**Q1 – Image Stitching**

1. In order to match between two images key-points we will use SIFT descriptors to find which two key points correspond in both images, using the provided SIFT scripts we found the descriptors for each given image in the "Stop Images" directory.
2. After getting the sift descriptors, and finding the matching key-points using the "siftmatch" function on the pairs (im1, im2), (im1, im3), (im1, im4), the results are:

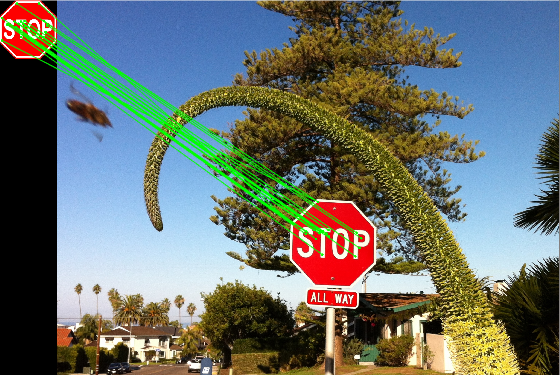


Figure 3: StopSign1-StopSign4

Figure 1: StopSign1-StopSign3

Figure 1: StopSign1-StopSign2

As you can see there are a few wrong matches that occurred, the most obvious of them is in Figure1, where there is a match between the left stop sign to the street sign.

1. The general affine matrix is: .

The key-points from the left image will be: , and from the right image will be: . (both homogenous).

As we can see we have six variables in matrix , and from every pair of matching key-points we get two equations, so in order to be able to solve it for , we need at least three key-points.

We will solve for using least squares method:

After finding the variables vector, we reshape it to be a matrix of 3x3 where the bottom row of the matrix is:

1. Using the RANSAC algorithm we will find the best affine transformation between the images, we will say that a point is an inlier if the projection error is smaller than 5 using L2 metric, the results are:



Figure 4: StopSign1-StopSign2, RANSAC



Figure 5: StopSign1-StopSign3, RANSAC



Figure 6: StopSign1-StopSign4, RANSAC

We can see that the outliers that were present earlier are now removed, and we are left with key points that correspond only to the stop sign.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **StopSign 1-2** | **StopSign 1-3** | **StopSign 1-4** |
| Number of Inliers | 17 | 18 | 15 |
| Affine Transformation |  |  |  |

1. We want to implement a warping function that will take the sign image and the affine transformation calculated earlier between the stop sign and the scene image with a stop sign. The problem when applying the transformation is that the pixels indexes are integers and after the transformation the indexes aren't necessarily integers, so:

* we will apply the transformation and find the size of the projected image.
* Find from where each pixel in the projected image comes in the original image (values aren't integers), by finding the inverse transformation.
* Interpolate (bilinear interpolation) the value of inversed pixels and assign it to the pixels in the projected image.





Figure 8: StopSign1-StopSign3, Warped

Figure 9: StopSign1-StopSign4, Warped

Figure 7: StopSign1-StopSign2, Warped

We can see in the results that the stop sign warped according to the orientation of the stop sign in the scene and translated to its location according to the top left corner.

1. We want to implement a stitching function that will take the warped sign image, the scene image and overlay the images so the warped stop sign will appear where the stop sign in the scene.



Figure 10: StopSign1-StopSign2, Stitched



Figure 11: StopSign1-StopSign3, Stitched



Figure 12: StopSign1-StopSign4, Stitched

We see that in figures 10,11 the results are good except for some artifacts that were added because in the overlay process we show from each color channel of the warped image only values that are different from zero, so we get the artifacts.

Also the reason that in Figure 12 the warped stop sign isn't aligned with the one from the scene image is that affine transformation breaks when there is big changes in the viewpoint and this is the case here, StopSign1 is captured straight and StopSign4 is captured from large different angle.

If we show from the warped image pixel values that are different from zero in **some channel** we get:



Figure 13: StopSign1-StopSign2, Stitched B



Figure 14: StopSign1-StopSign3, Stitched B



Figure 15: StopSign1-StopSign4, Stitched B

Here we see that the warped totally replaced the pixels in the scene image and we don't get the colored artifacts, instead we get black pixels.

(From the instructions this is what I thought needed to be done, but the example image in section 6 in the instructions shows otherwise).

**Full algorithm:**

We start with an image of only a stop sign and an image of a scene with a stop sign in it.



Figure 16: StopSign1



Figure 17: StopSign3

We find all sift descriptors for both images and match between them using the sift function given to us.

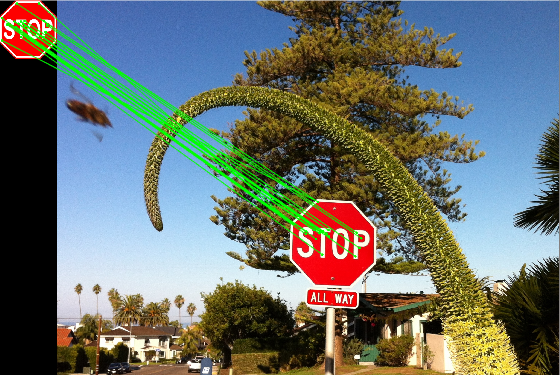


Figure 18: StopSign1-StopSign3

There are some outliers between the matches, in order to get rid of them and find the best transformation we run the RANSAC algorithm.



Figure 19: StopSign1-StopSign3, RANSAC

We are left only with the inliers and the best transformation H, now we perform the transformation we found for the left stop sign image.



Figure 20: StopSign1-StopSign3, Warped

The stop sign rotates, scales, translates, the image we get is bigger than before because of the translation to the location of the stop sign in the scene image. Now the images are aligned, (we pad the warped image to the size of the scene image).

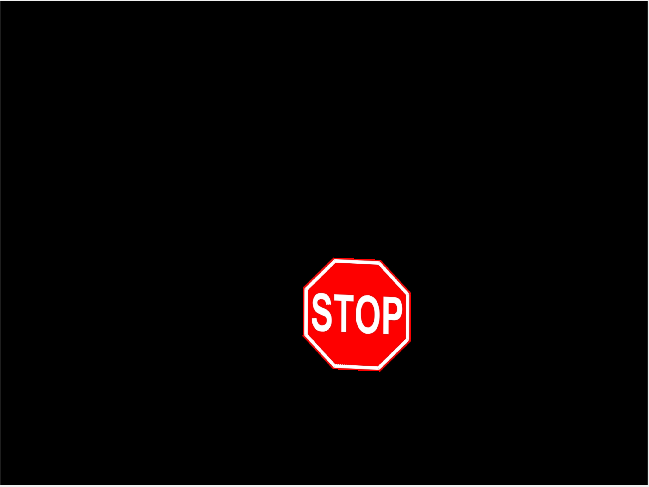


Figure 21: 20: StopSign1-StopSign3, padded

Now we overlay with the scene image by checking for each color channel if the value is different from zero then we change the value of the pixel in the corresponding channel in the scene image.



Figure 22: StopSign1-StopSign3, Overlay

The overlay result depends on the orientation of the camera when taking the scene image, for affine transformation to work we need small changes in camera angle between the two images and the result of the RANSAC algorithm.

**Q2 – Optical Flow**

1. In order to calculate Ix, Iy, It we spread all the frames into one continuous image in the size of [height, width\*number of frames], this makes it easy to calculate the derivatives for all the frames. For the right edge pixels, we will use the first column of the next frame pixels, for the bottom row we will use the first row of the same frame, for the time derivative of the last frame we will use the first frame.

For each region we check if the eigen values of are larger than zero, this is because in "Lucas-Kanade" we assume spatial coherence so if a region has only an edge in it becomes singular and if a region is homogeneous the derivatives in and are roughly zero so has eigen values that are close to zero and we can't track the direction of the flow.

If a region has texture in it so both eigen values will be large and the derivatives also which means we can calculate the flow of the region over frames.



Figure 23: Frame 1, N=16, T=1

We see that the valid regions are the ones with the cars this is because they have more texture. The direction of the arrows indicates the flow of the objects in the image for the next frame. The size of the arrows indicates the velocity of the region between frames. When viewing the video, we see that there is some salt and pepper noise in the image which can explain why regions on the road have high eigen values. We also see some arrows in the timer that has texture and it changes from frame to frame but doesn't have any true flow from frame to frame.



Figure 24: Frame1, N=16, T=0.1

Here we lowered the threshold T=0.1 for the minimum eigen value which means regions that have less texture are now valid, like those on the road and the tree, this translates to more arrows on the image. We can think about T as an indicator on how much texture needs to be in a region to make it valid.



Figure 25: Frame1, N=8, T=0.1

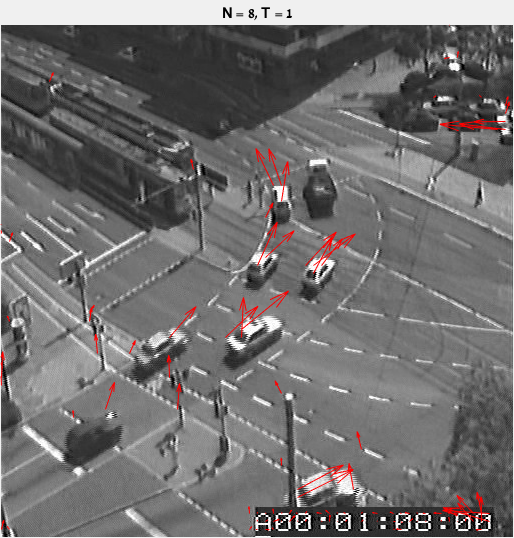


Figure 26: Frame1, N=8, T=1

Now we changed the size of the region we check to be 8x8, we see that for T=0.1 there are less valid regions than Figure 24 (N=16, T=0.1), this is because N decides the size of the region we look at, and for smaller regions there are less textures. We can see that the arrows point in the direction the car is headed in this frame, this is caused because of aperture ambiguity. N defines the aperture we look at.

For T=1 we still se strong responses but using a smaller region filtered out responses from regions that have more homogeneous textures and edges, this can be seen by observing that the location of arrows now is more at the center of cars and not in the area of the edges.

1. To summarize the previous sections:

T decides the validity of region by the amount of texture in it.

N decides the aperture, for larger N the direction of the velocity vector is more accurate.