EEE3032 – Computer Vision and Pattern Recognition

Lab Worksheet 4 – Features and Classification

Miroslaw Bober (m.bober@surrey.ac.uk)

Task and Learning Objective: In today's lab you will experiment with the detection and matching of SIFT features. You will also explore unsupervised clustering of data using KMeans. Finally you will experiment with classification using Support Vector Machines (SVMs).

Resources: The exercises will be performed using Matlab. Download the Matlab code for the labs, if you have not done so before, and unzip the code into a separate folder. Ensure the code folder is in the Matlab search path (File -> Set Path.. -> Add with subfolders).

Please note this sheet uses pre-compiled parts of the lab code that will only work on the Linux PCs

Ex1. Harris corner and SIFT keypoint detector (5 mins)

Run the program `demokeypoint`. This will load two images from the `testimages` folder (`wall1.jpg` and `wall2.jpg`) and detects both Harris corners and SIFT keypoints within them. These points are supposed to be stable across changes in viewpoint. By comparing the resulting Harris (blue) and SIFT (yellow) points visually between the views, do you agree?

The code saves the coordinates of the detected SIFT keypoints, and their corresponding SIFT descriptors, in the file `siftresults.mat`. These variables can be recalled by typing

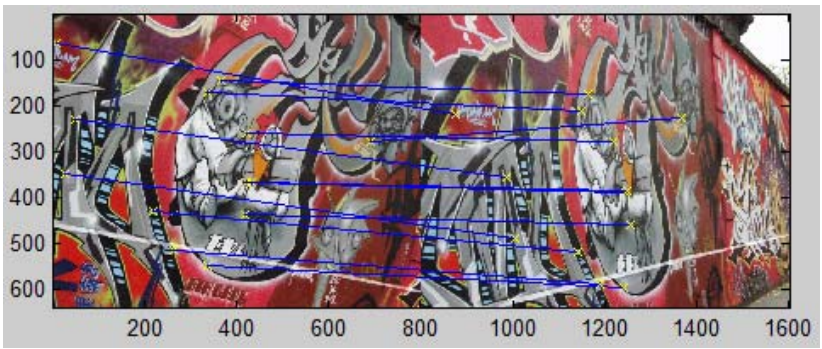
```
>> load siftresults.mat
```

The variable `keypoints1` contains the 2D coordinates of SIFT keypoints in `wall1.jpg` and the variable `keypoints2` contains those from `wall2.jpg` . The variables `descr1` and `descr2` contain the corresponding SIFT descriptors for each keypoint.

Ex2. SIFT descriptor matching (20 mins)

Run the program `demomatching`. This will load two images from the `testimages` folder (`wall1.jpg` and `wall2.jpg`) and uses the keypoints and their descriptors already computed in Ex.1 to create correspondences between the images. Vary the parameter STEPSIZE (line 31) which restricts the number of matches displayed. Do these matches appear visually correct?

Spend some time studying the code of `demokeypoint` and `demomatching` in detail, and ensure you thoroughly understand the steps involved in using the supplied SIFT code to detect keypoints in a pair of images and match them. You should recall / consider discussions on SIFT matching from the Lectures.



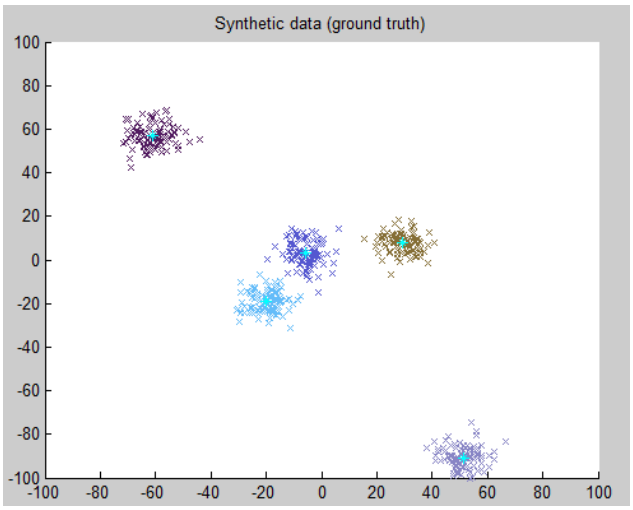
Ex3. Clustering via K Means (15 mins)

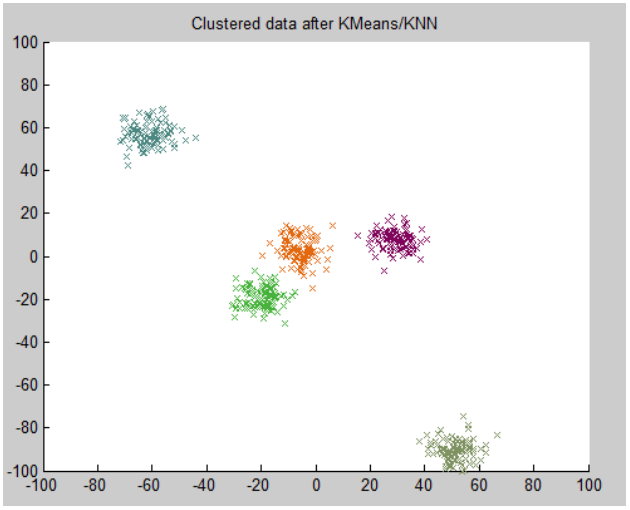
Run the program `kmeantest`. The program generates some random clusters of 2D data points, mixes the data together and then runs K-means to try to separate the data into the original clusters (i.e. performs unsupervised clustering).

Run the program a few times – how well does K-Means appear to perform if clusters touch or overlap?

Experiment by varying the parameters on the data generation routine (SPREAD which controls the spread i.e. size of cluster in the 2D space; NPOINTS the number of points per cluster; NCLUSTERS the number of clusters). How well does K-Means perform under extreme e.g. low or high values?

Disable the code for visualizing the points, and change DIMENSION to a higher number. Satisfy yourself that it is possible to use this code to cluster n-dimensional data using K-means.





Ex4. SVMs (10 min)

Run the program `testsvm`. The code uses a similar approach to `kmeantest` to generate random clusters of data, but here we use supervised learning (rather than unsupervised approach) to cluster the synthetic data.

Recall that in a supervised learning scenario, we "train" a classifier (here we use an SVM) to identify the characteristics of each data cluster. Here, the data is 2D – and we train the SVM on the (x,y) locations of points in each cluster. In this example we use 20% of our synthetic data to train the SVM.

We then feed the SVM the remaining 80% of our synthetic data, to see if the SVM can determine which cluster the data originally came from. Recall this process is "classification"

The program outputs a score based on classification of the training data, and of the test data. Unsurprisingly a classification over the training data outperforms classification over the unseen test data.

Similar to Ex.3, Vary the parameters of this code to explore conditions under which the classification performs well. You may wish to consider the "pure chance" classification as a baseline. For example, if you have 3 clusters a "pure chance" response will classify with accuracy 33% (i.e. one third).