CHAPTER 1

INTRODUCTION

1.1 Background

Augmented reality (AR) has emerged as a captivating technology that seamlessly blends virtual elements into the real world, creating an immersive and interactive experience. AR applications span a wide spectrum, encompassing entertainment, education, navigation, and industrial applications. A crucial component of AR systems is the ability to accurately detect and track fiducial markers, which serve as reference points for overlaying virtual content onto the physical environment.

Aruco markers, developed by the University of Zaragoza, are a type of fiducial marker specifically designed for AR applications. They offer several advantages over traditional markers, including high recognition speed, robustness to noise, and the ability to handle a large number of markers simultaneously. In this mini project, we will explore the development of an Aruco marker detector using a Raspberry Pi, a versatile and cost-effective single-board computer.

The project will involve utilizing the Raspberry Pi's camera module to capture images of the environment, employing OpenCV, a powerful computer vision library, to detect and track Aruco markers within the captured frames, and calculating the pose of each detected marker, which represents its location and orientation in three-dimensional space. This information will then be utilized to overlay virtual content onto the real-world environment, enabling interactive AR experiences.

1.2 Technology

1.2.1.Hardware Requirements

- **Raspberry Pi**: The Raspberry Pi is a small, low-cost computer that serves as the main platform for the project. It provides the processing power and hardware interfaces necessary to capture video from the camera and run the OpenCV and Python software.

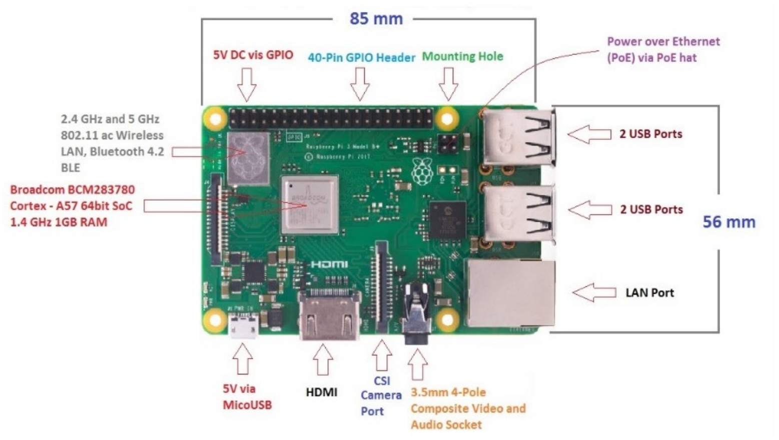

85 mm
5V DC vis GPIO
40-Pin GPIO Header
Mounting Hole
Power over Ethernet (PoE) via PoE hat
2.4 GHz and 5 GHz 802.11 ac Wireless LAN, Bluetooth 4.2 BLE
Broadcom BCM283780 Cortex - A57 64bit SoC 1.4 GHz 1GB RAM
2 USB Ports
2 USB Ports
56 mm
LAN Port
5V via MicoUSB
HDMI
CSI Camera Port
3.5mm 4-Pole Composite Video and Audio Socket

Figure 1.1: Raspberry Pi 3 B+

- **Raspberry Pi Camera Module**: The Raspberry Pi Camera Module is an add-on board that connects to the Raspberry Pi and provides the means to capture video footage. It converts real-world images into digital data for processing by the software.

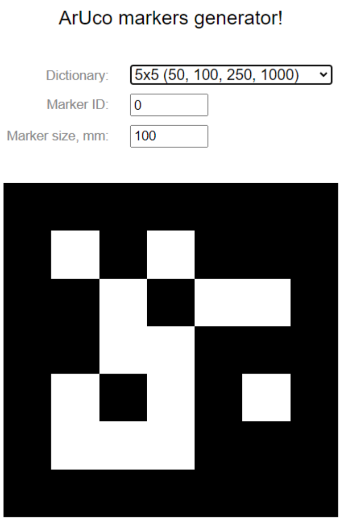
Figure 1.2: Raspberry Pi Camera Module

1.2.2.Software Requirements

- **OpenCV**: OpenCV (Open Source Computer Vision Library) is a powerful open-source library for computer vision tasks. It provides a comprehensive set of algorithms and functions for image processing, video analysis, and object detection. In this project, OpenCV is used to detect ArUco markers in the video frames captured from the Raspberry Pi camera.

- **Python**: Python is a popular programming language known for its versatility and ease of use. It is used in this project to write the script that controls the overall workflow of the system. Python's integration with OpenCV makes it an ideal choice for this project.

- **ArUco Library**: ArUco is a specific library within OpenCV that provides tools for generating and detecting ArUco markers. It simplifies the process of identifying and extracting information from these markers. [7.]


ArUco markers generator!

Dictionary: 5x5 (50, 100, 250, 1000)

Marker ID: 0

Marker size, mm: 100

Figure 1.3: Aruco Marker Generator

1.3 Summary

Aruco marker detection using Raspberry Pi is a project that utilizes the OpenCV library to identify and track Aruco markers in real-time. Aruco markers are square fiducial markers that are composed of a black border and an inner binary matrix. The binary matrix encodes the marker's ID, which can be used to distinguish between multiple markers. The project utilizes a Raspberry Pi camera to capture video frames, which are then processed by the OpenCV library to detect Aruco markers. Once a marker is detected, its ID and position are displayed on the screen. The project also demonstrates how to track Aruco markers over time, by maintaining a list of detected markers and updating their positions in each frame. This allows for applications such as augmented reality, where virtual objects can be overlaid onto the real world based on the position of Aruco markers.

CHAPTER 2
LITERATURE REVIEW

2.1 General Introduction

There has been a growing interest in the use of ArUco markers and Raspberry Pi for AR and computer vision applications in recent years. This is due in part to the ease of use and affordability of these technologies. Several studies have investigated the performance of ArUco marker detection on Raspberry Pi. These studies have shown that ArUco markers can be detected accurately and reliably under a variety of lighting conditions. They have also shown that ArUco markers can be detected in real-time, making them suitable for use in interactive AR applications.

There have also been a number of projects that have used ArUco markers and Raspberry Pi to create AR and computer vision applications. These projects have demonstrated the potential of these technologies for a wide variety of applications.

2.2 Previous Studies

In the paper published by Zoltán SIKI & Bence TAKÁCS [1], they report their experiments using OpenCV in specific land surveying projects. They concentrate on two areas. The first one is the application of close-range photogrammetry for deformation observations. Fitting a simple camera to the eyepiece of a total station we reached sub millimetre movement detection from few meters. This method was used for dynamic test load of a bridge and a floating platform. The second area is automatic ground control point (GCP) detection in images taken from drones. Optimizing the size and colours of the used ArUco markers we achieved nearly 100 percent results in different light conditions.

The paper published by [2.] Krzysztof Blachut, Michal Danilowicz1, Hubert Szolc1, Mateusz Wasala1, Tomasz Kryjak1, Mateusz Komorkiewicz [2], Testing and evaluation of an automotive perception system is a complicated task which requires special equipment and infrastructure. To compute key performance indicators and compare the results with real-world situation, some additional sensors and manual data labelling are often required. In this article, we propose a different approach, which is based on a UAV equipped with a 4K camera fying above a test track. Two computer vision methods are used to precisely determine the positions of the objects around the car – one based on ArUco markers and the other on a DCNN (we provide the algorithms used on GitHub). The detections are then correlated with the perception system readings. For the static and dynamic experiments, the differences between various systems are mostly below 0.5 m. The results of the experiments performed indicate that this approach could be an interesting alternative to existing evaluation solutions.

This paper is published by Atcha Daspan1, Anukoon Nimsongprasert1, Prathan Srichai2, and Pijirawuch Wiengchanda1 [3]. This article describes the novel design and implementation of the Robot Operating System (ROS) for the autonomous quadrotor landing on ArUco marker application. On a Raspberry Pi 4 companion computer, the ROS was set up using Ubuntu Mate 18.04. Then, to create communication between program nodes, ROS was put into practice together with autonomous landing. In the control approach, the Visual Inertial Odometry (VIO) technique, which uses vision-based localization, is employed to estimate the 3D posture. For computing the command control to direct movement quadrotor to landing on ArUco marker, the autolanding application is built. In experimental, 25 landing experiment trials were completed. The distance between the Drone's camera's center and the ArUco marker's center was calculated. In the results, the average distance accuracy during experimental validation was 11.12 cm, with a standard deviation of 3.67 cm.

The paper is published by Amonrat Prasitsupparote, Pakorn Pasitsuparoad [4]. A removal nasogastric (NG) tube of a patient is a critical problem especially the patients resist swallowing. To solve this problem, the conventional approach using a personal caretaker is a time-consuming and intense focus on the patient's hands. However, visual technology can decrease the intense focus of a personal caretaker by using image processing to evaluate the patient's gesture and warn the personal caretaker when the patient acts in a risky pose. This work illustrates the feasible solution to prevent a patient with nasogastric tube feeding on removing tube by applied the face detection using Haar and Fiducial markers which consist of color marker and ArUco marker. An image processing can evaluate the patient's gesture and warn the personal caretaker when the subject acts the risky pose. A Raspberry Pi 3 Model B and a Camera module with Python and Open CV package are applied to detect and evaluate the warning gestures with 648 measurements. Six detection methods to evaluate and warn when the patient on bed tries to remove a nasal feeding tube were performed and the results were analyzed. The results show that the detection method using ArUco marker is found to be a good candidate for the alarm system preventing nasogastric (NG) tube removal of a patient.

2.3 Summary

These research papers demonstrate the versatility and effectiveness of ArUco markers for various object tracking, positioning, and detection tasks. Their applications span from land surveying and automotive perception system testing to autonomous quadrotor landing and nasogastric tube removal prevention. ArUco markers offer a reliable, cost-effective, and accurate solution for a wide range of applications.

CHAPTER 3

PROBLEM STATEMENT

3.1 Problem Statement

Develop a real-time system for detecting and tracking Aruco markers using Raspberry Pi and OpenCV. The system should be able to identify and it should provide accurate pose estimation for each marker. The system should also be able to handle challenging lighting conditions and occlusions.

3.2 Objectives

Objective 1: Implement a robust Aruco marker detection algorithm using OpenCV.

- Successfully detect ArUco markers in real-time from camera input.
- Handle variations in marker size, orientation, and lighting conditions.
- Minimize false positives and ensure accurate marker identification.

Objective 2: Calculate accurate pose estimation for each tracked marker.

- Determine the 3D pose (position and orientation) of each tracked marker relative to the camera.
- Provide accurate orientation estimates for both in-plane and out-of-plane rotations.
- Compensate for lens distortions and perspective effects.

Objective 3: Evaluate the performance of the system under different lighting conditions and occlusions.

- Test the system's robustness to varying illumination levels and shadows.
- Assess the system's ability to track markers in partially occluded scenarios.
- Quantify the performance under different environmental conditions.

Additional considerations:

- Optimize the code for real-time performance on Raspberry Pi hardware.
- Implement a user-friendly interface for interacting with the system.
- Explore potential applications of the marker detection and tracking system.

CHAPTER 4

SYSTEM DESIGN

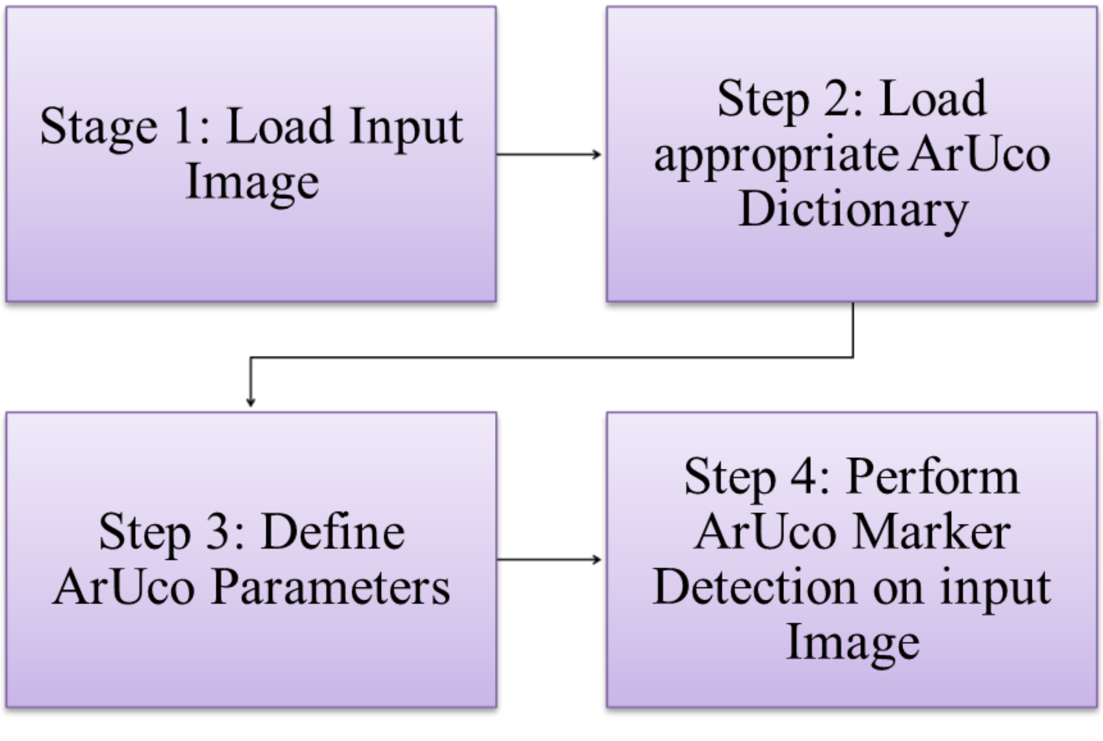
4.1 General Introduction

The system is designed with a focus on efficiency and affordability, utilizes aruco markers, visually distinctive patterns, to precisely identify and track objects within a specified area. This system, built around a raspberry pi as the central processing unit, leverages aruco detection algorithms to accurately locate aruco markers within captured images.

The system's design entails three primary components:

- **Raspberry Pi:** The raspberry pi serves as the system's computational core, responsible for executing the aruco marker detection algorithm.
- **Camera:** The camera captures images of the designated area, providing the visual data required for marker identification.
- **Software Application:** Developed using python and the opencv library, the software application processes the captured images and employs aruco detection algorithms to identify and locate aruco markers.

4.2 Block Diagram



Stage 1: Load Input Image
Step 2: Load appropriate ArUco Dictionary
Step 3: Define ArUco Parameters
Step 4: Perform ArUco Marker Detection on input Image

Figure 4.1: Flowchart of steps required to detect ArUco markers with OpenCV

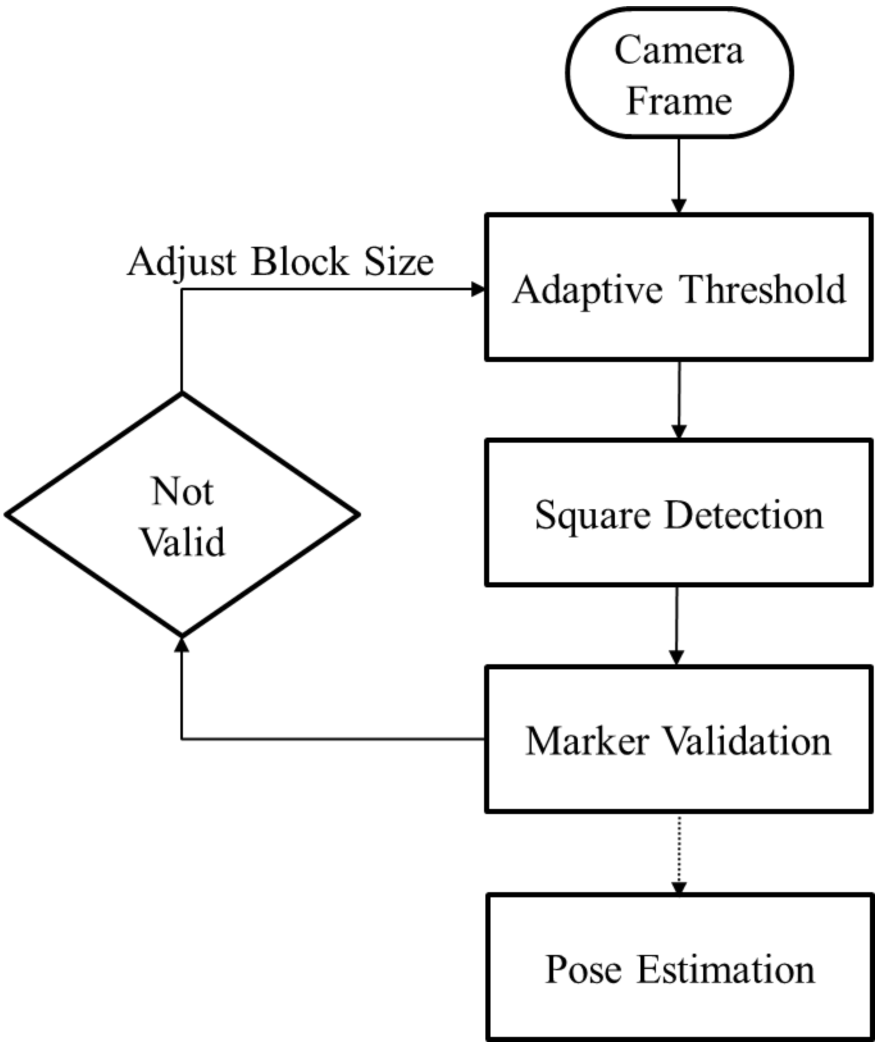


Figure 4.2: ArUco Marker Detector and Pose Estimation

4.3 Applications

1. **Augmented Reality (AR):** ArUco markers can be used to overlay digital objects onto real-world scenes, creating interactive AR experiences. For example, AR games can use ArUco markers to trigger virtual objects to appear in specific locations, while AR navigation apps can use markers to provide directions or information about landmarks.
2. **Object Tracking:** ArUco markers can be attached to objects to track their movement in real time. This is useful for applications such as robotics, where robots can use ArUco markers to track and manipulate objects, or for motion capture systems, where markers can be placed on actors to capture their movements for animation or special effects.
3. **Human-Computer Interaction (HCI):** ArUco markers can be used to create interactive interfaces that allow users to interact with computers using natural gestures or movements. For example, ArUco markers can be used to control

virtual objects or menus in AR applications, or they can be used to trigger specific actions or commands in interactive installations.

4. **Education and Training:** ArUco markers can be used to create interactive learning experiences that combine physical objects with digital content. For example, science experiments can use ArUco markers to trigger animations or explanations, or language learning apps can use markers to provide interactive exercises or pronunciation guides.
5. **Robotics and Automation:** ArUco markers can be used to provide robots with visual cues for navigation, object manipulation, and task completion. For example, robots can use ArUco markers to locate specific targets, avoid obstacles, or assemble components.
6. **Gesture Recognition**: ArUco markers can be incorporated into gesture recognition systems. By placing markers on objects or body parts, the system can interpret specific gestures, enabling hands-free control in applications such as gaming or home automation.
7. **IoT (Internet of Things):** Integrating the ArUco marker detector with Raspberry Pi allows for the development of IoT applications. For instance, markers could be used to trigger specific actions or events in a smart home environment.

4.4 Summary

The aruco marker detection system, utilizing aruco markers for object identification and tracking, leverages a raspberry pi as the central processing unit and employs aruco detection algorithms within a python-based software application. The system's design encompasses image preprocessing, marker detection, and pose estimation, enabling accurate object localization. Its cost-effectiveness, ease of implementation, and scalability make it ideal for diverse applications.

CHAPTER 5

IMPLEMEMTATION AND CODING

5.1. General Introduction

The implementation of the aruco marker detection system involves setting up the hardware components, installing the necessary software, and developing the aruco detection algorithm.

5.1.1 Hardware Setup

Raspberry Pi: The raspberry pi serves as the system's computational core and should be readily available with power supply, SD card, and network connectivity.

Camera: A compatible camera module should be connected to the raspberry pi, ensuring proper mounting and alignment.

5.1.2 Software Installation

Operating System: Install a suitable operating system, such as Raspbian, on the raspberry pi.

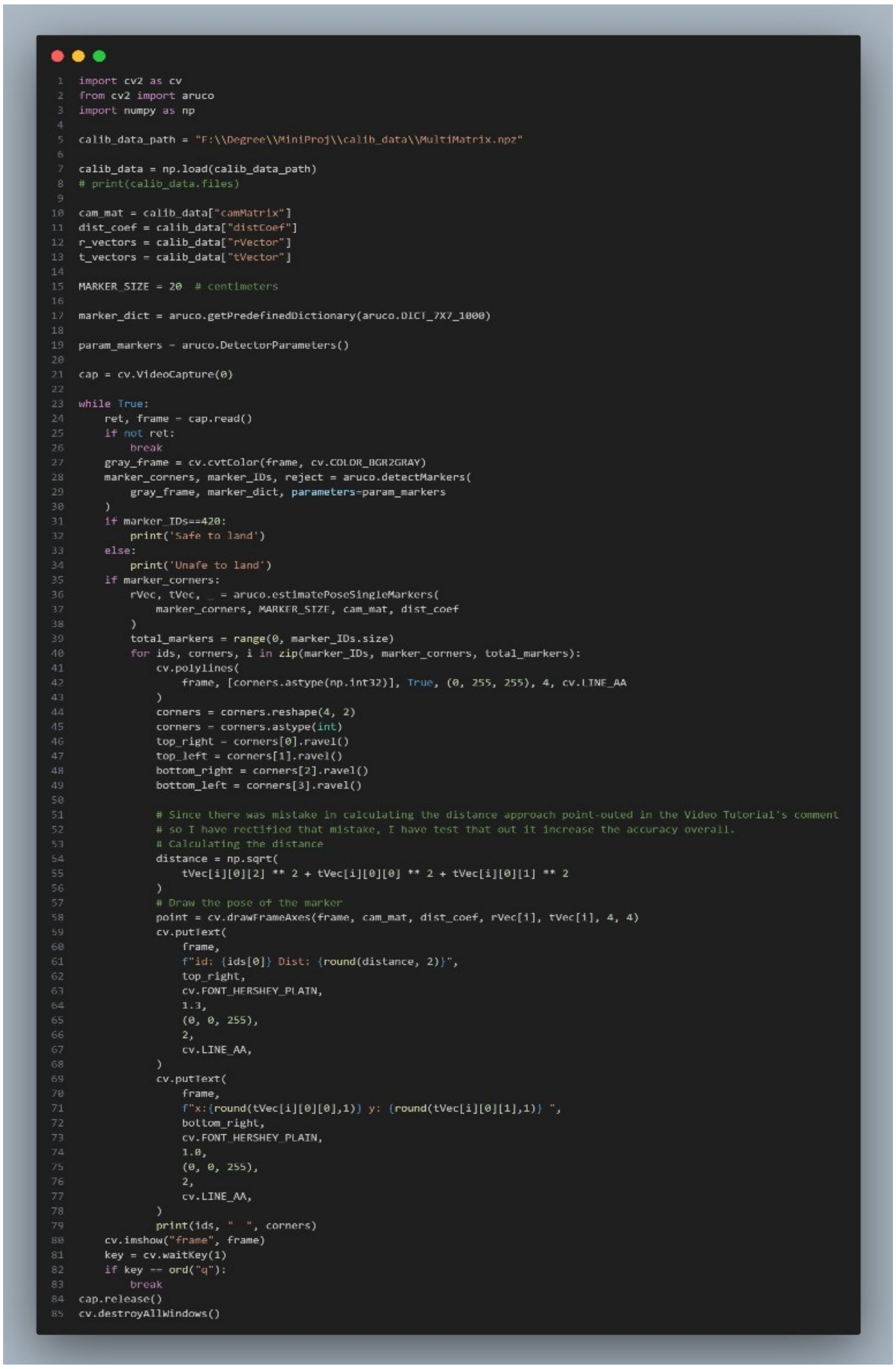
Python and OpenCV: Install Python and the OpenCV library, which provides the necessary tools for image processing and aruco marker detection.

5.2.Algorithm

1. Import the necessary libraries: NumPy, CV2, aruco from CV2, time, and math.
2. Define the ID and size of the Aruco marker.
3. Load the camera calibration data, including the camera matrix and distortion parameters.
4. Define the rotation matrix around the X-axis, taking into account the flipped reference frame.
5. Define the Aruco dictionary to be used (e.g., the original Aruco dictionary).
6. Capture the video frames from the camera, ensuring camera dimensions match the calibration procedure.
7. In a loop, read and convert the camera frame to grayscale.
8. Use the detect Markers function from the Aruco library to find markers in the frame, passing the grayscale image, dictionary, and calibration parameters.
9. If a marker is detected, use the estimatePoseSingleMarker function to estimate its rotation and position, passing the corners, marker size, and calibration parameters. Retrieve the rotation and position output.
10. Draw the detected marker and axes on the frame for visualization.
11. Print the marker's position and Euler angles on the frame.
12. Optionally, calculate the camera position and attitude in relation to the marker.

13. Repeat the loop to process subsequent frames.
14. Display the frames and set up a key to exit the process.

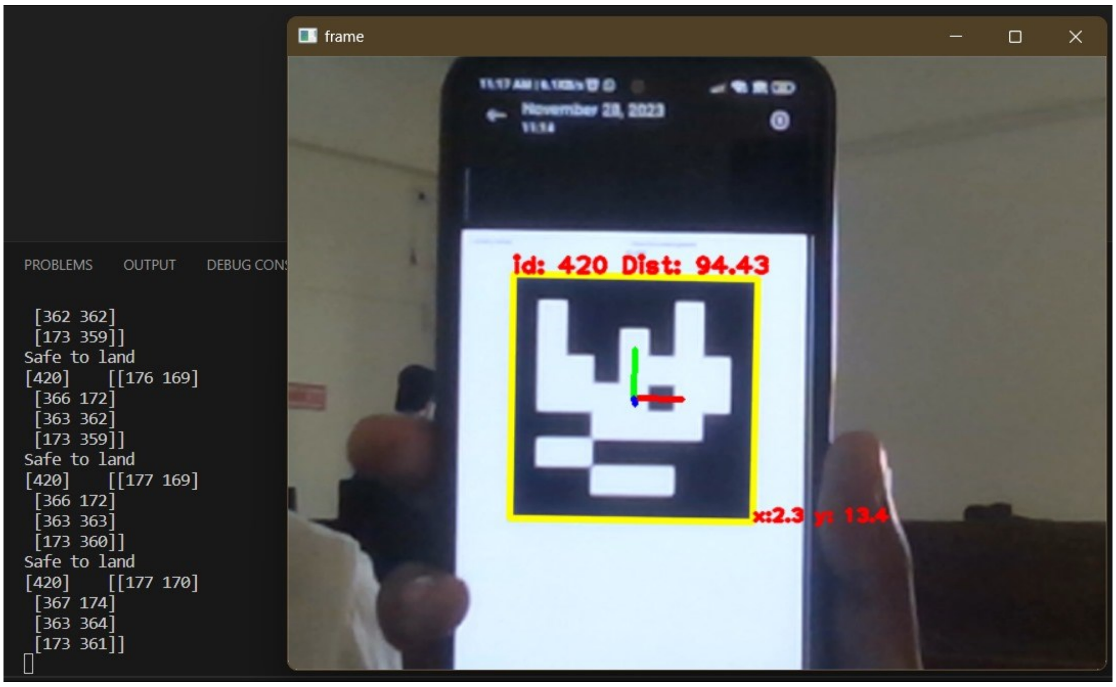
5.3.Coding



```python
import cv2 as cv
from cv2 import aruco
import numpy as np

calib_data_path = "F:\\Degree\\MiniProj\\calib_data\\MultiMatrix.npz"

calib_data = np.load(calib_data_path)
# print(calib_data.files)

cam_mat = calib_data["camMatrix"]
dist_coef = calib_data["distCoef"]
r_vectors = calib_data["rVector"]
t_vectors = calib_data["tVector"]

MARKER_SIZE = 20  # centimeters

marker_dict = aruco.getPredefinedDictionary(aruco.DICT_7X7_1000)

param_markers = aruco.DetectorParameters()

cap = cv.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray_frame = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    marker_corners, marker_IDs, reject = aruco.detectMarkers(
        gray_frame, marker_dict, parameters=param_markers
    )
    if marker_IDs==420:
        print('Safe to land')
    else:
        print('Unafe to land')
    if marker_corners:
        rVec, tVec, _ = aruco.estimatePoseSingleMarkers(
            marker_corners, MARKER_SIZE, cam_mat, dist_coef
        )
        total_markers = range(0, marker_IDs.size)
        for ids, corners, i in zip(marker_IDs, marker_corners, total_markers):
            cv.polylines(
                frame, [corners.astype(np.int32)], True, (0, 255, 255), 4, cv.LINE_AA
            )
            corners = corners.reshape(4, 2)
            corners = corners.astype(int)
            top_right = corners[0].ravel()
            top_left = corners[1].ravel()
            bottom_right = corners[2].ravel()
            bottom_left = corners[3].ravel()

            # Since there was mistake in calculating the distance approach point-outed in the Video Tutorial's comment
            # so I have rectified that mistake, I have test that out it increase the accuracy overall.
            # Calculating the distance
            distance = np.sqrt(
                tVec[i][0][2] ** 2 + tVec[i][0][0] ** 2 + tVec[i][0][1] ** 2
            )
            # Draw the pose of the marker
            point = cv.drawFrameAxes(frame, cam_mat, dist_coef, rVec[i], tVec[i], 4, 4)
            cv.putText(
                frame,
                f"id: {ids[0]} Dist: {round(distance, 2)}",
                top_right,
                cv.FONT_HERSHEY_PLAIN,
                1.3,
                (0, 0, 255),
                2,
                cv.LINE_AA,
            )
            cv.putText(
                frame,
                f"x:{round(tVec[i][0][0],1)} y: {round(tVec[i][0][1],1)} ",
                bottom_right,
                cv.FONT_HERSHEY_PLAIN,
                1.0,
                (0, 0, 255),
                2,
                cv.LINE_AA,
            )
            print(ids, "  ", corners)
    cv.imshow("frame", frame)
    key = cv.waitKey(1)
    if key == ord("q"):
        break
cap.release()
cv.destroyAllWindows()
```

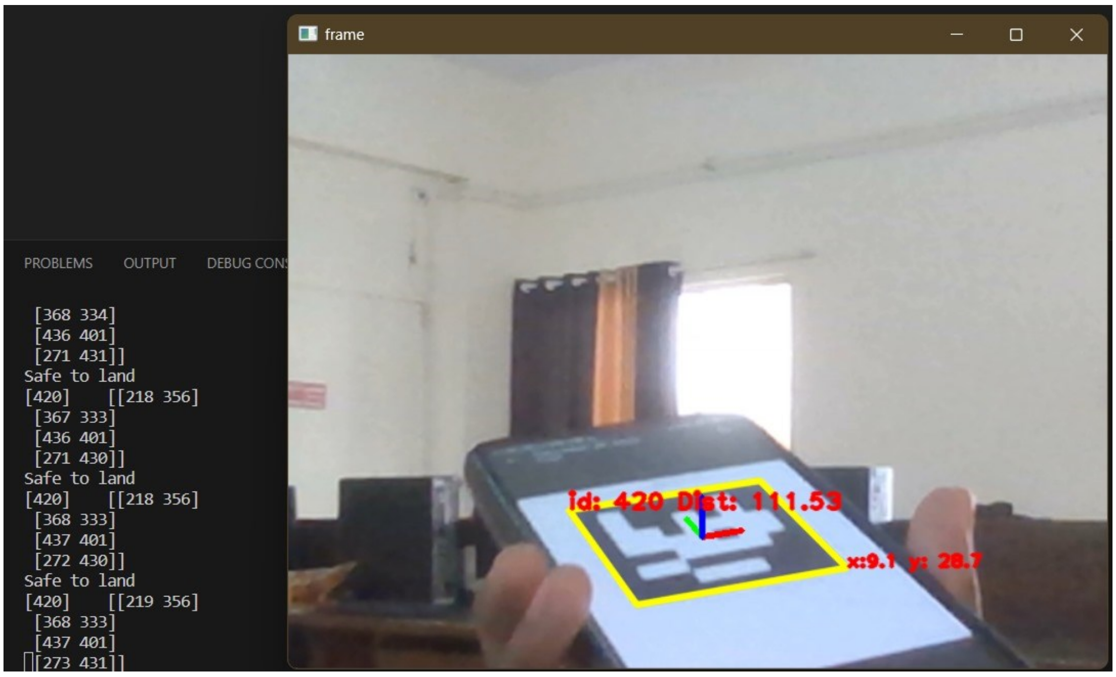
Figure 5.1: Implementation

5.4.Output



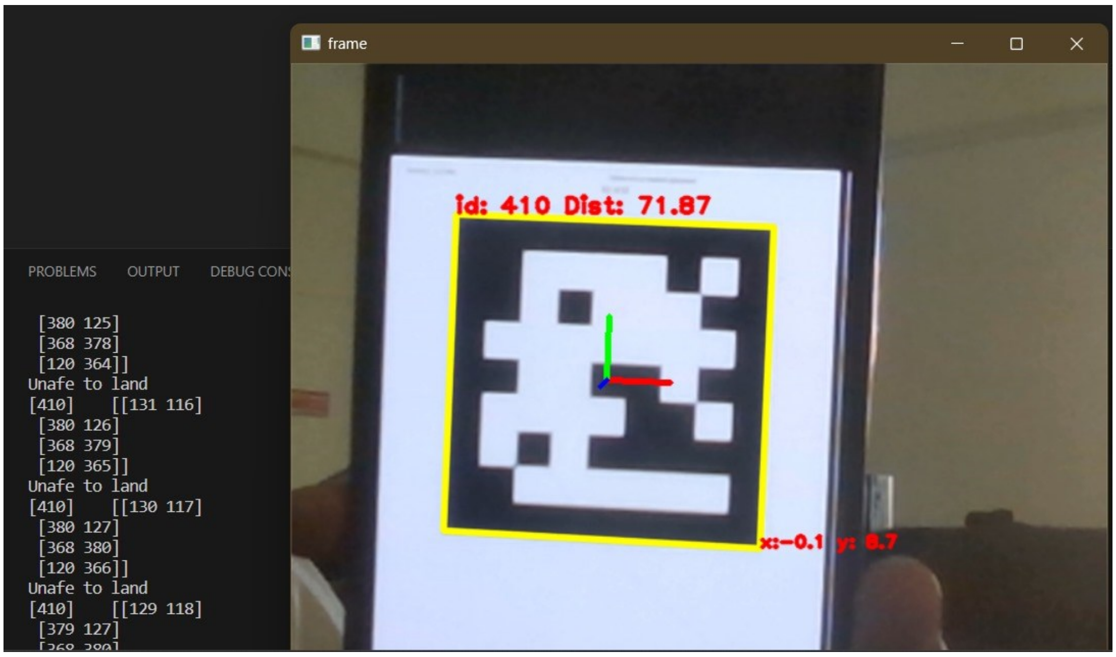
```
 [362 362]
 [173 359]]
Safe to land
[420]    [[176 169]
 [366 172]
 [363 362]
 [173 359]]
Safe to land
[420]    [[177 169]
 [366 172]
 [363 363]
 [173 360]]
Safe to land
[420]    [[177 170]
 [367 174]
 [363 364]
 [173 361]]
```

Figure 5.2: Safe Scenario 1



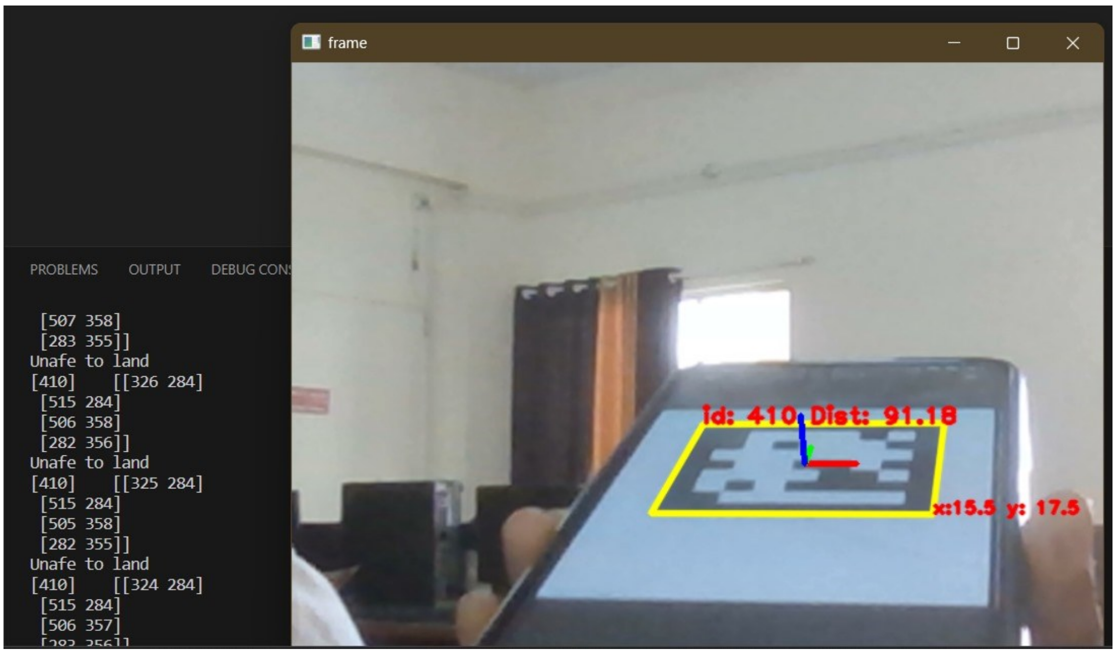
```
 [368 334]
 [436 401]
 [271 431]]
Safe to land
[420]    [[218 356]
 [367 333]
 [436 401]
 [271 430]]
Safe to land
[420]    [[218 356]
 [368 333]
 [437 401]
 [272 430]]
Safe to land
[420]    [[219 356]
 [368 333]
 [437 401]
 [273 431]]
```

Figure 5.3: Safe Scenario 2



```
 [380 125]
 [368 378]
 [120 364]]
Unafe to land
[410]    [[131 116]
 [380 126]
 [368 379]
 [120 365]]
Unafe to land
[410]    [[130 117]
 [380 127]
 [368 380]
 [120 366]]
Unafe to land
[410]    [[129 118]
 [379 127]
```

Figure 5.4: Unsafe Scenario 1



```
 [507 358]
 [283 355]]
Unafe to land
[410]    [[326 284]
 [515 284]
 [506 358]
 [282 356]]
Unafe to land
[410]    [[325 284]
 [515 284]
 [505 358]
 [282 355]]
Unafe to land
[410]    [[324 284]
 [515 284]
 [506 357]
```

Figure 5.5: Unsafe Scenario 2

5.5.Summary

The project utilizes the Raspberry Pi as the hardware platform, OpenCV and Python as the primary software tools, and the ArUco library as a specialized module for ArUco marker detection. The main aim is to achieve accurate and efficient object identification and tracking. By carefully integrating hardware and software components, the system can be successfully deployed in various applications, including object tracking in industrial settings and augmented reality experiences.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The ArUco marker detector project using Raspberry Pi and OpenCV offers a versatile and cost-effective solution for a wide range of applications in computer vision, robotics, augmented reality, and beyond. This project leverages the capabilities of the Raspberry Pi as a compact computing platform and utilizes the robust image processing functionalities provided by the OpenCV library. The detection of ArUco markers adds a valuable dimension to the project, enabling the recognition and tracking of physical markers in real-world environments.

The significance of this project lies in its practical applications across various domains. The ability to detect and decode ArUco markers opens the door to augmented reality experiences, robotics navigation, camera calibration, object tracking, human-computer interaction, and more. The flexibility of the system allows developers to adapt it to specific use cases, making it a valuable tool for education, research, and practical implementations in both industrial and consumer-oriented settings.

The project's educational value is noteworthy, providing a hands-on opportunity for learning about computer vision concepts, image processing techniques, and the practical challenges associated with real-time marker detection. It serves as a practical example for those interested in exploring the intersection of hardware and software in the context of computer vision applications.

In conclusion, the ArUco marker detector project on Raspberry Pi demonstrates the synergy between accessible hardware platforms, powerful open-source libraries, and the growing capabilities of computer vision. Its real-world applications and adaptability make it a compelling choice for individuals and organizations seeking to integrate computer vision into their projects and applications.

6.2 Future Scope

It has a promising future scope due to its versatility, affordability, and potential applications in various fields. Here are some potential future directions for this project:

1. **Enhanced marker detection and tracking**: Improving the accuracy and robustness of marker detection under challenging lighting conditions and complex environments will expand its applicability.

2. **Real-time object tracking and manipulation**: Integrating the marker detector with object tracking algorithms and robotic arms could enable real-time object manipulation and interaction.

3. **Augmented reality (AR) and mixed reality (MR) applications**: Combining the marker detector with AR and MR technologies could create immersive and interactive experiences for education, training, and entertainment.

4. **Indoor navigation and positioning**: Utilizing the marker detector for indoor navigation and positioning could provide accurate location tracking and guidance in various indoor environments.

5. **Robotics and automation**: Integrating the marker detector into robotic systems could enable precise object manipulation, path planning, and task execution.

6. **Real-time asset tracking and monitoring**: Utilizing the marker detector for real-time asset tracking and monitoring could provide valuable insights into asset location and status in various industries.

These potential future directions demonstrate the broad range of applications and advancements possible with the ArUco marker detector project using Raspberry Pi. This project has the potential to revolutionize various industries and enhance user experiences in numerous ways.

REFERENCES

[1.] Zoltán SIKI, Bence TAKÁCS "Automatic Recognition of ArUco Codes in Land Surveying Tasks" Baltic J. Modern Computing, Vol. 9 (2021), No. 1, 115-125.

[2.] Krzysztof Blachut, Michal Danilowicz1, Hubert Szolc1, Mateusz Wasala1, Tomasz Kryjak1, Mateusz Komorkiewicz "Automotive Perception System Evaluation with Reference Data from a UAV's Camera Using ArUco Markers and DCNN" Journal of Signal Processing Systems (2022) 94:675–692.

[3.] Atcha Daspan1, Anukoon Nimsongprasert1, Prathan Srichai2, and Pijirawuch Wiengchanda1 "Implementation of Robot Operating System in Raspberry Pi 4 for Autonomous Landing Quadrotor on ArUco Marker" International Journal of Mechanical Engineering and Robotics Research Vol. 12, No. 4, July 2023.

[4.] Amonrat Prasitsupparote, Pakorn Pasitsuparoad "Alarm System using Image Processing to Prevent a Patient with Nasogastric Tube Feeding from Removing Tube" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 13, No. 5, 2022.

[5.] William A. Freet "Image Recognition And Object Manipulation For Use In Autonomous Vehicles" May 2019.

[6.] JianPo Guo1*, PeiZhang Wu1, WeiRu Wang "A Precision Pose Measurement Technique Based On Multicooperative Logo" JianPo Guo et al 2020 J. Phys.: Conf. Ser. 1607 012047.

[7.] ArUco Marker Generator https://chev.me/arucogen/\

[8.] Adrian Rosebrock "Detecting ArUco markers with OpenCV and Python" https://pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-and-python/