Task 3.1: Detection and Pose Estimation of ArUco markers

Aryan A Kumar

***Abstract*— *"Pose estimation is of great importance in many computer vision applications: robot navigation, augmented reality, and many more."* - OpenCV.org**

Both object detection and pose estimation are pretty important problems in Computer Vision, their application being in military drones and other such UAVs for detection of enemy hideouts or estimation of likelihood of a tumor in medical image analysis, etc. For this task, I used OpenCV to implement the detection and pose estimation of ArUco markers in images and in live video capture.

I. INTRODUCTION

The problem can be divided into two subtasks:

1) The detection of ArUco markers
2) Estimation of their relative pose.

Detection has been performed using OpenCV by:

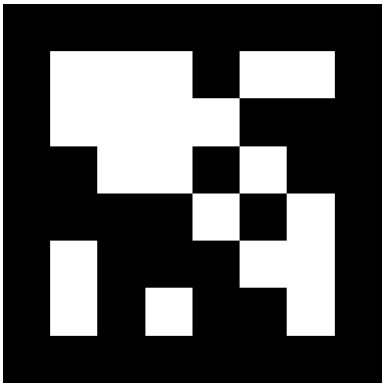
1) Specifying the ArUco dictionary.
2) Creating the parameters to the ArUco detector. (typically using the default values)
3) Applying the cv2.aruco.detectMarkers to actually detect the ArUco markers in the image or video stream.

The second part was done by generating calibration matrix and distortion coefficients for our camera and using them to estimate the relative pose.

II. PROBLEM STATEMENT

In this task, the following needs to be done:

1) Taking a picture of the ArUco marker given below and estimating its **relative pose**.



2) Printing the pose and tag id on the image.

3) Detecting and estimating the pose on a live video capture using webcam at different distances.

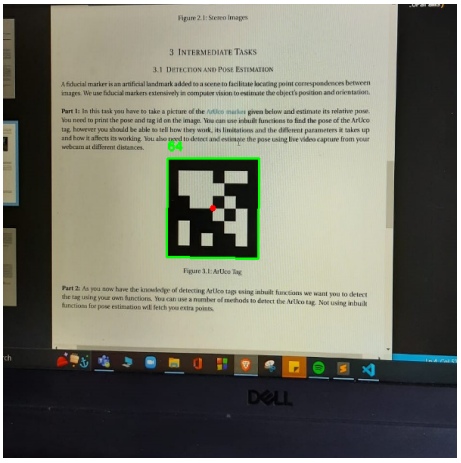
III. RELATED WORK

As for the first part, only the method using inbuilt functions in OpenCV like was considered because of its **efficiency** and **ease of implementation** compared to most other approaches. And, it worked. For the second part, a suitable repository was referred to due to the same reasons.

IV. APPROACH

The task can be divided into two parts - Marker **detection** and Camera **pose estimation**. Both parts were done using code from suitable repositories. The detection process involved:

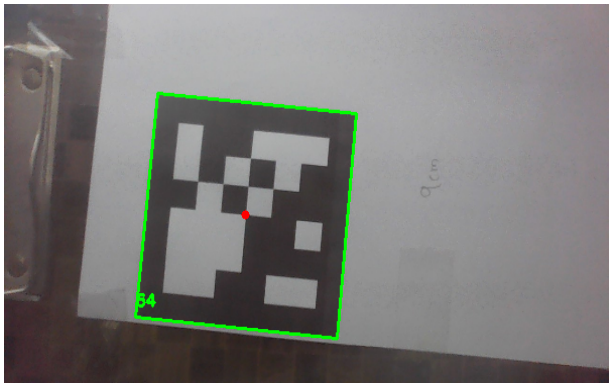
1) Loading the input image from disk.
2) Verifying that the supplied ArUCo tag exists and is supported by **OpenCV**.
3) Loading the ArUCo dictionary, grab the ArUCo parameters(set to default), and detecting the markers. For the ArUco parameters used for detection, the default parameters returned by *cv2.aruco.DetectorParameters_create* , are typically sufficient.
4) Identify marker corners and draw a box around them using *cv2.line*.
5) Display the final image.



After applying ArUco tag detection, the *cv2.aruco.detectMarkers* method returns three values:

1) corners: A list containing the (x, y)-coordinates of our detected ArUco markers.

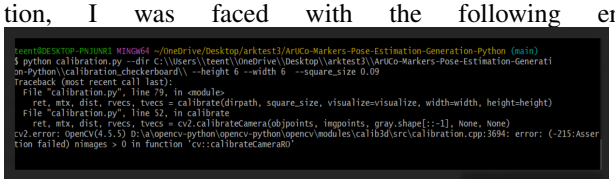
2) ids: The ArUco IDs of the detected markers.
3) rejected: A list of potential markers that were found but ultimately rejected.



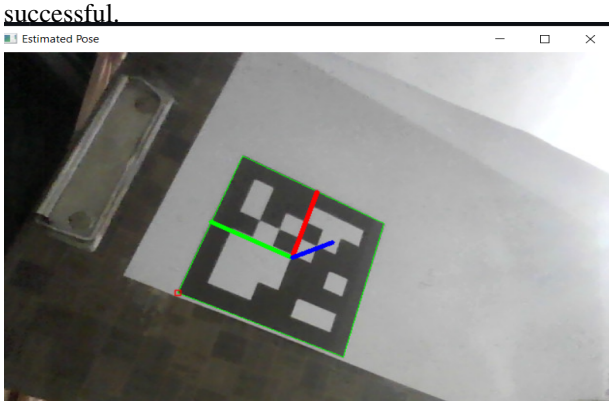
For successful detection, the type used to detect the ArUco tag should be the same type that was used for its generation. In this case the type that worked was *DICT 6X6 100.*

The next step was calibration. Here I tried to calibrate our camera by providing images of the marker in different poses. The goal was to generate the **calibration matrix** and the distortion coefficients using images from the webcam itself.

However, while performing the calibration, I was faced with the following error:

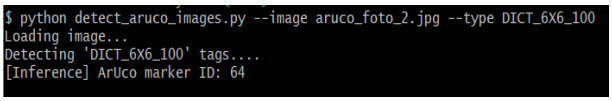


Ultimately I decided to use the existing calibration matrix and distortion coefficient files in the repository and to my surprise, it worked out fine and the **pose estimation** was successful.



The above image is a screenshot taken from a **live video** feed of my webcam.

V. Results and Observation



- The tag **ID** of the marker was 64. The marker dictionary was DICT_6X6_100.
- Both detection and pose estimation were successful for varied distances and orientations.

VI. Future Work

- Pose estimation was done through calibration matrix provided in the repository, and not from the webcam itself. This could generate problems if one uses a different webcam.
- The algorithm seemed to identify the marker only up to a certain inclination. There can scope for improvement there.
- ArUco markers are relatively simpler objects. It should be possible to detect more **complex markers** like QR Codes.

CONCLUSION

The task was ultimately one of **pose** estimation, which is a problem critical to most CV projects.
The process is based on finding correspondences between points in the real environment and their 2d image projection. A relatively easier way to perform this is using a marker. This task involved ArUco: a **binary** square marker. The main benefit of these is that a **single** marker provides enough correspondences (its four corners) to obtain the relative pose.

The pose estimation required detection, which was efficiently performed by *detectMarkers* function in OpenCV. Next was the generation of the calibration matrix. Finally pose estimation was performed using the *estimatePoseSingleMarkers* function.
The net result was a program capable of detecting ArUco markers and estimating their relative pose using images or live video feed over **different** distances.

As **aerial robotics** relies a lot on Computer Vision, the importance of object detection and pose estimation can be realised. The operations of **perception and control**(for example in case of **UAVs**) will heavily depend on these problems and so it becomes crucial that we solve them to an appreciable extent.

References

[1] https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
[2] https://github.com/GSNCodes/ArUCo-Markers-Pose-Estimation-Generation-Python
[3] https://pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-and-python/