
# ENPM 673 - Perception for Autonomous Robots

*Project - 1*

Kumara Ritvik Oruganti

117368963

okritvik@umd.edu

A. James Clark School of Engineering

University of Maryland

College Park - 20742

March 8, 2022

# Contents

# List of Figures

# Chapter 1

# Problem 1

## 1.1 Problem Statement A

The task here is to detect the April Tag in any frame of the video (just one frame). Notice that the background in the image needs to removed so that you can detect the April tag. You are supposed to use Fast fourier transform (inbuilt scipy function fft is allowed) to detect the April tag. Notice that you are expected to use inbuilt functions to calculate FFT and inverse FFT.

## 1.2 Solution

### 1.2.1 Flow Chart

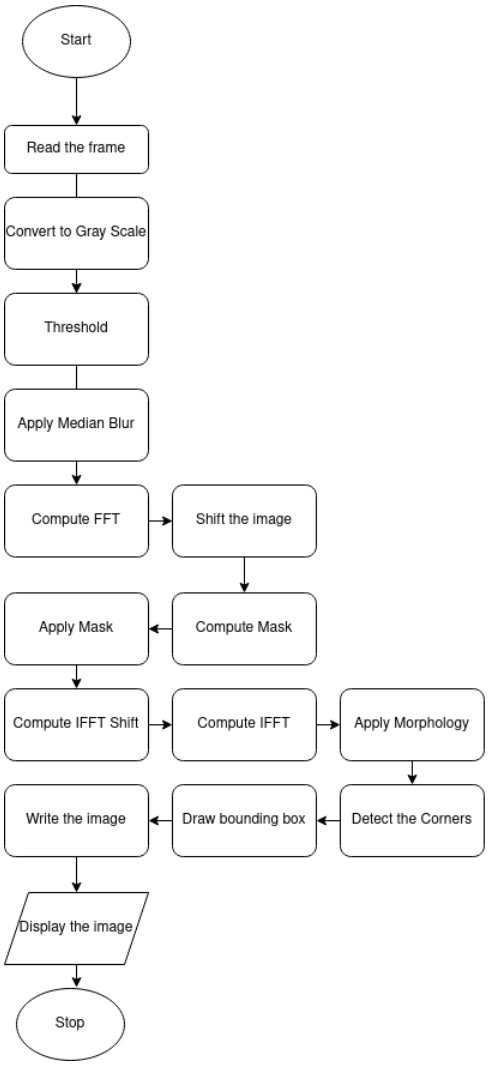
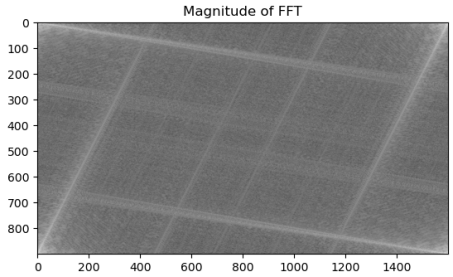


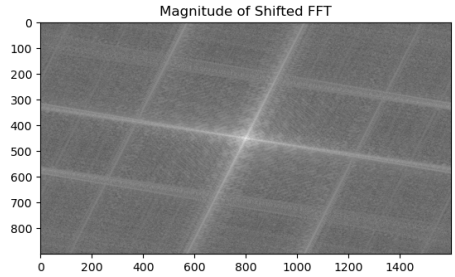

Figure 1.1: Flow Chart of the Algorithm to detect the AR Tag

The flow chart for the algorithm implementation is given in the figure 1.1. Since the inbuilt functions of CV2 cannot be used to detect any contours, the corners of the AR tag are determined using the min-max approach.

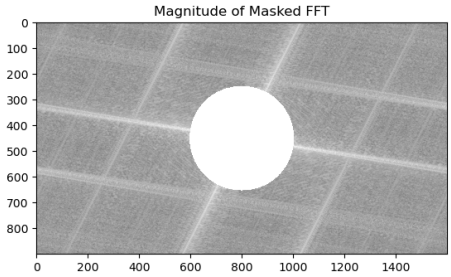
### 1.2.2 Results



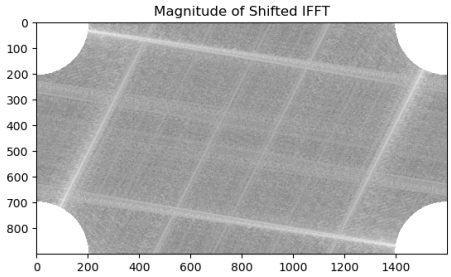

(a) FFT Magnitude



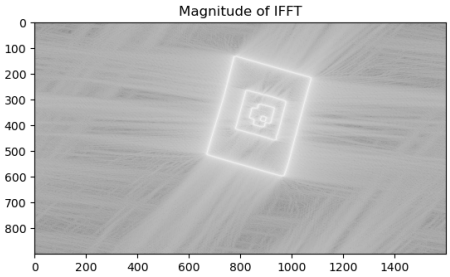

(b) FFT Shifted Magnitude




(c) Masked FFT




(d) Shifted IFFT




(e) IFFT

Figure 1.2: Magnitude outputs of the intermediate steps

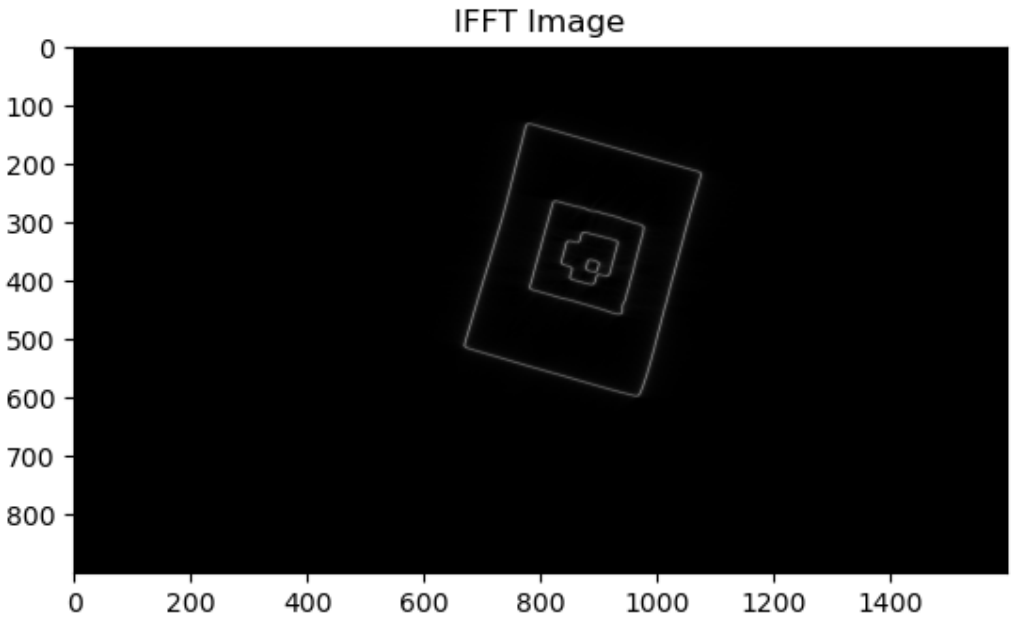



Figure 1.3: Edges Detected using FFT Algorithm

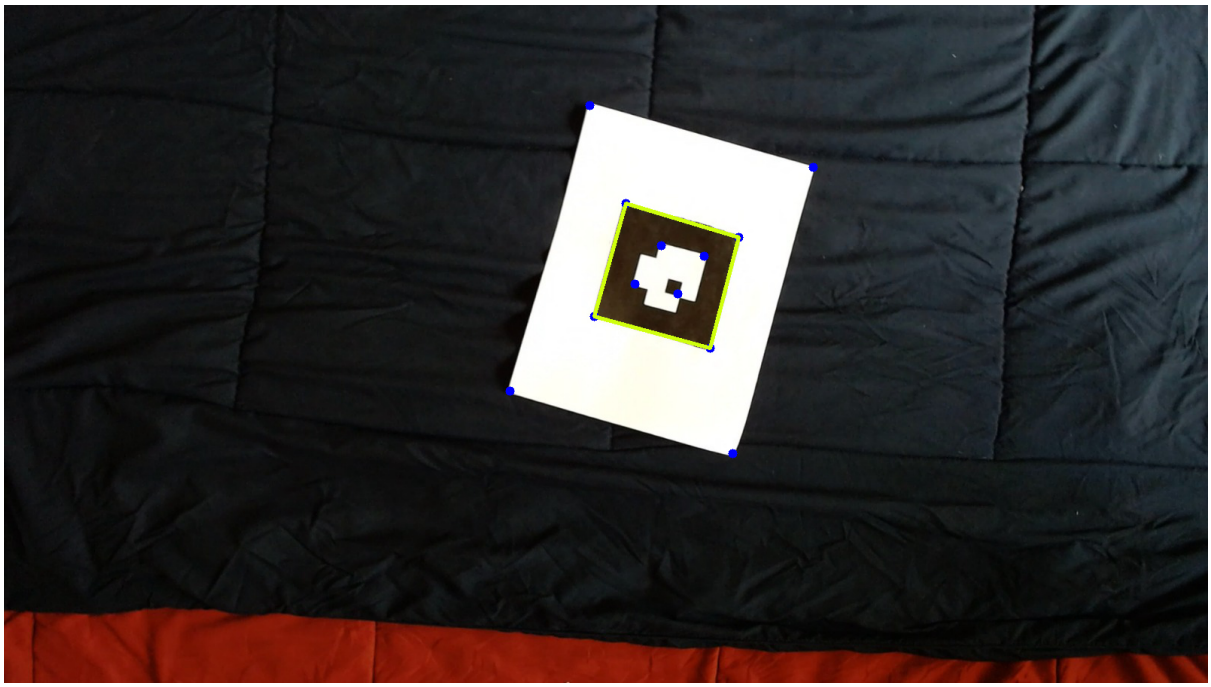


Figure 1.4: AR Tag detection using the edges detected from FFT Algorithm

### 1.2.3 Problems Faced

This pipeline is used for the next problem statements to detect the AR Tag from the video. Hence the parameters like the confidence level, distance between the corners, dilation, erosion, Morphology Operators, Kernel, Blurring, Thresholding, Computing the mask for FFT required a lot of trial and error to make sure that the pipeline can be reused for the next problems. The results for every iteration were compared and the best one is chosen which took ample amount of time for the project. The detection algorithm is also not perfect due to the corners detected at the edges of the white paper or the AR Tag and some white spots in the background which couldn't be removed.

## 1.3 Problem Statement B

You are given a custom AR Tag image that which is used as reference. This tag encodes both the orientation as well as the ID of the tag. Develop a detection pipeline to decode the information and orientation of the AR Tag for any given rotation.

## 1.4 Solution

### 1.4.1 Flow Chart

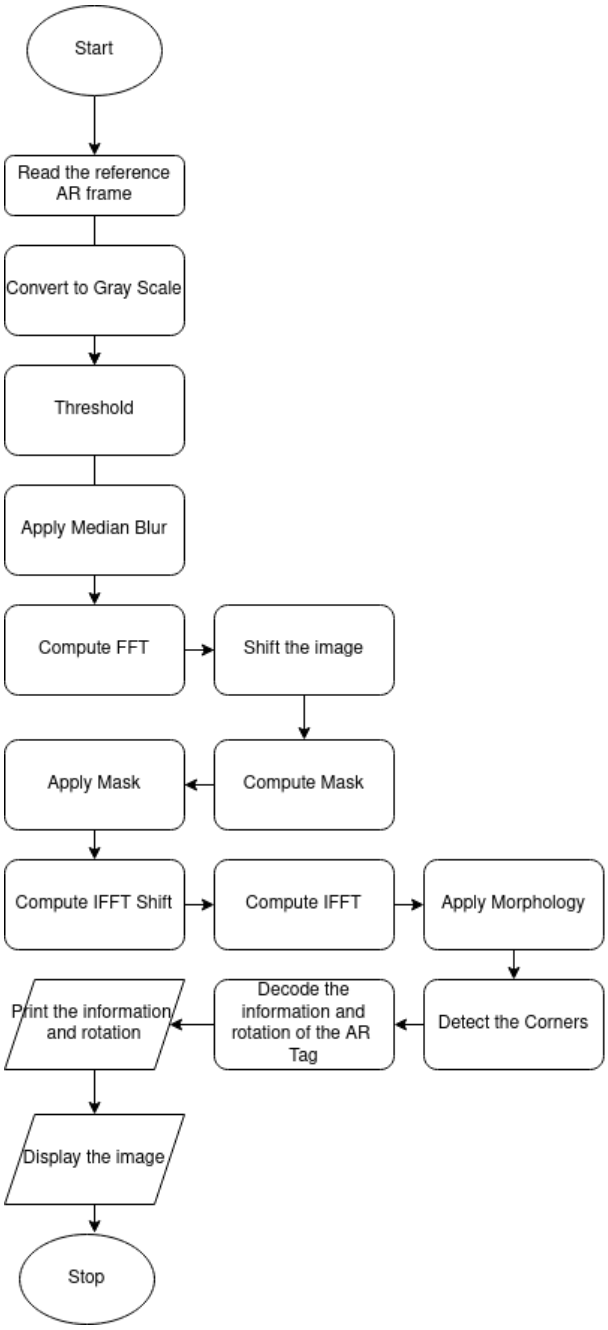



Figure 1.5: Flow chart for Tag detection and decoding using the edges detected from FFT Algorithm

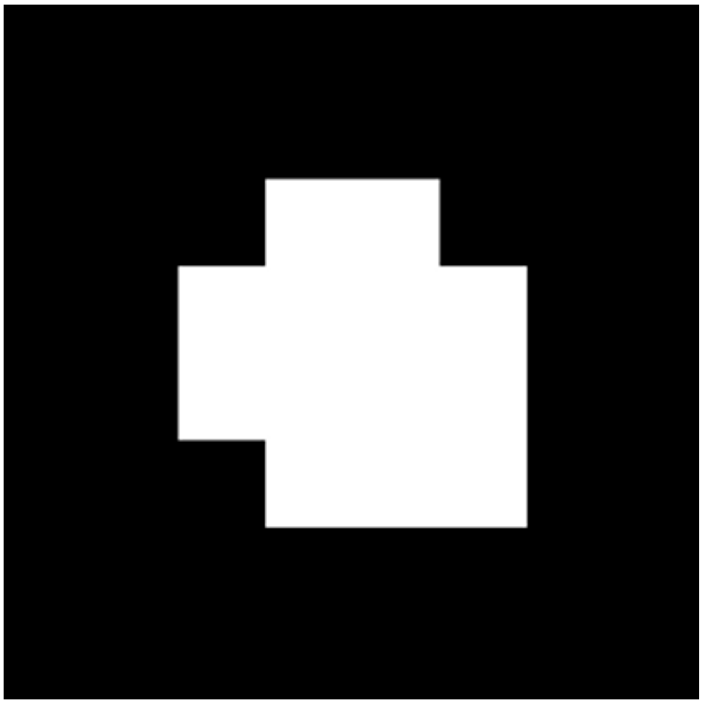


Figure 1.6: Reference AR Tag

### 1.4.2 Results

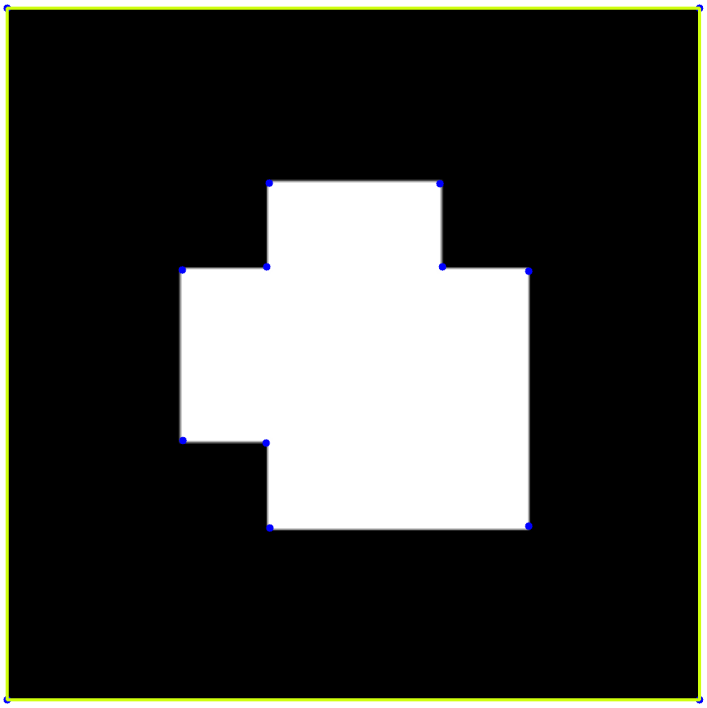


Figure 1.7: Detected AR Tag

```
(ENPM673-Project1) okritvik@okritvik-Alienware:/media/okritvik/DATA/UMD/Course_W
ork/ENPM673-Perception for Autonomous Robotics/Project_1$ python3 project1_1b.py

XmbTextListToTextProperty result code -2
XmbTextListToTextProperty result code -2
XmbTextListToTextProperty result code -2
XmbTextListToTextProperty result code -2
XmbTextListToTextProperty result code -2
XmbTextListToTextProperty result code -2
Normal Orientation
INFORMATION:  15
Done
```

Figure 1.8: Information and Rotation

Note that the Reference Tag (1.6) can be rotated and saved as ref_tag1.png in the workspace of the codes. The output of the console should give the information and the orientation of the given AR code as the pipeline is robust to detect the orientation of the AR Tag. The AR Code detection from the given video is done in the next chapter. This problem mainly focuses on dividing the AR Tag to 8 grids and calculating the position of the white grid on the corners after removing the black padding. The position of the white corner gives the rotation of the AR Tag. The inner 2x2 grid has the information and after the rotation, the AR Code information is decoded by using the LSB to MSB approach (base 2).

# Chapter 2

# Problem 2

## 2.1 Problem Statement A

Perform homography estimation on this in order to perform some image processing operations, such as superimposing an image over the tag. Superimpose a Testudo Image on the AR Tag. It is implied that the orientation of the transferred template image must match that of the tag at any given frame.

Figure 2.1: Testudo image to be superimposed

## 2.2 Solution

### 2.2.1 Flow Chart

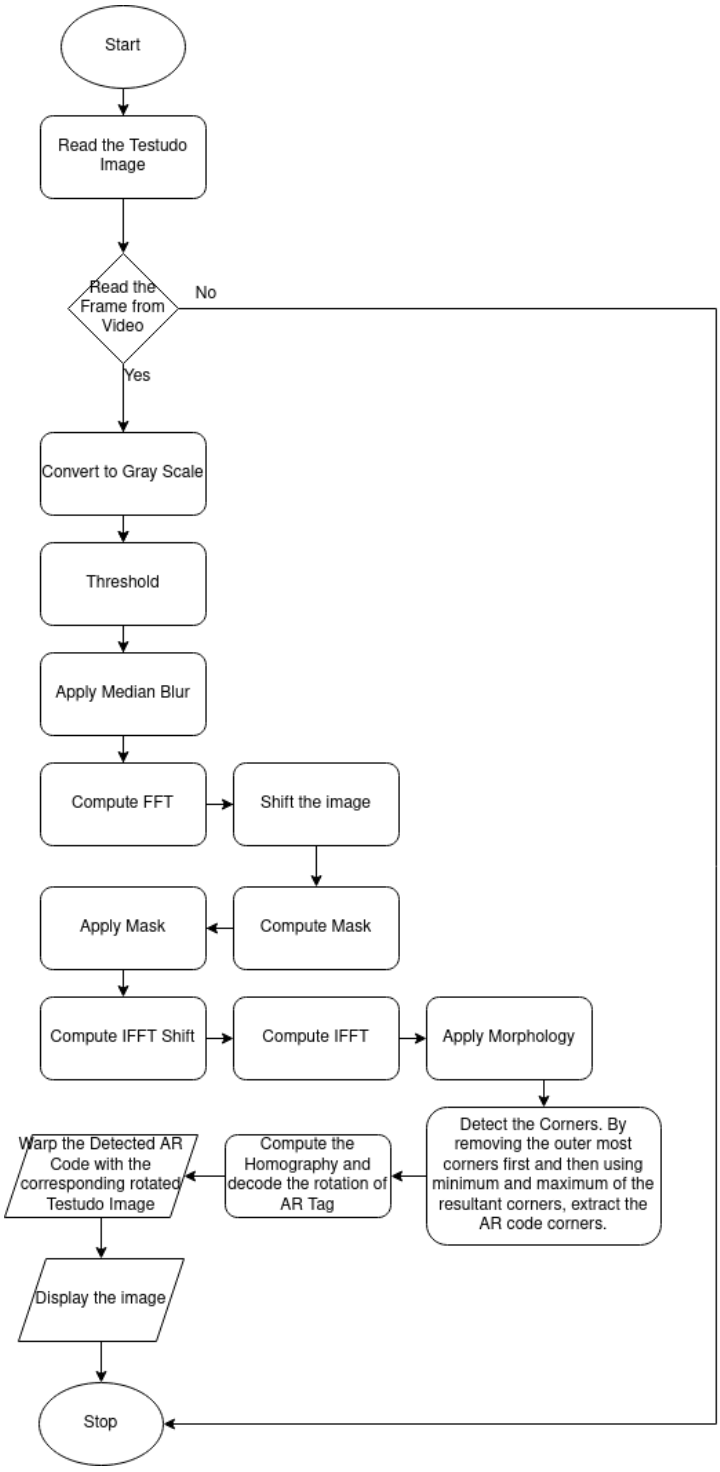



Figure 2.2: Flow chart to superimpose reference testudo image on the AR Tag

### 2.2.2 Results

The algorithm shown in figure 2.2 detects the corners of the AR Tag, decodes the information and orientation by projecting the corners of the AR Tag on to 160x160 image frame. By using the Homography and inverse warping, the testudo image 2.1 is super imposed on the AR Tag. Note that the Homography is computed between AR Tag and the 160x160 frame corners.


Figure 2.3: Testudo image on the AR Tag

Here is the ***YouTube link*** for the entire video.

### 2.2.3 Problems Encountered

The corners that are detected using the pipeline has some noise due to the presence of white pixels in the background. The algorithm to detect the AR Tag assumes that there are only four corners that are detected outside of the AR Tag which are the corners of the white paper. The edge detection using the FFT gave better results in compared with the canny edge detector for this algorithm because of the assumption of four corners that can be detected outside the AR Tag corners. During the whole video, corners were not perfectly detected for each and every frame. As we are not allowed to use any of the inbuilt functions, the tuning of the parameters for Shi-Tomasi, Median Blur, Kernels for Morphology operations, thresholding the image, mask radius plays a major role in

reducing the noise. With a lot of trial and error experiments with the parameters, the output was decent but not perfect due to the above said constraints. I also experimented to preserve the previous coordinates when there are no proper corners detected but the coordinates are changing fast from one frame to another and hence there needs to be a filtering technique to determine or estimate the next coordinates.

## 2.3 Problem Statement B

Augmented reality applications generally place 3D objects onto the real world, which maintain the three-dimensional properties such as rotation and scaling as you move around the "object" with your camera. In this part of the project you will implement a simple version of the same, by placing a 3D cube on the AR Tag.

## 2.4 Solution

### 2.4.1 Flow Chart

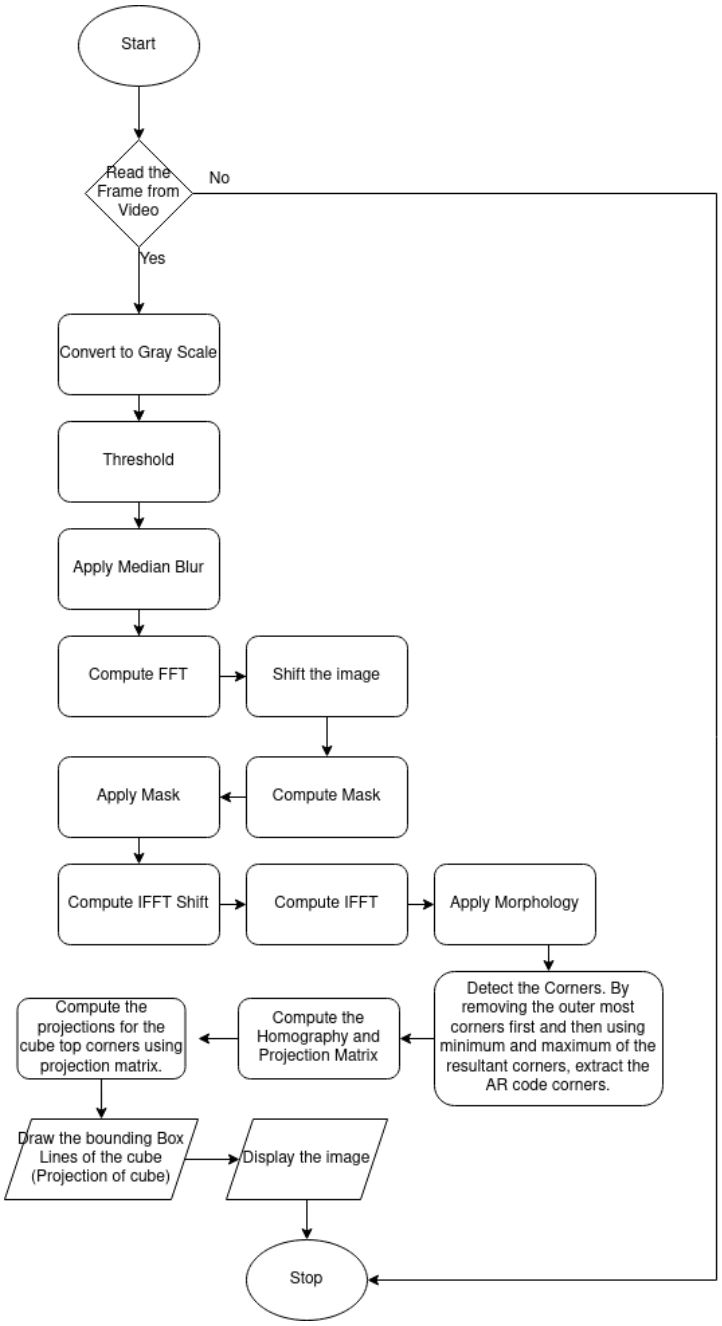



Figure 2.4: Flow chart to project a cube on the AR Tag

The algorithm shown in figure 2.4 detects the corners of the AR Tag, decodes the information and orientation by projecting the corners of the AR Tag on to 160x160 image

frame. By using the Homography and Camera Intrinsic Parameters Matrix, a 3D cube is projected on the AR Tag.

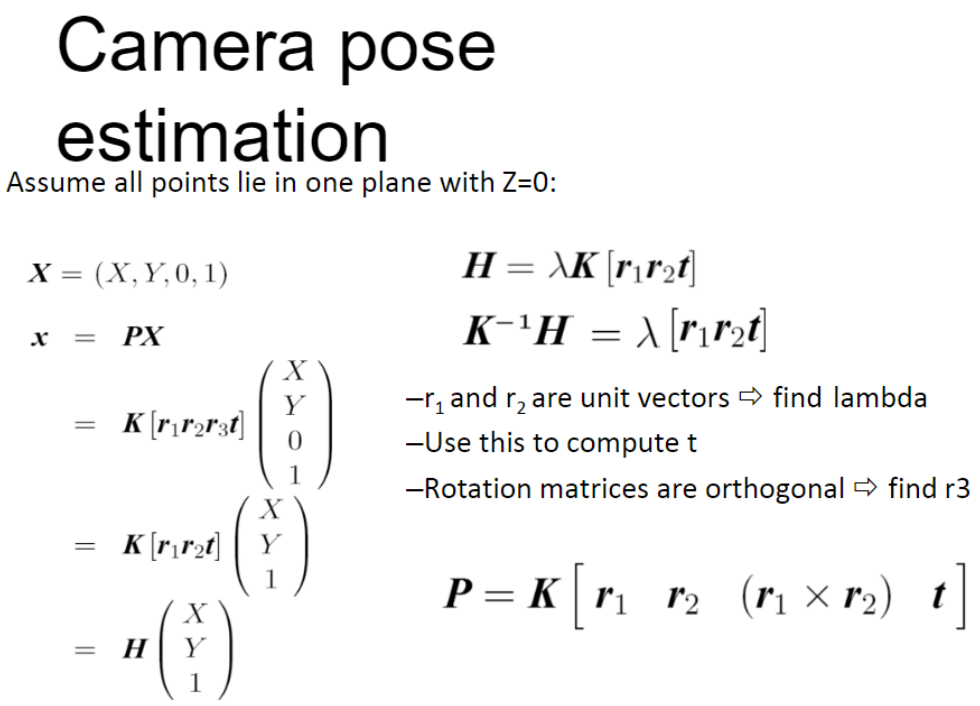



Figure 2.5: Equations to compute Projection Matrix [1]

### 2.4.2 Results

Figure 2.5 along with the supplementary document provided to compute the projection matrix is used to calculate the Rotation and Translation Matrix using the Homography matrix H and Camera Intrinsic Parameter Matrix K. The projection matrix is then multiplied with the 3D coordinates of the top corners of the cube to get corresponding x,y and z coordinates on the image plane. Note that there is no need recompute the bottom corners of the cube because the AR Tag corners will be the bottom corners of the cube and these can be used to construct the 3D cube projection. Below is a frame from the resultant output video.

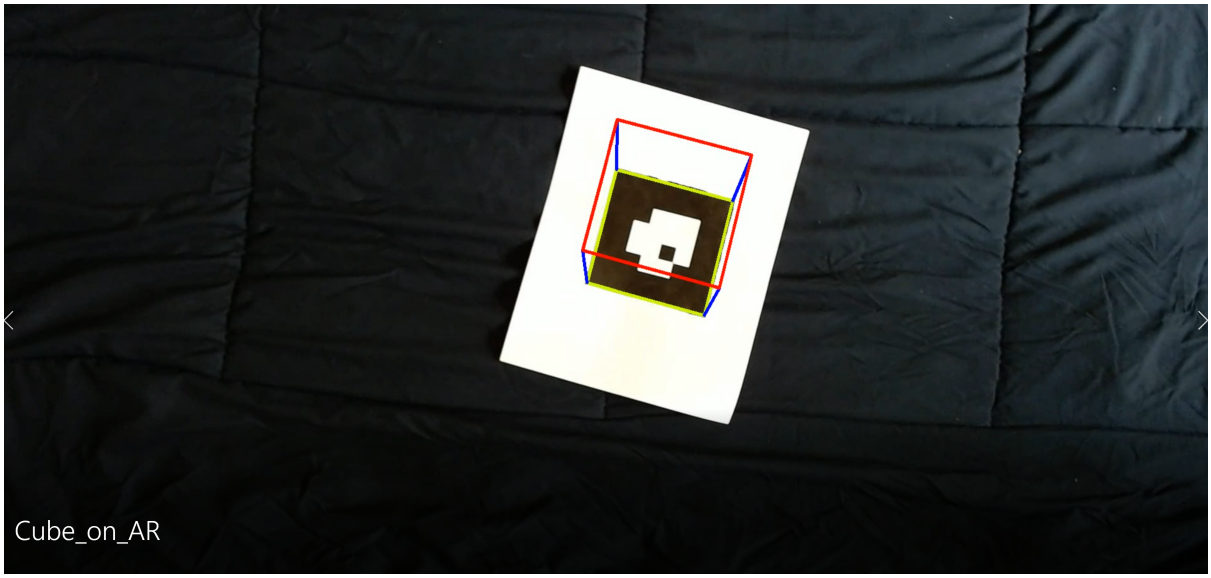



Figure 2.6: Flow chart to superimpose reference testudo image on the AR Tag

Here is the ***YouTube link*** for the output of the above problem.

### 2.4.3 Problems Encountered

The major problem is due to the detection of the corners along with the noise in the background. The corners are not perfect throughout the video and due to that the Homography matrix and Projection matrix change and thereby the coordinates of the top corners of the cube also changes. Another key point is that the homography should be computed between the 160x160 frame and the AR Tag corners, not the other way round as in the Section 2.2 which took good amount of time to figure.
The outputs can be improvised using the contour detections and corresponding area detection. This project helped to understand the concepts, color space, requirement for blurring, tuning different parameters to get the output and develop algorithm from scratch.

The code for the project can be found in the ***GitHub link***

# Bibliography

[1] Lecture Notes, ENPM 673.