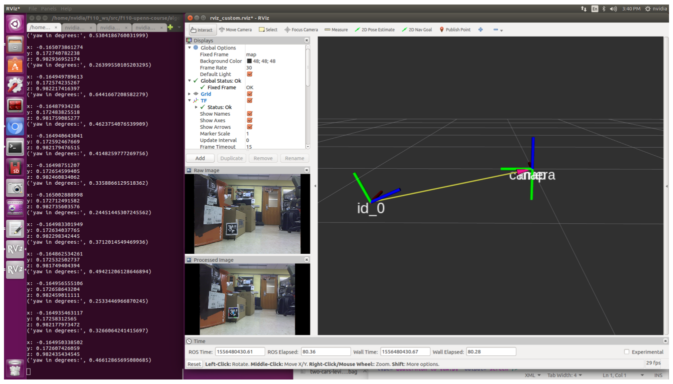
3 Working with AprilTags

3.1 Download the required packages

You can download the package containing the required repositories from this link. Link

3.2 Obtain camera parameters

The camera parameters and rviz parameters are present in the config/ folder of the repository.



```
('yaw in degrees:', 0.5304186760031999)

x: -0.165073861274
y: 0.17274078223 8
z: 0.982936952174
('yaw in degrees:', 0.2639955010520329 5)

x: -0.164949789613
y: 0.172574235267
z: 0.982217416397
('yaw in degrees:', 0.6441667208582279)

x: -0.16487934236
y: 0.172483825518
z: 0.981759085277
('yaw in degrees:', 0.462375407653990 9)

x: -0.164940643041
y: 0.172592467669
z: 0.982179476515
('yaw in degrees:', 0.4148259777269756)

x: -0.16498751207
y: 0.172654599405
z: 0.982460834062
('yaw in degrees:', 0.335886612951836 2)

x: -0.165002888998
y: 0.172712491582
z: 0.982735603576
('yaw in degrees:', 0.2445144530724556 2)

x: -0.164983301949
y: 0.172634037765
z: 0.982298342445
('yaw in degrees:', 0.371201454946993 6)

x: -0.164862534261
y: 0.172532502737
z: 0.981749404394
('yaw in degrees:', 0.4942120612864689 4)

x: -0.164956555106
y: 0.172658643204
z: 0.982459011111
('yaw in degrees:', 0.253344696687024 5)

x: -0.164935463117
y: 0.17258312565
z: 0.982177973472
('yaw in degrees:', 0.326606424141569 7)

x: -0.164950338502
y: 0.172607426059
z: 0.982435434545
('yaw in degrees:', 0.4661286569508068 5)
```

Global Options	
Fixed Frame	map
Background Color	48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: Ok	
Fixed Frame	OK
Grid	✓
TF	✓
Status: Ok	
Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Update Interval	0
Frame Timeout	15

ROS Time: 1556480430.61 ROS Elapsed: 80.36 Wall Time: 1556480430.67 Wall Elapsed: 80.28

Figure 2: Detecting April Tags over Rviz

3.3 Install apriltag_ros

Download the required base repositories from this link base April Tags. There are instructions for setting up the base april tags repository. First set up the AprilTag repositories, then You will just have to include the folders in the zip file in you src/ folder and compile your workspace.

```
$ catkin_make
```

3.4 Set up vehicle tracker and prediction repository

Once you have AprilTags setup you have to work on the vehicle_tracking_prediction repository. Verify that there are no missing dependancies.If the dependencies are installed correctly there should be no compiler errors.
Now you will be able to test the basic functionality of the AprilTags library. First, start playback of the recorded sensor data:

```
$ rosbag play <bagfile>
```

Then launch the AprilTag detector:

```
$ roslaunch vehicle_tracker_prediction_skeleton detect_apriltags.la
```

The next step is to modify the following files in the repository

```
1. Vehicle_Track.cpp
```

The following tasks need to be performed by the pose predictor node:

- It should subscribe to /tag_detections topic with AprilTagsArray message type.
- It should implement the weighted prediction algorithm described in lecture.
- You should tune the weighted prediction hyperparameters.
- It should publish the poses to a topic called /pose_predictions which will have a markerarray message type.

3.5 Deliverables

You should submit the updated vehicle tracker package and clear instructions on how to run your code including any additional dependencies or libraries (e.g. Pure Pursuit implementation).

4 Convert Nodes to Nodelets

Using nodelets is a standard practice when it comes to working with images in ROS. Transporting image messages over the standard TCP protocol for pub/sub in ROS creates significant delays. Instead of creating copies of images and passing the data around, nodelets creates shared memory between nodes and pass the pointer to the image in memory instead under a nodelet manager. The apriltag_ros wrapper we provided does not run on nodelets, and is not fast enough to provide a robust detection framework. You'll need to convert the bare apriltag_ros package to run on nodelets. **Warning:** Please do not copy any existing code that you can find on the internet to convert to nodelets. We will know if you copied from existing repositories online.

References:
ROS wiki
Blog on porting nodes to nodelets
Example nodelet
Clearpath robotics tutorial

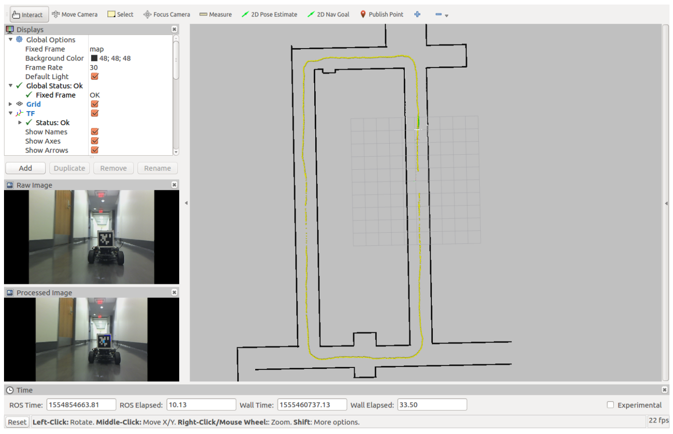
Here are a few steps and tips that you can take to get started.

- Everything you need to care about is in the ROS package inside the folder apriltags2_ros
- Since we are only using the continuous_detector node you can only convert this node into a nodelet.
- First, edit the continous_detector.h file. You will have to make a shared pointer object for the TagDetector Object and also a shared pointed object image_transoirt object.
- You will have to add the nodelet spec will ifications in the main function in the apriltags2_ros_continous_node.cpp file.
- In continuous_node.cpp you will have explort the pluginlib class list macros and specify the class to be exported. You will have to make the oninit() function and initialize the node-handles, tag detector objects, image transport objects, etc. inside it. You can now leave the constructor empty. Ypu might have to make some changes in the image callback function too.
- in the cmake list you will have to take care of the dependencies such as pluginlib and nodelet. Make sure your executable and libraries are defined correctly for the continuous node detector
- You will create a new .xml file called the nodelet_plugins.xml file and specify the library path and the class path.
- You will change the package.xml file to specify the dependencies of nodelets and pluginlib and also export the nodelet_plugins.xml file. This will be loaded during runtime.

5 Test Framework

We have provided a launch file `bag_prediction.launch` to test your system using a bag file. You may have to change some parameters in the launch file. The launch file will launch RViz in order to display the markers showing the poses which are estimated by the algorithm. Record a video of

your predictions as displayed in RViz. In addition compare the frequency of messages published on the detected poses topic with and without the use of nodelets.
In the launch file you will have to add the launch commands for the executable you create. The scripts/ folder also contains utilities that you can use to save and/or publish markers of the waypoints, predicted poses and actual poses. Feel free to use them.



Global Options
Fixed Frame map
Background Color ■ 48; 48; 48
Frame Rate 30
Default Light ✓
Global Status: Ok
Fixed Frame OK
Grid ✓
TF ✓
Status: Ok
Show Names ✓
Show Axes ✓
Show Arrows ✓

ROS Time: 1554854663.81 ROS Elapsed: 10.13 Wall Time: 1555460737.13 Wall Elapsed: 33.50 Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options. 22 fps

Figure 3: Predicting Poses over Rviz

6 Extra Credit

You may already be doing some of these things for your project, this is some encouragement to get started early. The mechanical design and TX2 kernel support tasks have slightly flexible deadlines, see your TAs. The other tasks must be submitted with your lab to receive the bonus points. Extra credit will be awarded on an individual basis. If multiple team members participate you must use git to track contributions and supply the teaching staff with relevant commit history and a summary of who did what. The teaching staff's decisions about partially completed tasks and sufficiency of contributions by an individual working in a team are final.

6.1 Mechanical Design

(10 pts added to lowest lab score)
Create a 3D printed mount for the realsense or logitech cameras

(10 pts added to lowest lab score)
Create a mount for placing an AprilTag on the back of the car (3D printed and laser cut)

(10 pts added to lowest lab score)
Fabricate the designs for the rest of the class once submitted (you need certification from MEAM to use these tools).

6.2 Improved Prediction

(5 pts added to lowest lab score)

Use RRT, RRT*, or follow-the-gap instead of pure-pursuit within the prediction algorithm. Can you weight the predictions from multiple algorithms? How would you decide the weighting etc.

(10 pts added to lowest lab score)
Use MPCC within the prediction algorithm.

6.3 Update the TX2 Kernel to support RealSense Camera

(5 pts added to lowest lab score)
Provide a script and verify with the TAs so we can distribute this to the rest of the class. Be careful, hacking without understanding can brick your TX2.

6.4 YOLO pipeline

(10 pts added to lowest lab score)
Replace the AprilTags pipeline with YOLO keep output of detected poses the same etc.

7 Deliverables and Submission

Submit the following as `groupnumber_lab8.zip` (replace `number` with your groupnumber):

1. Your nodelet implementation of the vehicle tracker package and clear instructions on how to run your code including any additional dependencies or libraries (e.g. Pure Pursuit implementation).
2. A youtube video of your detection and prediction pipeline. Add this link to a text file named `groupnumber_lab8_video.txt`
3. A plot showing the mean and variance of the detected poses publishing frequency.

8 Grading

8.1 Rubric

Topics	**Points**
Compilation	10
Tracking implementation	30
tracking performance on turns	10
Nodelets Implementation	25
Nodelets: improvement in performance	10
Video Provided	10
Clear Instructions and Readme	5
Extra credit	0
Total	100