ENPM673 PERCEPTION FOR AUTONOMOUS ROBOTS

PROJECT 1

AR Tag Detection and Tracking

Aditya Khopkar (116911627)
Nalin Das (116698290)
Nidhi Bhojak (116787529)

Contents

List of Figures

1 Introduction

The purpose of this project is to understand the concepts pertaining to homography and projective geometry. There are multiple input video files. The problem statement of the project defines that a template image and a virtual 3D cube needs to be superimposed onto the original video. This involves first detecting the tag from the video and subsequently tracking the video through each frame. The detection and tracking should run for all the video files. The Project has been divided into three parts:

1. Detection of AR Tags and ID
2. Overlaying the given template image on the AR Tags
3. Placing a 3D Cube over the AR Tags

2 Part 1: Detection and Tracking

The main aim of this part of the project is to detect and decode the given AR Tag. A video is

Figure 1: AR Tag

a sequence of image and thus, a video may be processed similar to any image by applying similar image processing functions. In-order to detect the AR tag properly there are certain pre-processes being applied beforehand. The order we followed is mentioned below:

- Convert the input file to grayscale so that it becomes easier to process.
- Perform Binary thresholding to retrive the image in a binary format. This step will make it easier to define contours.
- Detecting the AR Tag using the findContour Function.

2.1 Finding Contours and Detecting Corners

Contours are the regions that are formed around the edges in the image. Firstly, the frame is being converted to grayscale. After that, binary thresholding is applied in order to convert the frame into a binary frame (All blacks and All whites)to seperate the tags from the background.

The findContour function is used to mark out the contours in the frame. The function returns a list of all the contours in the frame and hierarchy of the contours. While there are multiple parameters to this function which have different return schemes, we are specifically interested in the argument cv2.RETR_TREE which returns all the hierarchy of the contours. We eliminate these contours by understanding the working of the hierarchy. We do this using approxPolyDP and arcLength functions. These functions help us to find the perimeter of the contour and subsequently its corner points based on a tuning parameter of 'epsilon', which is given by epsilon = 0.02*perimeter. This suggests how closely the closed contour points need to be bounded. We look for those contours which have number of edges = 4 and they have no parents in the hierarchy list, which effectively

means, the contour is within a contour, in this case, the page in the frame. To detect the correct corners, we imply another criteria of area. This eliminates all other contours from the list.

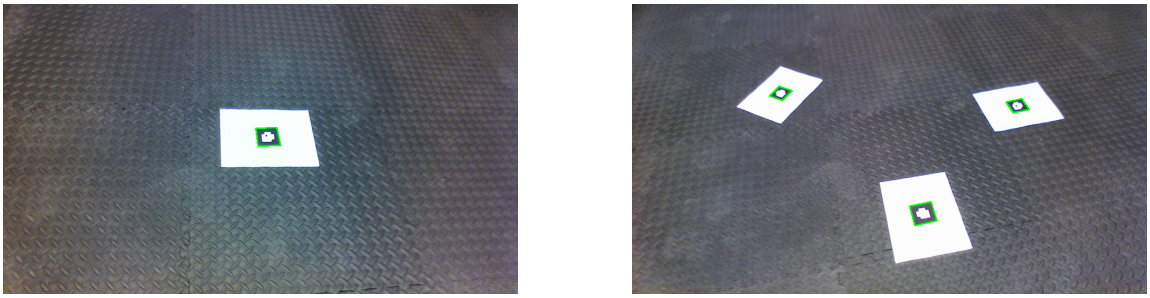

Figure 2: Detecting Tags

2.2 Finding Homography

Homography relates any two images of same planar surface in space. Given two projective spaces P(V) and P(W) of the same dimension, homography is the mapping from P(V) to P(W) which is induced by isomorphism of vector spaces. The working of the Homography function can be divided into following steps:

- The function takes the corner points of the source and destination image as arguments.
- Initialising "A" matrix of dimensions 8×9 with zeros.
- Constructing the A matrix using all the points
- Computing the Singular Valued Decomposition of the A matrix using the inbuilt function which gives us the parameters U, S ,Vh.
- Normalising the Vh matrix to get the 3×3 homography matrix.

The camera frame co-ordinates can be obtained by multiplying the world frame co-ordinates with Homography matrix.

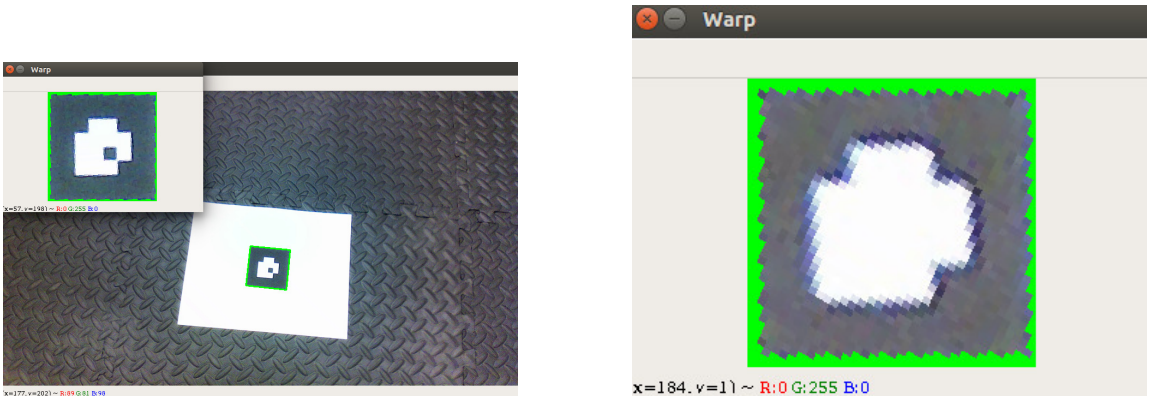

Figure 3: Warped AR Tag

2.3 Decoding Tag ID and Orientation

To decode the Tag ID and orientation properly, we need the bird's eye view of the tag itself. To implement that we use the concepts of homography. After that, the corresponding set of points in the source image can be mapped to the destination image using the 'homography' function.

To decode the ID, we convert the detected tag into grayscale and apply bilateral filter to filter the tag from the noise. We take the region of interest of the tag and convert it into a 4×4 matrix of binary. Black block is detected as 0 and White block is detected as 1. The inner 2×2 matrix can be decoded based on its corresponding Most Significant Bit and Least Significant Bit. The resultant is the decimal value of the TagID.

If there is a white tile in the bottom right corner, then the orientation of the tag is bottom right. Similarly the same applies for all the four corners of the 4x4 grid in the AR Tag.

The orientation of the AR Tag is important to realise the rotation of the AR Tag and subsequently map the Lena Image on to the AR Tag with the correct orientation.

3 Part 2 :Overlaying the Template Image on AR Tag

The aim here is to superimpose the template image onto the AR Tag. Firstly, the image is resized to dimensions 200×200 to match the world co-ordinates.
To overlay the "Lena" image on the AR Tag, again the homography matrix is computed to transform the co-ordinates from the world frame to camera frame. After finding the relation between the camera frame and the world frame, the pixels from the camera frame are simply replaced by the pixels from the world frame.
Also, to match the orientation in the transformed image, we have used the orientation bit of the AR Tag as previously mention in section 2.4.
The coding scheme works as, when the frame rotates, the rotated orientation bit is mapped to a particular corner. And accordingly the template image is reoriented to that position.

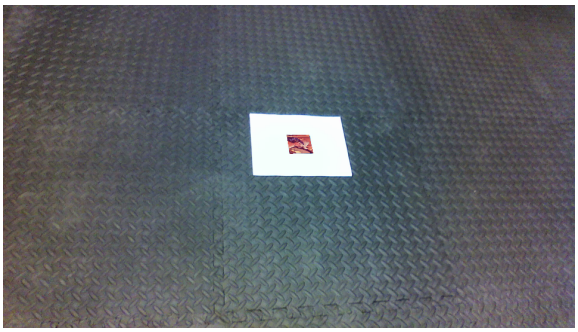

Figure 4: Superimposing Lena Image

4 Part 3: Placing a Virtual Cube on the AR Tag

For placing the cube on the tag, we first define the cube co-ordinates in the world co-ordinate frame and then calculate the projection matrix which comprises of Rotation and Translation. Below steps are to be followed inorder to project the cube:

- Initially ,obtain a matrix by multiplying the camera intrinsic properties and the homography matrix, column vise.
- Then, we find the Scale factor λ which is used for the scaling of the transformation matrix.
- The projection matrix is composed of the respective rotations and translation which are in-turn proportional to the B matrix.
- Comparing B matrix column vise, we get our rotation 1, rotation 2 and translation.
- Defining the co-ordinates in the world frame where the cube needs to projected. This forms the vertices of the cube.
- Multiplying each vertex by the Projection matrix will give us new co-ordinates in the camera frame.
- Normalizing the co-ordinates
- Using cv2.circle() and cv2.drawContour() to draw the cube.

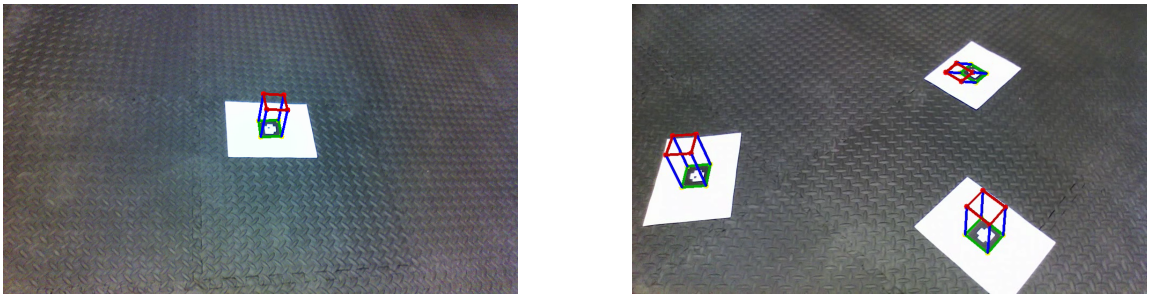

Figure 5: Projecting Virtual Cube

5 Conclusion

The output video files can be accessed from the link provided in the Readme.md file. The project helped us learn the concepts of Image Processing and Projective Geometry. We learned how homography relates to how one perceives the image from a frame. There were certain challenges faced while carrying out the project whic are mentioned below:

- Scrapping through all the contour points and hierarchies to detect the encoded portion of the AR Tag
- Creating a function similar to the warpPerspective() of openCV which transforms the image based on homograpghy
- After warping the image, information loss was observed which was corrected by multiplying the pixels with inverse of Homography matrix.