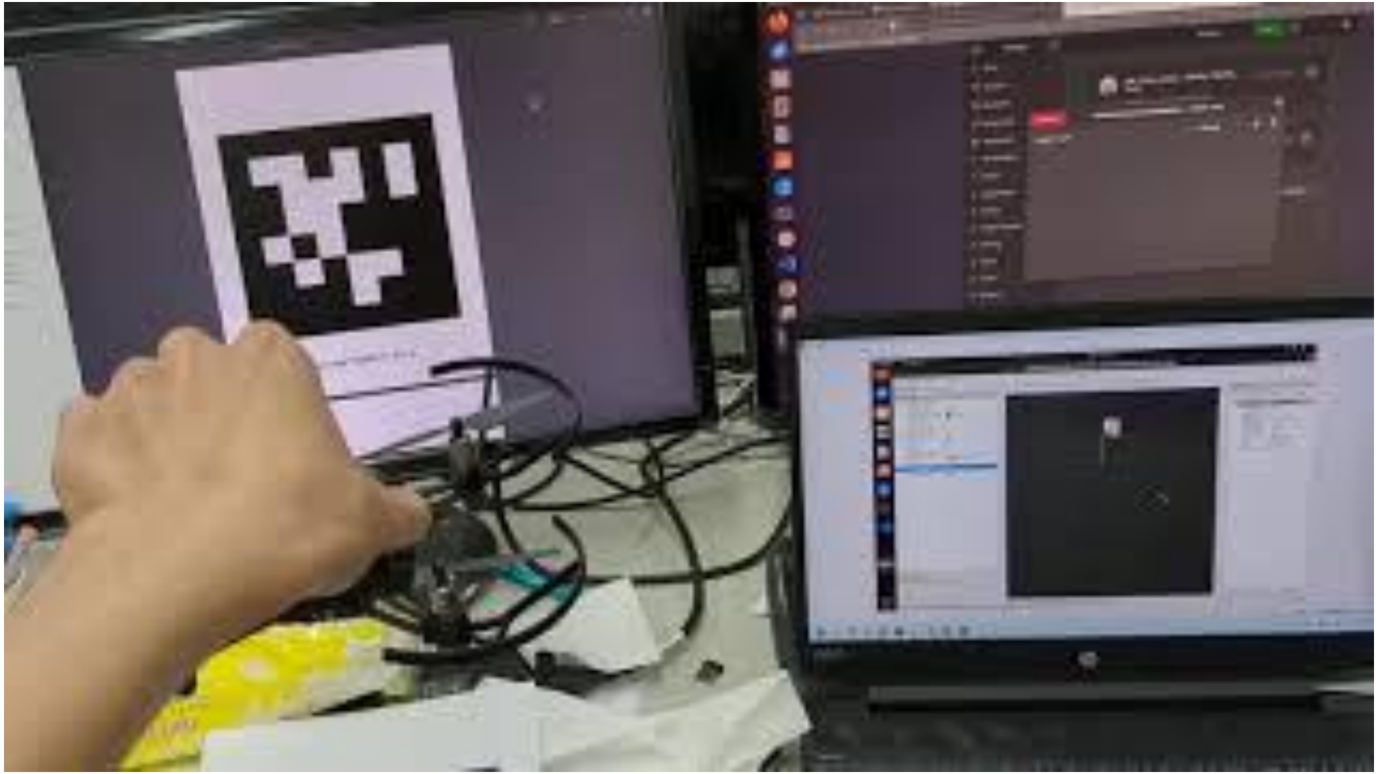
State Estimation Lab

Lab 1

Overview

- **ROS2 Introduction**
- Coordinate Systems (Extrinsic, Intrinsic Matrix)
- AprilTag, Calibration application
- Lab
  - 1.1 Get camera's intrinsic matrix and distortion coefficient.
  - 1.2 Run Tello driver to see drone streaming.
  - 1.3 Pose estimation using single AprilTag.

Today's Goal



2025 HCC State Estimation Lab1 test

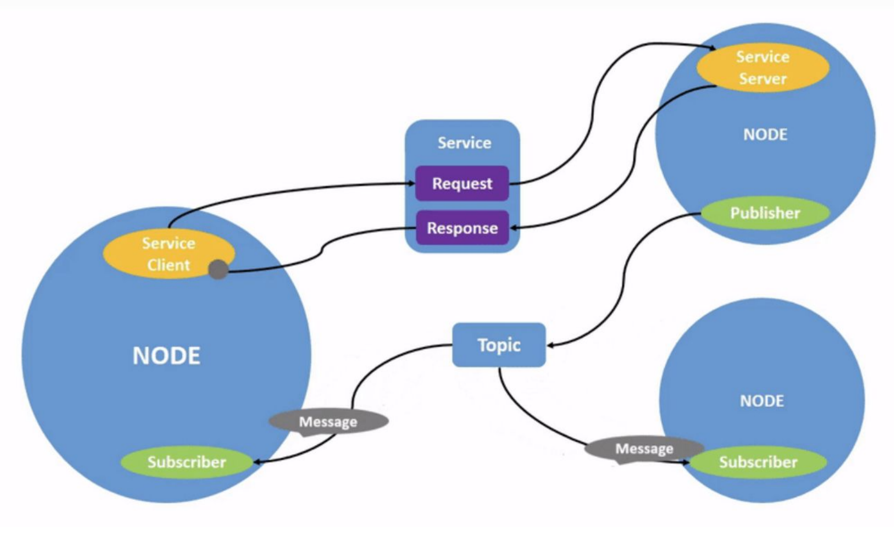
What is ROS? Why use ROS?

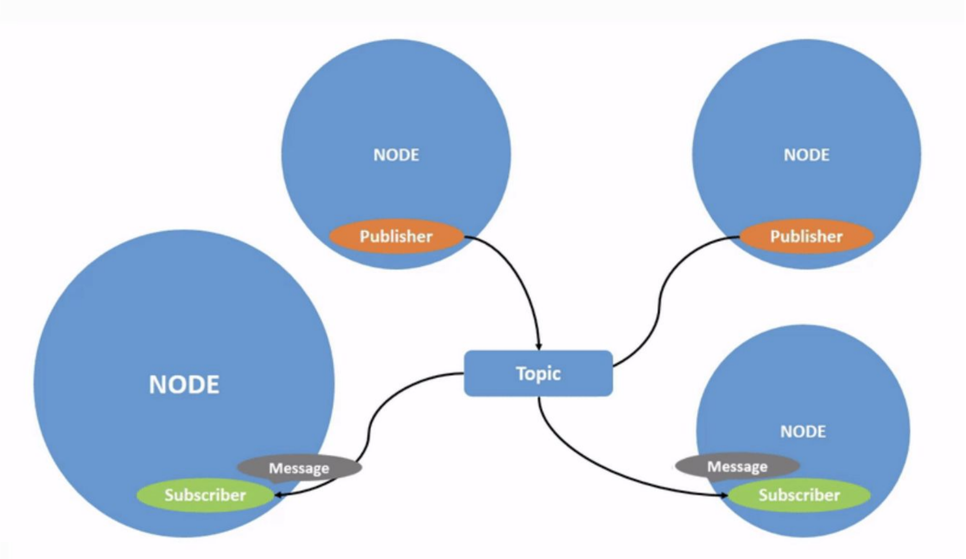
- **ROS (Robot Operating System)** is a set of tools and code that helps you build and control robots more easily.
- It lets different parts of a robot (like cameras, motors, or sensors) talk to each other.
- Each part runs separately, but ROS helps them work together like a team.
- People use ROS because it saves time, has many ready-to-use packages, and a big community for support.

Please ensure that the ROS2 installation is completed in the prelab.

ROS2 Node and Topic Overview

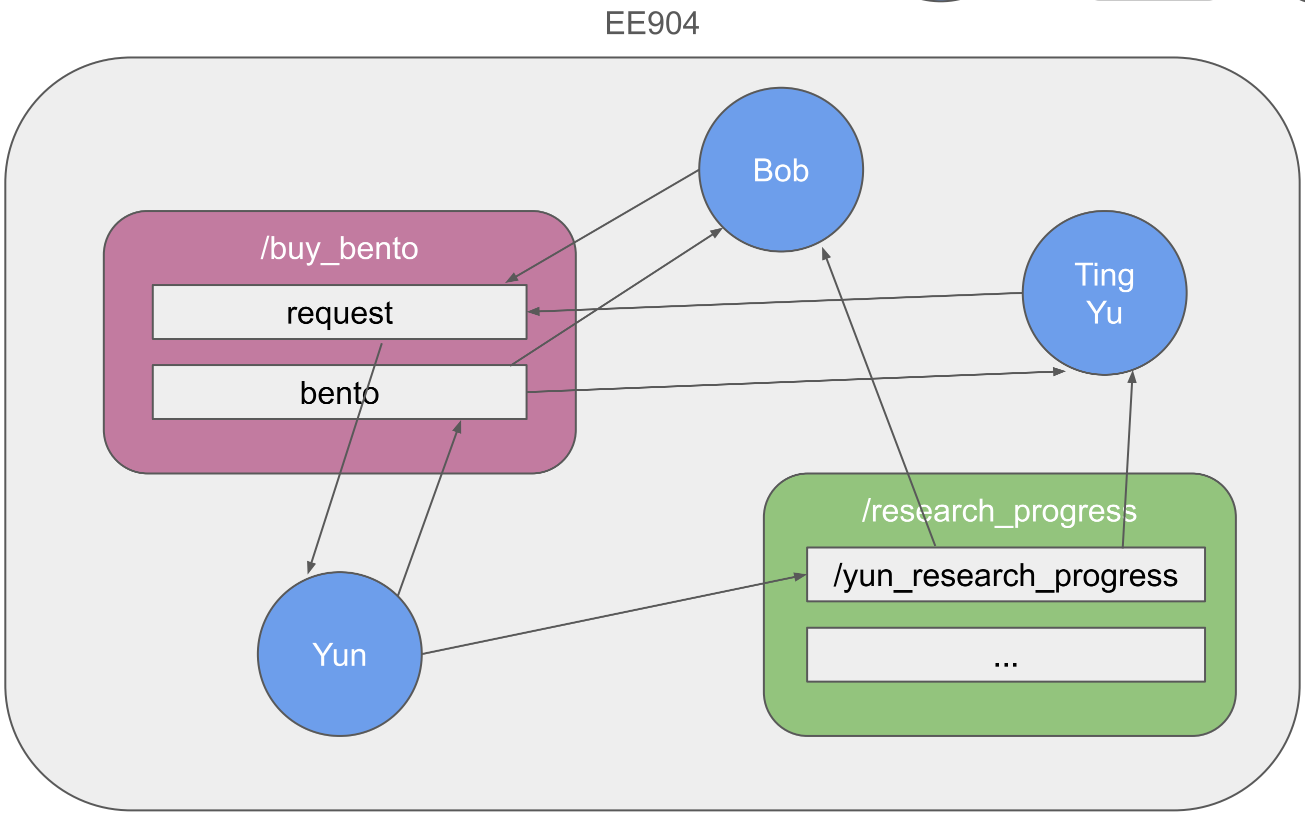
- Nodes perform specific tasks and communicate using topics, services, or actions.
- Topics and Services allow flexible data exchange between multiple publishers and subscribers.



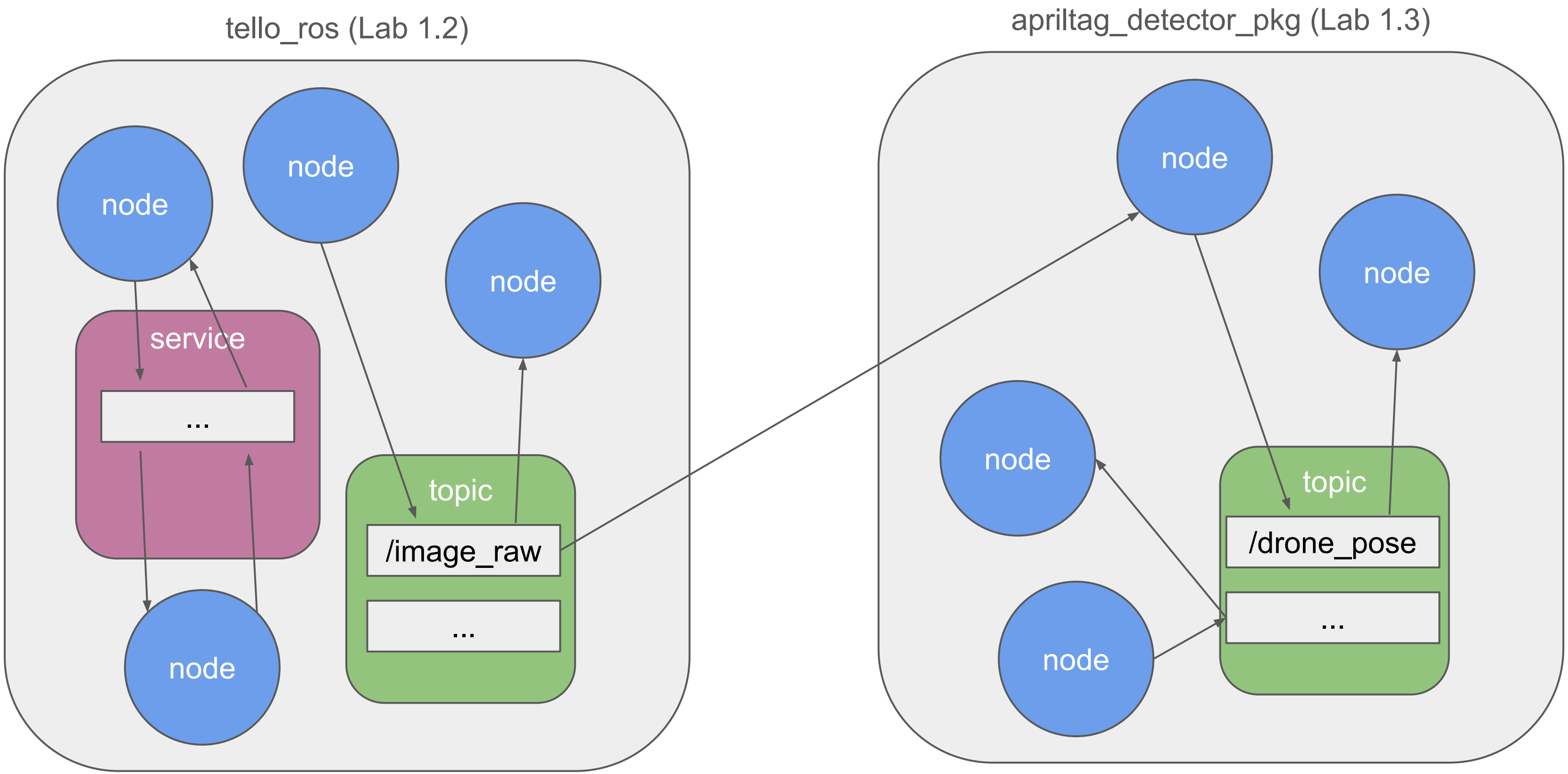


https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html
https://docs.ros.org/en/rolling/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html

What is service? What is topic?



Example: Lab 1.2 & 1.3

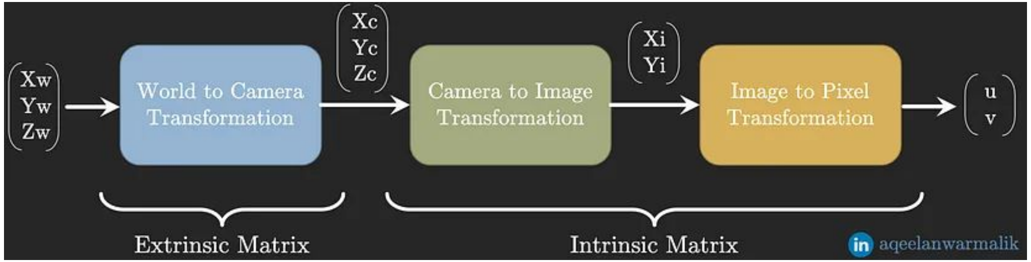


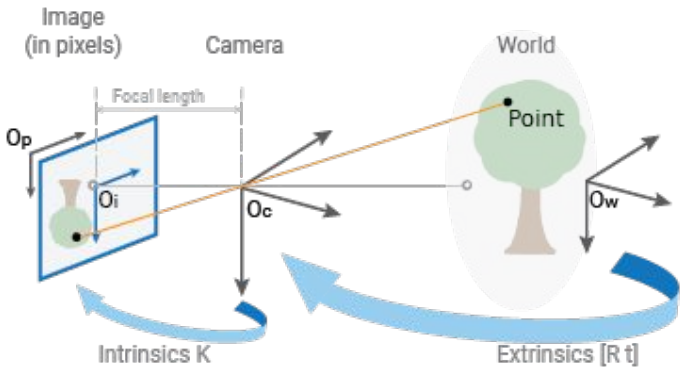
Overview

- ROS2 Introduction
- **Coordinate Systems (Extrinsic, Intrinsic Matrix)**
- AprilTag, Calibration application
- Lab
  - 1.1 Get camera’s intrinsic matrix and distortion coefficient.
  - 1.2 Run Tello driver to see drone streaming.
  - 1.3 Pose estimation using single AprilTag.

Coordinate Systems

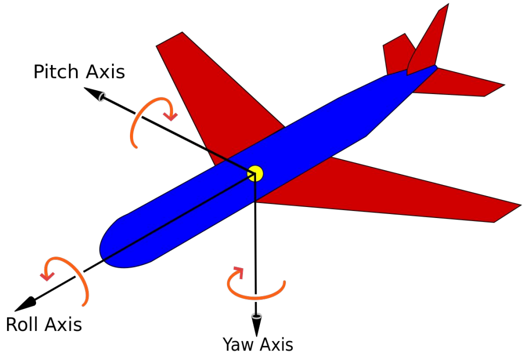
- Extrinsic: parameters of a camera depend on location and orientation (Pose)
  - World (3D) → Camera coordinate system (3D)
- Intrinsic: parameters of a camera depend on how it captures the images, such as focal length, aperture, field-of-view, resolution
  - Camera (3D) → Image (2D) → Pixel coordinate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



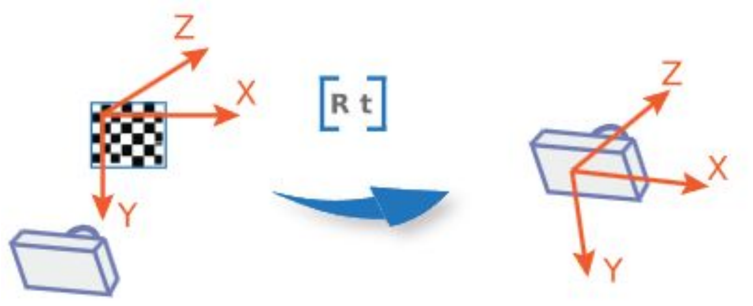


World (3D) → Pixel coordinate system (2D)

Extrinsic Matrix

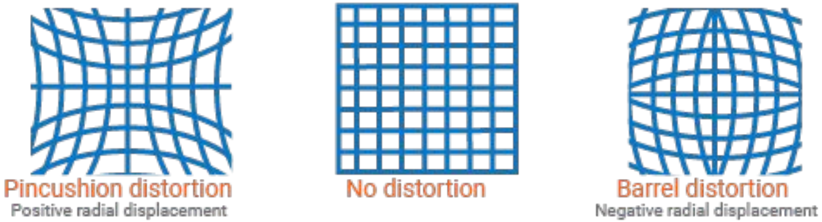


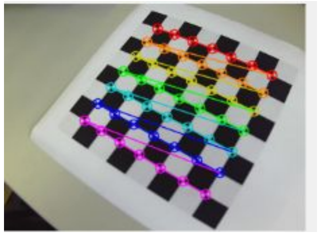
- World (3D) → Camera coordinate system (3D)
- Consist of a rotation matrix **R** and a translation matrix **t**.
- Pose estimation



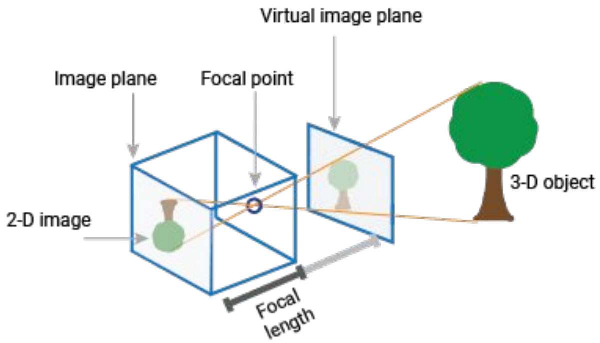
$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$

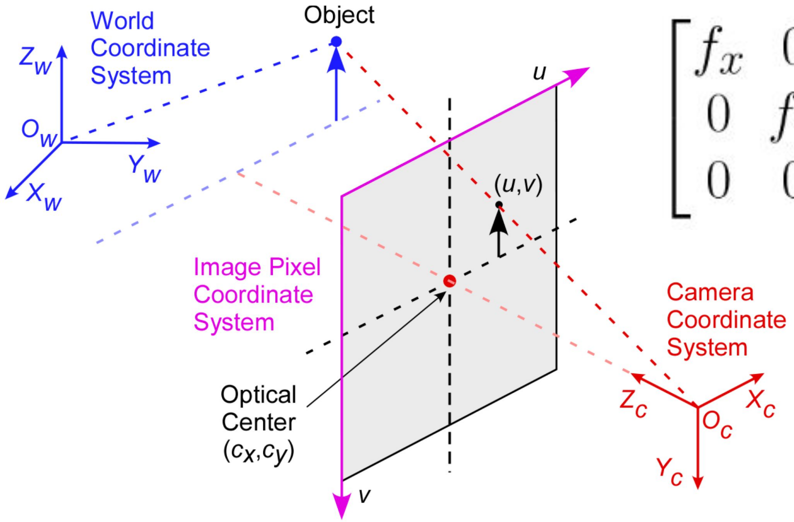
Intrinsic Matrix





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





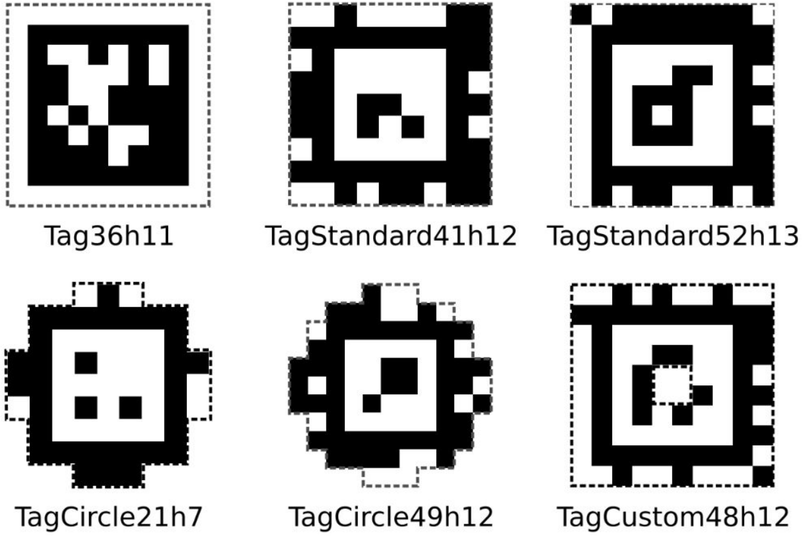
$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Overview

- ROS2 Introduction
- Coordinate Systems (Extrinsic, Intrinsic Matrix)
- **AprilTag, Calibration application**
- Lab
    - 1.1 Get camera's intrinsic matrix and distortion coefficient.
    - 1.2 Run Tello driver to see drone streaming.
    - 1.3 Pose estimation using single AprilTag.

AprilTag

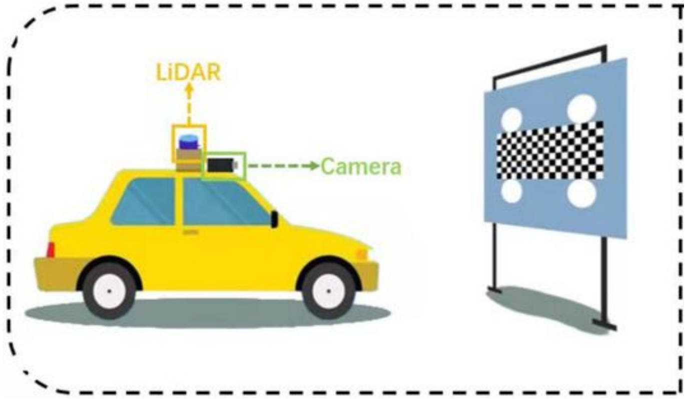
- Two-dimensional bar code, similar to QR Codes.
- Smaller data payloads, while still being detectable at longer distances.
- Can compute 3D pose (position, orientation), and tags ID relative to camera.

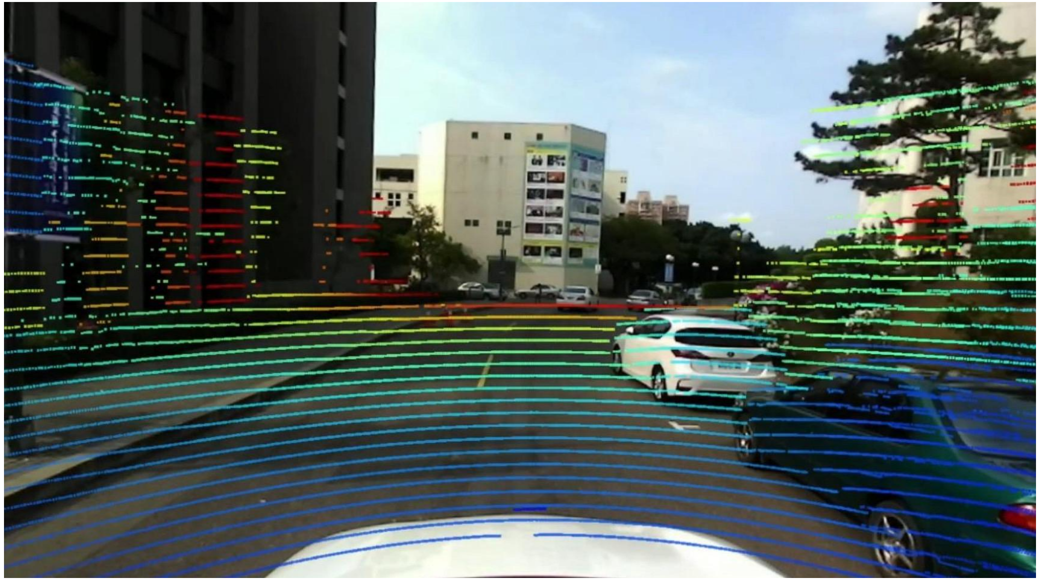


extrinsic

Calibration application

Intrinsic, Extrinsic, Temporal





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

Overview

- ROS2 Introduction
- Coordinate Systems (Extrinsic, Intrinsic Matrix)
- AprilTag, Calibration application
- **Lab**
  - 1.1 Get camera's intrinsic matrix and distortion coefficient.
  - 1.2 Run Tello driver to see drone streaming.
  - 1.3 Pose estimation using single AprilTag.

Lab

Lab

1.1 Get camera’s intrinsic matrix and distortion coefficient.

1.2 Run Tello driver to see drone streaming.

1.3 Pose estimation using single AprilTag.

Lab 1.1 - Find camera intrinsic matrix

Download “camera_calibration.py” and “take_photo_flying.py” from here:

https://drive.google.com/drive/folders/14ky1Woz_MrXBU_clXLOt1t04xkMtqGvb?usp=drive_link

Take photos with “take_photo_flying.py”, you can just hold the drone by hand, or fly the drone if you want.

Put the photos in a folder.

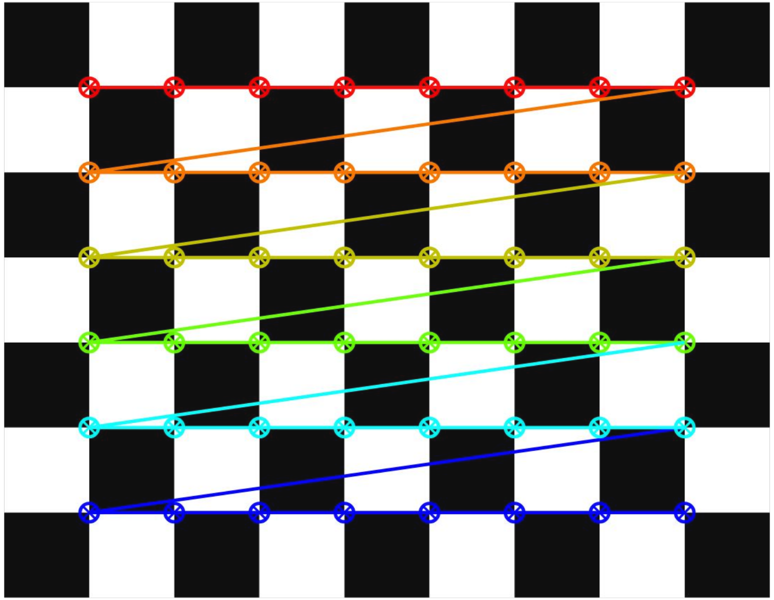
Measure the length of a square.

Fill in the blanks in “camera_calibration.py” and calculate the parameters.

Lab 1.1 - Find camera intrinsic matrix

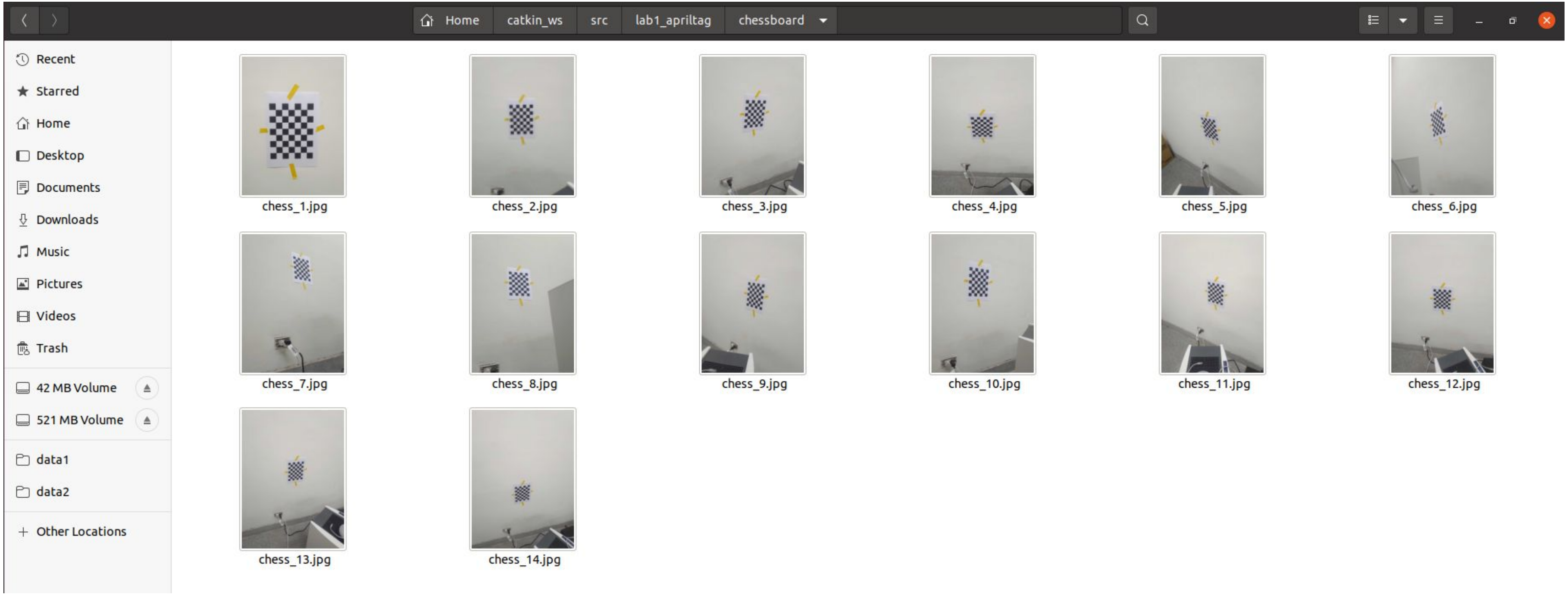
Capture multiple images of a chessboard using the camera and save them in a folder.

Measure the length of a square. (2.2 cm? you can measure by yourself)



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.

```
# 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.

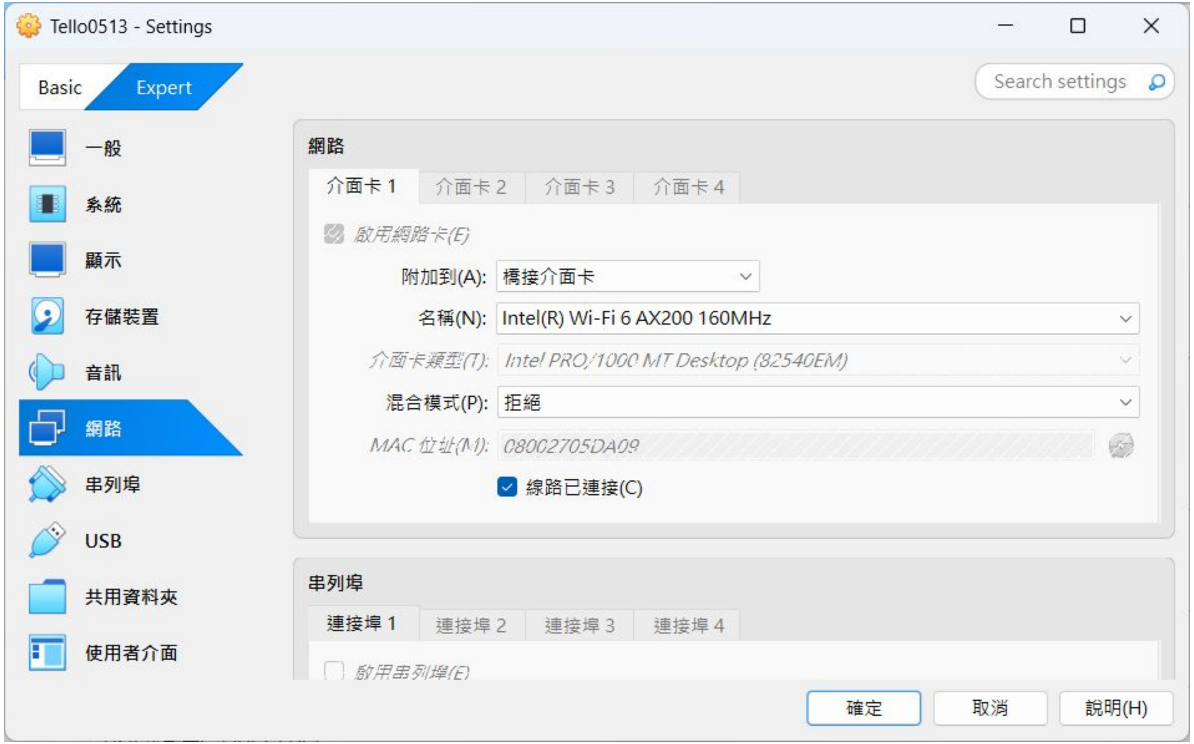
```
~/catkin_ws/src/lab1_apriltag/src ❯ python camera_calibration.py
Objective function value: 0.7488936896790611

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 - Run Tello driver to see drone streaming

Set the network configuration to “橋接介面卡” on your VirtualBox.

It you haven’t install, refer to this prelab.



Lab 1.2 - Run Tello driver to see drone streaming

Do the "Installation" section of this repository:

[clydemcqueen/tello_ros: C++ ROS2 driver for DJI Tello drones [requires Foxy and Gazebo Classic -- both EOL]]

1. Set up your Linux environment

Set up a Ubuntu 20.04 box or VM.

Also install asio:

```
sudo apt install libasio-dev
```

2. Set up your ROS environment

Install ROS2 Foxy with the `ros-foxy-desktop` option.

If you install binaries, be sure to also install the development tools and ROS tools from the source installation instructions.

Install these additional packages:

```
sudo apt install ros-foxy-cv-bridge ros-foxy-camera-calibration-parsers
```

3. Install `tello_ros`

Download, compile and install `tello_ros`:

```
mkdir -p ~/tello_ros_ws/src
cd ~/tello_ros_ws/src
git clone https://github.com/clydemcqueen/tello_ros.git
git clone https://github.com/ptrmu/ros2_shared.git
cd ..
source /opt/ros/foxy/setup.bash
# If you didn't intall Gazebo, skip tello_gazebo while building:
colcon build --event-handlers console_direct+ --packages-skip tello_gazebo
```

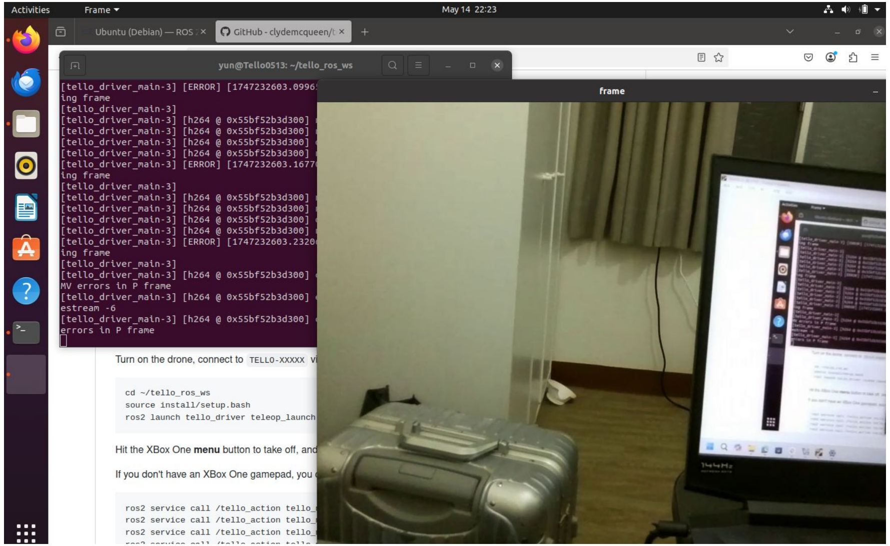
Lab 1.2 - Run Tello driver to see drone streaming

Turn on the drone, connect to its wifi, do the following commands:

cd ~/tello_ros_ws

source install/setup.bash

ros2 launch tello_driver teleop_launch.py



Lab 1.3 - Pose estimation using single AprilTag

Do the first step of 1.3 from this guide:

https://hackmd.io/@L-In544OTxi8tkHC4JCwsg/SJ9h3MM-gg

Fill in the blanks in code.

```
# Add publisher
self.pose_pub = self.create_publisher(PoseStamped, '/drone_pose', 10)

# === Camera intrinsics === (You must adjust based on your Tello's camera)
self.fx = ???  # Focal length x
self.fy = ???  # Focal length y
self.cx = ???  # Principal point x (image center)
self.cy = ???  # Principal point y

# Tag size in meters
self.tag_size = ???

self.get_logger().info("Pose publisher Initialized.")
```

Then continue with the guide.