Fall 2020 ECE/CS 5582 Computer Vision Homework-1

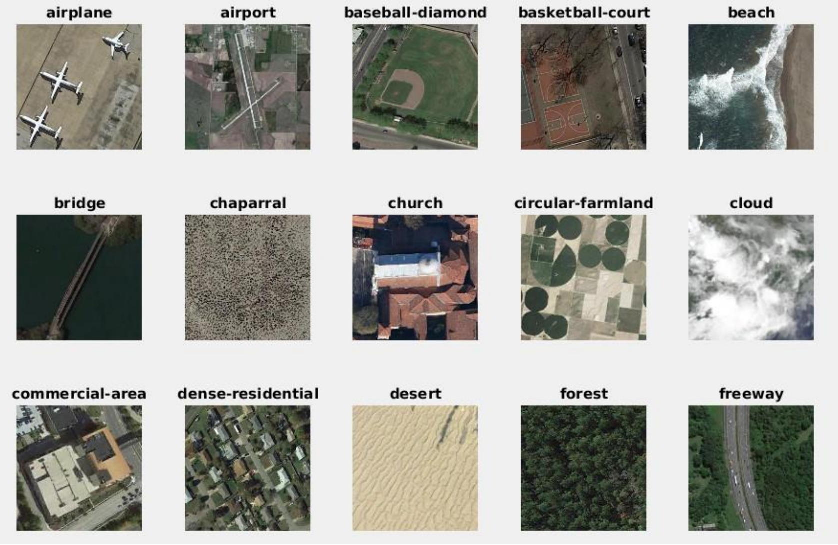
Student ID: 16281273 Name: Vladislav Dubrovenski

---

[1] Compute Histogram for image recognition (50pts)

Data set: the NWPU aerial data set contains approximately 45 categories of 700 images for each category in 256x256 RGB format. The data set can be downloaded from:

https://umkc.box.com/s/fxvzh5qq2tiob6eklfxfwn89kg3e1io1

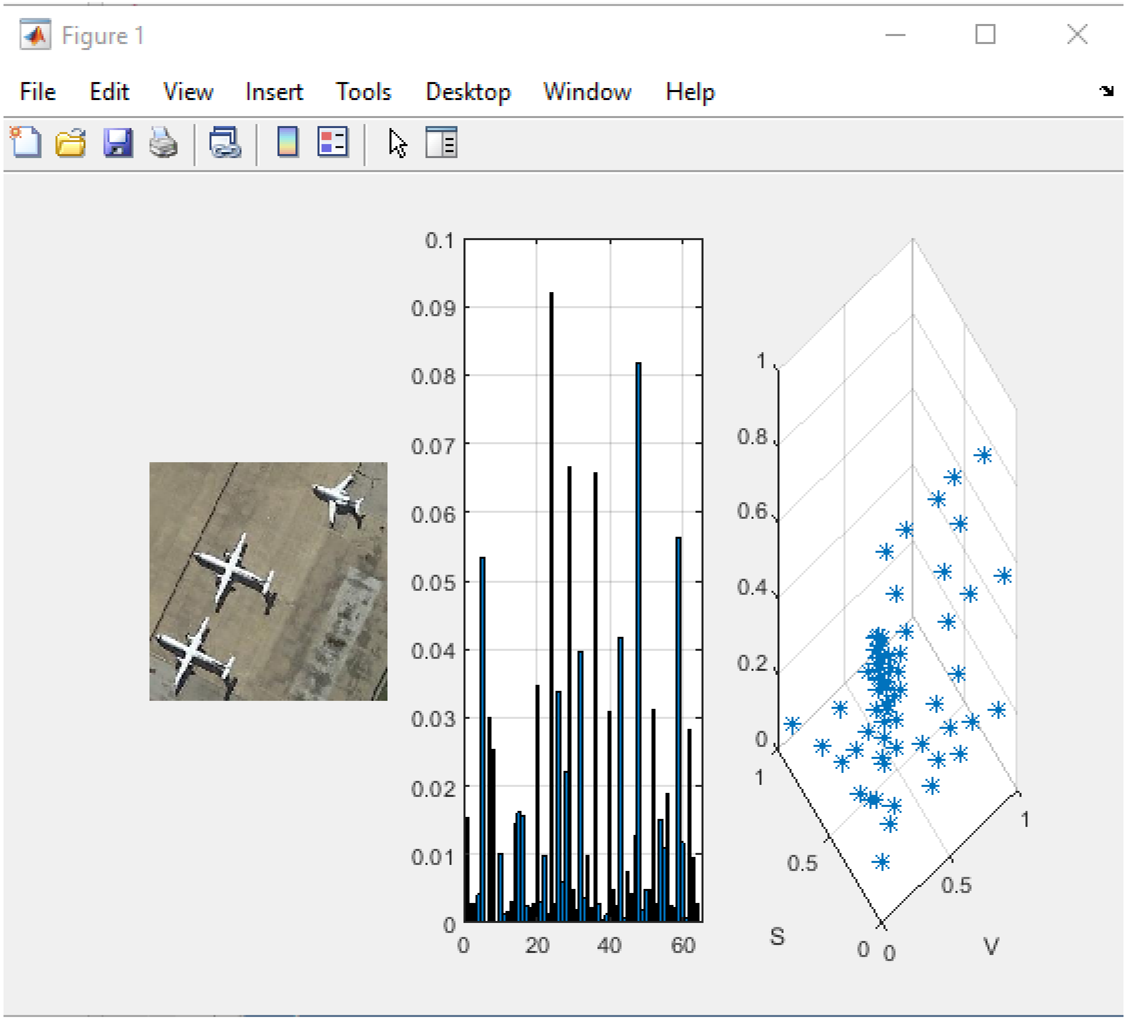


For HW-1, let us just take the first 15 classes (shown above) from the full data set, and 100 images per class and form a data set of 1500 images with 15 labels. Objectives are,

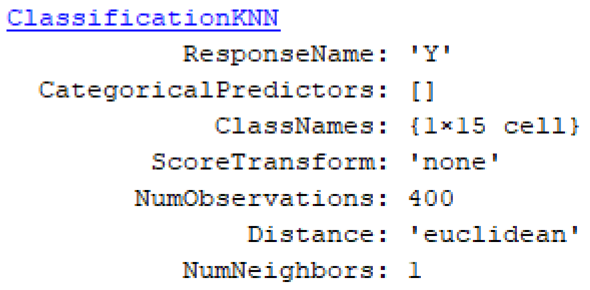
a) compute a HSV kmeans model with total number of entry K=64, b) for each image compute a color histogram, c) use Euclidean, and KL Distances to measure the similarity between two images, i.e, have a matlab/python function,[d]=getHSVDistance(im1, im2, t); where t is the table from Kmeans. d) randomly select 400 images from the data set, and compute its 1-NN label prediction from histogram distance and plot its confusion map.

In this question, I had to perform several computations. First I had to create a function to compute HSV kmeans that was consequently used to calculate histograms.

I computed a histogram for 1 picture to demonstrate the diagrams of histograms and centroid. The following is the screenshot of the diagrams:



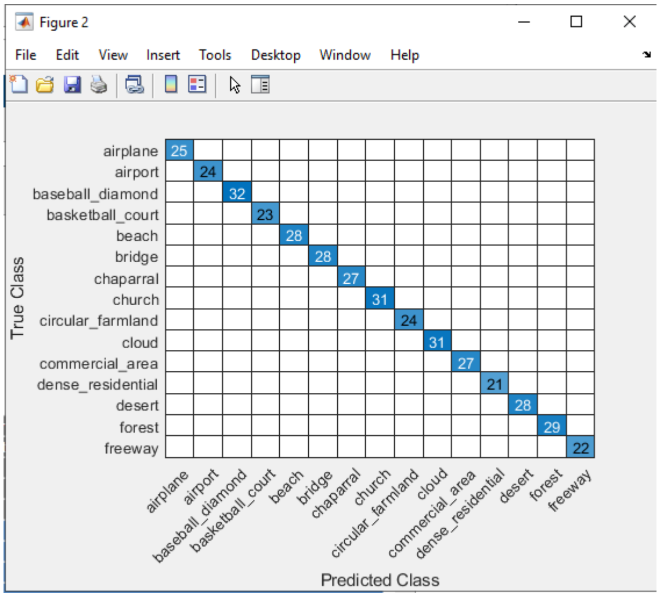
After that I calculated the histograms of each picture in the dataset of 1,500 and saved it for future utilization. After these two parts were completed, I consulted with the professor about finding a distance. I received a permission to use fitcknn which creates a model with the following properties:



```
 ClassificationKNN
           ResponseName: 'Y'
  CategoricalPredictors: []
             ClassNames: {1×15 cell}
         ScoreTransform: 'none'
        NumObservations: 400
               Distance: 'euclidean'
           NumNeighbors: 1
```

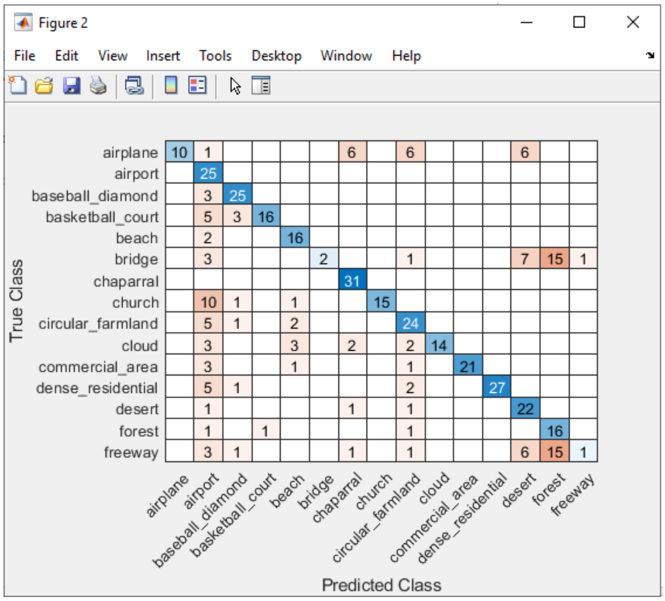
As we can see, the fitcknn is using Euclidean distance by default in calculating KNN which was one of the requirements. Below you will find the confusion charts for k=1(required in the assignment), k=2, k=5, k=10. The reason I use confusion chart instead of confusion matrix as it looks better.

K1:



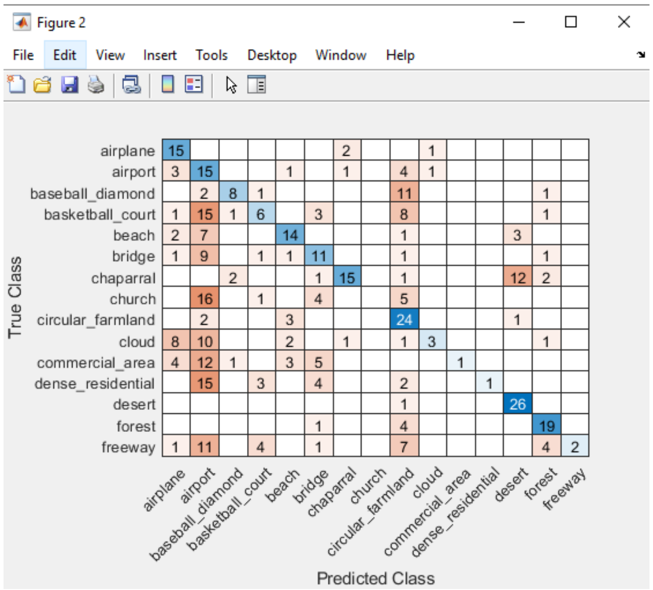
	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	25														
airport		24													
baseball_diamond			32												
basketball_court				23											
beach					28										
bridge						28									
chaparral							27								
church								31							
circular_farmland									24						
cloud										31					
commercial_area											27				
dense_residential												21			
desert													28		
forest														29	
freeway															22

K2:



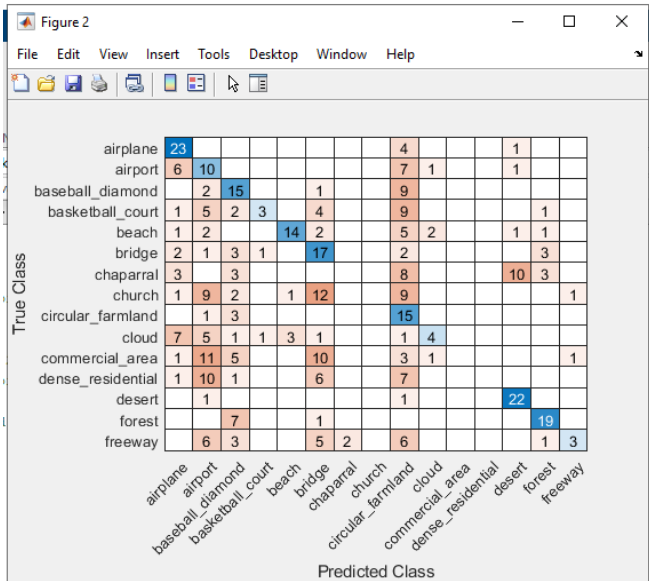
	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	10	1					6		6				6		
airport		25													
baseball_diamond		3	25												
basketball_court		5	3	16											
beach		2			16										
bridge		3				2			1				7	15	1
chaparral							31								
church		10	1		1			15							
circular_farmland		5	1		2				24						
cloud		3			3		2		2	14					
commercial_area		3			1				1		21				
dense_residential		5	1						2			27			
desert		1					1		1				22		
forest		1		1					1					16	
freeway		3	1				1		1				6	15	1

K5:



	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	15						2			1					
airport	3	15			1		1		4	1					
baseball_diamond		2	8	1					11					1	
basketball_court	1	15	1	6		3			8					1	
beach	2	7			14				1				3		
bridge	1	9		1	1	11			1					1	
chaparral			2			1	15		1				12	2	
church		16		1		4			5						
circular_farmland		2			3				24				1		
cloud	8	10			2		1		1	3				1	
commercial_area	4	12	1		3	5					1				
dense_residential		15		3		4			2			1			
desert									1				26		
forest						1			4					19	
freeway	1	11		4		1			7					4	2

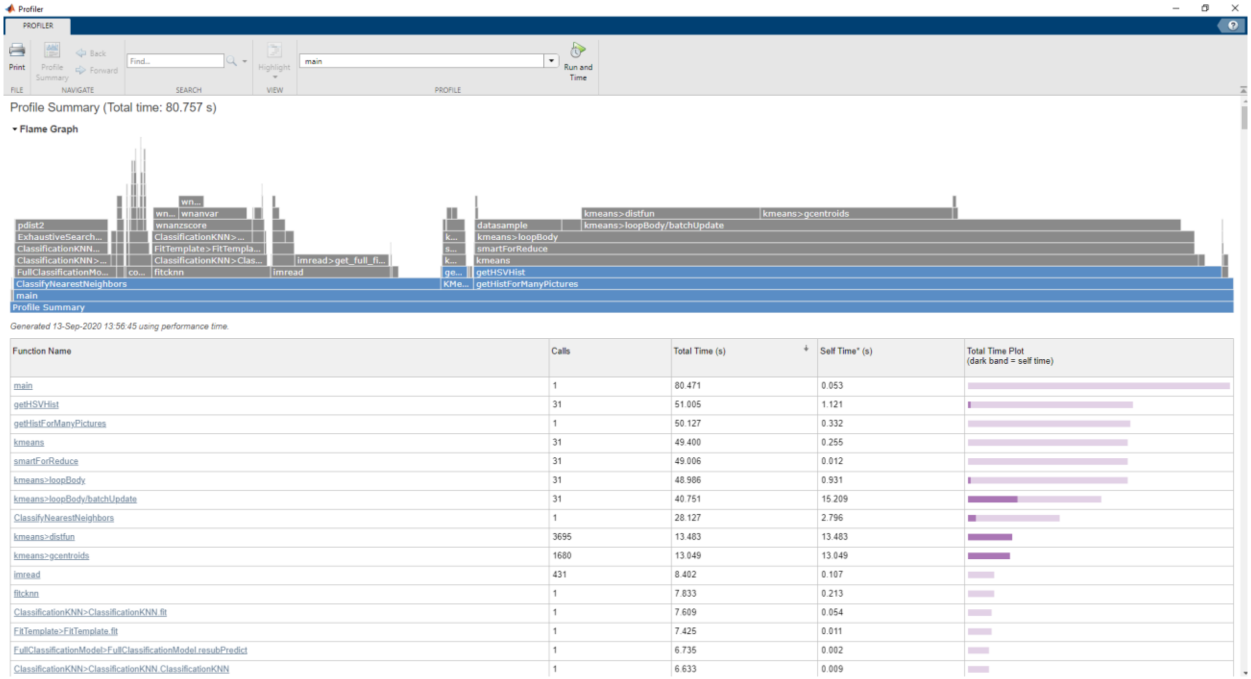
K10:



	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	23								4				1		
airport	6	10							7	1			1		
baseball_diamond		2	15			1			9						
basketball_court	1	5	2	3		4			9					1	
beach	1	2			14	2			5	2			1	1	
bridge	2	1	3	1		17			2					3	
chaparral	3		3						8				10	3	
church	1	9	2		1	12			9						1
circular_farmland		1	3						15						
cloud	7	5	1	1	3	1			1	4					
commercial_area	1	11	5			10			3	1					1
dense_residential	1	10	1			6			7						
desert		1							1				22		
forest			7			1								19	
freeway		6	3			5	2		6					1	3

To get the random images, I used “randperm” which got me 400 of unique indexes out of 1,500 images. I stored both classes and those images at the same index, which let me use them with knnclassifier as they had the same first dimension size(400). In fact, the classes were represented by a vector, while images were represented by 2 dimension matrix.

This was the first time I worked with Matlab on perform image data analytics. I was pleasantly surprised with the features this software provides. For example, it has a really nice “Profiler” which can be used to analyze how fast each function runs, and is very useful to find bottlenecks. For example, I figured out that I need to preallocate few data matrices with empty values just so the matlab does not have to change the size each time which is expansive. I did that on almost every matrix and cut the execution time by a lot. The profiler screenshot is below:

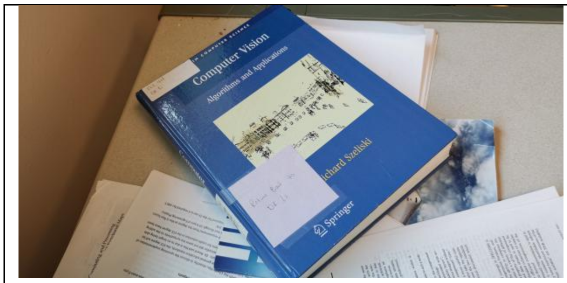


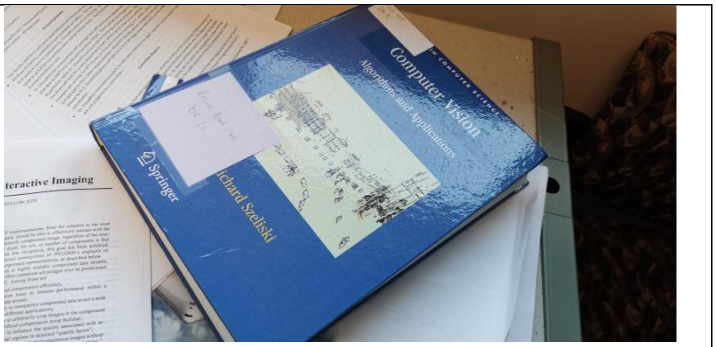
Profile Summary (Total time: 80.757 s)

Generated 13-Sep-2020 13:56:45 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
main	1	80.471	0.053	
getHSVHist	31	51.005	1.121	
getHistForManyPictures	1	50.127	0.332	
kmeans	31	49.400	0.255	
smartForReduce	31	49.006	0.012	
kmeans>loopBody	31	48.986	0.931	
kmeans>loopBody/batchUpdate	31	40.751	15.209	
ClassifyNearestNeighbors	1	28.127	2.796	
kmeans>distfun	3695	13.483	13.483	
kmeans>gcentroids	1680	13.049	13.049	
imread	431	8.402	0.107	
fitcknn	1	7.833	0.213	
ClassificationKNN>ClassificationKNN.fit	1	7.609	0.054	
FitTemplate>FitTemplate.fit	1	7.425	0.011	
FullClassificationModel>FullClassificationModel.resubPredict	1	6.735	0.002	
ClassificationKNN>ClassificationKNN.ClassificationKNN	1	6.633	0.009	

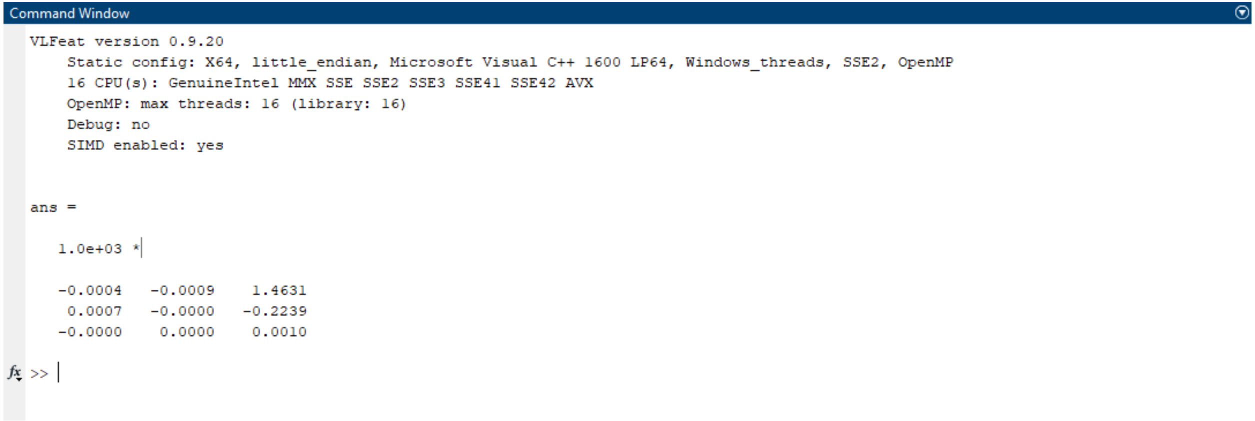
[2] Homograph estimation: use the SVD to solve the over-fitting Ah=0 as covered in class. Use the sample code as template: http://www.vlfeat.org/applications/sift-mosaic-code.html , compute homograph for the following two images:





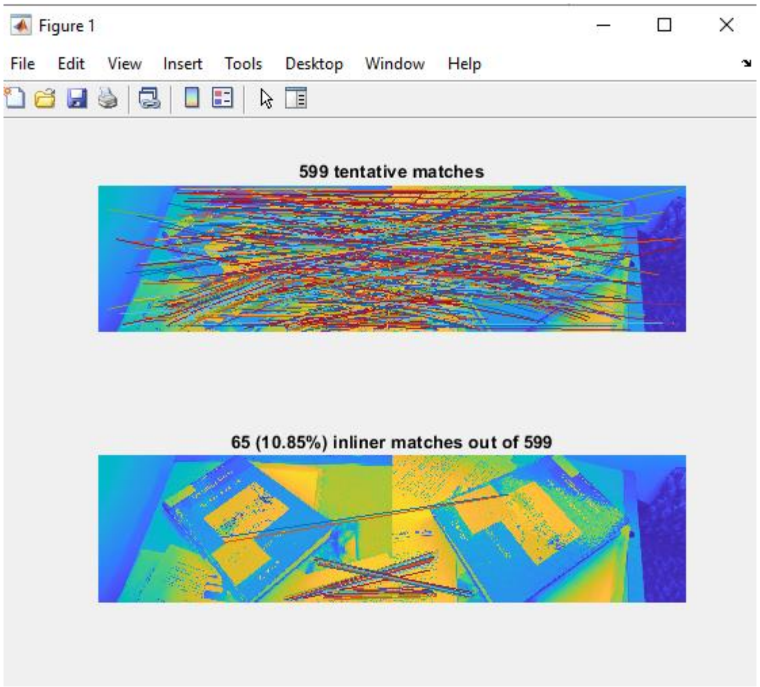
First compute SIFT and SIFT matches, then select good matches to solve homography.

To solve homography, we try to map every point of the first image to every point of the other image with transformation of homography. I did use vlfeat and the example provided in the question. I first computed the SIFT points for image 1 and image 2. Then I find matching points out of those SIFT points. Then I calculate the homography matrix. The following is the matrix I received:

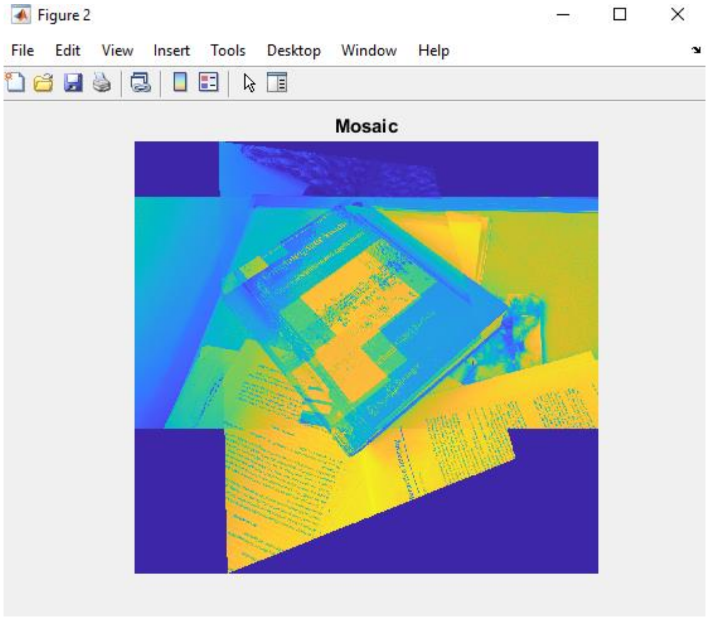


```
VLFeat version 0.9.20
    Static config: X64, little_endian, Microsoft Visual C++ 1600 LP64, Windows_threads, SSE2, OpenMP
    16 CPU(s): GenuineIntel MMX SSE SSE2 SSE3 SSE41 SSE42 AVX
    OpenMP: max threads: 16 (library: 16)
    Debug: no
    SIMD enabled: yes


ans =

   1.0e+03 *

  -0.0004   -0.0009    1.4631
   0.0007   -0.0000   -0.2239
  -0.0000    0.0000    0.0010

>> 
```

The following are the matches I got:



And the Mosaic solution:



The second part of the assignment was not very hard as I received an example code. However, I learned how easy it is to compute homography and find SIFT points using VLFeat package in matlab.