Scene Segmentation and Interpretation
Pascal Project

Taner GUNGOR
Al ARAFAT

May 31, 2015

Deadline:	June 01, 2015
Instructor:	Xavier Llad
	Arnau Oliver

1 Introduction and problem definition

1.1 Introduction

For human, it is easier to perceive the environment around him but considering Computer vision system, it is that much harder to recognize its surrounding environment. Thus, detecting and recognizing objects has become one of the core tasks of computer vision.

In computer vision, standard object recognition paradigm relies on matching and geometric verification. For generic object categorization, a statistical model of appearance or shape learns from examples. For the categorization problem, learning visual objects entails gathering training images of the given category, and then extracting or learning a model that can make new predictions for object presence or localization in novel images. Models are often constructed via supervised classification methods, with some specialization to the visual representation when necessary. [1]

Concerning recognition, the big challenge comes with visual recognition. Since different visual object classes can have same perceived shape or same object class can have different shapes, the recognition problem includes identifying instances of a particular object or scene as well as generalizing to categorize instances at a basic level like, cat or dog or *etc.*.

For recognition problem, another big concern is the computational complexity and scalability. Highly efficient algorithms are necessary to implement to work efficiently over high-dimensional image representations, to search large image databases, or to extend recognition to thousands of category types. While scalability problem associates with designing training dataset. Thus the better algorithm implementation should consider both the factors.

In this project work, we implement a best suited strategy to recognize objects from a number of realistic scene of visual object classes. As our strategy, we implement different combinations of descriptors and classifiers to obtain best result. We use,

1. *Development kit* which simplifies the work as it has already implemented the file access portion.
2. *Sift* library to automatically compute *sift* descriptors from an image.
3. *PRtools5* - Matlab toolbox for Pattern Recognition.
4. *K-Nearest Neighbor, AdaBoost* and *Support Vector Clustering* methods as classification algorithms to check which one gives better result.
5. *ROC* curve to compare the results form different classification algorithms and choose the best strategy.

The implementation of our strategy is described briefly in the later sections.

1.2 Problem definition

This project work asks for designing and implementing an object recognition paradigm that would efficiently recognize object from a large dataset. We follow and also use the database of the well known Pascal Challenge 2006 `http://www.pascal-network.org/challenges/VOC/voc2006/index.html`, where 10 object classes have to be recognized,

- bicycle
- bus
- car
- motorbike
- cat
- cow
- dog
- horse
- sheep
- human

The main goal of this project can be stated as a sequence of steps as following:

i Information search.

ii Design, analyze and implement the strategy to follow, which is a combination of descriptors and classifier with better result.

iii Test the classification with the provided images sets. Analyze the problems and offer some possible improvements.

We implement our strategy over both the **gray level** images and **RGB color space** images. And then we compare our classification result using the ROC curve analysis.

2 Strategy analysis

Since the project work is an open work, we can choose any strategy to solve the problem, the descriptors, the classifiers. We use *Sift* as descriptor, *k-means* as clustering algorithm and *Bag of words* to select features in both *dense* and *sparse* patches. And as classifier we use *K-Nearest Neighbor*, *AdaBoost* and *Support Vector Clustering.* All of the topics are described below.

2.1 SIFT descriptor

First step in building the *SIFT* feature vector is to locate the keypoint. The following figures show the process of getting the *SIFT* keypoint,

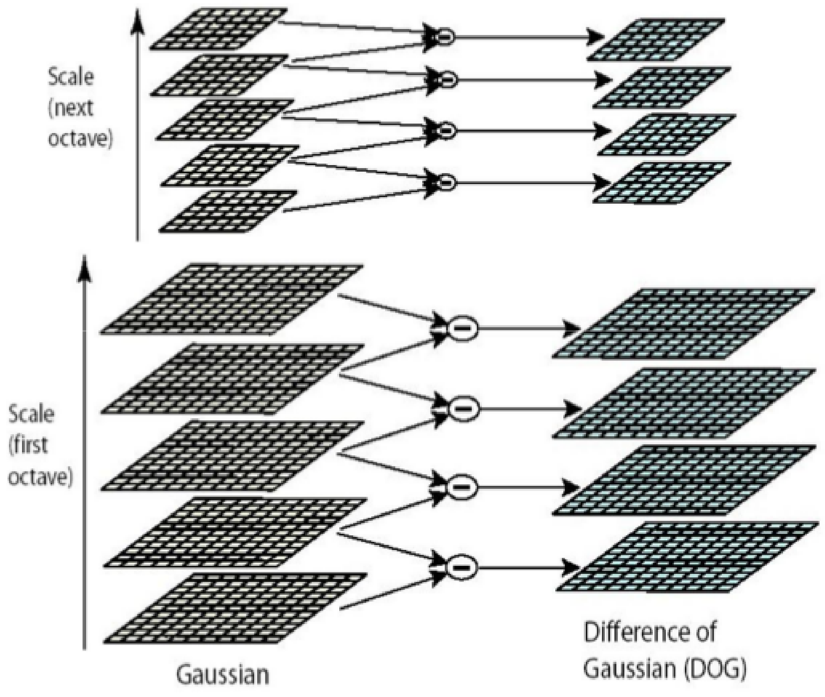

Figure 1: Difference of Gaussian from Gaussian pyramid. [2]

At first, the original image is convolved with incremental Gaussian to produce images separated by a constant value. Then the differences between them are computed. Then follows the following step,

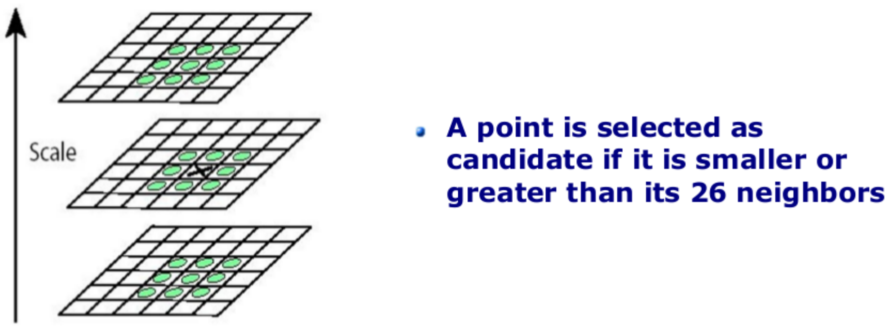
Scale
A point is selected as candidate if it is smaller or greater than its 26 neighbors

Figure 2: Selecting SIFT keypoint. [2]

Next step is concerned about detecting the local extrema in 3D, that is, to detect points that are smaller or larger than all its neighbors, and the neighborhood includes the 8 neighbors in its same image, and the $9 + 9$ neighbors in its superior and inferior scale within the same octave.

The *SIFT* feature descriptor is a vector that contains the values of the gradient orientation histogram entries. The following figure depicts the way of computing *SIFT* descriptor.

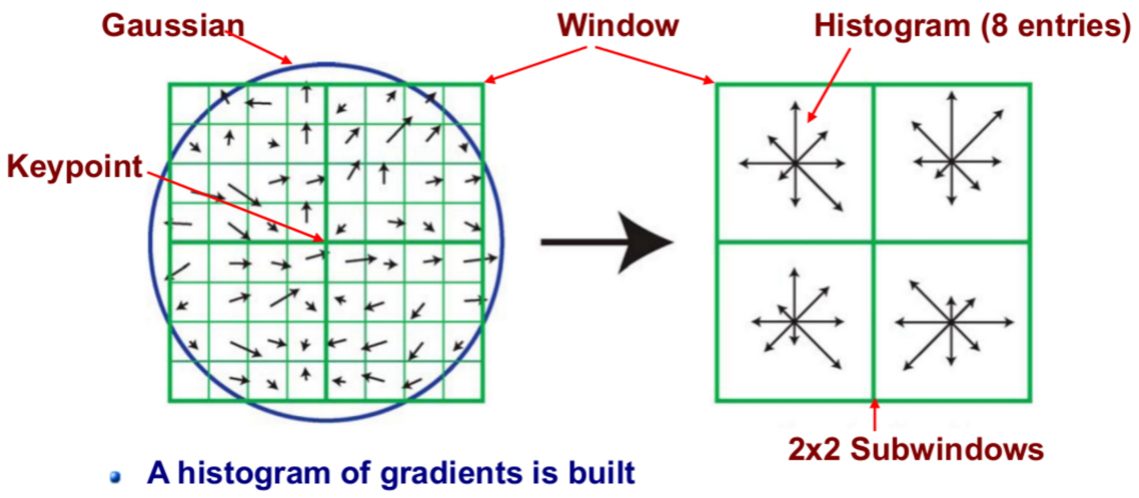

Figure 3: SIFT keypoint descriptor. [2]

The steps of building *SIFT* keypoint descriptor vector are:

1. At each point where a *SIFT keypoint* is detected, the descriptor is constructed by computing a set of 16 orientation histograms based on 4×4 windows within a 16×16 pixel neighborhood centered around the *keypoint*.
2. At each pixel in the neighborhood, the gradient direction (quantized to 8 directions) is computed using a Gaussian with s equal to 0.5 times the scale of the *keypoint*.
3. The orientation histograms are computed relative to the orientation at the *keypoint*, with values weighted by the gradient magnitude of each pixel in the window.
4. This results in a vector of 128 ($= 16 \times 8$) feature values in the SIFT descriptor. (The values in the vector are also normalized to enhance invariance to illumination changes.)

Thus, we get the feature vector from this step.

2.2 K-means clustering

K-means is a clustering algorithm that tries to partition a set of points into K sets or clusters in a way that the points in each cluster tend to near each other. It is unsupervised because the points have no external classification. The principle tends to minimize the sum of the squares of distances between data and the corresponding cluster centroids.

For example, if we have a dataset of 5 players who need to be grouped into k distinct groups based on their belonging department. To solve this problem, we can use *k-means*. Here *k* is the number of clusters we want to have in the end. k = 2 means we have 2 clusters, or distinct groups or for this particular example, number of departments. The following figure shows the steps of *k-means* clustering algorithm,

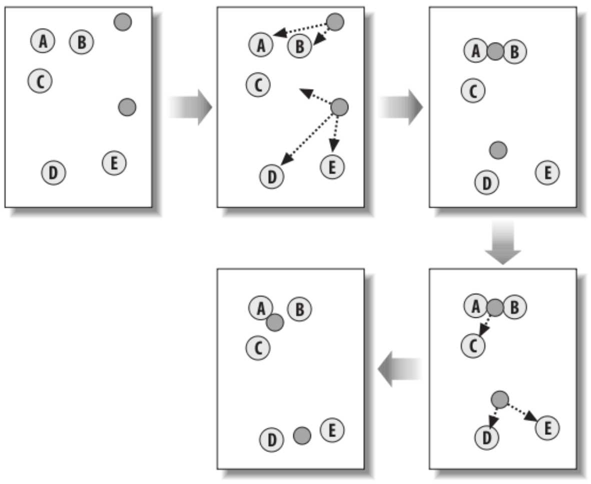

Figure 4: *K-means* clustering algorithm. [3]

2.3 Bag of words

In computer vision, bag of words resembles the idea of representing an object as a bag of visual words called *patches* that are described by a certain descriptor. The following well-known figure gives an idea about bug of words,

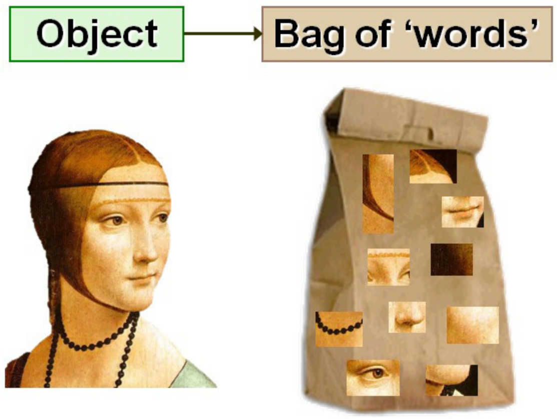

Figure 5: Bag of words model. [4]

We can use the bag of words model for object categorization by constructing a large vocabulary of many visual words and represent each image as a histogram of the frequency words that are in the image. The following figure illustrates this idea:

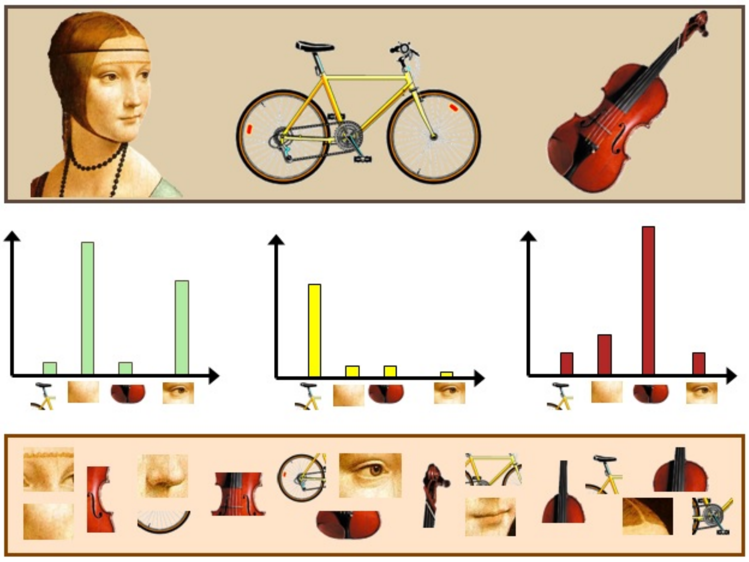
Figure 6: Representing objects as histogram of words in *Bag of words*. [4]

The steps to build the dictionary are:

- From a large set of object images, the descriptors are extracted (*SIFT* for our implementation).
- The descriptors are then clustered (for this implementation *K-means*). The cluster centers act as the dictionary's visual words.
- Build a histogram of length k where i^{th} value is the frequency of the i^{th} dictionary words.

The following figure show the steps of building the *Bag of words* model,

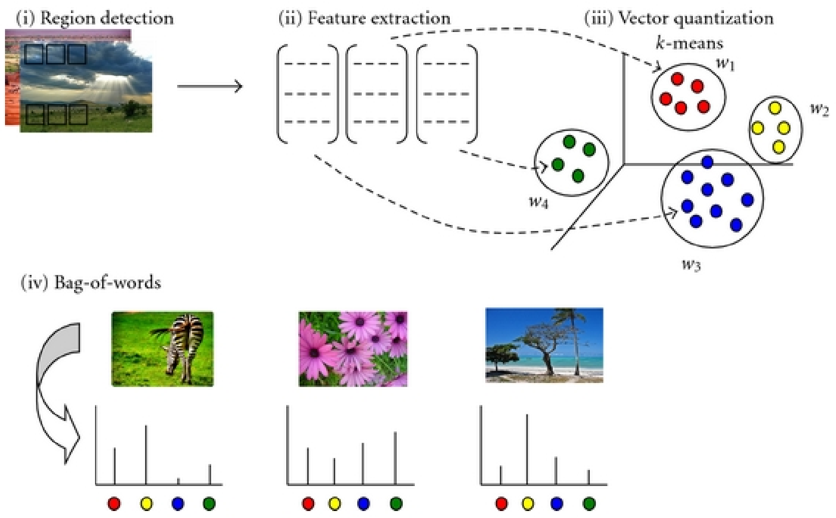

Figure 7: Steps in building *Bag of words*. [5]

The next step is to classify the dataset.

2.4 K-Nearest Neighbor

K-Nearest Neighbor is one of the simplest of classification algorithms available for supervised learning. The idea is to search for closest match of the test data in feature space. The following figure gives an idea about *K-NN*,

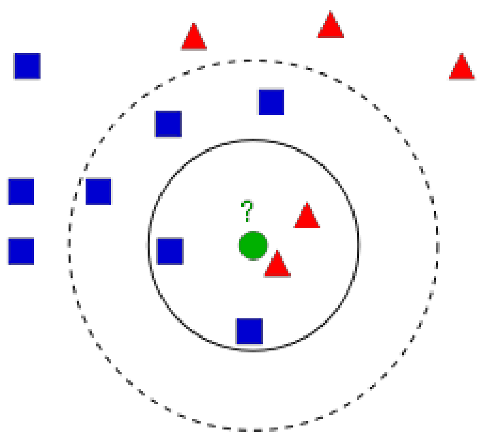

Figure 8: K-NN theory.

In the image, there are two families, *Blue Squares* and *Red Triangles*. For example, let us call each family as Class. Their houses are shown in their town map which we call feature space. For this example this one is a 2D feature space. Now a new member comes into the town and creates a new home, which is shown as green circle. He should be added to one of these Blue/Red families. We call that process, Classification. Since we are dealing with K-NN, let us apply this algorithm.

From the figure, *Red Triangle* may be the nearest. If there are lot of *Blue Squares* near to him then *Blue Squares* have more strength in that locality than *Red Triangle*. Since just checking nearest one is not sufficient, we check some k-nearest families. Then whoever is majority in them, the new guy belongs to that family.

In our image, let us take k=3, *i.e.* 3 nearest families. The new comer has two Red and one Blue (there are two Blues equidistant, but since k = 3, we take only one of them). Thus, again, he should be added to Red family. But if we take k=7, then the new comer has 5 Blue families and 2 Red families. Now he should be added to Blue family. So it all changes with value of *k*. This method is called *k-Nearest Neighbor* classification algorithm since classification depends on *k-nearest neighbors*.

2.5 Adaptive Boosting

AdaBoost stands for *Adaptive Boosting*, consists of two parts: a simple weak classifier and a boosting part. The *weak classifier* tries to find the best threshold in one of the data dimensions to separate the data into two classes -1 and 1. The *boosting* part calls the classifier iteratively, after every classification step. It changes the weights of miss-classified examples. This creates a cascade of "weak classifiers" which behaves like a "strong classifier". The following figure shows the way, how *AdaBoost* works,

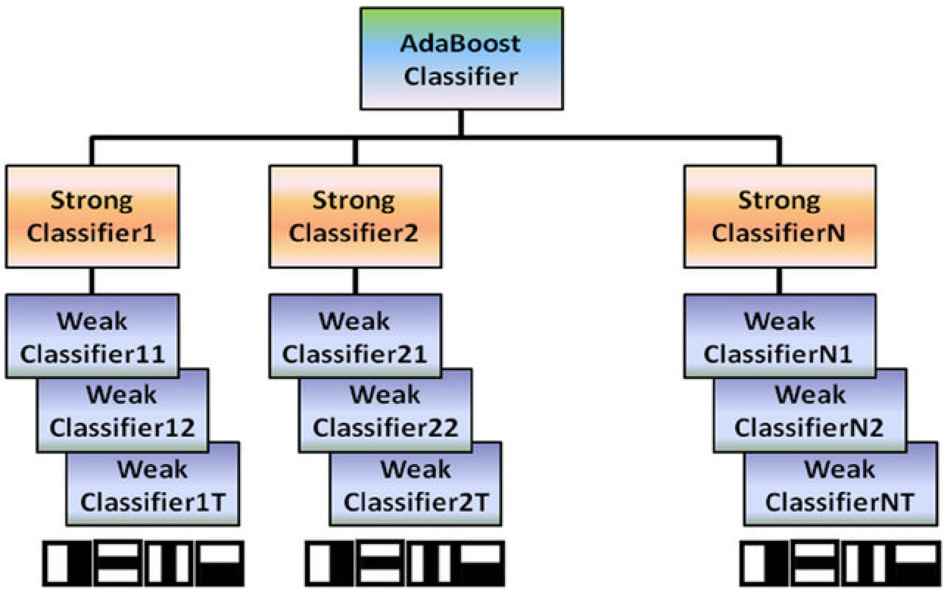

Figure 9: Adaptive Boosting classification algorithm. [6]

2.6 Support Vector Clustering

This algorithm learns an item-pair similarity measure to optimize performance of correlation clustering on a variety of performance measures. *Support Vector Clustering (SVC)* algorithm maps data points from data space to a high dimensional feature space using a Gaussian kernel.The working steps of *Support Vector Clustering* are:

1. In feature space we look for the smallest sphere that encloses the image of the data.
2. This sphere is mapped back to data space, where it forms a set of contours which enclose the data points.
3. These contours are interpreted as cluster boundaries. Points enclosed by each separate contour are associated with the same cluster.
4. As the width parameter of the Gaussian kernel is decreased, the number of disconnected contours in data space increases, leading to an increasing number of clusters. [7]

The following image shows how *SVC* classifies features in feature space,

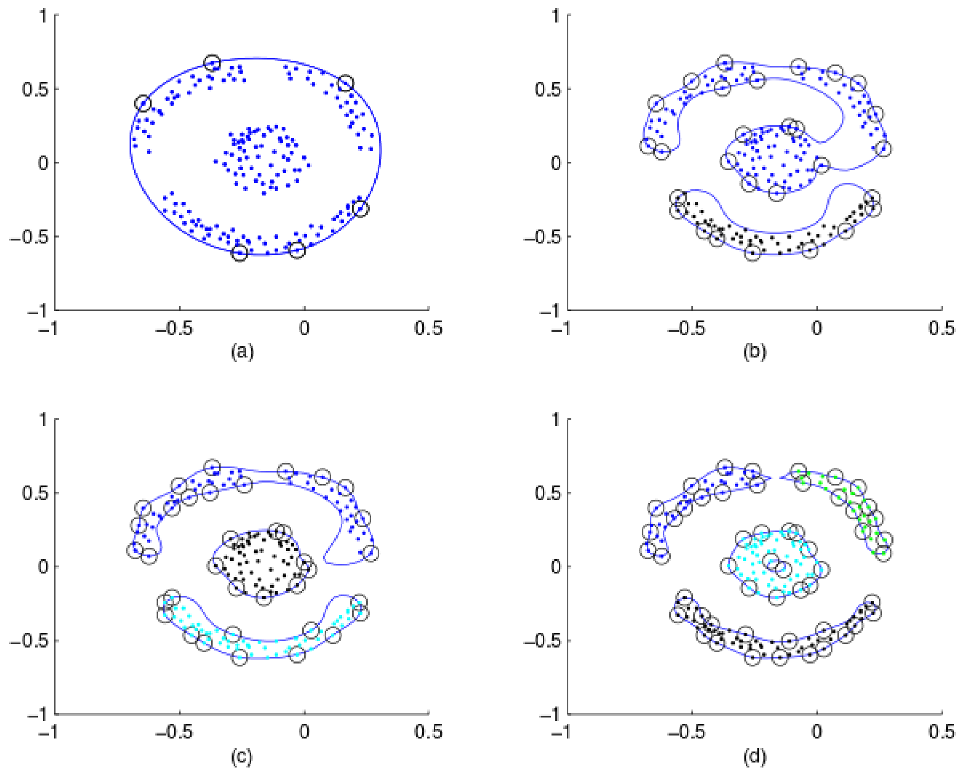

Figure 10: Clustering of a data set containing 183 points using SVC with C = 1. Support vectors are designated by small circles, and cluster assignments are represented by different gray scales of the data points (a) q = 1, (b) q = 20, (c) q = 24, (d) q = 48. Here, q is the scale parameter of the Gaussian kernel and C is soft margin constant. [7]

2.7 Receiver operating characteristic

A *Receiver Operating Characteristic (ROC)* graph is a technique for visualizing, organizing and selecting classifiers based on their performance. The properties of *ROC* curve are:

1. *ROC* graphs have long been used in signal detection theory to depict the trade-off between hit rates and false alarm rates of classifiers.
2. When evaluating binary decision problems, ROC curves show how the number of correctly classified positive examples varies with the number of incorrectly negative examples (*True Positive Rate* vs *False Positive Rate*).

The best use of *ROC* curve is carried out when the *Area Under Curve (AUC)* is measured. *AUC* is just the area under the *ROC* curve. The bigger *AUC* means that best classification result is obtained. Thus, comparing the *AUC* of several classification algorithms from their *ROC* curve suggest the better classification algorithm to use.

The following figure show how *ROC* curve evaluates classifiers,

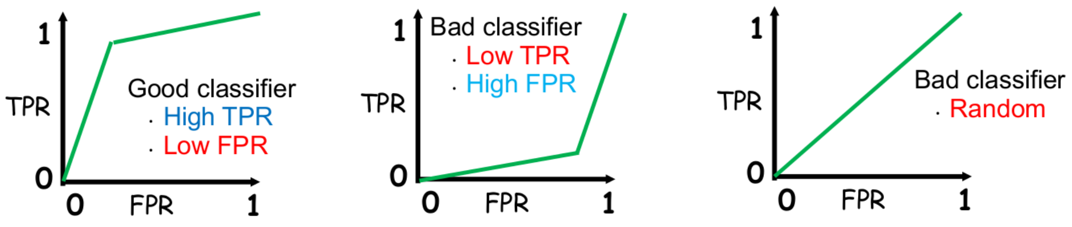

Figure 11: Evaluating classifiers according to the AUC for the ROC curve. [8]

3 Design and implementation of the proposed solution.

In this section, we discuss about the design of our strategy to solve the problem and our approaches. For this project work, we use the large database images from PASCAL 2006. From the images, we use 15 images from each class for training to define the center. After training, all the images are used for testing. The whole project work can be separated into 5 different tasks:

1. Generating feature vector.
2. Clustering the features.
3. Selecting vocabulary for the classifiers.
4. Classification.
5. Evaluation.

For each of the task, our approach is discussed below:

1. **Generating feature vector:** To generate feature vector, we use *SIFT* descriptor. We use the *vl-feat* library for MATLAB. For gray level images, it is simple to compute the feature vector for each image. But for RGB images, the R,G,B channels are separated and the feature vector for each channel is computed. Since the feature vector for each channel can be different in shape, we use the minimum size of the 3 to generate the feature vector for an RGB image.
2. **Clustering the features:** The features are clustered using *k-means*. We use the build-in *kmeans* function of MATLAB.
3. **Selecting vocabulary for the classifier:** We use different vocabulary size and evaluate the result. For all of the cases, we use vocabulary size of 50, 100 and 200. After *dictionary* size = 100, increasing vocabulary size decreases the performance slowly and also increases computational cost.
4. **Classification:** We use 3 different classifier to classify the objects. We use *K-NN, AdaBoost* and *SVC* and then evaluate their results. They have libraries for MATLAB.
5. **Evaluation:** To evaluate the classified results, we use the *ROC* curve. We also computed the time complexity for each of the classifier.

4 Experimental section and results analysis (speed, ROCs, Az values, etc).

After running our implementation, it generates *ROC* curve for each class. Then the *ROC* curves are stored in a separate folder named *ROC*. The experimental results are shown below:

Object class: Bicycle. **Image: RGB**

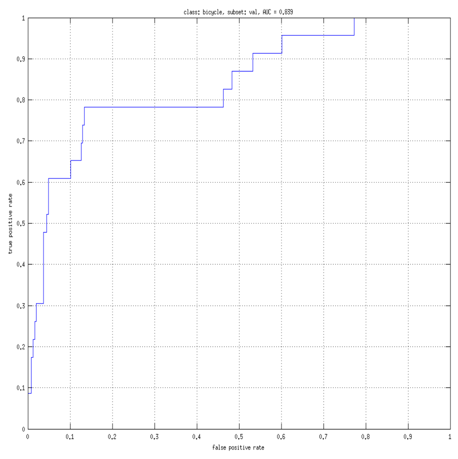

Figure 12: *AdaBoost* classifier with vocabulary size = 50.

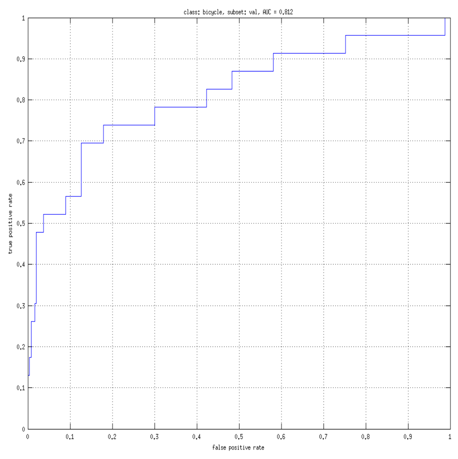

Figure 13: *AdaBoost* classifier with vocabulary size = 100.

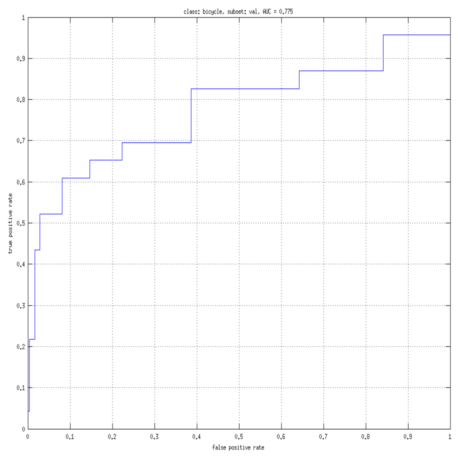

Figure 14: *K-NN* classifier with vocabulary size = 50.

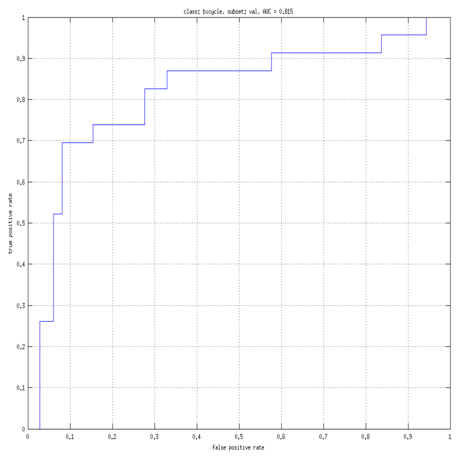

Figure 15: *K-NN* classifier with vocabulary size = 100.

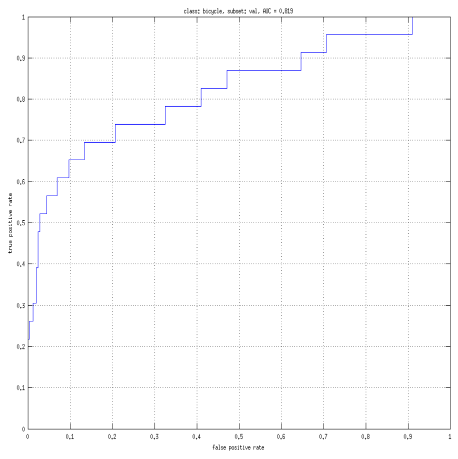

Figure 16: *SVC* classifier with vocabulary size = 50.

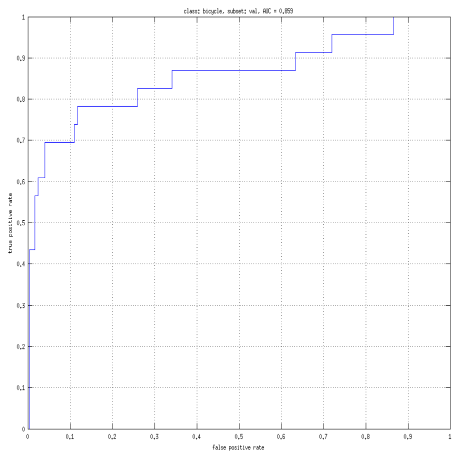

Figure 17: *SVC* classifier with vocabulary size = 100.

Object class: Bicycle. **Image: Gray level**

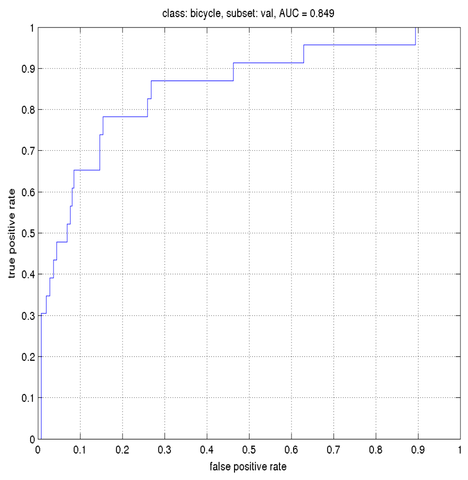

Figure 18: *AdaBoost* classifier with vocabulary size = 100.

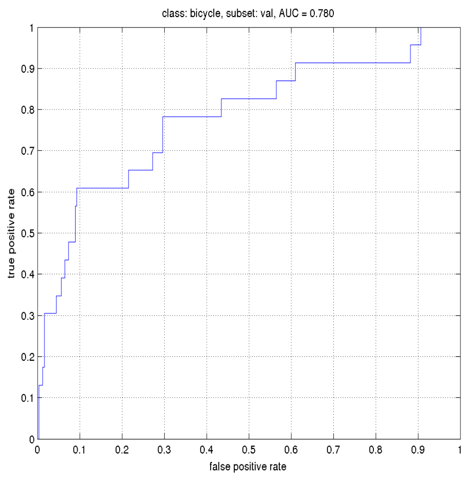

Figure 19: *AdaBoost* classifier with vocabulary size = 200.

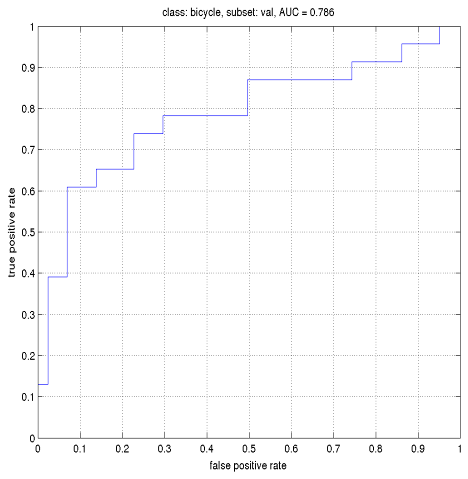

Figure 20: *K-NN* classifier with vocabulary size = 100.

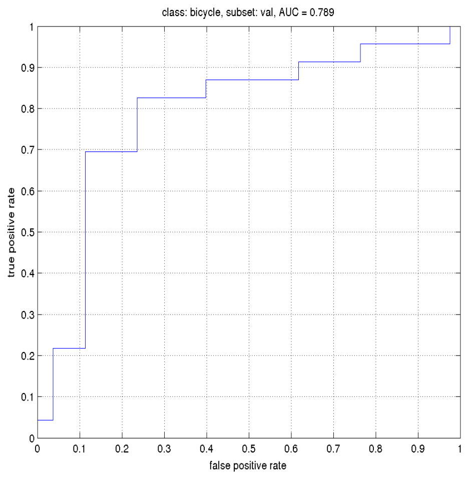

Figure 21: *K-NN* classifier with vocabulary size = 200.

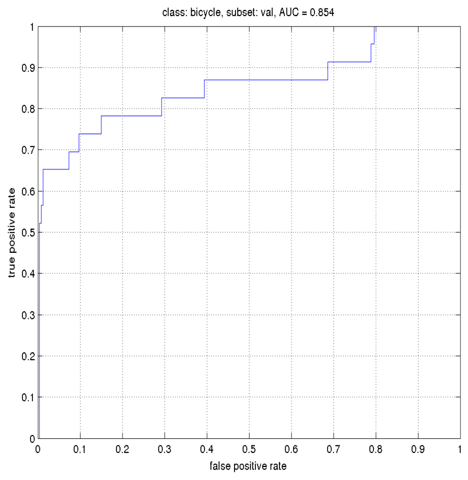

Figure 22: *SVC* classifier with vocabulary size = 100.

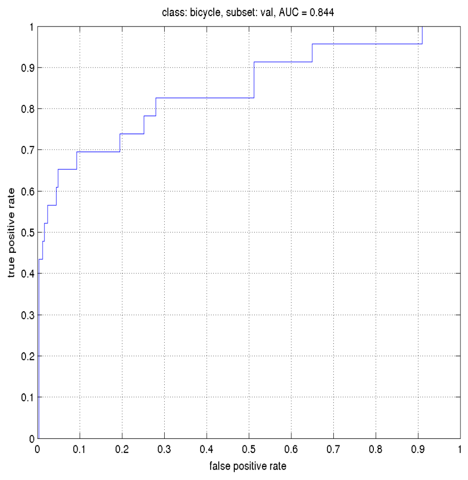

Figure 23: *SVC* classifier with vocabulary size = 200.

Comparison Results :
Comparing the AUC and time complexity for each of the classifier algorithm we found the following result:

Comparing the computational time

Classifier	RGB image	Gray level image
K-NN	3.1285	3.2479
SVC	3.0777	3.2644
AdaBoost	3.1917	5.5731

For each of the object class, we found the best result using the *Support Vector Clustering.* And considering time complexity, *K-Nearest Neighbor* gives better result. And the

- **AdaBoost :**
  AUC : Better than *K-NN* but worse than *SVC*,
  Computational time : Slower than both *K-NN* and *SVC* for both RGB and gray level images.
- **K-NN :**
  AUC : Worse than both *AdaBoost* and *SVC*,
  Computational time : Faster than both *AdaBoost* and *SVC* for gray level images but slower than *SVC* for RGB images.
- **SVC :**
  AUC : Better than both *AdaBoost* and *K-NN*,
  Computational time : Slower than *K-NN* but faster than *AdaBoost* for gray level images but faster than both the *K-NN* and *AdaBoost* for RGB images.

Thus considering both the AUC and *computational time*, we chose **Support Vector Clustering** as our classifier. For the challenge, we chose the implementation over **gray level** images with **dictionary** size = 100, since for this combination, we got better result from the SVC classifier.

The **SVC** classifier results for all the object classes using **gray level** images and **dictionary** size = 100 are shown below,

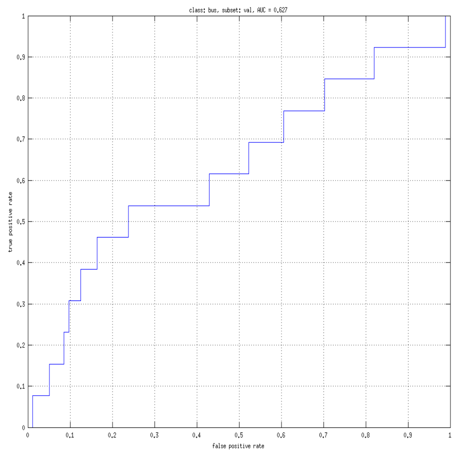

Figure 24: ROC curve for object class **bus**.

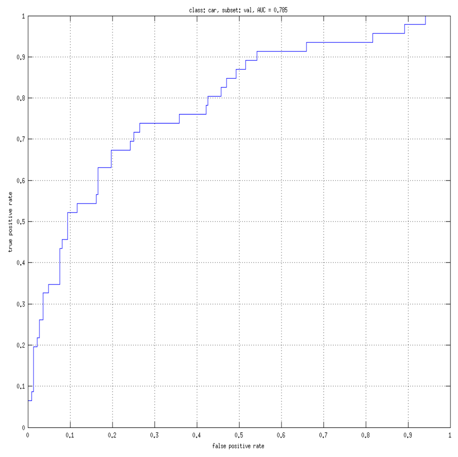

Figure 25: ROC curve for object class **car**.

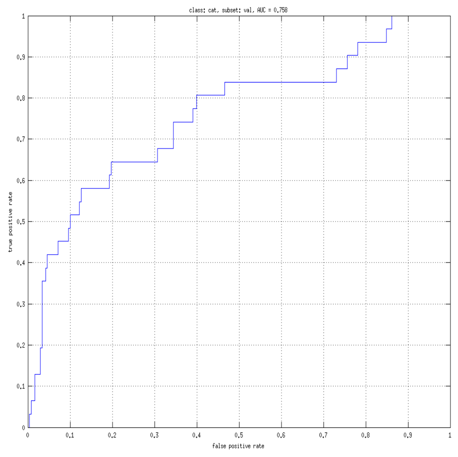

Figure 26: ROC curve for object class **cat**.

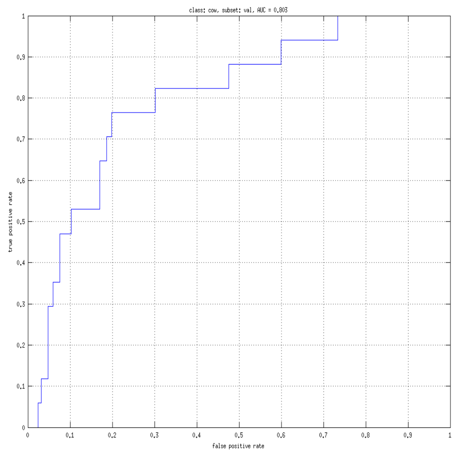

Figure 27: ROC curve for object class **cow**.

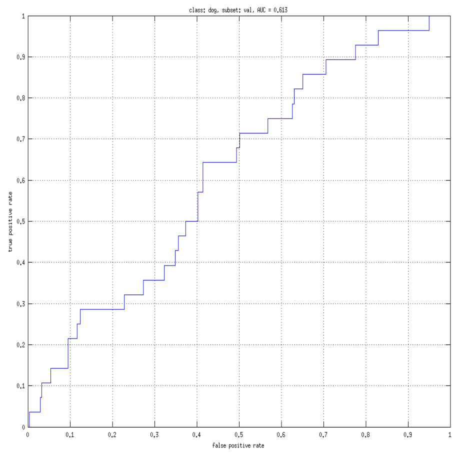

Figure 28: ROC curve for object class **dog**.

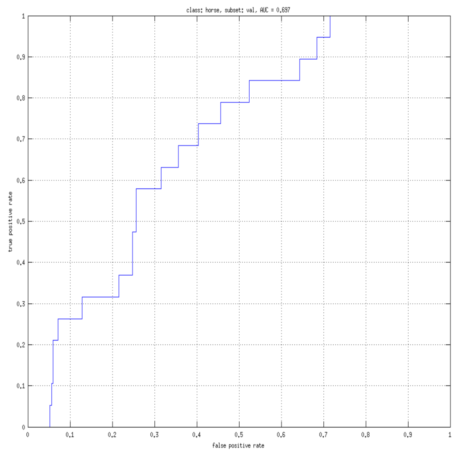

Figure 29: ROC curve for object class **horse**.

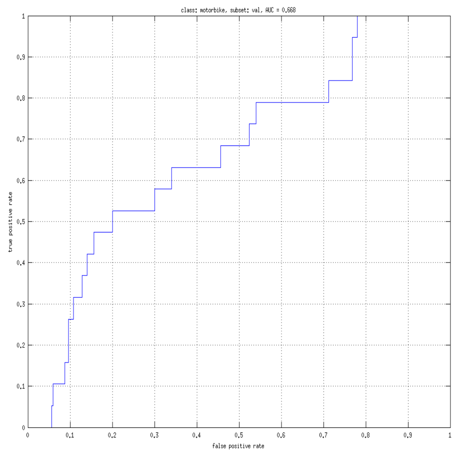

Figure 30: ROC curve for object class **motorbike**.

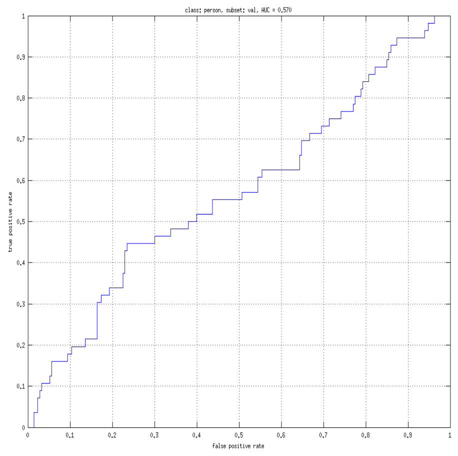

Figure 31: ROC curve for object class **person**.

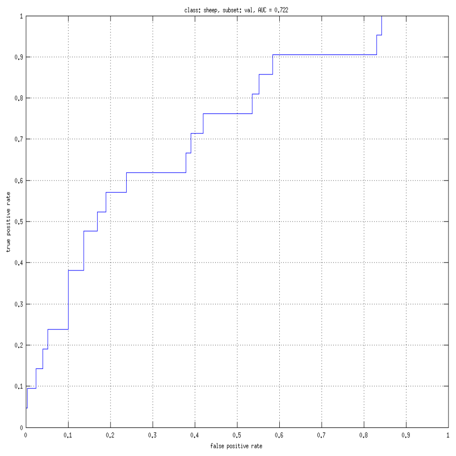

Figure 32: ROC curve for object class **sheep**.

Another major point came up while we were evaluating the results for each classifiers for different object classes. It is about the object class. As we know that, some object classes can show same perceived shape which may result in an ambiguous result. This ambiguity came out for *horse* and *human* classes. For this two classes, classification result is poorer than other classes. The following two results show the difference,

Classifier: SVC. **Dictionary size: 100**

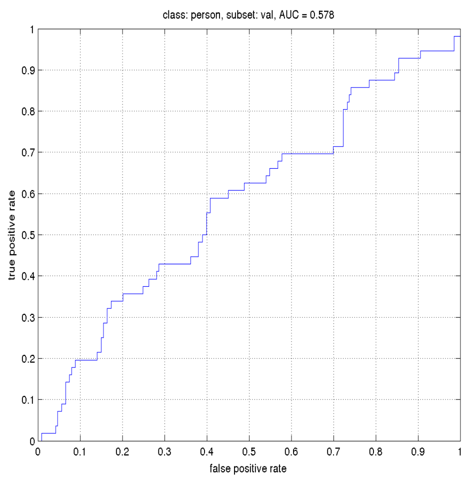

Figure 33: ROC curve for object class *person* for gray level image.

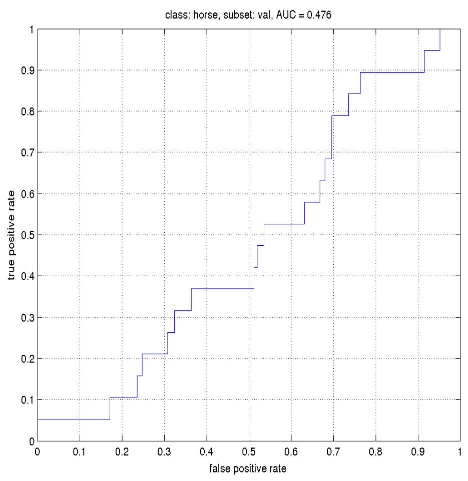

Figure 34: ROC curve for object class *horse* for gray level image.

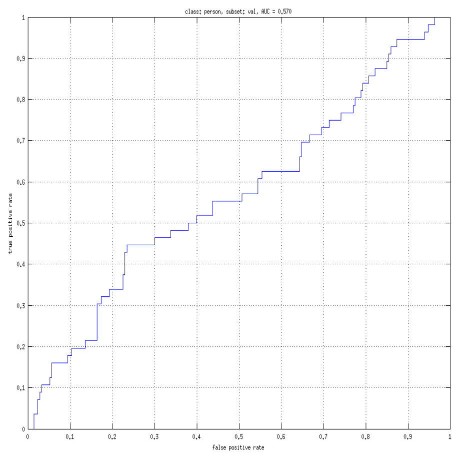

Figure 35: ROC curve for object class *person* for RGB level image.

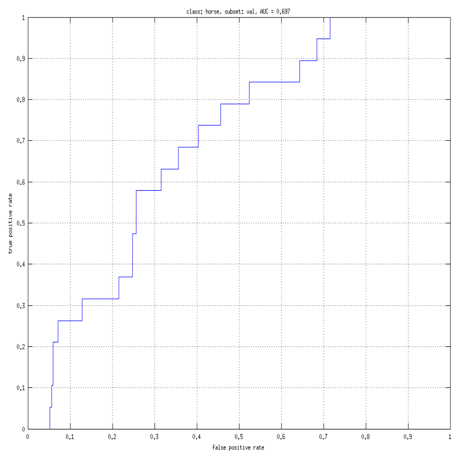

Figure 36: ROC curve for object class *horse* for RGB level image.

5 Organization and development of the coursework (tasks, time estimations and real dedication)

In this section, we present how we organized our project work. We had the lecture related to this project work before we started our project work. After assigning the project work, we got almost 5 weeks to complete the project work with report. We divided our time into the following time-frame.

1. First week: Studying and analyzing the problem. Run the given code over the large dataset.
2. Second week: Implementing the *SIFT* descriptor portion to get the feature vector.
3. Third week: Selecting from the list of classifiers which can we implement. Trying to implement at least one of them.
4. Fourth week: Implementing the classifiers and evaluate the result for each classifier.
5. Fifth week: Evaluating the result and writing the report.

We finished almost each of the tasks in time which led us to finish our final project timely.

6 Conclusions

This paper has described the implementation of the project work. The project work was a open one which led us using different classifiers. We evaluated the classifiers upon their results to chose the best one. Although we got good results using *Support Vector Clustering*, we can extend our implementation for some other classifiers. And also we can use other feature descriptors like: *HOG*. We can also add some other measures to evaluate the classifiers. Although this implementation can be extended for some other measures, our implementation is also offering nice results.

References

[1] Kristen Graumen and Bastian Leibe, *"Visual Object Recognition"*, 2011.

[2] Visual Perception, *Lecture6_Correspondence and Planar transformations*, 2015, Practical Exercise Instruction, UdG, Spain.

[3] `http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/sphilip/kmeans.html`

[4] `https://gilscvblog.wordpress.com/2013/08/23/bag-of-words-models-for-visual-categorization/`

[5] `http://www.hindawi.com/journals/isrn/2012/376804/fig1/`

[6] Chin-Teng Lin, Sheng-Chih Hsu, Ja-Fan Lee, Chien-Ting Yang, *Boosted Vehicle Detection Using Local and Global Features*, Journal of Signal and Information Processing, Vol.4 No.3, (2013).

[7] Asa Ben-Hur, David Horn, Hava T. Siegelmann and Vladimir Vapnik, *Support Vector Clustering*, Journal of Machine Learning Research 2 (2001).

[8] Medical Image Processing, *Lecture 5_Algorithm Evaluation*, 2015, Practical Exercise Instruction, UdG, Spain.