Project 1 Report

ENPM 673 - Perception for Autonomous Robots

Team 5

Prof. Dr. Mohammed Charifa

UNIVERSITY OF
MARYLAND

Nikhil Lal Kolangara (116830768)
Kartik Venkat (116830751)
Kushagra Agrawal(116700191)

This paper represents our own work in accordance with University regulations.

Contents

1 Introduction

This project will focus on detecting a custom AR Tag (a form of fiducial marker), that is used for obtaining a point of reference in the real world, such as in augmented reality applications. There are two aspects to using an AR Tag, namely detection and tracking, both of which will be implemented in this project. The detection stage will involve finding the AR Tag from a given image sequence while the tracking stage will involve keeping the tag in "view" throughout the sequence and performing image processing operations based on the tag's orientation and position (a.k.a. the pose). You are provided multiple video sequences on which you have to test your code.

2 Problem 1 - Detection

Question 1 *You are given a custom AR Tag image, as shown in Fig. 1, to be used as reference. This tag encodes both the orientation as well as the ID of the tag.*

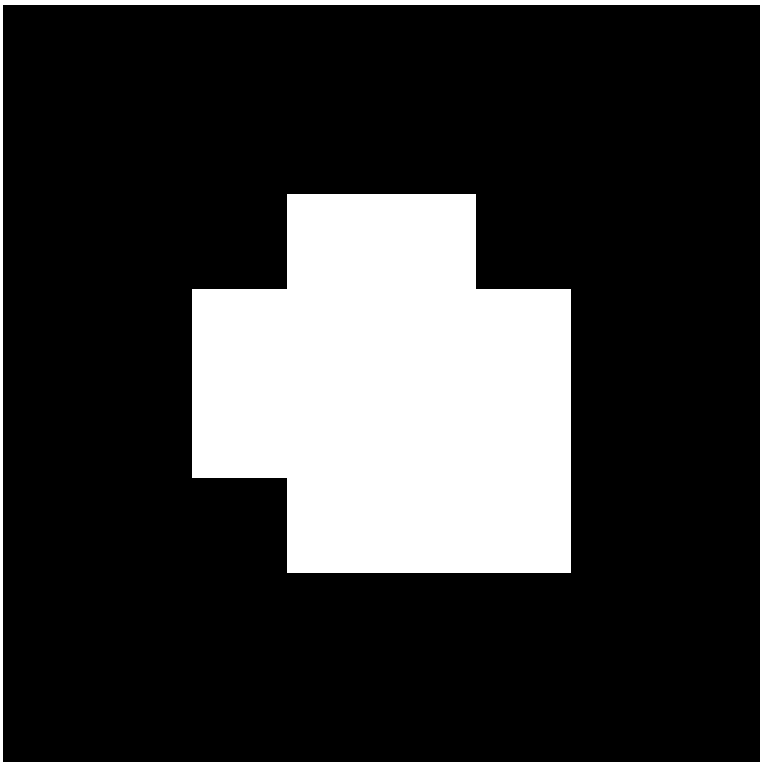

Figure 1: Figure 1: Reference AR Tag to be detected and tracked

Encoding Scheme

In order to properly use the tag, it is necessary to understand how the data is encoded in the tag. Consider the marker shown in Fig. 2

The tag can be decomposed into an 8 8 grid of squares, which includes a padding of 2 squares width (outer black cells in the image) along the borders. This allows easy detection of the tag when placed on any contrasting background. The inner 4 4 grid (i.e. after removing the padding) has the orientation depicted by a white square in the lower-right corner. This represents the upright position of the tag.

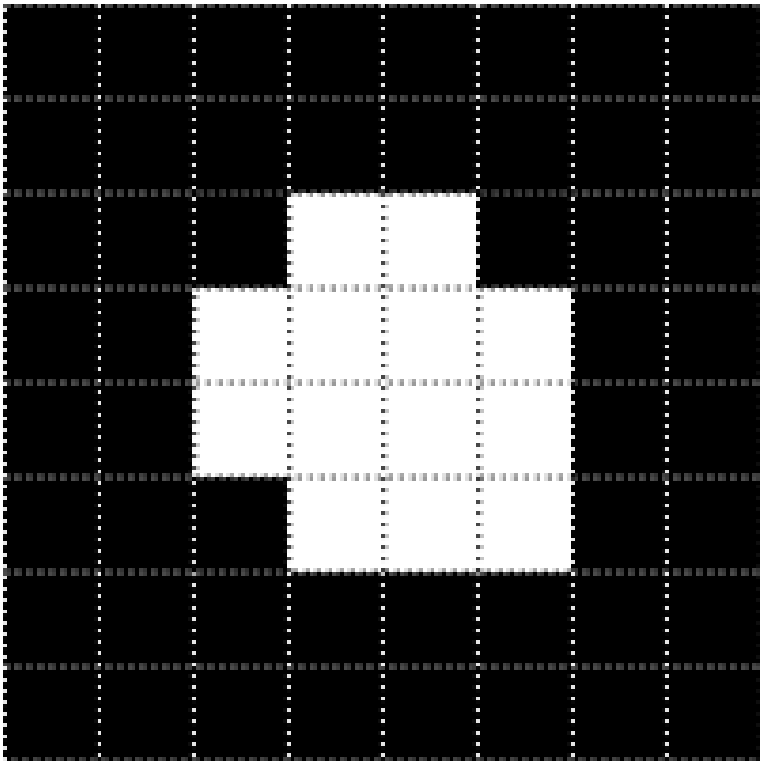

Figure 2: Figure 2: Grid pattern overlayed on reference AR Tag

2.1 Approach

In order to extract the tag, the first step was to detect its four corners and orient it

The steps followed for detecting the corners as follows:

Step 1: The algorithm takes in one frame at a time from the the given tag videos.

Step 2: Once the frame is considered, we use the cv2.cvtColor function in the OpenCV library to convert the frame to grayscale. The next procedure was to turn the grayscale frame into binary format.

Step 3: After the previous processes, the contours are determined from the binary image. The contours are selected by determining the pixels which has no parent contour but should contain minimum two child contours.

Step 4: Therefore, the results from the previous step helps extract the white sheets containing the tags from the video, ignoring the background and white objects.

Step 5: The tag is recognized by selecting the child contours of that parent contour which lies under a certain perimeter.

Step 6: Using the approxPolyDP function, the contour points are approximated by tuning the Epsilon which reduces the contour points to just four points marking it the corners of the tag.

After detecting the tag, the desired orientation of the tag is obtained by following steps:

Step 1: The homography matrix is used to change the perspective warp of an image. Therefore the homography matrix is determined between the tag corners and the planar upright position calculated of the tag contours from the previous process.

Step 2: A warp function is defined which takes in the matrix as one parameter and the frame as the next parameter. The warping of the image leads to generation of voids in the image which is fixed by filling up the void. The output of the warp function is an upright image.

Step 3: The resulting upright image is then converted to a grayscale image. Thresholding and Blurring is applied to retain the characteristics of the image but also smoothen out the borders.

Step 4: Again, the contours of the resultant image is determined and by approximation, the four required corner points of the region is found which contains the tag.

Step 5: Distance between each corner point and the corresponding upright image is found. The shortest distance returns the corner of the white rectangle which specifies the orientation.

Step 6: Finally the tag is rotated to change the perspective and keep the tag upright.

Step 7: The tag ID is detected by taking in the binary format of the upright tag and calculating the four corner points where an image is to be encoded.

Step 8: The four points help in determining the center pixel's intensity value for each four squares and converts them into binary values.

Step 9: These binary values are then sorted from MSB to LSB which is converted to a decimal value to give an ID.

Step 10: The ID is then printed on the video.

3 Problem 2(a) - Superimposing an image onto the tag

Question 2 *Superimposing an image onto the tag*

Once you have the four corners of the tag, you can perform homography estimation on this in order to perform some image processing operations, such as superimposing an image over the tag. The image you will use is the Lena.png file provided, see Fig. 3. Let us call this the template image.

The first step is to compute the homography between the corners of the template and the four corners of the tag. You will then transform the template image onto the tag, such that the tag is "replaced" with the template.

It is implied that the orientation of the transfered template image must match that of the tag at any given frame. Note: For transforming the image using homography you have to develop your own version of

"cv2.warpPerspecive" function. In doing so you will observe holes in your output image which depict loss of information.

Figure 3: Figure 3: Lena.png image used as template

3.1 Approach

Steps to superimpose Lena image on Tag

Step 1: After we have calculated the Tag orientation, we calculate the homography matrix between the corners of the tag detected and the Lena image gievn to us for reference.

Step 2: Then this homography matrix is passed to the warped perspective function made by us.

This warp perspective function places the Lena image in a black background of the size of the frame.

Step 3: The transformed Lena image which we have obtained in the previous step is used to create a mask by threshholding it.

Step 4: The next step is to inverse the mask and then take bitwise_and of the input frame and the inverted mask.

Step 5: We have further taken bitwise_and of transformed Lena image and the mask.

Step 6: Then we add both the image obtained in step 4 and step 5 and obtain an image with AR tag replaced with the Lena image.

4 Problem 2(b) - Placing a virtual cube on the tag

Question 3 *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 tag. This is the process of "projecting" a 3D shape onto a 2D image. The "cube" is a simple structure made up of 8 points and lines joining them. There is no need for a complex shape with planes and shading. However, feel free to experiment.*

You will first comupte the homography between the world coordinates(the reference AR tag) and the image plane (the tag in the image sequence). You will then build the projection matrix from the camera calibration matrix provided and the homography matrix. Assuming that the virtual cube is sitting on the top of the marker, and that the Z axis is negative in the upwards direction, you will be able to obtain the coordinates of the other four corners of the cube. This allows you to now transform all the corners of the cube onto the image plane using the projection matrix.

Please refer to the supplementary document on homography to understand how to compute the projection matrix from the homograhy matrix.

4.1 Approach

Superimposing Virtual cube on Tag

Step 1: The first step is to calculate Projection Matrix.

Step 2: To calculate the projection matrix we need the homography matrix between the four points of the reference AR tag and the corners of the tag detected in the previous steps.

Step 3: Using this homography matrix and the camera intrinsic parameters we calculate the first two columns of the Rotation matrix and the translation vector.

Step 4: The cross product of the first two columns of the rotation matrix gives us the third column of the rotation matrix.

Step 5: Using the rotation matrix and the translation vector we form the projection matrix. The projection matrix is given by $P = [K] * [R|t]$ where K is the camera intrinsic parameters, R is the rotation matrix and t is the translation vector.

Step 6: The next step is to draw a cube.

Step 7: To draw the cube first we calculate the four corner points of the top plane of the cube. These points are multiplied by the projection matrix.

Step 8: The four points which we get are the four corner points on the image plane.

Step 9: We then make a contour of the top four and the bottom four points and draw lines between them.

Step 10: Then we place the virtual cube formed in the above step in place of the tag.

5 Output

5.1 Superimposing Lena Image on the Tag

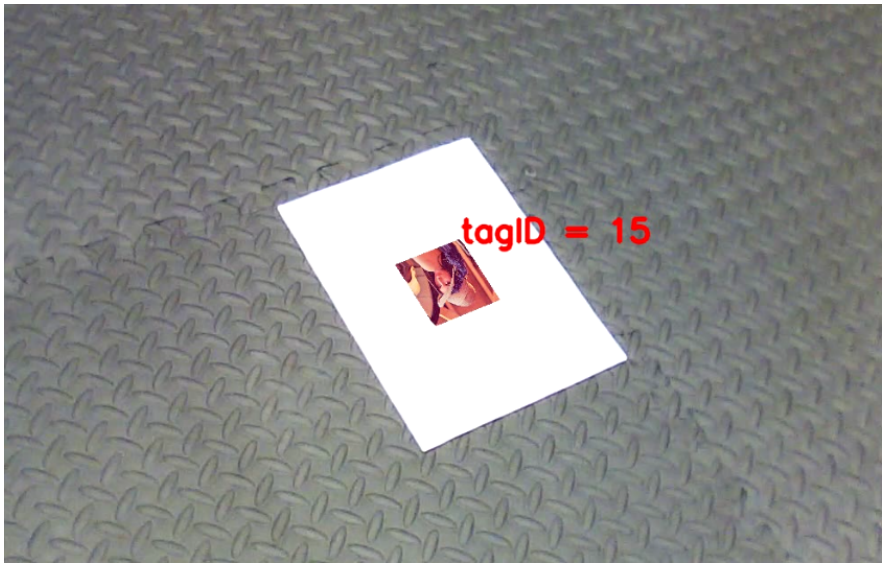

Figure 4: Lena Image on Tag0

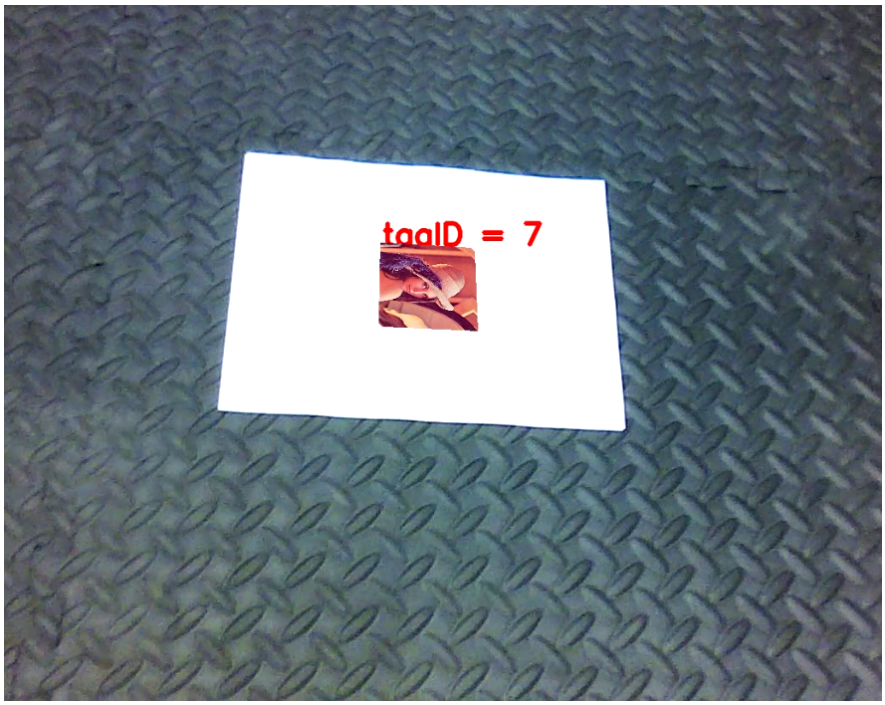

Figure 5: Lena Image on Tag1

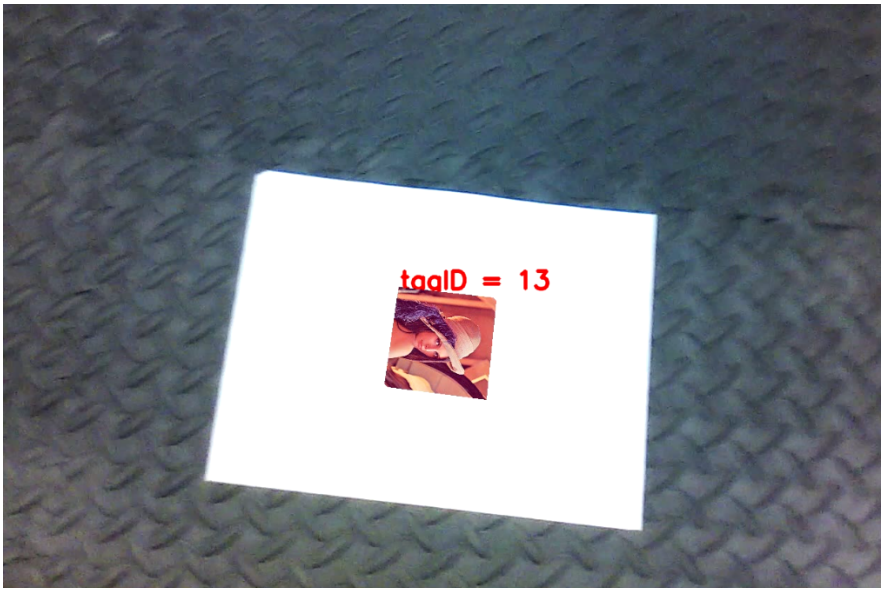

Figure 6: Lena Image on Tag2

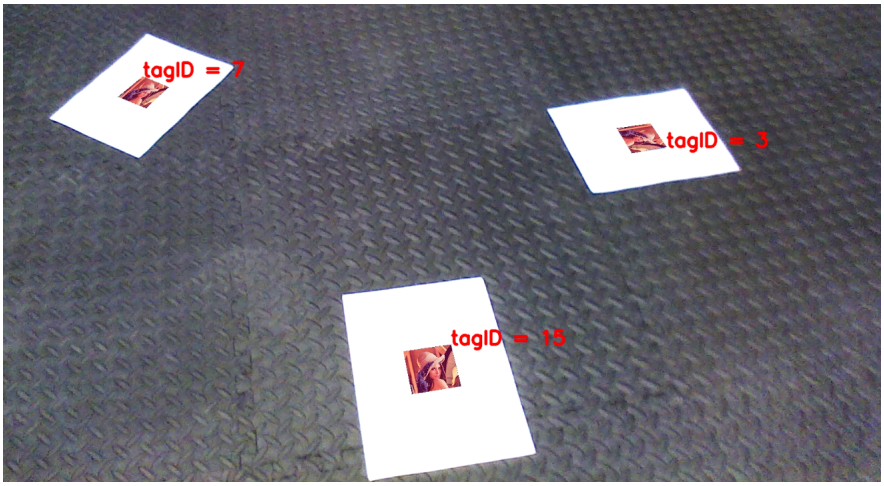

Figure 7: Lena Image on multipleTags

5.2 Placing a Virtual Cube on Tag

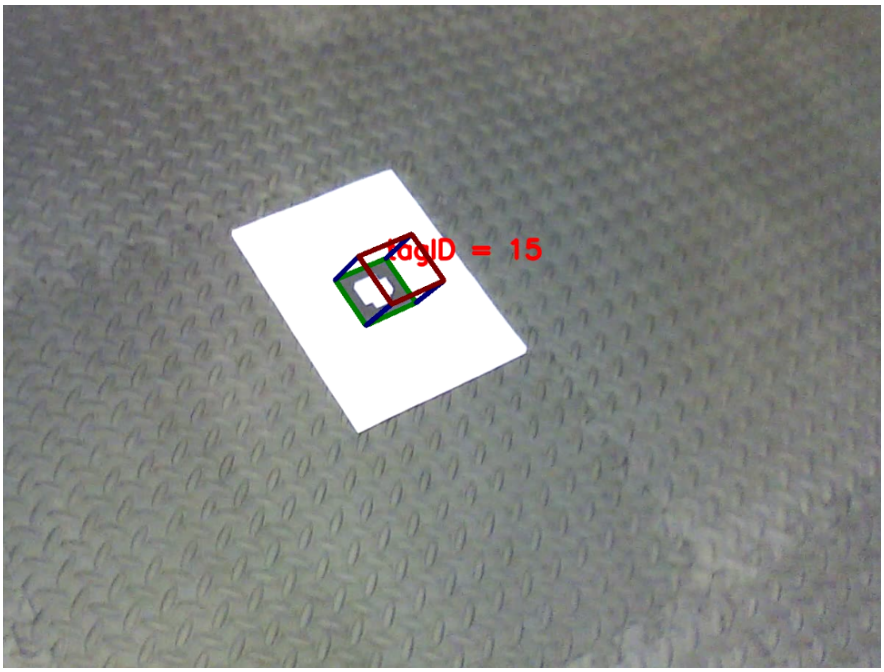

Figure 8: Virtual Cube on Tag0

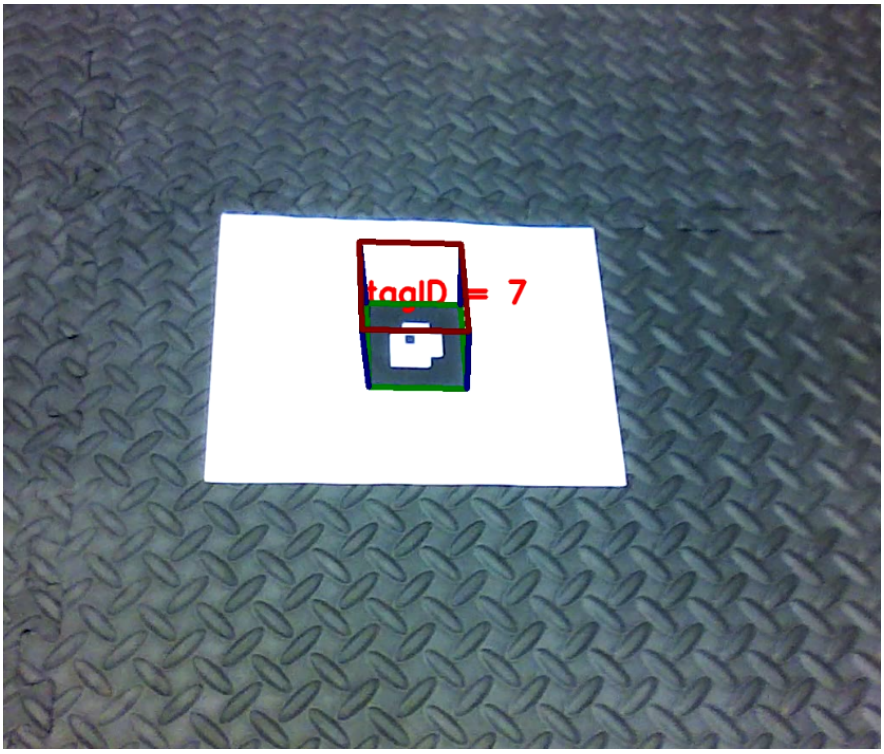

Figure 9: Virtual Cube on Tag1

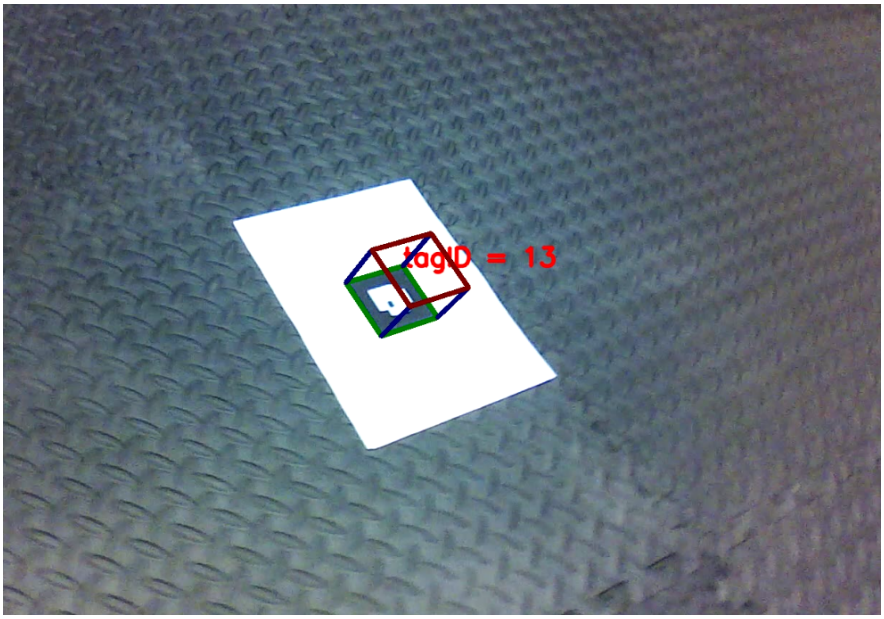

Figure 10: Virtual Cube on Tag2

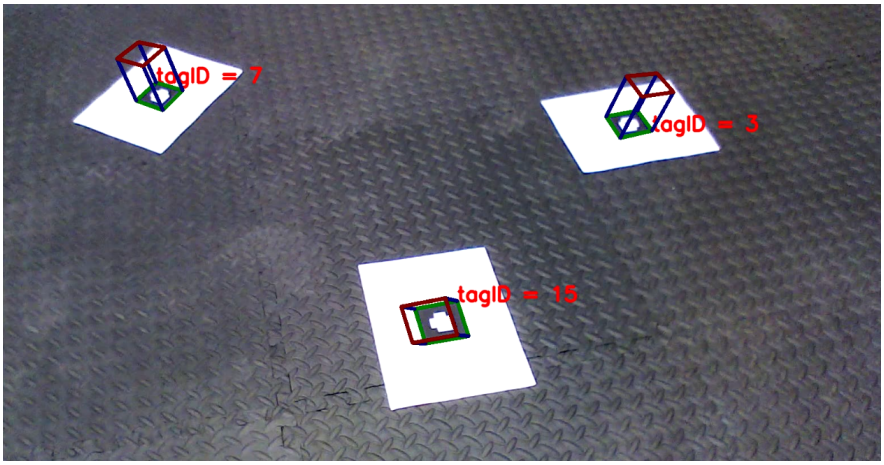

Figure 11: Virtual Cube on multipleTags

5.3 Drive link with OUTPUT Videos

`https://drive.google.com/drive/folders/19VEktSxPgUtj-co-xke_uFBXf5aZQbbe?usp=sharing`