Assignment 3: Projection and Homography

Daniel Sawyer
University of South Florida
danielsawyer@mail.usf.edu

Abstract

Using projection methods along with calculating homography using iterative algorithm least squares in order to compute the homography matrix. From there, we can easily project a flat image onto a perspective skewed space inside another image. For example, we can replace a billboard image with something else.

1. Algorithm

I will be referring to the two images as scene image and input image. We project the input image within the polygon (usually the whole image but not always) into the scene image's polygon. The polygon is created by choosing corresponding points in both images matched in a clockwise fashion. The algorithm is broken into three parts. The first part is finding all the points that are inside the polygons from both images. This is an iteritave algorithm that iterates through all the points clockwise back to the origin point. I found the algorithm on GeeksForGeeks[1] website and made some modifications to it in order to search the whole image at once and return a list of points (x, y) that are within the polygon. Next, I calculate the homography matrix H. This is the majority of the assignment. What we have to do here is calculate the error between estimated point and the actual point which gives us a delta P vector that we continuously add to the P vector as we iterate through all the matching points. We solve for delta P each iteration, an iteration is going through all matching points once while summing up A matrix and B vector. A matrix is an 8x8 matrix while B and P are 8x1 column vectors. We get Ai by calculating the Jacobian, J which is 2x8, for the set of matching points and dividing it by D. Ai then equals Transpose(J)(J). This gives us an 8x8 matrix which we then add to A sum. Bi is the error between estimated point, ri, and the Transpose(J). We then add this to the running sum of B. After getting the sum of A and B from all matching points, we then solve for P by using least squares algorithm for (A + d*Diagonal(A))P = B where d is a dampending factor from [0, 1]. Using least squares, we get delta P. We iterate through this constantly changing P until there is no change to delta P and the points start to converge. Once the values have converged, we return H which can be used to map one set of points to the other. The last part of the algorithm is maping the points. This is the easy part. From here, we just take a scene point from within the polygon and calculate the point to pull from the input image by creating a homogeneous point vector Xs = [xs, ys, 1] where (xs, ys) are scene points. We do matrix multiplication H(Xs) and get a 3x1 vector Xi which can be used to calculate input points (xi, yi) where xi = Xi[0]/Xi[2] and yi = Xi[1]/Xi[2] and then replace SceneImage[ys][xs] with InputImage[yi][xi]. After doing this for all pixels inside the scene polygon, we save the resulting image which is the input image projected onto the scene image's outlined polygon.

2. Related Work

I used the information obtained in the class and book. I also used a find if point is inside polygon algorithm from GeeksForGeeks[1] website online.

3. Examples

Here are some examples of input and scene images with the input projected onto the scene.

Figure 1. Pepsi Truck With Coke Sign

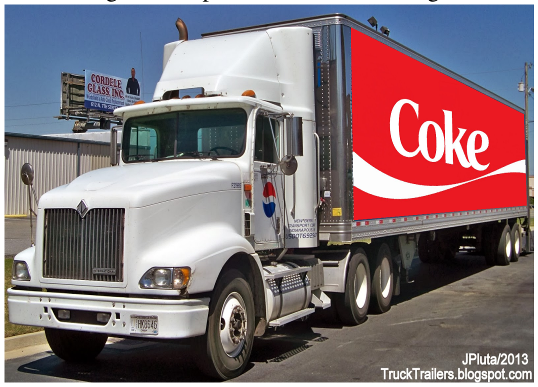


Figure 2. Coke On Stop Sign

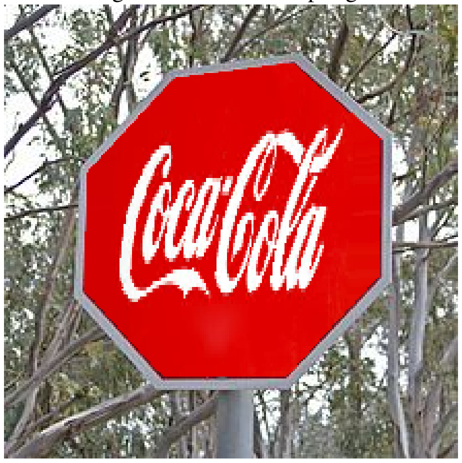


Figure 3. Pocahontas On Stop Sign

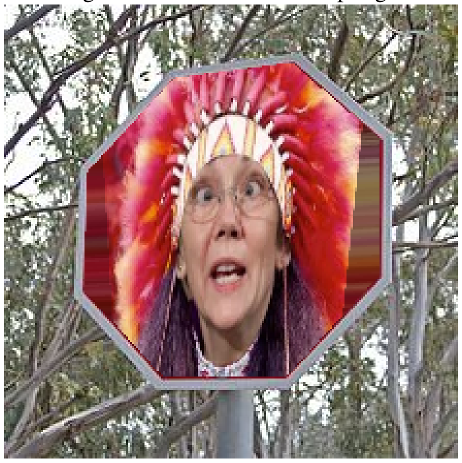
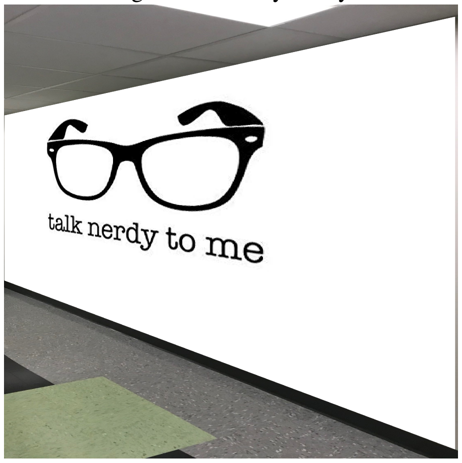


Figure 4. Hallway Nerdy



4. Lessons Learned

I learned how you can transform a flat ”2D” image into a perspective but still flat ”3D” image. Calculating homography was also very useful and can see were this can be used in my research. Projecting images onto another or using it to calculate distances can also be very useful.

References

[1] Point inside polygon. 2019.