State Estimation Lab

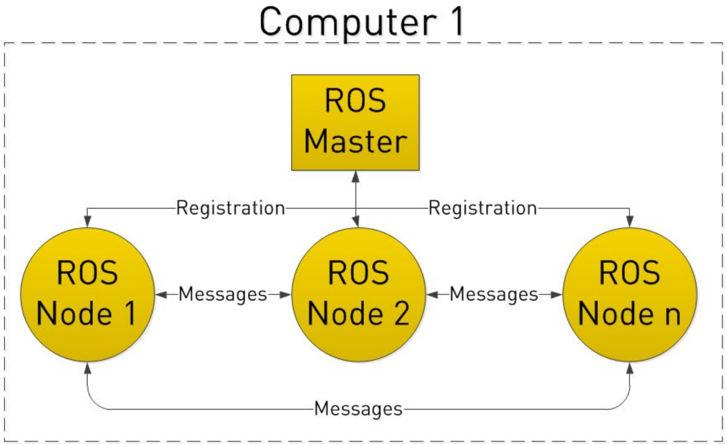
Lab 1

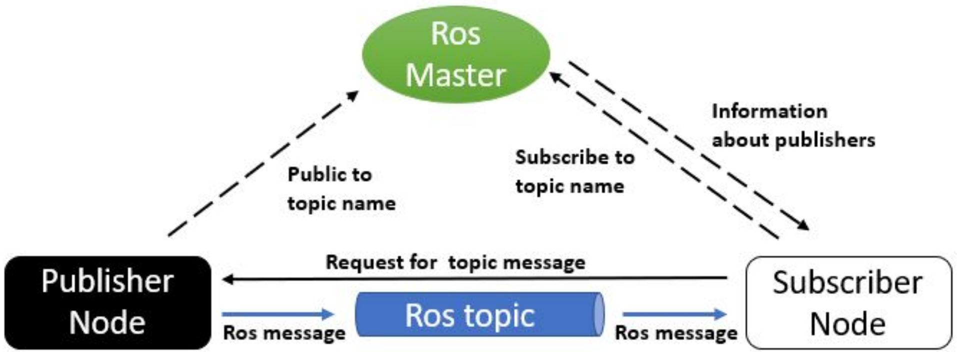
Overview

- ROS Introduction
- Coordinate Systems (Extrinsic, Intrinsic Matrix)
- AprilTag, Calibration application
- Lab
  - Lab 1.1 Find camera intrinsic matrix
  - Lab 1.2 Pose estimation using single or multiple Apriltags

ROS Introduction

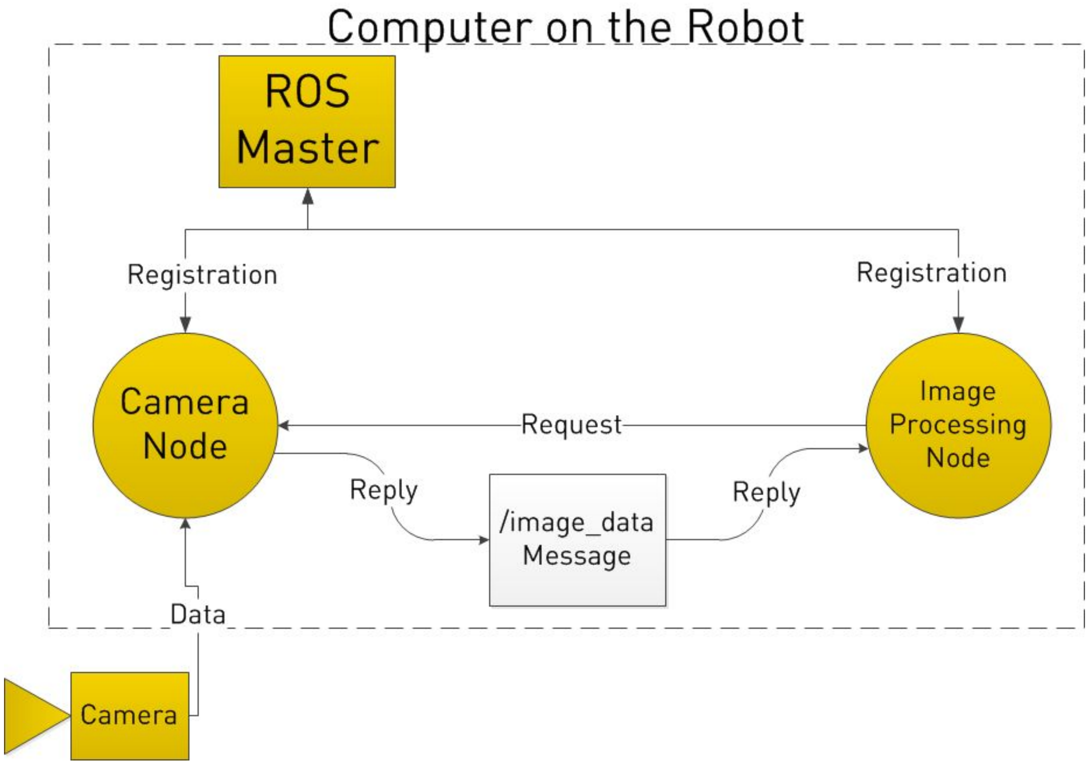
- Comprised of a number of independent nodes.
- Each communicates with others by publishing and subscribing to topics.





Please ensure that the ROS installation is completed in the prelab.

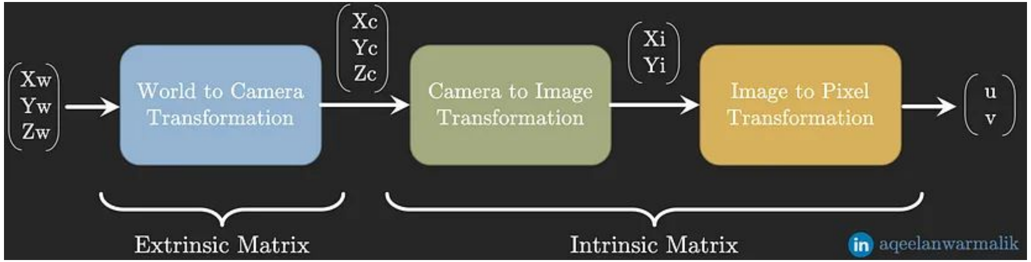
Example

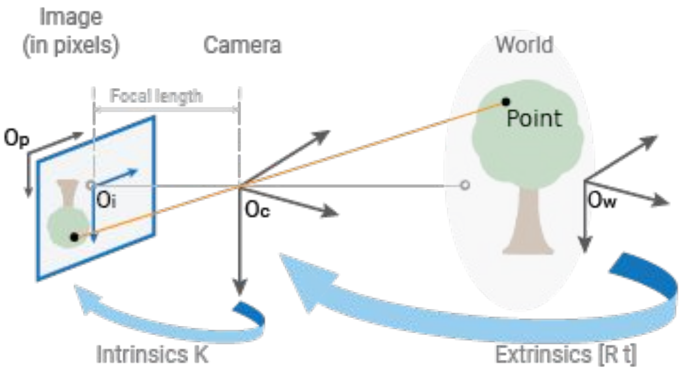


Computer on the Robot
ROS Master
Registration
Registration
Camera Node
Request
Image Processing Node
Reply
/image_data Message
Reply
Data
Camera

Coordinate Systems

- Extrinsic: parameters of a camera depend on location and orientation (Pose)
  - World (3D) → Camera coordinate system (3D)
- Intrinsic: parameters of a camera depend on how it captures the images, such as focal length, aperture, field-of-view, resolution
  - Camera (3D) → Image (2D) → Pixel coordinate system (2D)

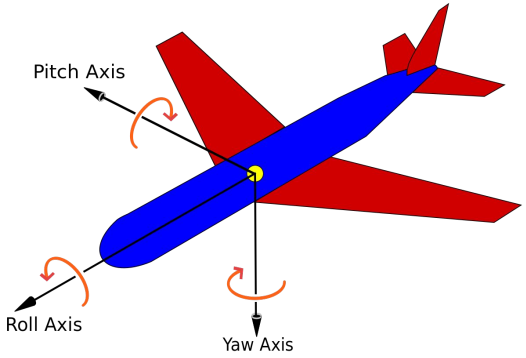


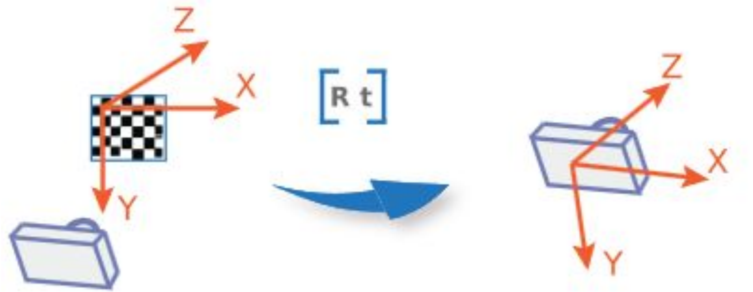


World (3D) → Pixel coordinate system (2D)

Extrinsic Matrix

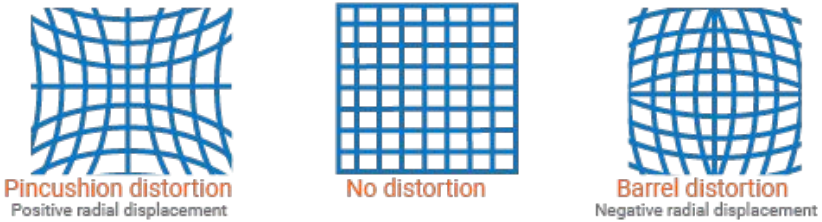
- World (3D) → Camera coordinate system (3D)
- Consist of a rotation matrix **R** and a translation matrix **t**.
- Pose estimation

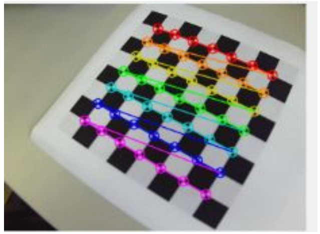




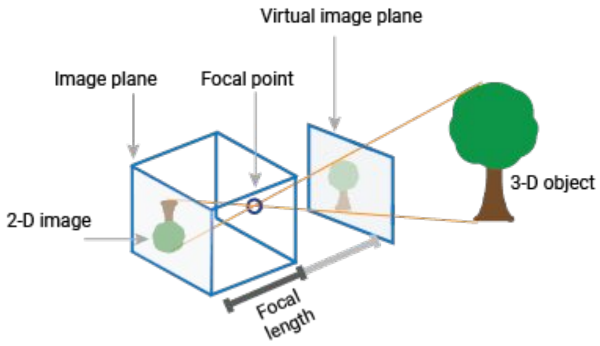
$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$

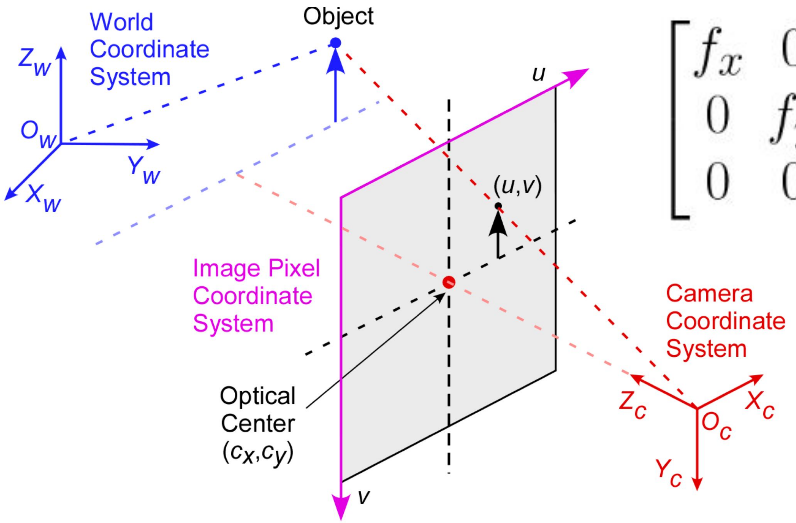
Intrinsic Matrix





- Camera (3D) → Image (2D) → Pixel coordinate system (2D)
- Pinhole camera model
- Distortion

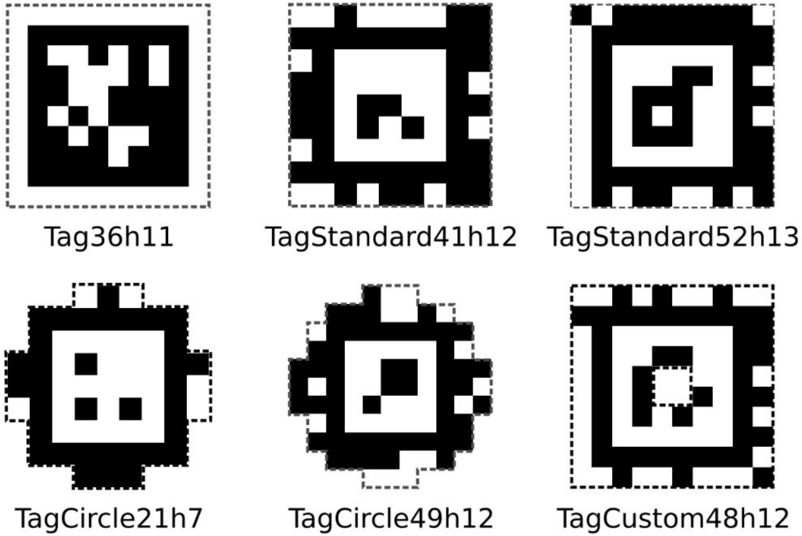




$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

AprilTag

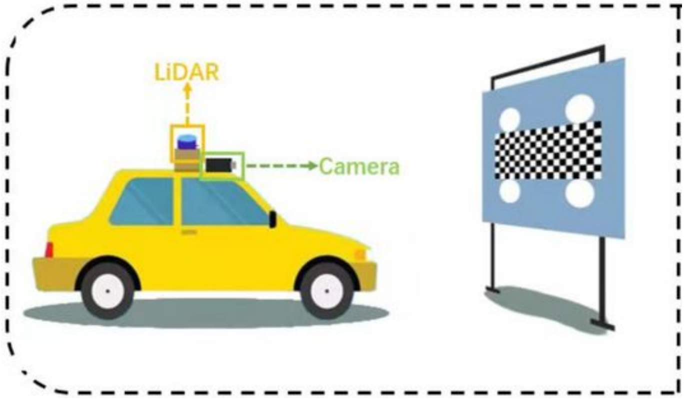
- Two-dimensional bar code, similar to QR Codes.
- Smaller data payloads, while still being detectable at longer distances.
- Can compute 3D pose (position, orientation), and tags ID relative to camera.

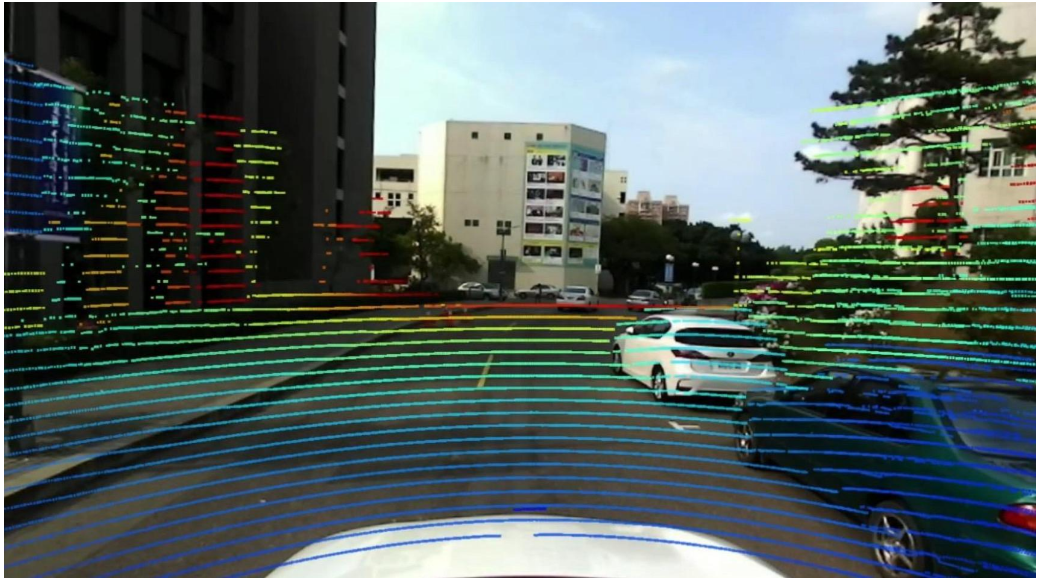


extrinsic

Calibration application

Intrinsic, Extrinsic, Temporal





[Hella Gutmann - 360° calibration for Advanced Driver Assist Systems (ADAS) (youtube.com)]

Lab

Preparations before Lab

Download files from E3

Install packages:

pip install rospkg

pip install pupil-apriltags

pip install opencv-python

Install VSCode (Optional):

sudo apt update

sudo apt install snapd

sudo snap install code --classic

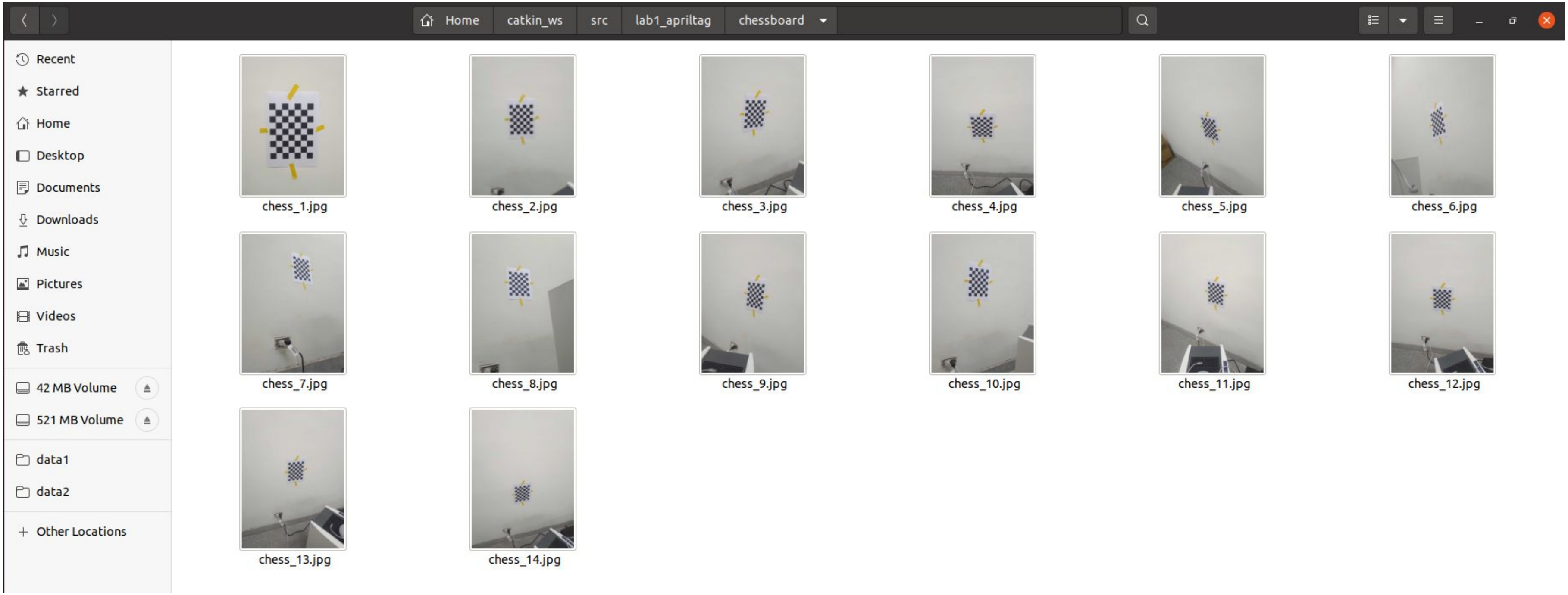
Lab

1.1 Get camera’s intrinsic matrix and distortion coefficient

1.2 Pose estimation using single or multiple AprilTags.

Lab 1.1 - Find camera intrinsic matrix

Capture multiple images of a chessboard using the camera and save them in a folder.



Lab 1.1 - Find camera intrinsic matrix

Open camera_calibration.py and modify the path to the folder containing the chessboard images.

```python
# Load images
images_folder = ???
images_list = os.listdir(images_folder)
```

Lab 1.1 - Find camera intrinsic matrix

Finished the code here, so it can show the value of fx, fy, cx, cy.

```python
# Print the camera matrix
print(f"Objective function value: {ret}\n")
print(f"Distorition coefficient: {dist}\n")
print("Camera Matrix:")
print(mtx)

fx = mtx[?, ?]
fy = mtx[?, ?]
cx = mtx[?, ?]
cy = mtx[?, ?]

print(f"\nfx: {fx}, fy: {fy}, cx: {cx}, cy: {cy}")
```

Lab 1.1 - Find camera intrinsic matrix

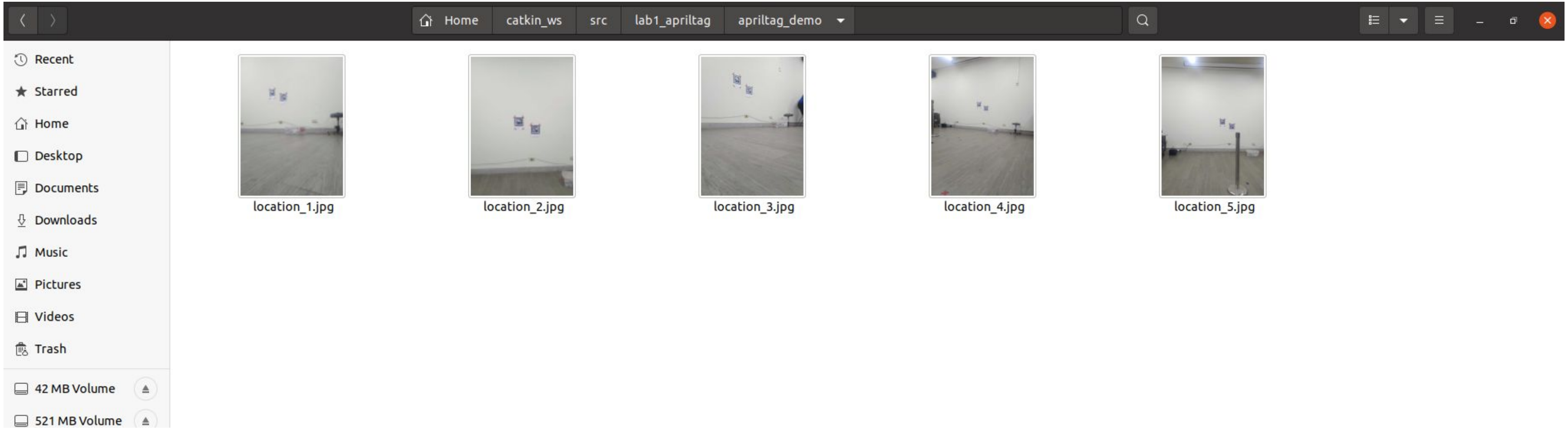
Open the terminal and type "python camera_calibration.py" Then the program will start executing and computing the intrinsic matrix and distortion coefficients.

```
~/catkin_ws/src/lab1_apriltag/src ❯ python camera_calibration.py
Objective function value: 0.7488936896790611

Distorition coefficient: [[ 0.04884705  0.08326766  0.00990691 -0.00287781  0.51442997]]

Camera Matrix:
[[3.13524341e+03 0.00000000e+00 1.48682923e+03]
 [0.00000000e+00 3.13972523e+03 2.06891569e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

fx: 3135.243406828593, fy: 3139.7252280414996, cx: 1486.8292332587046, cy: 2068.915688921479
```

Record the values of fx, fy, cx, cy, and distortion coefficient. These values will be used in the next program.

Lab 1.2 - Pose estimation using single or multiple Apriltags

Take pictures at the designated locations (5 places) and save them in a folder. Each image should include two AprilTags.



Lab 1.2 - Pose estimation using single or multiple Apriltags

1. apriltag_id_3.csv

   This file contains the world coordinates of the center of the AprilTag with ID = 3.

2. apriltag_id_4.csv

   This file contains the world coordinates of the center of the AprilTag with ID = 4.

3. apriltag_id_3_4.csv

   This file contains the world coordinates of the center of the AprilTag with ID = 3 and ID = 4.

4. benchmark.csv

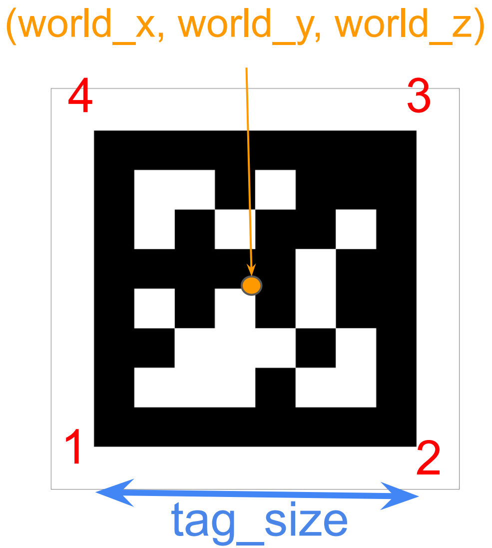
   This file contains the benchmark pose information.

Lab 1.2 - Pose estimation using single or multiple Apriltags

Please finish the function below. It takes the center of the AprilTag in the world coordinates and the size of the tag as input. The output of this function will be the corner points of the AprilTag in the world coordinates.

Note: The order of the points is important! In the counter-clockwise order, the first point will be at the bottom left.

```
def get_apriltag_corners(world_x, world_y, world_z, tag_size):

    return np.array([[??, ??, ??],   # First point
                     [??, ??, ??],   # Second point
                     [??, ??, ??],   # Third point
                     [??, ??, ??]],  # Fourth point
                     dtype = np.float32)
```



Lab 1.2 - Pose estimation using single or multiple Apriltags

Setting the following parameters

- APRILTAG_SIZE :
  width of the AprilTag, measured in meters.

- APRILTAG_LOCATION_PATH:
  path to apriltag_id_3.csv 、apriltag_id_4.csv or apriltag_id_3_4.csv.

  When the path is set to apriltag_id_3.csv, it means that we estimate the pose by only using the apriltag with id = 3. When the path is set to apriltag_id_3_4.csv, it means we estimate the pose by using two apriltags.

```
# Apriltag parameter
APRILTAG_SIZE = ???
APRILTAG_LOCATION_PATH = ???
```

Lab 1.2 - Pose estimation using single or multiple Apriltags

- camera parameters :
  FX, FY, CX, CY, DIST_COEFF.

```
# Camera parameters
FX = ???
FY = ???
CX = ???
CY = ???

DIST_COEFF = np.array([?, ?, ?, ?, ?])
```

- benchmark_path:
  path to benchmark.csv

```
# Get Benchmark
bench_mark_path = ???
bench_mark_pose_list = read_csv(bench_mark_path)
```

Lab 1.2 - Pose estimation using single or multiple Apriltags

- Image_folder:
  Path to the folder that contains the AprilTags image

```
# Get camera pose from image
Image_folder = ???
image_name_list = os.listdir(Image_folder)
```

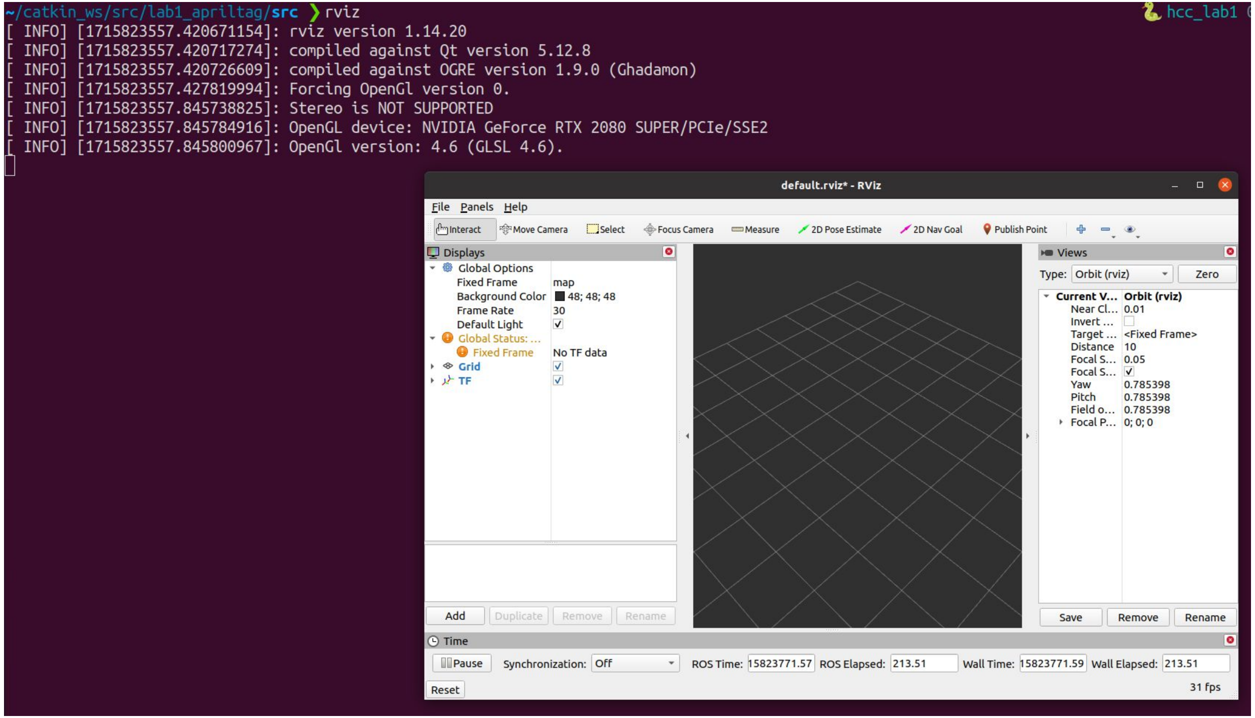
- Output_csv_file
  Path to the estimated output pose

```
## Write the result to csv
Output_csv_file = ???
header = ['x', 'y', 'z', 'roll', 'pitch', 'yaw']
with open(Output_csv_file, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(header)
    writer.writerows(content)
```

Lab 1.2 - Pose estimation using single or multiple Apriltags

execute the code

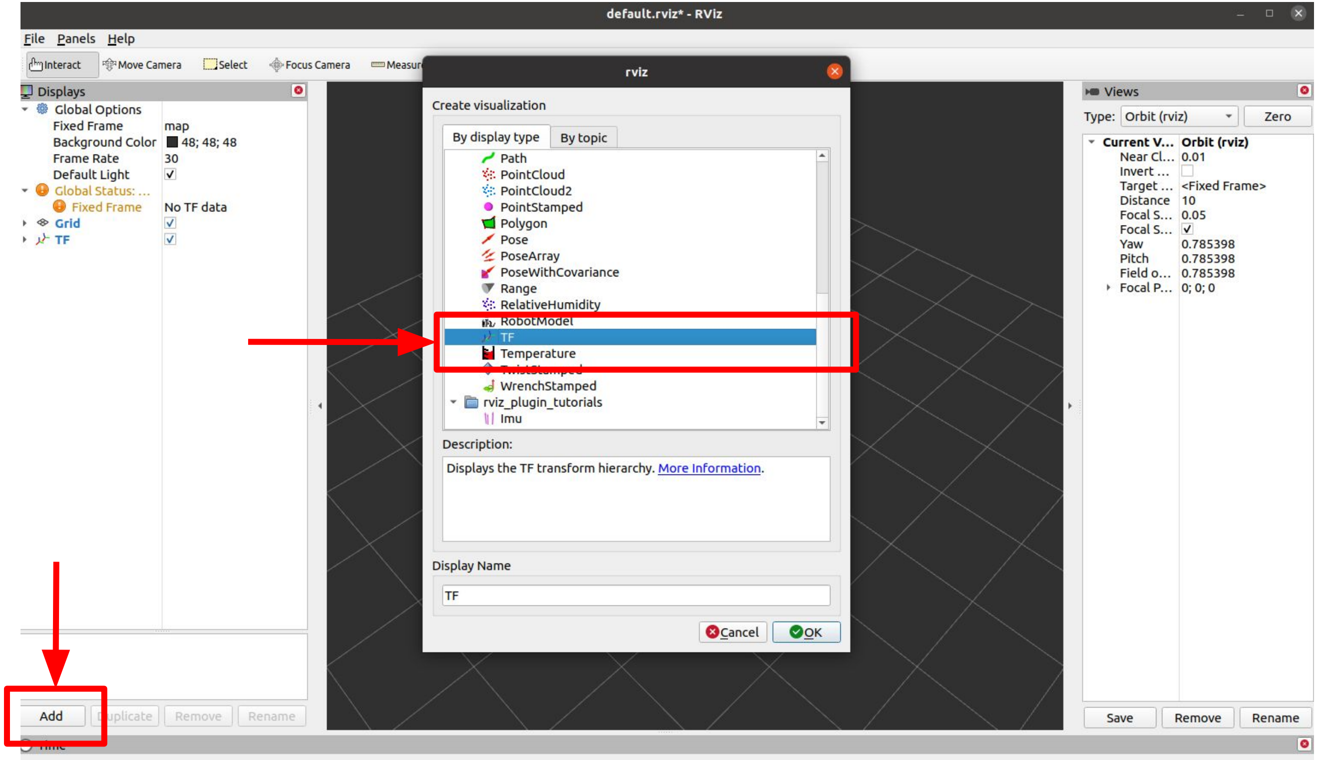
- open the terminal and type “rviz”



Lab 1.2 - Pose estimation using single or multiple Apriltags

execute the code

- add the TF topic



Lab 1.2 - Pose estimation using single or multiple Apriltags

execute the code

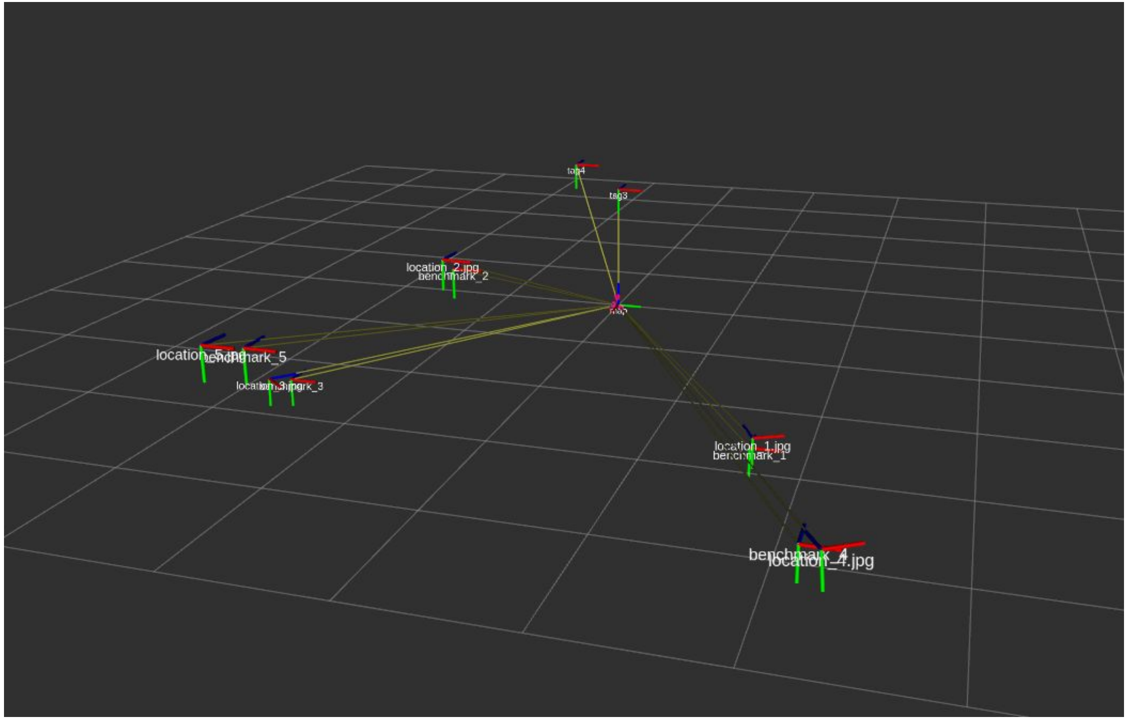
- open the terminal and type “python lab1_apriltag.py”

```
~/catkin_ws/src/lab1_apriltag/src ❯ python lab1_apriltag.py
[INFO] [1715824058.396524]: Publish tag3 coordinate
[INFO] [1715824059.399218]: Publish tag4 coordinate
[INFO] [1715824060.404208]: Publish benchmark_1
[INFO] [1715824061.409240]: Publish benchmark_2
[INFO] [1715824062.414610]: Publish benchmark_3
[INFO] [1715824063.419648]: Publish benchmark_4
[INFO] [1715824064.424326]: Publish benchmark_5
[INFO] [1715824065.734655]: Publish location_3.jpg
[INFO] [1715824066.975111]: Publish location_2.jpg
[INFO] [1715824068.262690]: Publish location_4.jpg
[INFO] [1715824069.543793]: Publish location_5.jpg
[INFO] [1715824070.807137]: Publish location_1.jpg
[INFO] [1715824071.810216]: CSV file saved to /home/ee904/catkin_ws/src/lab1_apriltag/result/camera_pose.csv
```

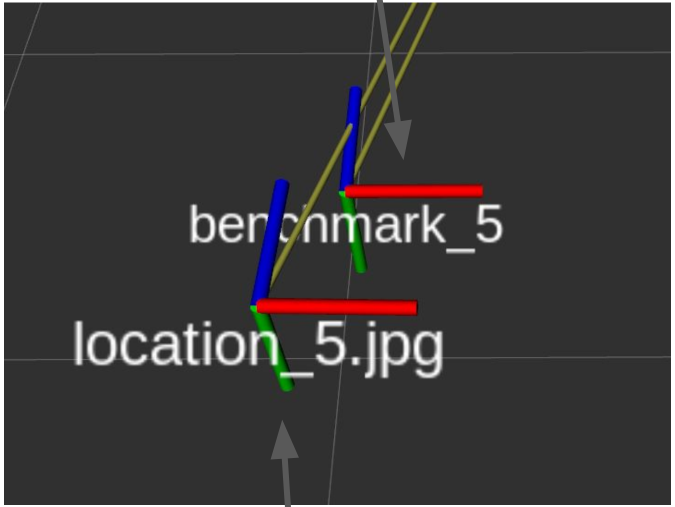
Lab 1.2 - Pose estimation using single or multiple Apriltags

execute the code

- In the rviz window



Benchmark pose



Estimation pose

Lab 1.2 - Pose estimation using single or multiple Apriltags

After the code stops executing, please change the APRILTAG_LOCATION_PATH parameter so that we can observe the difference by using a different number of tags to calculate the pose.

You will receive 3 CSV results by using different parameters of APRILTAG_LOCATION_PATH.

Lab 1.2 - Pose estimation using single or multiple Apriltags

Evaluate the pose using a benchmark.

Open the evaluate.py, and modify the following parameters.

- BENCHMARK_PATH: Path to benchmark.csv
- PRED_PATH: The output csv file from lab1_apriltag.py

```
BENCHMARK_PATH = ???
PRED_PATH = ???
```

Open the terminal and type “python evaluate.py”,

The result shows the average distance error along x-axis, y-axis and z-axis.

```
~/catkin_ws/src/lab1_apriltag/src ❯ python evaluate.py
Average error:
x: 0.18468398000000014 m
y: 0.06532835799999995 m
z: 0.098115924 m
```

Lab 1.2 - Pose estimation using single or multiple Apriltags

Answer the following question.

1. Which result is better: using one Apriltag or using two Apriltags for pose estimation?

2. If there were no benchmark provided today, what result would you prefer to use as the result for pose estimation, and why?

Reference

Intro to ROS — ROS Tutorials 0.5.2 documentation (clearpathrobotics.com)

AprilTag Introduction — FIRST Tech Challenge Docs 0.2 documentation (firstinspires.org)

What Is Camera Calibration? - MATLAB & Simulink (mathworks.com)