Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

Computer Vision 236873 – HW3:

Q1 – Image Stitching

1) In order to match between two images, we used SIFT descriptors to find the key-points of the images and then find which two key points correspond in both images. Using the provided SIFT descriptors code, we found the descriptors for each given image in the "Stop Images" directory (after converting them to double and grey values representation).

2) After getting the sift descriptors, we found the matching key-points by using the "siftmatch" function on the pairs (im1, im2), (im1, im3), (im1, im4).
The results are:

Figure 1: StopSign1-StopSign2

Figure 1: StopSign1-StopSign3

Figure 3: StopSign1-StopSign4

As we 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.

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

3) The general affine matrix is: $H = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$.

The key-points from the left image will be annotated as: X, and from the right image will be annotated as: $\hat{X}$ (both homogenous).

$$\hat{X} = H * X = \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} => \underbrace{\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}}_{P} * \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}}_{h} = \underbrace{\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}}_{\tilde{P}}$$

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

We solved for h using least squares method:
$h = (P^T P)^{-1} P^T \tilde{P}$.
After finding the variables vector, we reshape it to be a matrix of 3x3 where the bottom row of the matrix is: $[0 \quad 0 \quad 1]$.

4) Using the RANSAC algorithm we found the affine transformation corresponding to the maximal number of inliers, we determined that a point is an inlier if the projection error was smaller than the threshold 5, where an error was calculated using L2 metric.

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

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 itself, which are the inliers.

The results:

	StopSign 1-2	**StopSign 1-3**	**StopSign 1-4**
Number of Inliers	17	18	15
Affine Transformation	0.9384 0.0569 321.2264 0.0114 1.2092 71.5203 0 0 1	1.4432 -0.0058 1.2108e+03 0.0700 1.4953 1.0185e+03 0 0 1	2.5988 -0.1929 157.7543 0.3407 2.3940 81.3857 0 0 1

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

5) We wanted 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).
   Indexes are integers, but after the transformation the indexes aren't necessarily integers, therefore the steps were:
   - 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.

   The results:

Figure 7: StopSign1-StopSign2, Warped

Figure 8: StopSign1-StopSign3, Warped

Figure 9: StopSign1-StopSign4, 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 bottom left corner.

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

6) We wanted to implement a stitching function that will take the warped sign image and the scene image as inputs, and overlay the images so that the warped stop sign will appear where the stop sign appeared 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 relatively good, except of some artifacts that were added since in the overlay process, from each color channel of the warped image, we showed only values that were different from zero, so we got 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 a big changes in the viewpoint and this is the case here, StopSign1 is captured straight and StopSign4 is captured from large different angle.
Particularly, StopSign4 has the projectivity behavior – it has convergence to a vanishing point. Therefore, probably a projective transformation would have better result.

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

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 sign totally replaced the pixels in the scene image and we didn't get the colored artifacts, instead we got black pixels.
(From the instructions this is what we thought needed to be done, but the example image in section 6 in the instructions shows otherwise. Therefore, we presented the results for both cases).

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

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 left only with the inliers when using the best transformation H, and now we perform this transformation for the left stop sign image (StopSign1).

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

Figure 20: StopSign1-StopSign3, Warped

The stop sign rotates, scales and translates, so that 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).

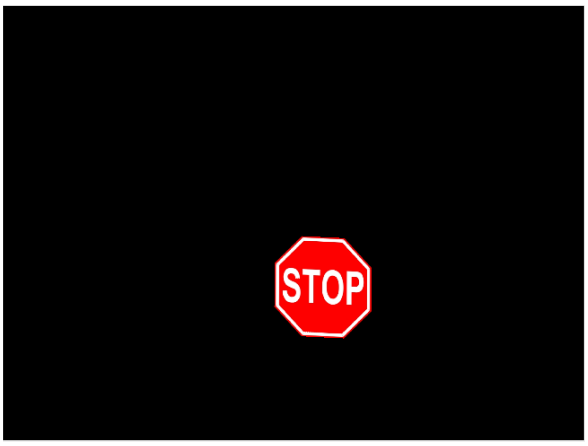

Figure 21: 20: StopSign1-StopSign3, padded

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 to be as it is in the StopSign1 image.

Figure 22: StopSign1-StopSign3, Overlay

The overlay result depends on the orientation of the camera when taking the scene image. It seems that better result can be achieved when using an affine transformation for small changes in camera angle between the two images.

Q2 – Optical Flow

1) In order to calculate the approximated derivation in each dimension - 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.

   The calculation was done by simple subtraction from each value the next one, for each of the 3 dimensions.
   In order to deal with the last element (e.g. the rightest in the X dimension) we did so:
   For the right edge pixels, we used the first column of the next frame pixels, for the bottom row we used the first row of the same frame, for the time derivative of the last frame we will use the first frame.

   We marked groups of pixels as regions of size NxN pixels, which are not overlapping, in this step N=16.

   For each region we checked if the eigen values of $A^T A$ are larger than zero, this is because in "Lucas-Kanade" we assume spatial coherence, therefore:
   1. If a region has only an edge in it - $A^T A$ becomes singular.
   2. If a region is homogeneous - the derivatives in X and Y are roughly zero, so $A^T A$ has eigen values that are close to zero, and we can't track the direction of the flow.
   3. 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.

   In this step, we wanted to identify cases where the eigen values larger than a threshold of 1.

   For them, we calculated the (u,v) vectors of by solving a least squares problem.

2) From the estimations of (u,v) at each region, we displayed the flow over the full frame.
   We did it by mapping the (u,v) of each region to pixels in the original dimensions.
   We printed the output over the frame itself, e.g.:

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

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

We see that the valid regions are the ones where the cars, this is because they have more texture, which is as we excepted and wanted.
The direction of the arrows indicates the flow of the objects in the image for the next frame. It also seems to be logic for most of cases.
The size of the arrows indicates the velocity of the region between frames.

When watching 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, which is unwanted phenomena.
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. But it makes sense due to Optical Flow algorithm attributes and the fact that these are areas with texture.

Example of the output for some frames - [1,10,20,30,40,49]:

Example of the output for 5 sequential frames - [1,2,3,4,5,6]:

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

3) Repeating for t = 0.1:

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 trees. This translates to more arrows on the image. We can think about T as an threshold on how much texture needs to be in a region to make it considered as relevant.

Example of the output for some frames - [1,10,20,30,40,49]:

Example of the output for 5 sequential frames - [1,2,3,4,5,6]:

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

4) For N = 8 and t = 1 or t = 0.1:

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

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

Now we changed the size of the regions to be 8x8.
We can see that for T=0.1 there are less valid regions than when N=16, T=0.1. This is because N affect the size of the regions we look at, and for smaller regions there are less pixels considered. Therefore, we can assume smaller difference between pixels. Therefore, the derivation values are smaller, there is less textures and it ended with less valid values.
We can see that the arrows point in the direction of the car is headed in this frame, this is probably because of aperture ambiguity. N defines the aperture we look at.
For T=1, we still see 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. E.g., for N=16 we had more values on the road and trees.

Example for N = 8 and t = 1 of the output for some frames - [1,10,20,30,40,49]:

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

Example for N = 8 and t = 1 of the output for 5 sequential frames - [1,2,3,4,5,6]:

Example for N=8 and t=0.1 of the output for some frames - [1,10,20,30,40,49]:

Example for N=8 and t=0.1 of the output for 5 sequential frames - [1,2,3,4,5,6]:

5) To summarize the previous sections:
   The size of T affects the threshold which the eigen values must comply in order to be considered as valid.
   As much as T is higher, so the threshold is higher, therefore the eigen values must be bigger in order to be considered as valid.
   As a result, there are less valid values.
   Particularly, only regions where there was relatively significant texture considered as valid.
   We can think about T as a sensitivity threshold which determines which regions are significant.

Aram Gasparian | 310410865 | aram89@campus.technion.ac.il
Hana Matatov | 203608302| shanama@campus.technion.ac.il

The size of N affects the squared size of each region.
As much as N is lower, so the considered regions are smaller.
As the region is smaller, all calculations depend on less pixels.
Particularly, as there are less pixels, we can assume that the pixels in each region have smaller differences between them.
As a result, the values of the derivations Ix Iy It become smaller.
Therefore, there are less valid values.

Note that these 2 values (T and N) should be fine-tuned in order to deal with those trade-offs, in addition to the regular time and space complexity trade-offs.