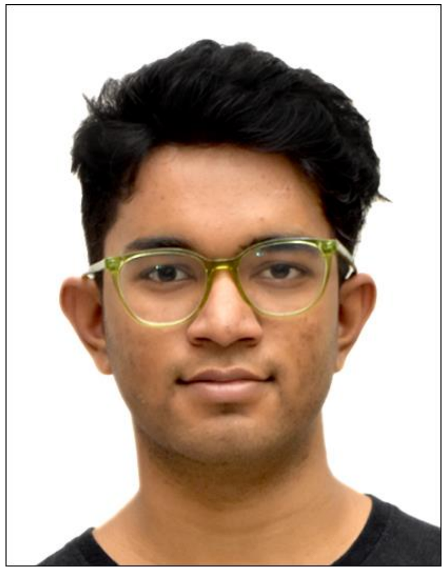
Development of Drone Systems for Indoor Navigation, Object Tracking, and ArUco-based Inventory Management with ROS2

Under Guidance of

Dr Abderrahim Benslimane (Professor, Computer Science Department, University of Avignon, France)

Prof. Pallavi Dongare (Assistant Professor, Civil Engineering Department, Vishwaniketan’s iMEET, India)

Our Team



Adyansh Gupta

Tejal Ubale

Anushka Kadam

Introduction

Objective: Develop and implement autonomous navigation and object detection capabilities for Crazyflie and Tello drones.

Technologies Used: ROS2, SLAM Toolbox, Navigation Stack, Computer Vision (OpenCV), and Machine Learning.

Applications: Enhancing drone autonomy for tasks like obstacle avoidance, object tracking, and efficient warehouse management.

Goals:

- Enable autonomous navigation for Crazyflie in dynamic environments.
- Implement robust object detection and tracking for Tello.
- Utilize ArUco markers for inventory management in warehouse settings.

Impact: Demonstrating the potential of drones in autonomous operations and smart logistics solutions.

Crazyflie 2.1 Drone

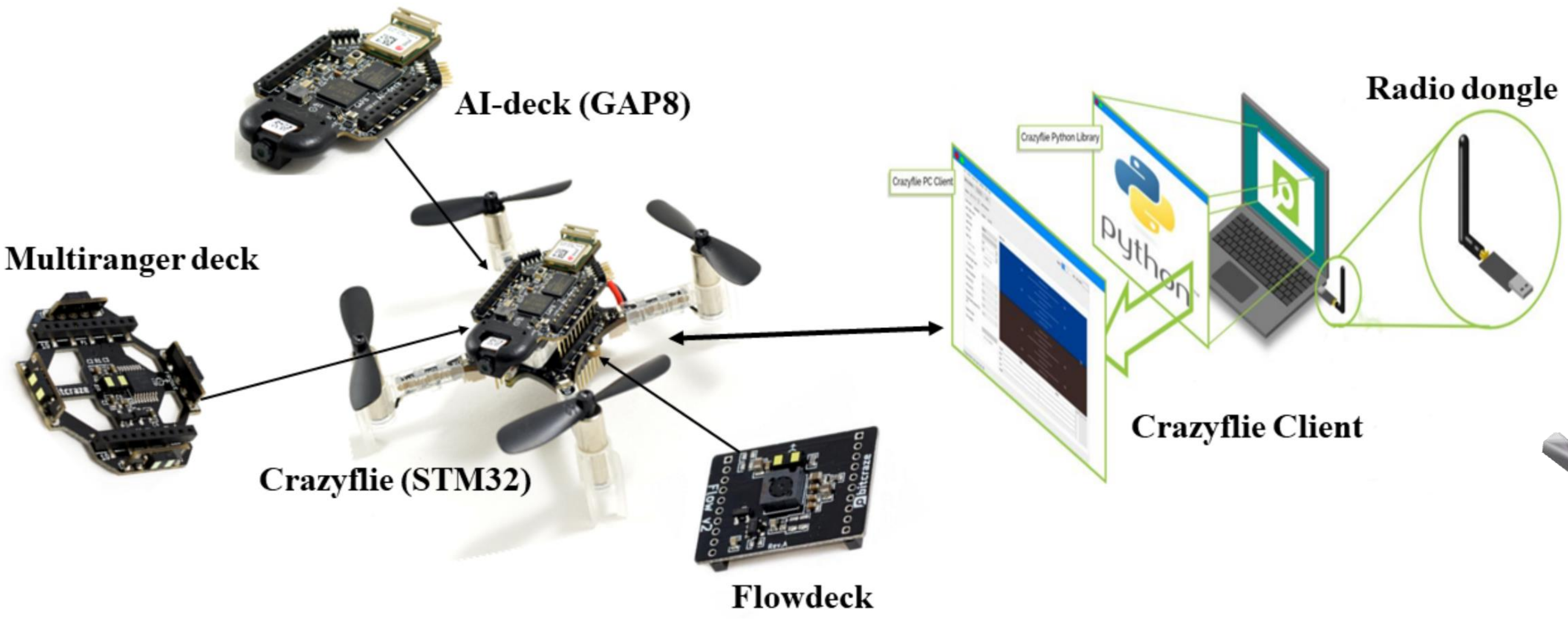
The Crazyflie 2.1 is a small, open-source, and customizable drone platform ideal for research and development in areas such as autonomous navigation, control systems, and sensor integration.

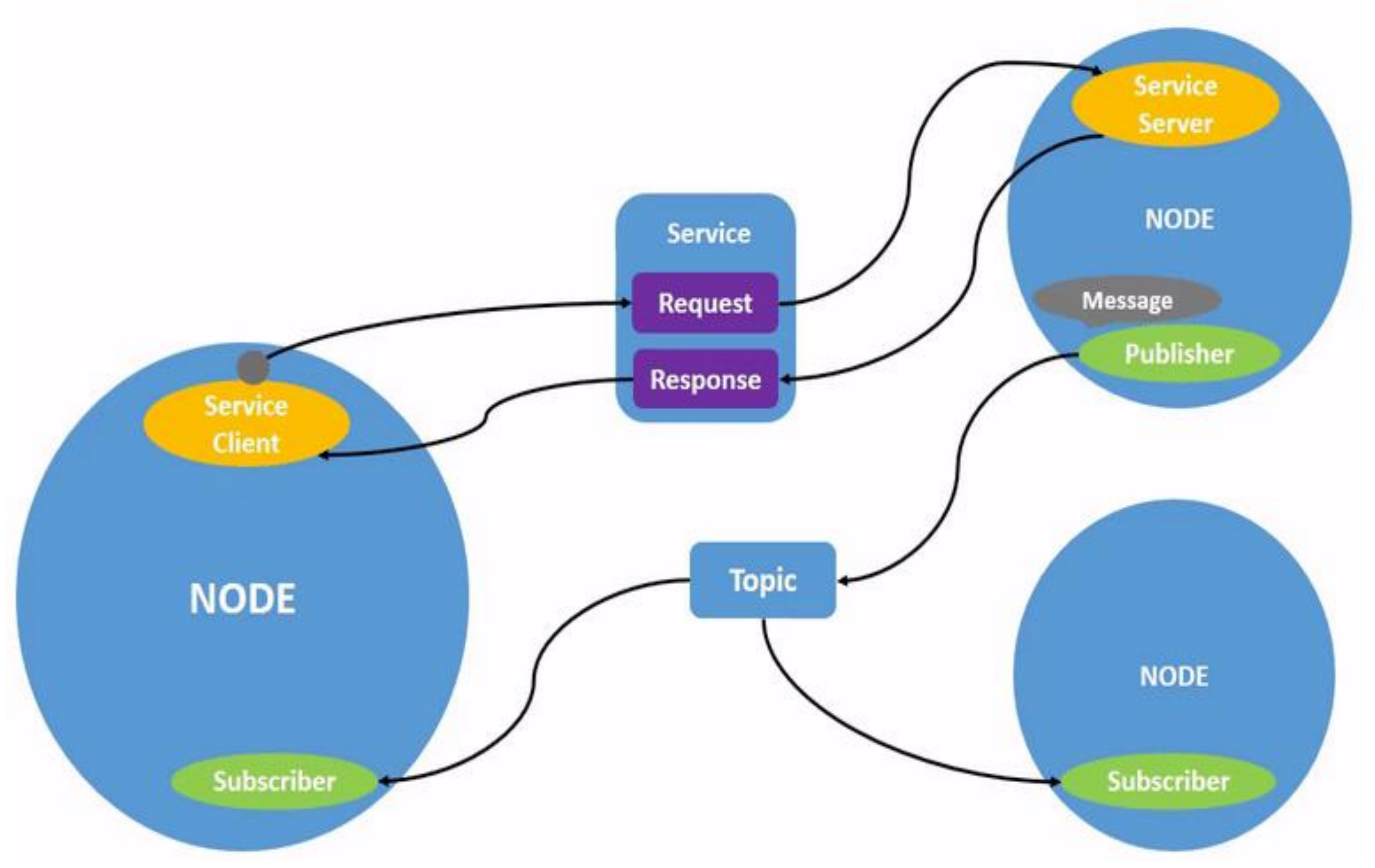
This versatile drone features a compact design and lightweight construction, making it an excellent choice for indoor applications.

Crazyflie 2.1

Tasks and Objectives:

- ✓ Autonomous Navigation
- ✓ Obstacle Avoidance
- ✓ Mapping and Exploration





ROS2: Simplifying Multi-Drone Control and Programming

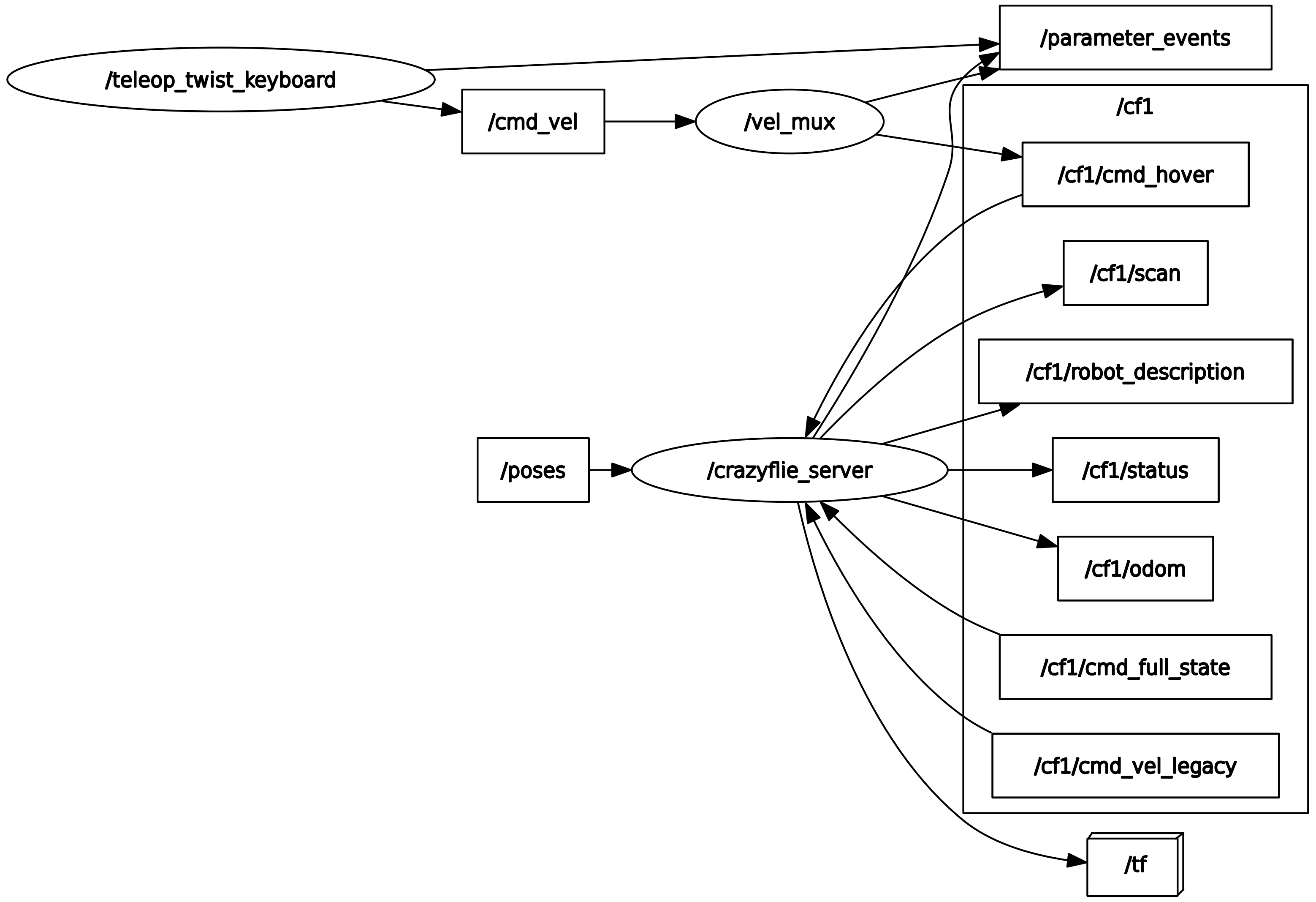
ROS2 (Robot Operating System 2): An open-source framework for robot software development.

Bridge for Drone Control: Provides a unified communication platform for controlling and programming both Crazyflie and Tello drones.

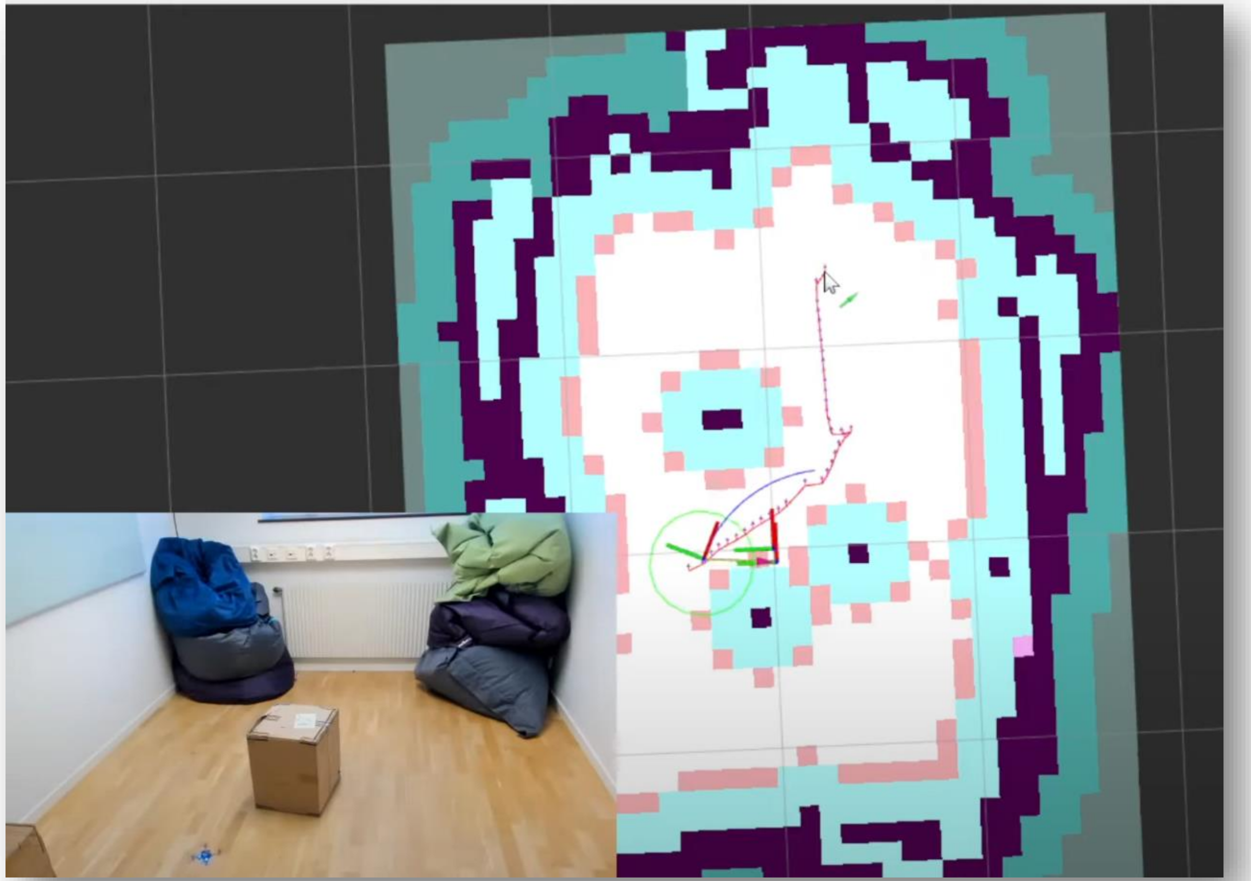
Advantages of ROS2:

- Modular Design
- Scalability
- Rich Ecosystem
- Multi-Language Support

Crazyflie 2.1 - Architecture



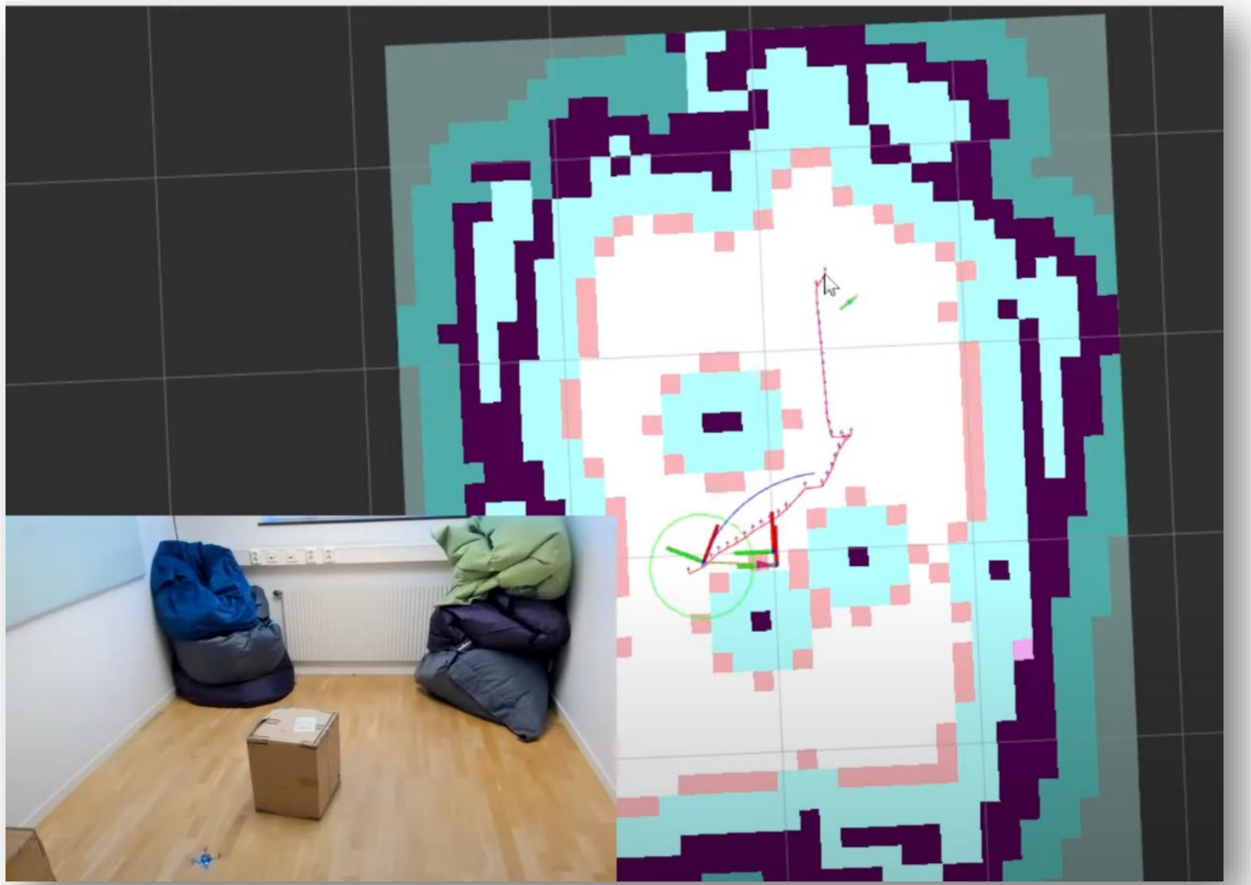
ROS2 SLAM Toolbox Integration for Map Building



- ✓ Package for building maps and enabling robot navigation.
- ✓ Muliranger and Flow Deck -> /scan topic
- ✓ IMU -> /odom topic
- ✓ Uses Transforms to represent relationship between coordinate frames of sensors and robot

ROS2 Navigation Stack for Autonomous Navigation

- ✓ A collection of ROS2 nodes for planning and executing robot motion.
- ✓ The map represents obstacles, walls, and free space the robot can navigate through.
- ✓ Navigation algorithms for collision free paths.
- ✓ Receives the planned path from the navigation stack.
- ✓ Breaks down the path into smaller achievable waypoints.



Tello EDU Drone

Tello EDU is an impressive and programmable drone perfect for education. With an upgraded SDK 2.0, Tello EDU comes with more advanced commands and increased data interfaces.

Tello EDU

Camera-equipped Drone: Designed for object and face detection, tracking, and warehouse management applications.

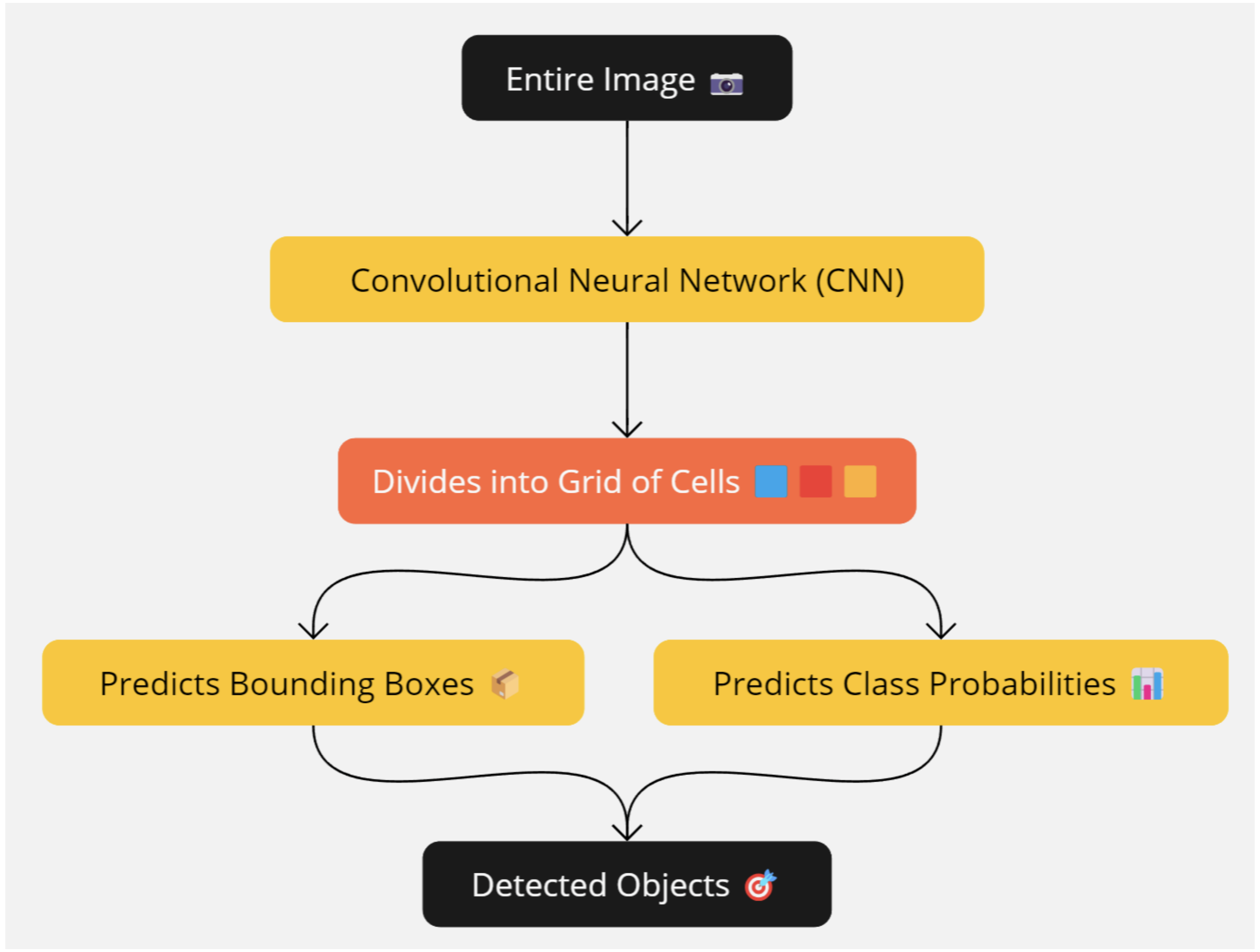
Advanced Features:

- Object Detection
- Face Detection
- ArUco Marker Detection

Benefits:

- Improves warehouse efficiency and accuracy in inventory management.
- Automates tedious manual tasks for streamlined operations.
- Provides real-time data for better inventory control and decision-making.

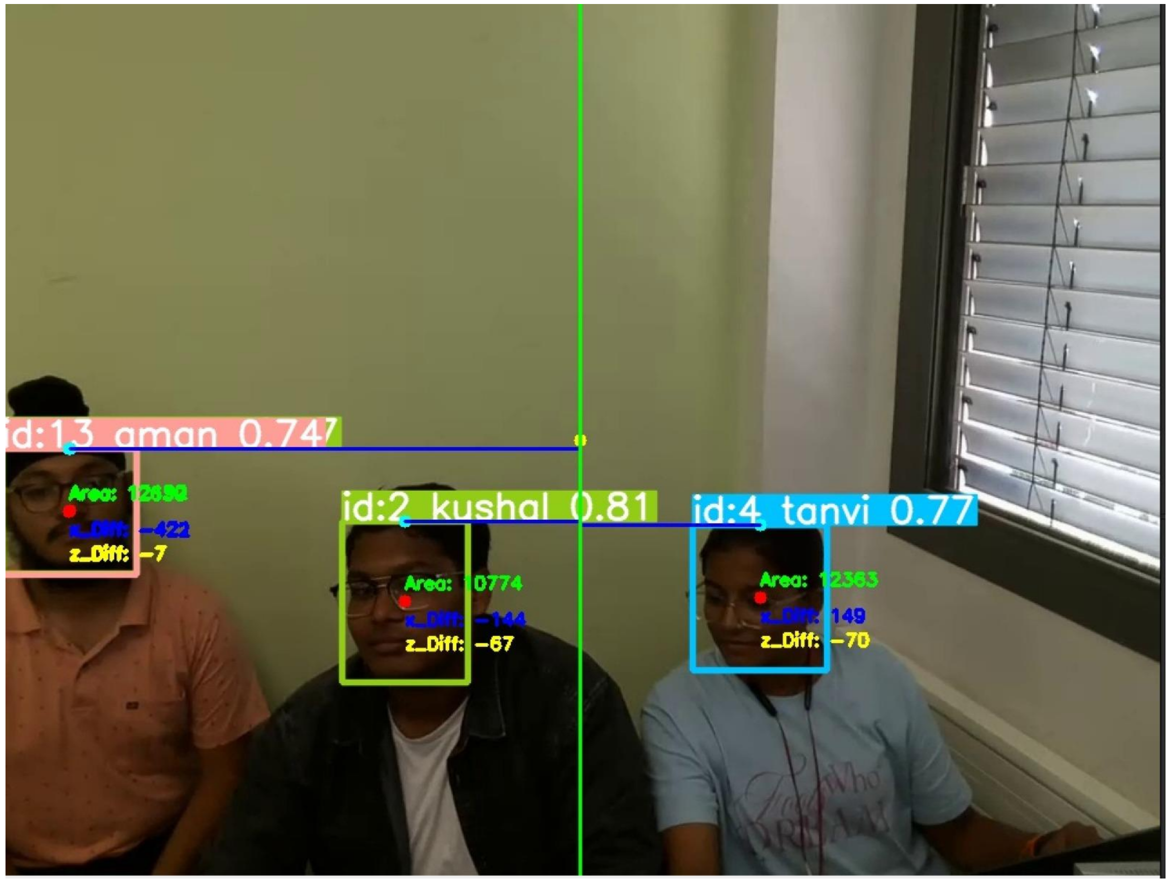
YOLO Object Detection for Tello EDU Drone

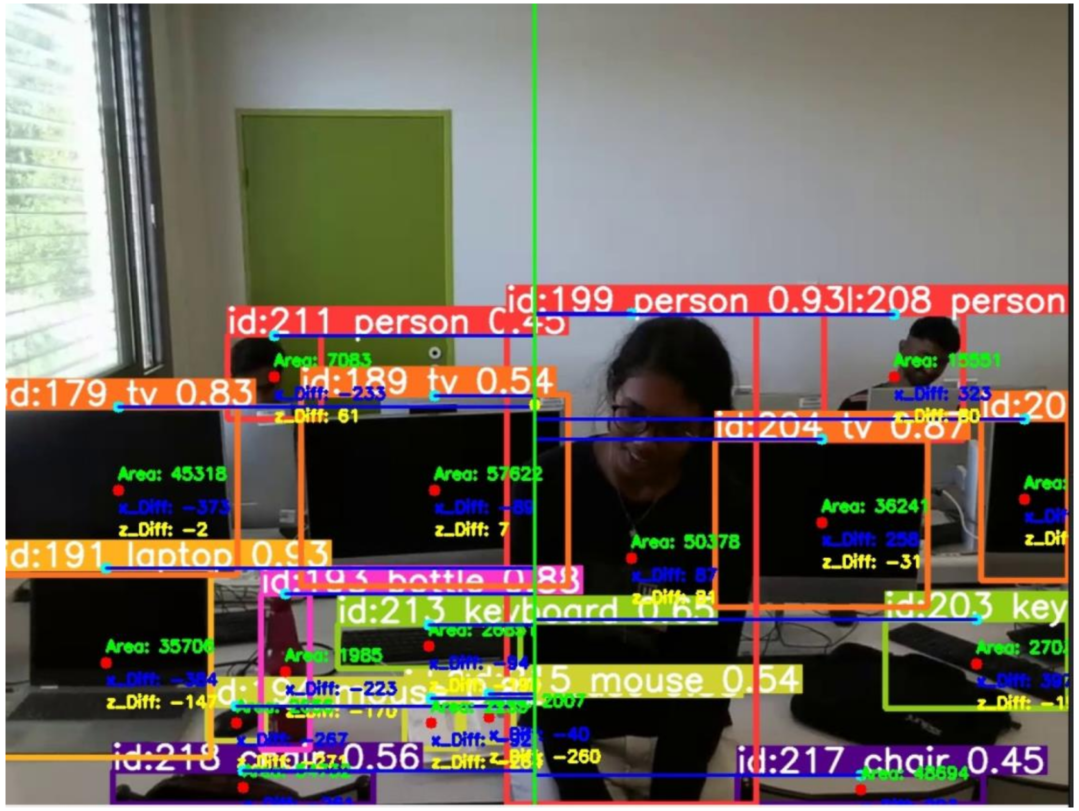

Entire Image
Convolutional Neural Network (CNN)
Divides into Grid of Cells
Predicts Bounding Boxes
Predicts Class Probabilities
Detected Objects

Key Features:

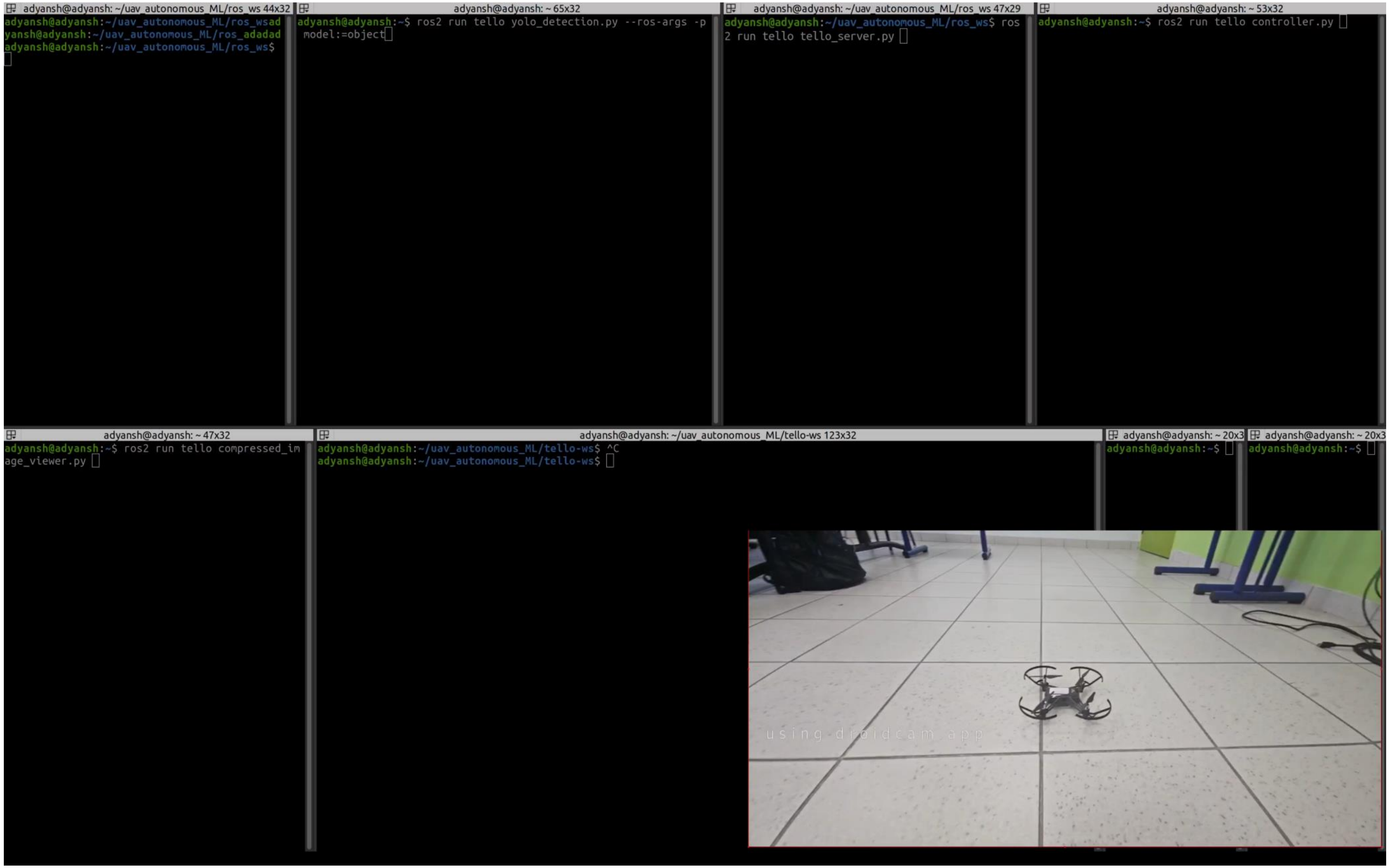
- **Real-time Processing:** Enables fast object detection on the Tello drone, crucial for applications requiring immediate responses.
- **Single-Stage Detection:** Processes the image once for both object localization (bounding boxes) and classification (class probabilities). Makes it faster compared to multi-stage detectors.
- **High Accuracy:** Achieves good accuracy in object detection despite its speed, making it suitable for various real-world scenarios.

YOLO Object Detection for Tello EDU Drone





Tello – Object Detection Demo



```
adyansh@adyansh: ~/uav_autonomous_ML/ros_ws 44x32
adyansh@adyansh:~/uav_autonomous_ML/ros_wsad
yansh@adyansh:~/uav_autonomous_ML/ros_adadad
adyansh@adyansh:~/uav_autonomous_ML/ros_ws$
```

```
adyansh@adyansh: ~ 65x32
adyansh@adyansh:~$ ros2 run tello yolo_detection.py --ros-args -p
 model:=object
```

```
adyansh@adyansh: ~/uav_autonomous_ML/ros_ws 47x29
adyansh@adyansh:~/uav_autonomous_ML/ros_ws$ ros
2 run tello tello_server.py
```

```
adyansh@adyansh: ~ 53x32
adyansh@adyansh:~$ ros2 run tello controller.py
```

```
adyansh@adyansh: ~ 47x32
adyansh@adyansh:~$ ros2 run tello compressed_im
age_viewer.py
```

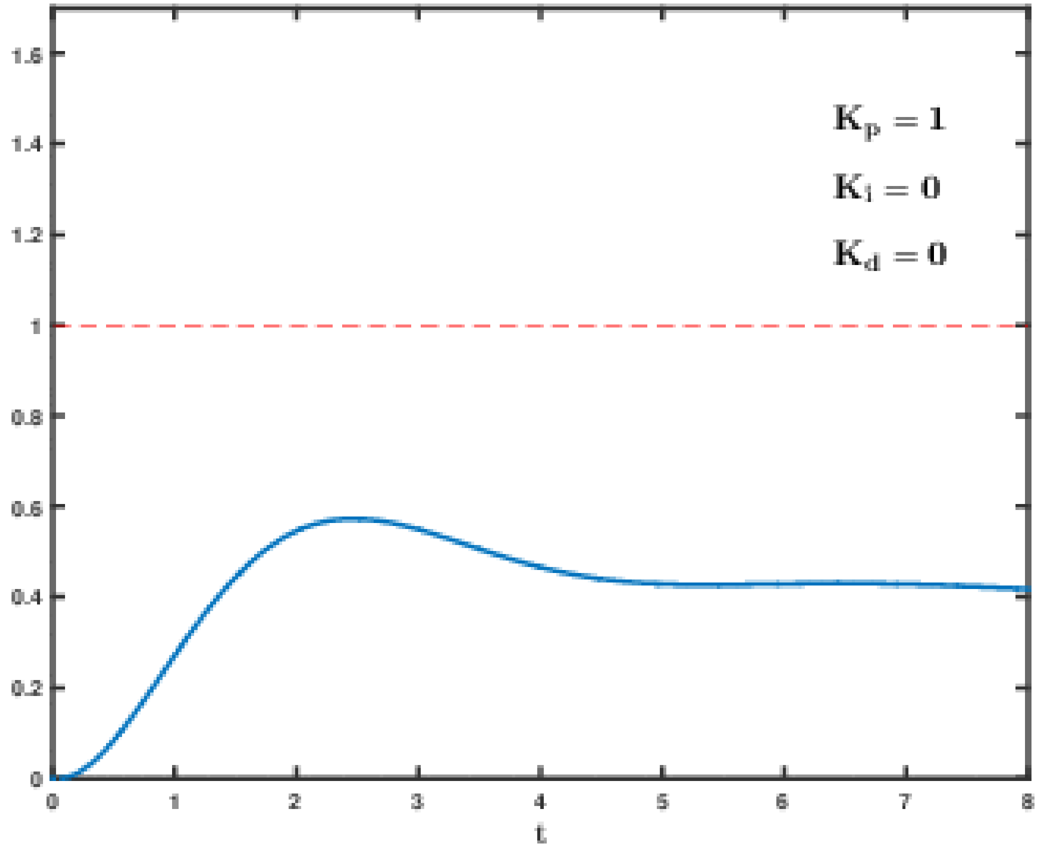
```
adyansh@adyansh: ~/uav_autonomous_ML/tello-ws 123x32
adyansh@adyansh:~/uav_autonomous_ML/tello-ws$ ^C
adyansh@adyansh:~/uav_autonomous_ML/tello-ws$
```

```
adyansh@adyansh: ~ 20x3
adyansh@adyansh:~$
```

```
adyansh@adyansh: ~ 20x3
adyansh@adyansh:~$
```

PID Controller

- **Proportional (P) Control:** difference between desired and actual value
- **Integral (I) Control:** object remaining off-center even after initial correction
- **Derivative (D) Control:** how quickly the system reaches the desired state



Pseudocode for Basic PID Control:

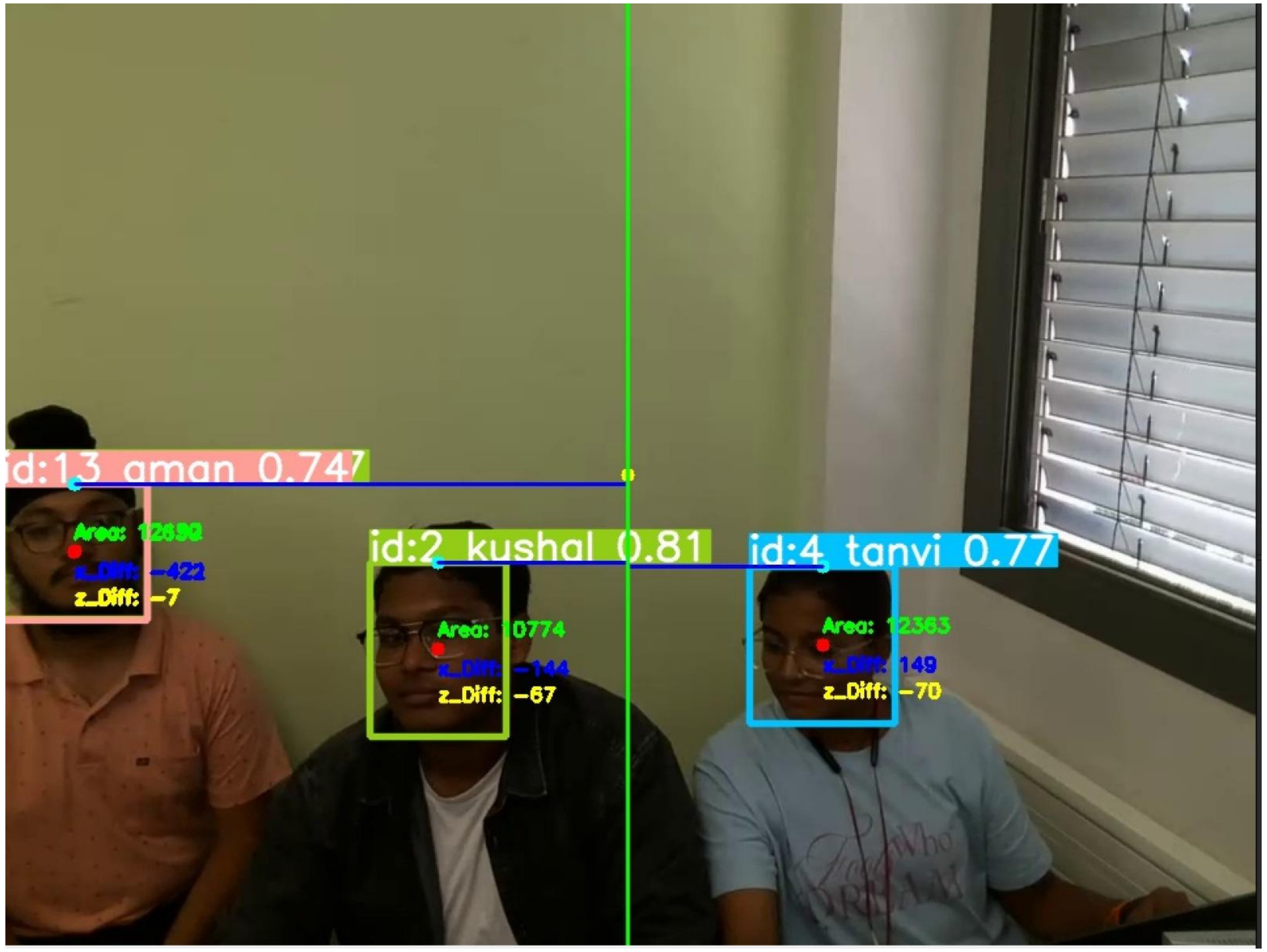
```
function PID_control(desired_value, current_value):
error = desired_value - current_value
P_term = error * Kp    # Kp is the proportional gain
I_term += error * Ki * dt    # Ki is the integral gain, dt is the time step
D_term = (error - previous_error) / dt * Kd
# Kd is the derivative gain, previous_error is stored from the last iteration
output = P_term + I_term + D_term
previous_error = error    # Update previous error for next iteration
return output
```

Object Following with PID Control

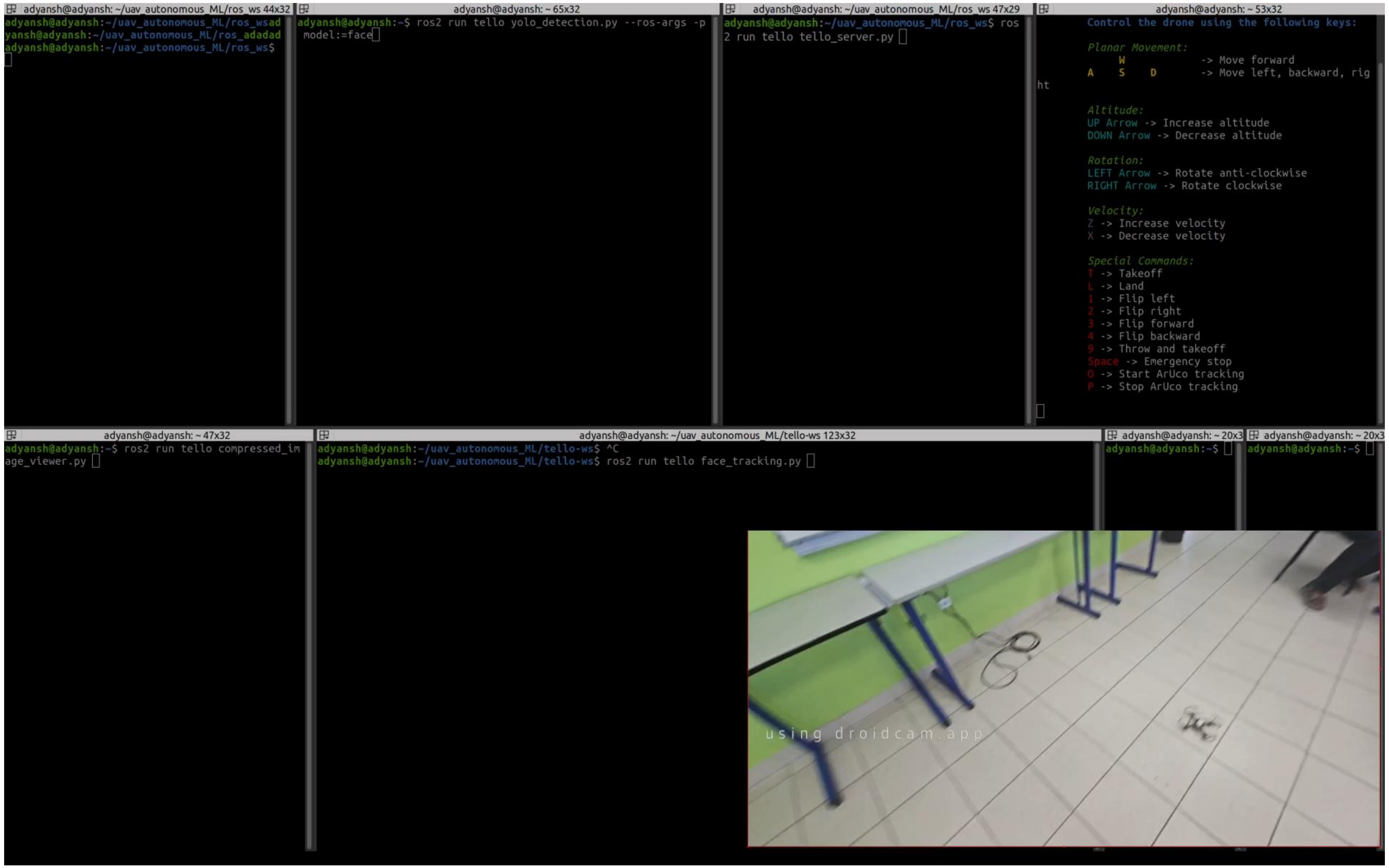
- **Linear X Velocity:** difference between the desired and actual contour area of the object's bounding box.

- **Angular Z Velocity (Yaw):** horizontal pixel difference between the bounding box center and the camera center.

- **Linear Z Velocity (Up/Down):** vertical pixel difference between the bounding box center and the camera center.



Tello – Face Tracking Demo



```
adyansh@adyansh: ~/uav_autonomous_ML/ros_ws 44x32
adyansh@adyansh:~/uav_autonomous_ML/ros_wsad
yansh@adyansh:~/uav_autonomous_ML/ros_adadad
adyansh@adyansh:~/uav_autonomous_ML/ros_ws$
```

```
adyansh@adyansh: ~ 65x32
adyansh@adyansh:~$ ros2 run tello yolo_detection.py --ros-args -p
 model:=face
```

```
adyansh@adyansh: ~/uav_autonomous_ML/ros_ws 47x29
adyansh@adyansh:~/uav_autonomous_ML/ros_ws$ ros
2 run tello tello_server.py
```

```
adyansh@adyansh: ~ 53x32
        Control the drone using the following keys:

        Planar Movement:
             W              -> Move forward
        A    S    D         -> Move left, backward, rig
ht

        Altitude:
        UP Arrow -> Increase altitude
        DOWN Arrow -> Decrease altitude

        Rotation:
        LEFT Arrow -> Rotate anti-clockwise
        RIGHT Arrow -> Rotate clockwise

        Velocity:
        Z -> Increase velocity
        X -> Decrease velocity

        Special Commands:
        T -> Takeoff
        L -> Land
        1 -> Flip left
        2 -> Flip right
        3 -> Flip forward
        4 -> Flip backward
        9 -> Throw and takeoff
        Space -> Emergency stop
        O -> Start ArUco tracking
        P -> Stop ArUco tracking
```

```
adyansh@adyansh: ~ 47x32
adyansh@adyansh:~$ ros2 run tello compressed_im
age_viewer.py
```

```
adyansh@adyansh: ~/uav_autonomous_ML/tello-ws 123x32
adyansh@adyansh:~/uav_autonomous_ML/tello-ws$ ^C
adyansh@adyansh:~/uav_autonomous_ML/tello-ws$ ros2 run tello face_tracking.py
```

```
adyansh@adyansh: ~ 20x3
adyansh@adyansh:~$
```

```
adyansh@adyansh: ~ 20x3
adyansh@adyansh:~$
```

Warehouse Management

This system utilizes ArUco markers for efficient inventory management in your Tello EDU drone.

Workflow:

1. Marker Detection:
2. Image Processing
3. Marker ID Extraction
4. Database Lookup
5. Data Retrieval
6. Database Update

Warehouse Management

Applications in Warehouse Management:

✓Streamlined Inventory Tracking

✓Real-time Data Acquisition

✓Improved Accuracy

✓Enhanced Efficiency

✓Scalability

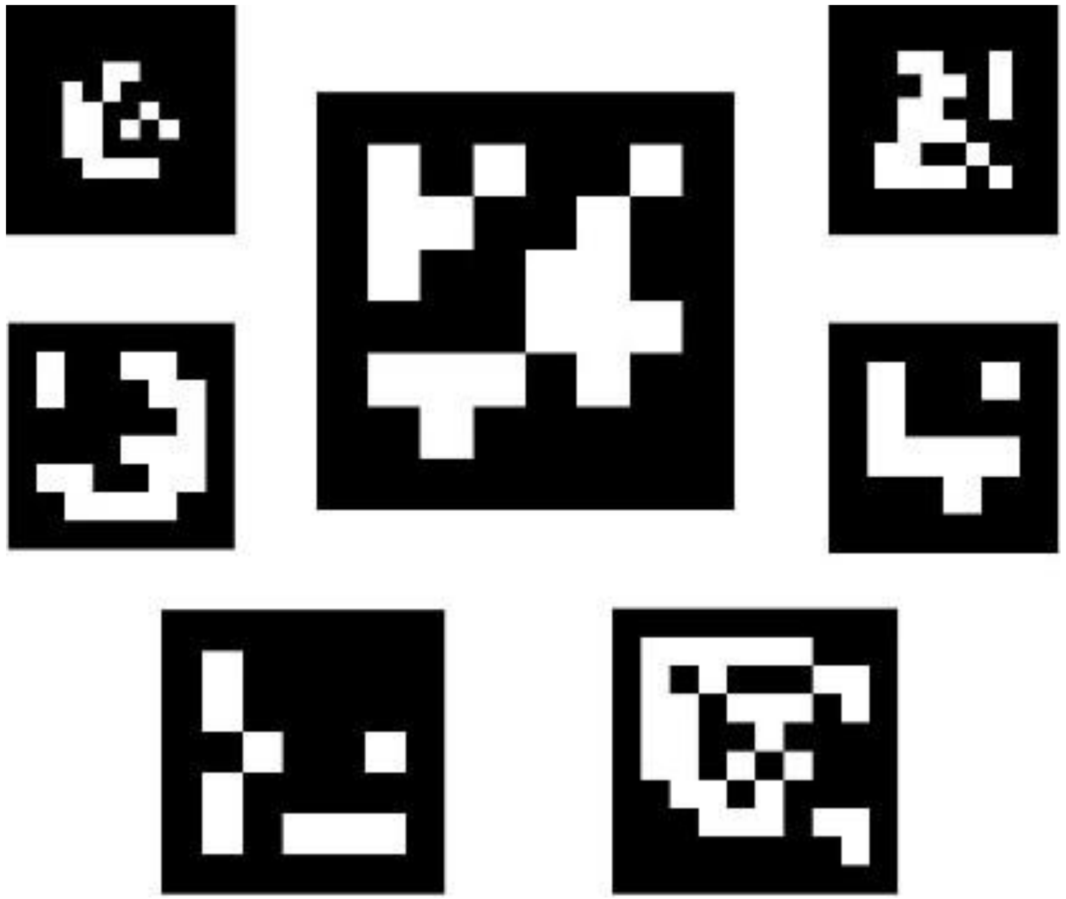
Aruco Marker Detection Algorithm

Markers and Dictionaries

- ✓ A **dictionary of markers** is the set of markers that are considered in a specific application. It is simply the list of binary codifications of each of its markers.

- ✓ The **dictionary size** is the number of markers that compose the dictionary.

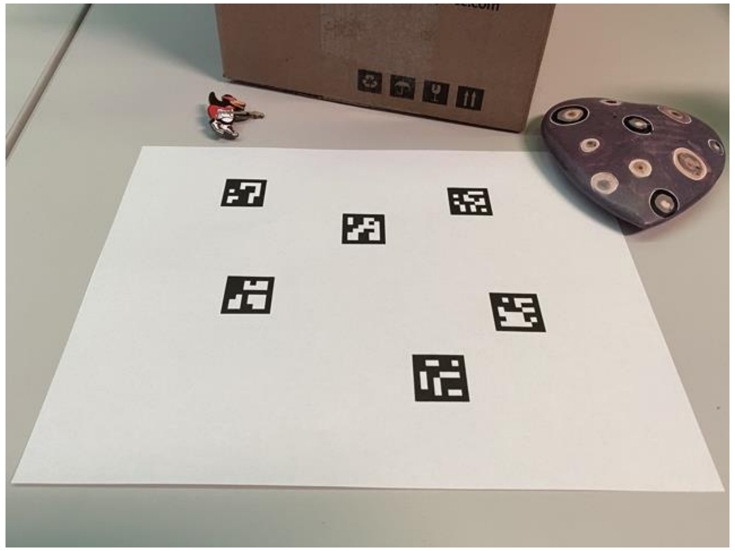
- ✓ The **marker size** is the size of those markers (the number of bits/modules).



Aruco Marker Detection Algorithm

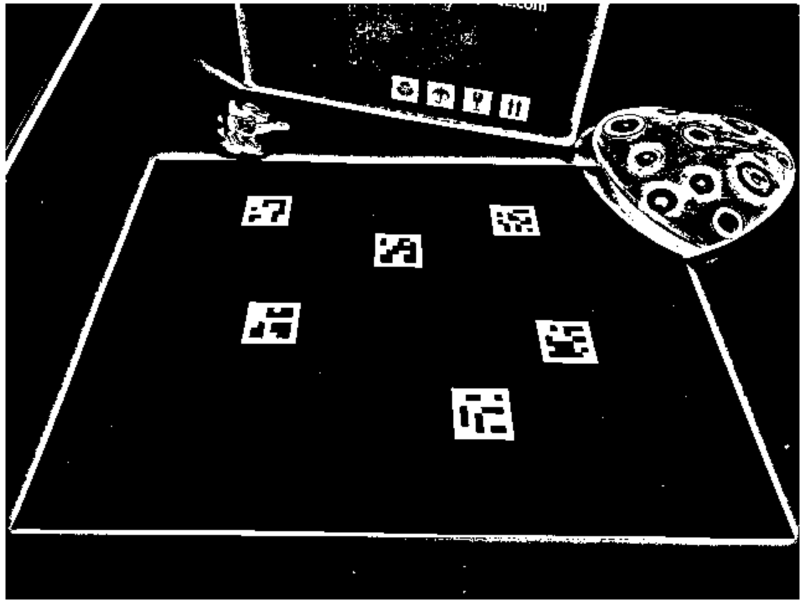
Thresholding

- ✓ One of the first steps in the marker detection process is adaptive thresholding of the input image.
- ✓ adaptiveThreshWinSizeMin, adaptiveThreshWinSizeMax, and adaptiveThreshWinSizeStep



Contour filtering

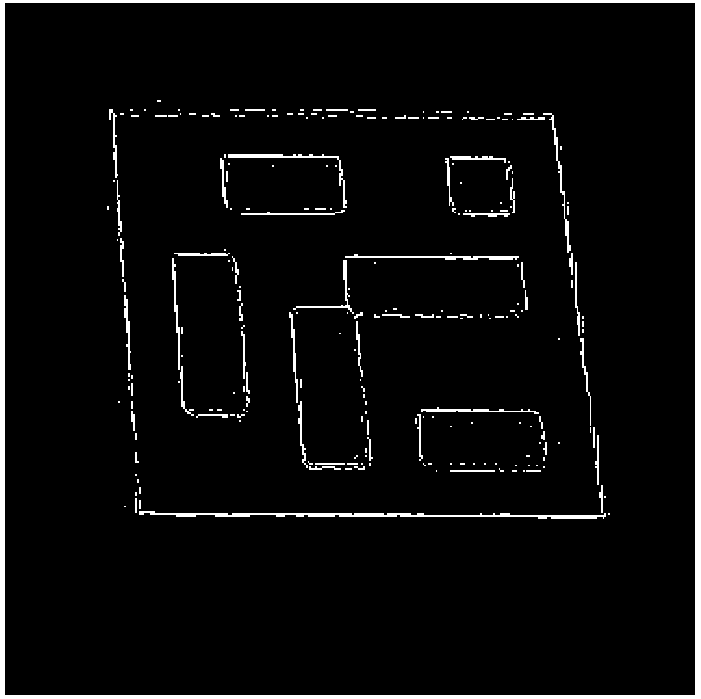
- ✓ After thresholding, contours are detected.
- ✓ However, not all contours are considered as marker candidates.
- ✓ They are filtered out in different steps so that contours that are very unlikely to be markers are discarded.



Aruco Marker Detection Algorithm

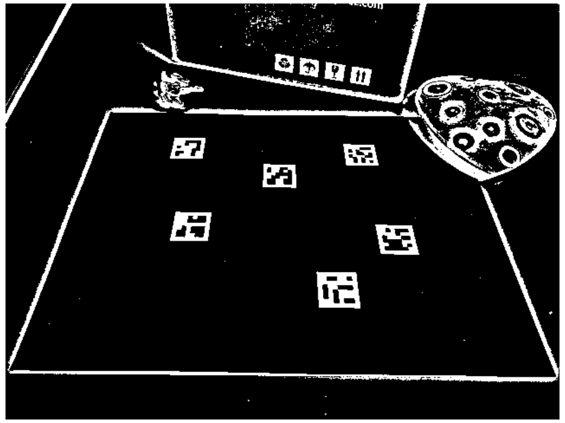
Thresholding

- ✓ One of the first steps in the marker detection process is adaptive thresholding of the input image.
- ✓ adaptiveThreshWinSizeMin, adaptiveThreshWinSizeMax, and adaptiveThreshWinSizeStep


Broken Marker Image

Contour filtering

- ✓ After thresholding, contours are detected.
- ✓ However, not all contours are considered as marker candidates.
- ✓ They are filtered out in different steps so that contours that are very unlikely to be markers are discarded.

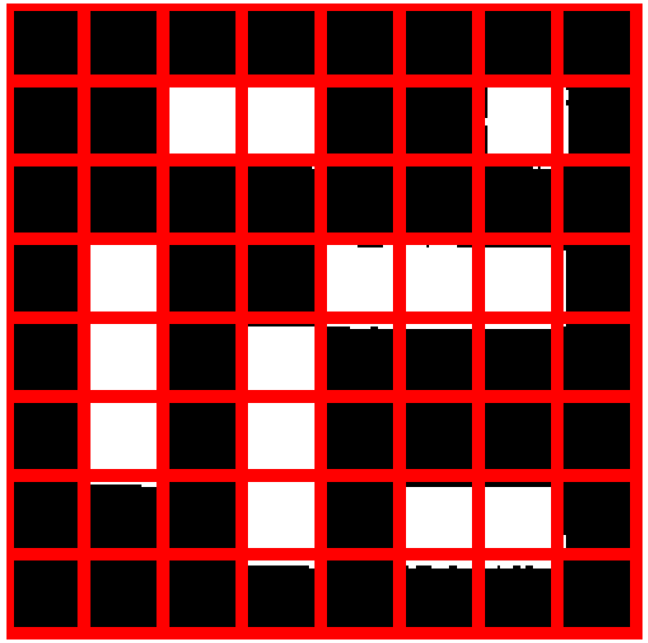


Aruco Marker Detection Algorithm

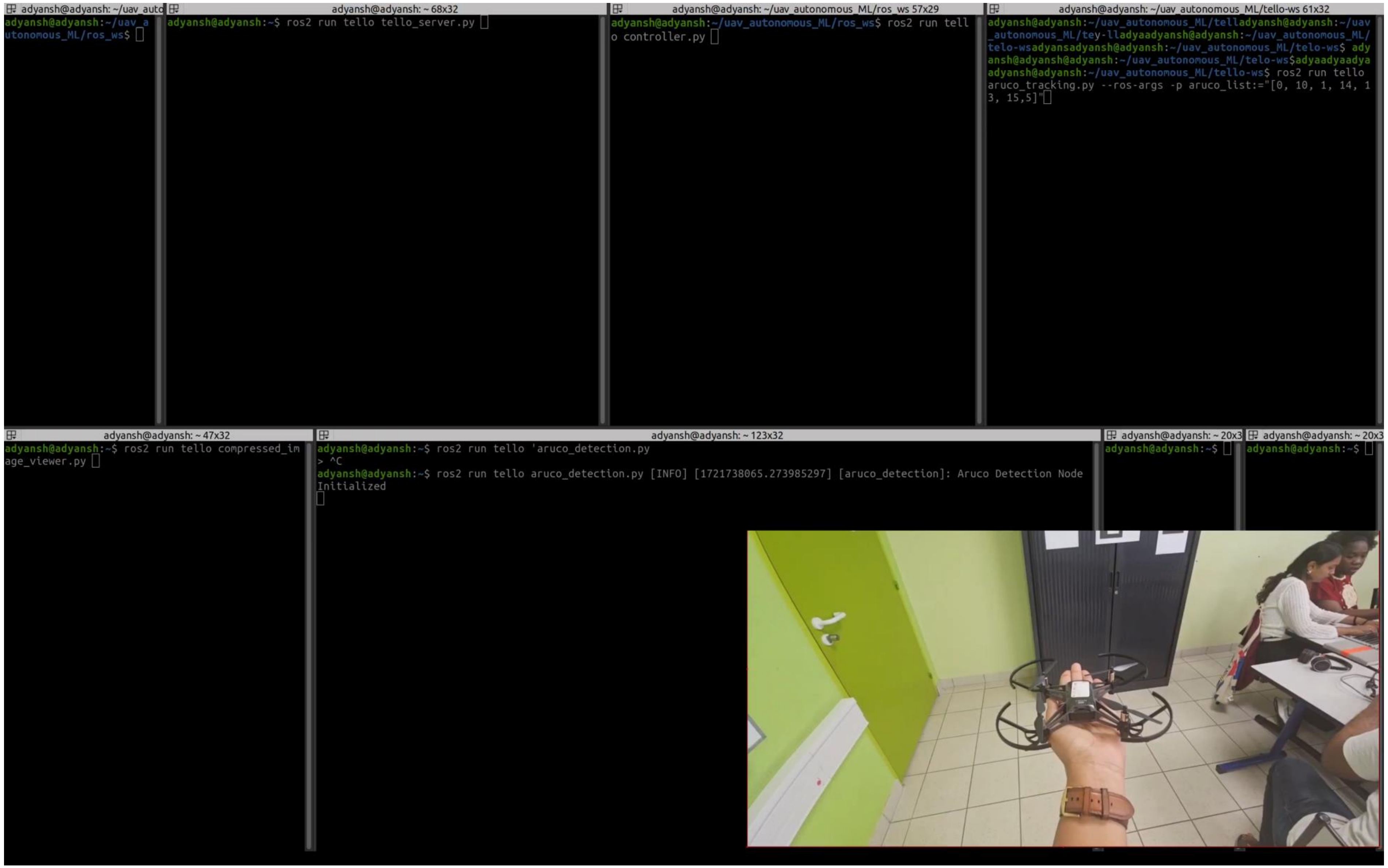
Bits Extraction

- ✓ After candidate detection, the bits of each candidate are analyzed in order to determine if they are markers or not.

- ✓ Then, the image is divided into a grid with the same number of cells as the number of bits in the marker. In each cell, the number of black and white pixels are counted to determine the bit value assigned to the cell (from the majority value):



Tello – Aruco Detection Demo



```
adyansh@adyansh: ~/uav_auto
adyansh@adyansh:~/uav_a
utonomous_ML/ros_ws$
```

```
adyansh@adyansh: ~ 68x32
adyansh@adyansh:~$ ros2 run tello tello_server.py
```

```
adyansh@adyansh: ~/uav_autonomous_ML/ros_ws 57x29
adyansh@adyansh:~/uav_autonomous_ML/ros_ws$ ros2 run tell
o controller.py
```

```
adyansh@adyansh: ~/uav_autonomous_ML/tello-ws 61x32
adyansh@adyansh:~/uav_autonomous_ML/telladyansh@adyansh:~/uav
_autonomous_ML/tey-lladyaadyansh@adyansh:~/uav_autonomous_ML/
telo-wsadyansadyansh@adyansh:~/uav_autonomous_ML/telo-ws$ ady
ansh@adyansh@adyansh:~/uav_autonomous_ML/telo-ws$adyaadyaadya
adyansh@adyansh:~/uav_autonomous_ML/tello-ws$ ros2 run tello
aruco_tracking.py --ros-args -p aruco_list:="[0, 10, 1, 14, 1
3, 15,5]"
```

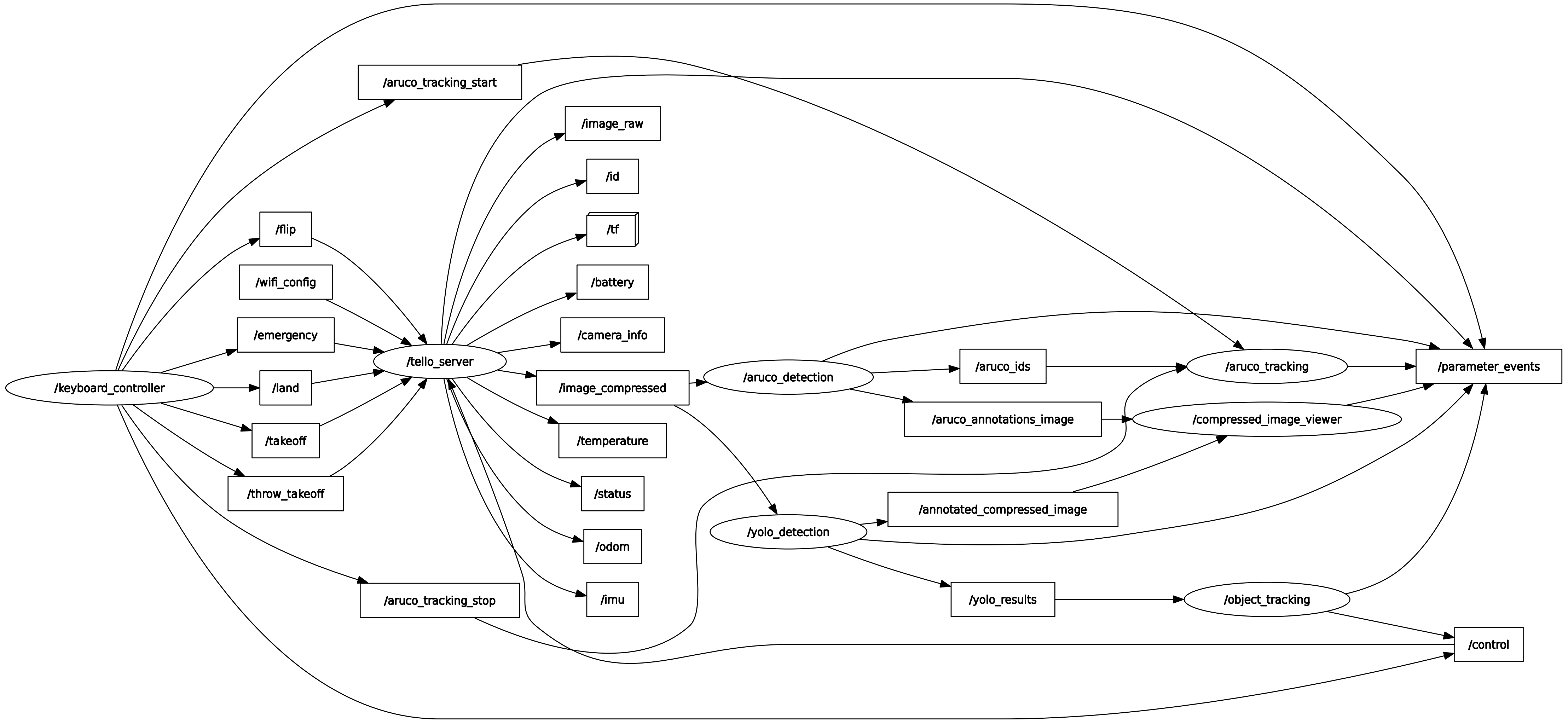
```
adyansh@adyansh: ~ 47x32
adyansh@adyansh:~$ ros2 run tello compressed_im
age_viewer.py
```

```
adyansh@adyansh: ~ 123x32
adyansh@adyansh:~$ ros2 run tello 'aruco_detection.py
> ^C
adyansh@adyansh:~$ ros2 run tello aruco_detection.py [INFO] [1721738065.273985297] [aruco_detection]: Aruco Detection Node
Initialized
```

```
adyansh@adyansh: ~ 20x3
adyansh@adyansh:~$
```

```
adyansh@adyansh: ~ 20x3
adyansh@adyansh:~$
```

Tello EDU – Architecture



Tello Swarm

Scaling the Drone Project for Real-Life Applications

Enhanced Hardware:

- ❑ Payload Capacity
- ❑ Weatherproofing
- ❑ Battery Life

Applications in Diverse Fields:

- ❑ Search and Rescue
- ❑ Delivery Services
- ❑ Infrastructure Inspection
- ❑ Precision Agriculture
- ❑ Environmental Monitoring

Conclusion: Advancing Drone Autonomy and Object Detection

This project successfully demonstrated the capabilities of drones for autonomous navigation and object detection. We utilized two distinct platforms:

- **Crazyflie 2.1:** Equipped with Flow Deck and Multiranger Deck for precise indoor navigation using ROS2 SLAM and Navigation Stack.
- **Tello EDU Drone:** Integrated with a camera for object detection, face tracking, and warehouse management applications using ArUco markers.

By combining these functionalities, we achieved:

- **Autonomous Navigation**
- **Object Detection and Object Tracking**
- **Warehouse Management with ArUco Markers**

References

1. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable markers under occlusion. IEEE. (https://ieeexplore.ieee.org/abstract/document/7105819)

2. Chadehumbe, C., & Sjöberg, J. (2020). Autonomous flight of the micro drone Crazyflie 2.1 through an obstacle course. Uppsala University, Diva Portal. (https://www.diva-portal.org/smash/record.jsf?pid=diva2:1440184&dswid=5579)

3. Pang, Y., Deng, Y., Liu, L., & Zhuang, Z. (2023). Real-time deep learning-based visual object detection and tracking for drones. IEEE (https://ieeexplore.ieee.org/abstract/document/10272390)

4. Kis, K., & Lepkova, N. (2018). Drones in logistics: A comprehensive analysis of their use and integration in the industry. Aviation (https://journals.vilniustech.lt/index.php/Aviation/article/view/10681)

THANK YOU
02:88
.36