ENPM 673 - Project 1

Chandrakanth bodapati

Abstract

Detecting a custom AR Tag (a form of fiducial marker) is done in this project and this is used in augmented reality applications. There are two components in using an AR Tag, namely detection and tracking. Both these techniques are used in this project The detection stage will involve finding the location and identity of AR Tag from a given image sequence while the tracking stage will involve keeping the tag in "view" which involved superimposing an image and placing a 3D cube over the tag through- out the sequence and performing image processing operations based on the tag's orientation and position (a.k.a. the pose).

Keywords · AR Tag · Image Processing · Detection · Tracking

AR Code Detection

The boundary of any object is formed by a sharp contrast, and this occurs right at the edge. This is true for any image. In an Image, an edge is the place where sharp contrast occurs. This facilitates us to detect the boundary and thereby the any object in the image. It is a known fact that to perform edge detection, we need to take the first derivative (Discrete Derivatives since we are dealing with pixels) along both the axis to detect the places of high contrast in an image along both the axis. Convolution is done over the entire image by various type of operators like Sobel,Prewitt. we can also use Canny Edge algorithm which involves non-maximal suppression thinning. All these algorithms involve high computations as we need to convolve masks like sobel over an entire image. Since the computations are high and exhaustive, there is a need to reduce the number of computations by using the property of linearity. We can also use the Fourier Transforms to detect edges. Any Periodic signal can be expressed in terms of Sines and Cosines which is the essence of Fourier Transforms. So, by assuming our whole image as repeating periodic sequence we can express our image in terms of a Sine-Cosine wave form. This property allows us to reduce the computations to perform the edge detection, As multiplication in frequency domain is equivalent to Convolution in Spatial Domain. So by use of the Scipy functions i converted the image to the Frequency Domain. Later, i shifted all the frequencies by placing the center at zero. Now, There are far less edges in the image as compared to objects thus all the edges form a high frequency components of a signal. But Noise in the image will also be part of this high frequency signals so to reduce the noise, i multiplied a Gaussian kernel to blur the image thus reducing the noise. An important property of Fourier Signal is that we can reconstruct the image if we know the amplitude, frequency, and phase of the signal. So, Later converted the signal by using Inverse FFT to give a Blurry image. Now, Threshold-ed the image to give a Binary image and converted the image to a Fourier Signal. Now, the high frequency components of this image form the edges of the image as there are far too less edges than objects in the image and less noise. So by multiplication with a circular mask and by using the high pass signals we get the Fourier Signal of an Edge Image. How good an edge is determined by the radius of the circular mask but there will be trade off between the number of edges detected vs number of edges in the image. Later, converted this signal back to the spatial domain.

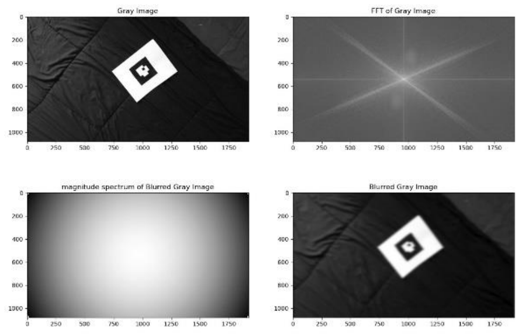

Figure 1: Image Blur using FFT

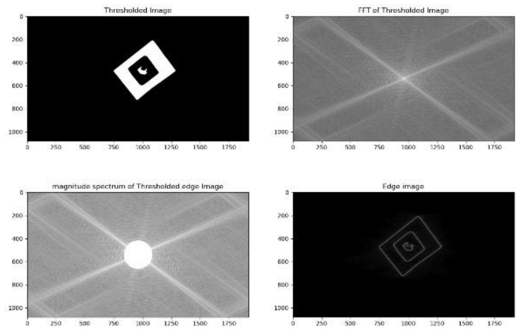

Figure 2: Image Edges using FFT

Decode custom AR tag

The tag has been resized into a 160 x 160 grid. Now, as mentioned i have converted the 160 x 160 grid to an 8 x 8 grid so that each grid contains 20 pixels. Then i calculated mean of each grid and if the mean of the grid > 255/2, then i set the value of the grid to 255 otherwise it is set to 0. now, I extracted the inner 4 x 4 grid where the Rotational and ID value is encoded. Now, if in a video frame if the tag corners are found rightly we would have only one high value at the corners of the 4x4 matrix. Now, the 4 x 4 grid is rotated until the bottom right corner is high i.e 255 which represents the upright position of the AR tag. Then, the 2 x 2 grid is Extracted in which the LSB to MSB are ordered clockwise from the left corner of 2 x 2 Grid. The ID of the reference tag is found to be 15.

Corner Detection Corners are the points in the image where there is a change in both x and y directions. This points can be used to detect the objects where there is change in scale,rotations etc by various methods such as Harris corners and Shi-Thomasi version of GoodFeatures to Track. I have used both of these to detect corners and settled on Harris corners algorithm given the video.Shi-Thomasi version of corner detectors considers
the minimum eigen value of the auto correlation matrix where the direction of change is rapid and computes maxima in the direction of smaller eigen value. As it considers only the one eigen value its shape may change with respect to rotation. But, the features provided will be robust for in the case of tracking where we can use Lucas Kanade tracking algorithm which involves Optical flow which assumes all the pixels in the image undergo same motion in the image so that we can track the detected features. I tried to implement the algorithm but i couldn't get good results with the algorithm implemented, if i had some more time i might have got good results. The Shi-Thomasi corners involved 3 parameters the clarity of the corner, the Euclidean distance between the corners and number of corners. Tried to tune these values and found the best to 10 corners of 0.1 clarity with minimum of 100 Euclidean distance between the coordinates.
The Harris Corner Detector algorithm on the other hand considers both the eigen values of the auto correlation matrix which makes it insensitive to change in rotation. As our video has predominantly rotational motion i choose to continue with the Harris Corner Detectors. But, the initial few frames couldn't detect the corners properly and some of the frames through out the video does not detect the corners properly. The Harris corners involved the tuning parameter alpha, sobel operator kernel size and neighbourhood window size. Tried to tune these values to detect good corners. I performed the morphological operations to the gray image to remove noise and detect the rectangles properly but i found to miss some initial frames and where there is a change in rotation speed predominantly. found the Harris corners are also robust to illumination changes in the image. All in all, with more fine tuning the result can be improved over time.

Decoding AR tag from the Video

- For every frame in the video, the corners are found using the Harris Corner Detector.
- Once the corners are found, we use the rectangle properties and we detect the corners of the AR tag by popping out the Xmin,Xmax,Ymin,Ymax of the white sheet.
- Sort the corners in anti clockwise starting from top left of rectangle.
- Compute the Homography between the image corner coordinates and the desired 160 x 160 coordinates. Now after computing the coordinates, we have two ways to retrieve the information for the upright image to be shown.
- The Homography computed is a 3 x 3 matrix, so we are doing a Perspective Transform which
has 8 degrees of freedom. A quadrilateral can be transformed to a different plane allowing it to distort the quadrilateral.
- After computing the coordinates, we have two ways to get the intensity values of our upright image.
- One method is to find for every pixel intensity in the image plane coordinates where does these pixel values maps to the desired image. This is called Forward warping. This causes holes in the desired upright image because the pixels in image plane may be mapped to in between pixels in the desired coordinates.
- There is another method, which is
called inverse warping. Here, we say, for every pixel coordinate in the desired image figure out where it came from in the image plane. In order to do this we multiply with the inverse of the homography matrix.
- There are methods like Bilinear interpolation and cubic interpolation which can plug in the holes. Bilinear Interpolation is used to superimpose Testudo. A simple inverse warping is used for AR tag Detection. This method produced good results as most of
the holes are plugged and we get a complete binary image while decoding the tag in the video when we detect the corners accurately. Later, we flip the AR tag by 90 degrees until the right most corner is found to be 255. If the corners are not detected properly after four rotations the rotation is stopped and we get an incorrect value and orientation of the AR tag.

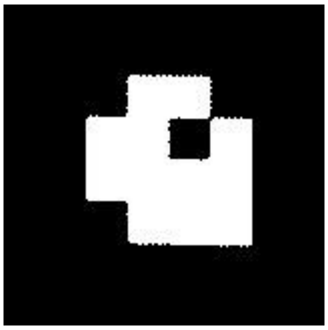

Figure 3: AR tag decoded in one of the frames

Superimposing Testudo onto the AR tag

Let us see how we can superimpose the Testudo. It is similar in detecting the AR code tag in with respect to the pipeline.

- Detect the corners of the AR tag
- Compute the orientation of the tag
- Sort the corners according to the orientation and then perform homography between the corners of Testudo image and the AR tag.
- Compute the image plane coordinates for the testudo image to be superimposed.
- Perform inverse warping from the image plane to the testudo
- Do a bi-linear interpolation between pixels and calculate the intensity value at the corresponding image plane coordinates.

Problems encountered:

The detected corners were found to be out of bounds of the image plane in some frames. The square is also found to be distorted in some owing to the lack of proper corner detection. To rectify this, several attempts had been made to adjust previous orientation and corner values but to no avail.

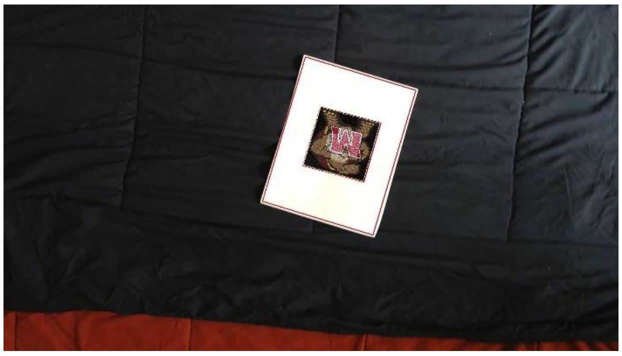

Figure 4: Testudo superimposed on AR tag with orientation

Superimposing a virtual Cube onto tag

To project a 3D object onto the camera plane, we need 3d coordinates. So in order to obtain the co-ordinates we need to compute Projection Matrix P. where P = K[R|t]. where K is the intrinsic camera matrix. Consider the rows of the homography matrix as h1,h2,h3. Now, $\lambda H = K \; B$ where $B = [r_1, r_2, t]$.

Now the scale λ is given by $\lambda = \left(\frac{\|K^{-1} h1\| + \|K^{-1} h1\|}{2} \right)^{-1}$

Next step is to compute, $B = \lambda K^{-1} H$.

later, $B = B(-1)^{|B|<0}$.

if the det is less than 0, we multiply in front of the camera. We need the solution in front of the camera. Here, $r_1 = b_1$, $r_2 = b_2$, $r_3 = r_1 x r_2$, $t = b_3$

The above is the code written for the projection matrix as it is.

Now, to draw the cube define the coordinates of cube in reference frame and then multiply with the projection matrix to get the coordinates of the cube in the image plane. Now, we draw lines to connect these coordinates to get the desired result of cube on the AR tag.

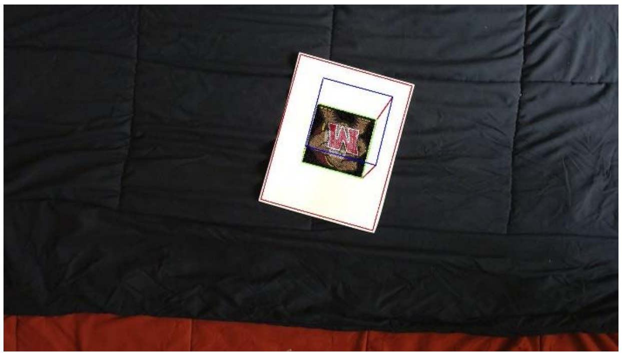

Figure 5: Testudo superimposed on AR tag with orientatio