Report Project Datamining

Alper Çuğun
Leon van der Ree
Melle Sieswerda
Pascal Vreeburg
Oliver Verver

Contents

Foreword	3
Introduction	4
Problem Description	5
Research	6
Implementation	7
Features	8
Graphical User Interface	9
Relevance Feedback	10
Discussion	11
Teamwork	12
References	13
Appendix A: Testing runs	14
Apepndix B: Self evaluations	19

Foreword

You are currently reading the final report for the Project Datamining. This project has a strong foundation in the Media and Knowledge Engineering program. It focuses on applying skills from previous courses like ‘Image Processing’ and ‘Project Beeldverwerking’ and it provides the opportunity to apply the knowledge acquired in the parallel course ‘Datamining’.

The goal of the project is to implement a system, capable of image retrieval based on relevance feedback: the user presents a system with an example image and the system should retrieve (all) similar images from a dataset. When the user is not completely satisfied with the results, he has the option to tell the system which results are relevant. Based on this new information the system tries once again to retrieve similar images.

Introduction

Given a set of images humans generally have no problems making sense of the contents, recognizing the objects displayed and using associations to classify them in categories. Automation of all but the simplest of these tasks is daunting but very much needed when looking for specific types of images in large databases (e.g. the Internet).

We have built a system that is capable of retrieving images while taking into account the feedback of the user requesting the images. In the following chapters the workings of the system and the process to this implementation will be explained.

Problem description

The system retains a collection of images each of which is a member of a class (e.g. birds, buildings, flowers).

The user selects an image from the collection and inputs it to the system. The system then returns those images which it thinks are most similar to the given image (and its associated class).

After this first step the user gives feedback to the system marking those images which are from the correct class. The system uses this feedback to improve the results in the next iteration.

These iterations continue until an acceptable number of correct images has been returned or a specified maximum number of iterations have taken place.

Research

In researching this problem we found several systems already implementing most parts of our problem. Image analysis and classification seems to be a hot research topic.

As seen in [Iqbal], Q. Iqbal and K.K. Aggarwal developed an application called CIRES which offers the capability to query the system online with images and returns quite impressive results. Reading Iqbal's research [Iqbal99] inspired features based on histograms and gabor filters.

An approach to the relevance feedback problem was found in [Yu76]. It uses features to represent documents in an n-dimensional space. A query vector is built from an input-document and as the user provides the system with information on which documents are relevant and which are not, the query vector is updated to reflect this information. According to [Yu76] the formula used to update the query vector was originally proposed by Rocchio and Salton.

The Quicklook image search engine is another example of a system which already solves our problem. This system [Ciocca00] gave us a good pointer on Gaussian feature normalization and another failed approach to the relevance feedback.

Implementation

The problem can be divided into two main sub-problems:

1. Finding features that can distinguish the different categories
2. Finding a way to implement the relevance feedback

In the following thee sections we will discuss:

- The features extracted.
- The GUI used to interact with the system
- The relevance feedback engine

Features

The problem of finding good features proved quite difficult testing not only our technical aptitude but also our creativity. After a lot of trial and error we came up with the following features:

Variance: The purpose of this function is to show the distribution of the R, G and B colors. The variance function takes a color-image as input and returns an array with 3 elements containing the variance of the R, G and B channel of the input image.

Uniformity: The purpose of this function is to return a number, describing the uniformity between different parts of the picture. The uniformity function takes a color-image as input and divides every channel in 3x3 sub-images (nonants). After that, the mean of every part is then calculated and then the mean of every horizontal en vertical neighbor is subtracted. The mean of these absolute transitions is then returned.

Edge-method: The purpose of this function is to return a low an a high value describing the amount of fuzzy edges (low) and hard edges (high). The edgemethod function takes a color-image as input and combines its 3 channels to make a gray-value image. It then applies an edge detection filter and counts the pixels below an above a threshold, and returns those numbers divided by the total number of pixels.

Color distribution: One of features used for the recognition of the pictures is the color distribution in various color spectra. These pictures in these spectra are displayed in histograms using the diphist function. From these histograms the percentage of total amount of color was calculated over a number of six different bins, namely:

1 – 30	1 – 50	51 – 100	101 – 150	151 – 200	201 – 255

This distribution has been calculated for: the grayscale spectrum, the RGB spectrum per individual color and the chroma part of the HCV (Hue, Chroma, Value) spectrum. This component is composed of the difference between the greatest and smallest value of the RGB spectrum.

Gabor extraction: Textures in an image usually are a prime indicator for natural objects. This is a stark contrast to man-made objects which generally exhibit an artificial regularity in their surfaces and their edges.
The gabor feature was extracted using the dipimage gabor function to calculate the gabor response for different settings of standard-deviation, frequency and direction. For each different value the responses were summed and returned.

Blob count: For each color (red, green, blue, etc.) we are interested in the frequency and quantity of its occurrences in an image. For example to count the number of red blobs in an image the color values are processed according to the following formula and the resulting values are stored in a greyscale image:

$$\sqrt[3]{(RED * (255 - GREEN) * (255 - BLUE))}$$

After this the resulting image is thresholded (see images left) resulting in a value only on those locations which satisfy the requested color. These locations are labeled and counted which results in a frequency measure for a specific color.

Graphical User Interface

The Graphical User Interface (GUI) is an important part of our program, because in this assignment, the user has to select the images that are relevant for his or her query. We chose to use Java for the GUI, because the Matlab GUI development tool (GUIDE) is not suitable for serious GUI building.
We developed three Java classes, handling the three dialogues that we needed:

SelectImageDir

```
SelectImageDir.select( 'F:\Plaatjes' )
```

Displays a simple Java dialog to select the directory that contains the images. The argument is the directory in which the dialog starts, so we did not have to select the right directory every time during the testing phase.

SelectImageFrame

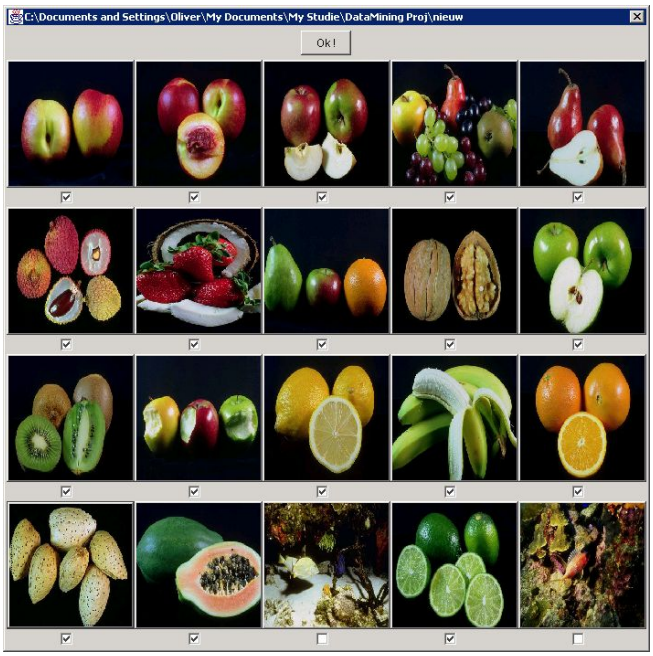
```
SelectImageFrame.getImage
( 'F:\Plaatjes' )
```

This frame (see right) displays thumbnails of all the images so the user can pick one of them as his request.
We have noticed that loading the 200 images takes a lot of time, so we have also implemented a caching system that saves the thumbnails to a file in the directory. On start-up, if this file is found, the class will use it; else, it will load all the images, create the thumbnails and writes them to the cache file. This way the application will start up faster on the next invocation.

SelectMultiImageFrame

```
SelectMultiImageFrame.getImages
( 'F:\Plaatjes', [14; 15; … ;
157] )
```

Displays the thumbnails (see right) of the 20 'relevant' images returned by the system. Beneath each image is a checkbox the user can mark to indicate the returned image is relevant. When the user presses the Ok button, the indices of the selected images are returned for further processing.



Relevance Feedback

The first method we tried was based on Rocchio's formula. It starts out by representing all documents as points in an n-dimensional space, the image-space D. The query-image, is represented as a vector, Q. After the initial run D is split into a set of relevant images and irrelevant images, D_r and D_{irr} respectively. It then updates the query-vector Q using the following formula:

$$Q^* = Q + \alpha \sum D_r - \beta \sum D_{irr}$$

After some experimentation this method was rejected. It didn't produce the results we'd hoped for. This algorithm uses a purely nearest neighbor approach where the query point gets bumped through the image-space at each iteration. This does not work if the classes are not well separated in the feature space.

The next method had an individual weight for each feature individually instead of for each image as a whole. Based on the relevant and the irrelevant images a feature-weight vector was updated to reflect the relevance or irrelevance of certain features for the current requested image.
The idea was to have the algorithm focus on those features which were important to the classification problem at hand and ignore features that were not important. This approach was promising but did not provide the desired results and left us stranded.

In the final method the user starts by selecting his initial query image out of all the images in the image folder. The system then makes a list of relevant images L_r (initialized containing the query image) and a list of irrelevant images L_{irr} (initialized empty).
For each iteration the system will loop through all the images in the system and rate them. This rating is determined by calculating the squared Euclidean distance of all images against the images in L_r and L_{irr}. The higher the resemblance of an image with an image in L_r, the higher a positive rating it will get. And the higher the resemblance of an image with an image in L_{irr}, the higher a negative rating it will get.
This way the system does not search new images from a single query point like in the previous two approaches, but it will use all the positive and negative images it knows to calculate a score. The ratings of an image with all the images in the L_r and L_{irr} lists are summed and the 20 images with the highest summed ratings will be returned.
When the user selects the images that belong to the sought class the system will add the selected images to L_r and it will add the images that were not selected to L_{irr}. After this the ratings are calculated again and this process repeats until the user is satisfied with the results returned.

Discussion

Discussion on the performance of the developed software,
Suggestions for improvement: What would you do differently if you had to do the project again?

The performance tests on the engine were done using an automated testing script. The script tested each image from our collection and counted how many iterations were needed to return 18 out of the 20 possible relevant images.

The results of these testing runs are summarized in the following table:

Label	Min. Iterations	Average Iterations	Max. Iterations
1 Birds	11	18.5	25
2 Underwater	3	8.5	29
3 Ice Landscapes	3	8.35	17
4 Flowers	4	8.9	17
5 Sculptures	4	14.55	46
6 Fruit	1	2.25	6
7 Ice flowers	4	15.95	26
8 Textures	4	6.95	20
9 Fractals	5	7.2	15
10 Buildings	16	25.9	42

The number of iterations mentioned in the table is Maximum/Minimum/Average of the number of iterations that were needed for all images in the class to reach the mark of 18 (90%) images. Performance graphs for each class can be found in the Appendix A.

We might have gotten better results if we also had tried different distance methods. Currently we have only used one (the Euclidean) and focused on other aspects of the implementation.
Using more sophisticated features to classify on would probably also improve results. This area was under-utilized because good and useful features are hard to come up with.

Team Work

We opted not to organize our team work using formalized team meetings. Our preference was to have all communication to proceed informally. This may have caused some miscommunications along the way but these were usually quickly cleared up again. The informal atmosphere also resulted in a more fun approach to the project where everyone acted according to their ability.

The general consensus is that the work put in by each team member was balanced. For more detailed accounts on how everyone experienced the process see Appendix B.

References

[Ciocca00] "The Quicklook image search engine" by G. Ciocca, R. Schettini

[Iqbal] "CIRES: Content Based Image Retrieval System" http://amazon.ece.utexas.edu/~qasim/research.htm

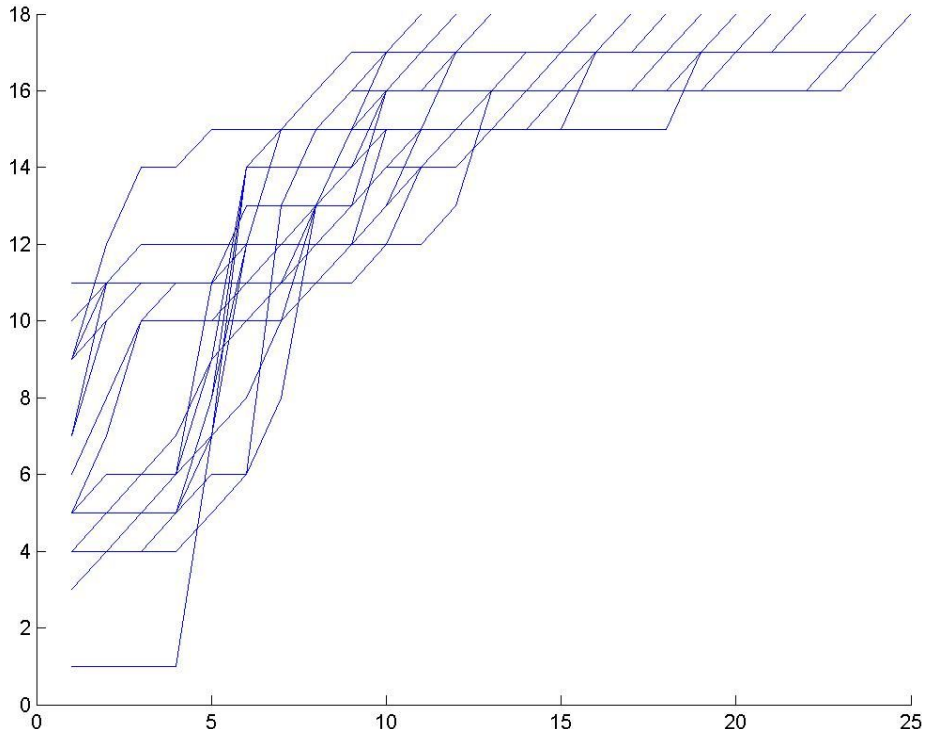
[Iqbal99] "Using Structure in Content-based Image Retrieval" by Qasim Iqbal and J. K. Aggarwal

[Yu76] "A Statistical Model for Relevance Feedback in Information Retrieval" by C. T. Yu, W. S. Luke and T. Y. Cheung

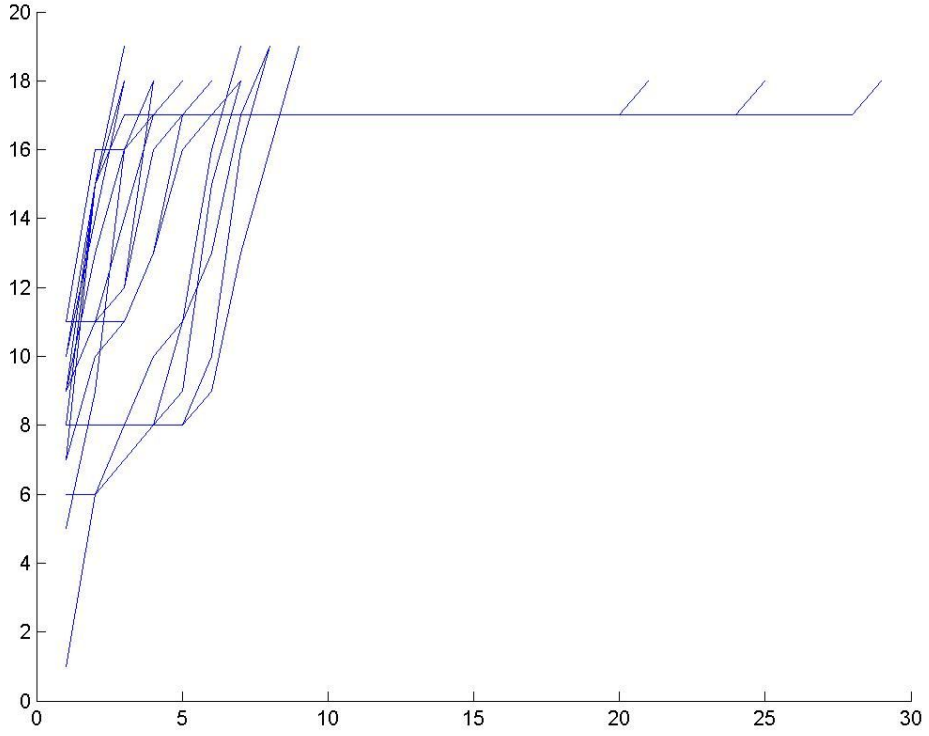
Appendix A: Testing Runs

In this section for each class of images the number of iterations needed to reach 18 relevant images is graphed. The number of iterations is plotted on the horizontal axis and the number of relevant images on the vertical axis.

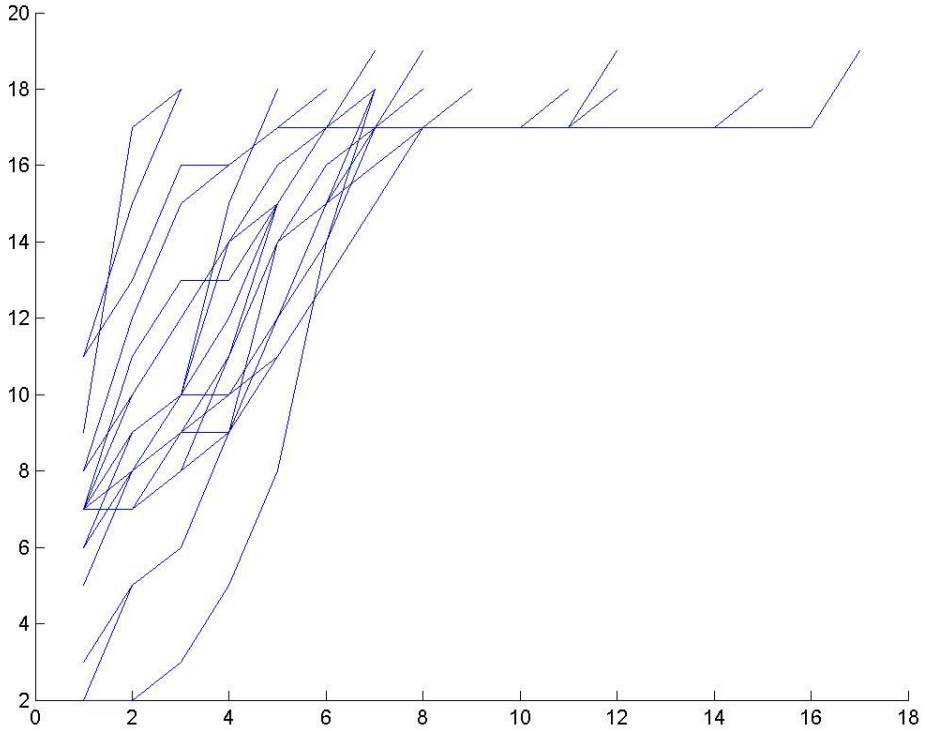
Label 1
Birds



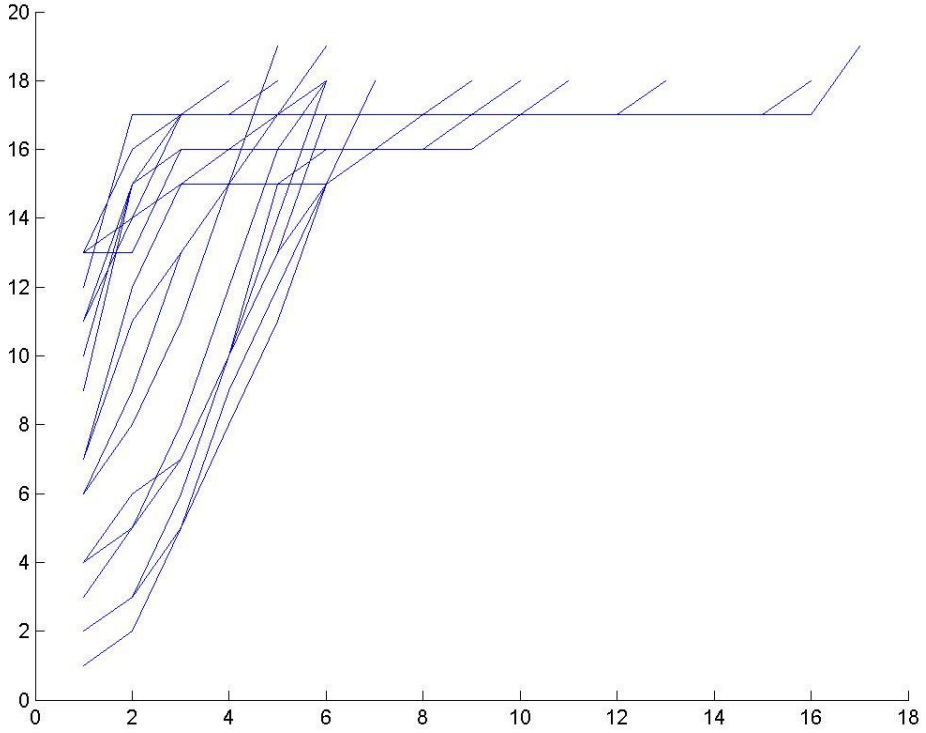
Label 2
Underwater



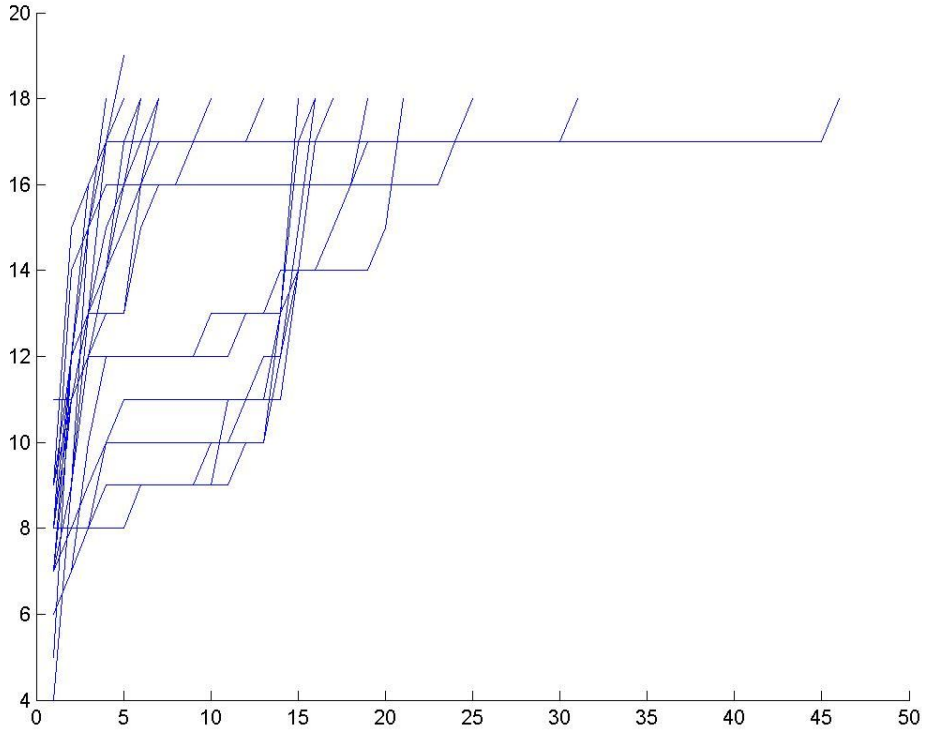
Label 3
Ice landscapes



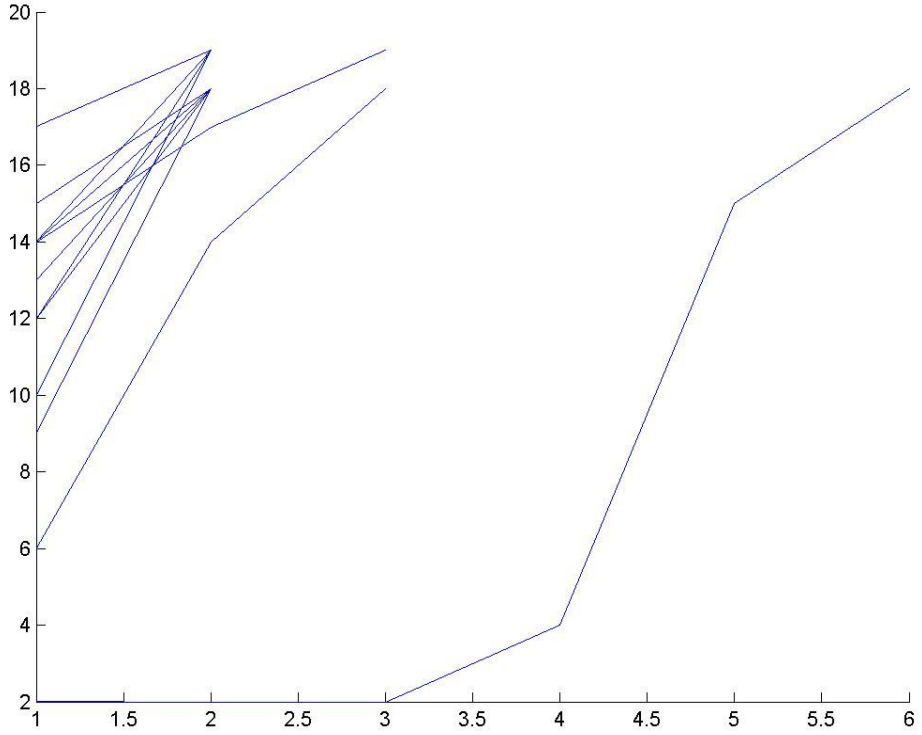
Label 4
Flowers



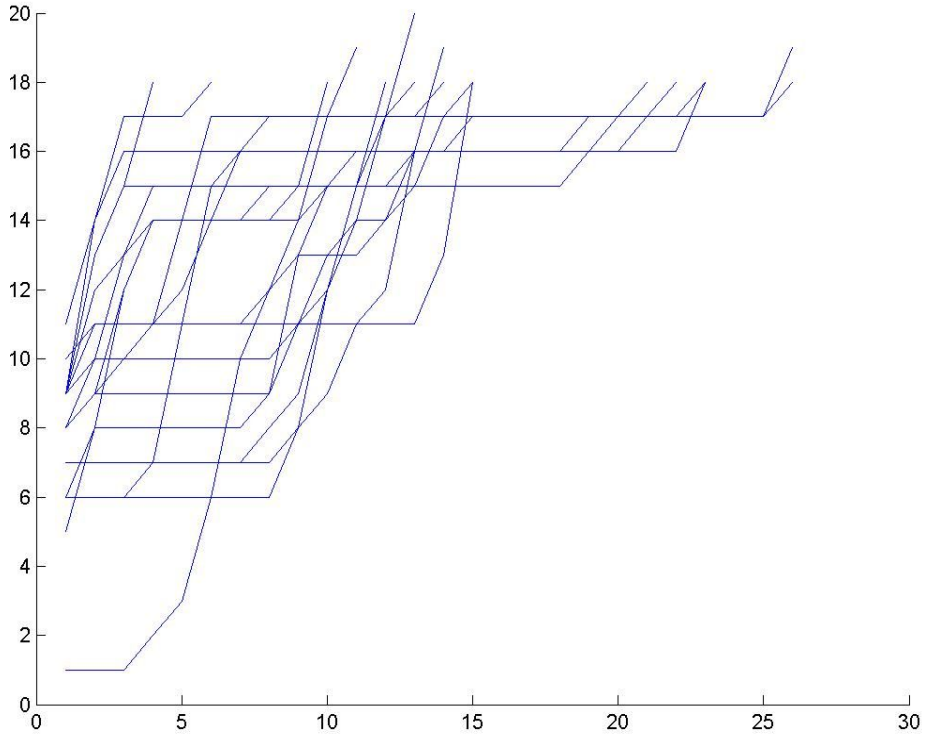
Label 5
Sculptures



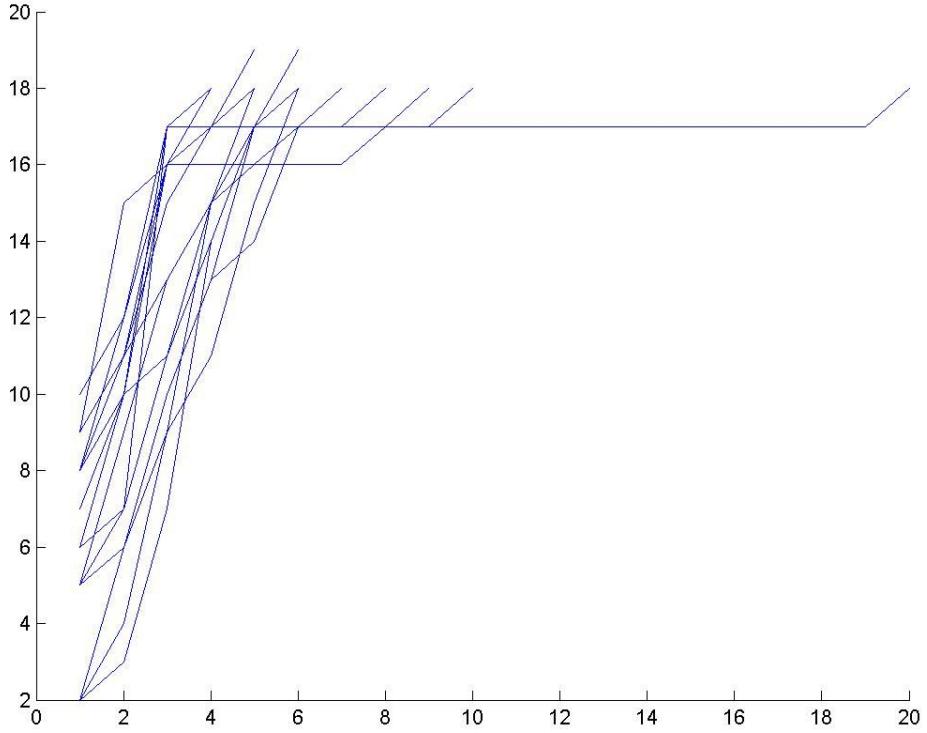
Label 6
Fruit



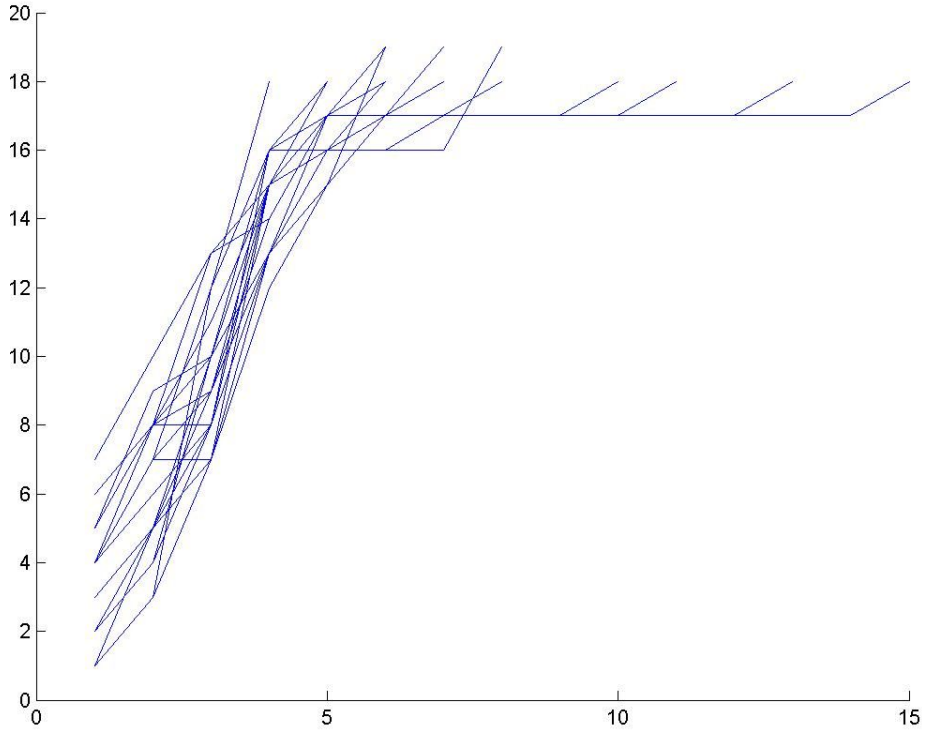
Label 7
Ice flowers



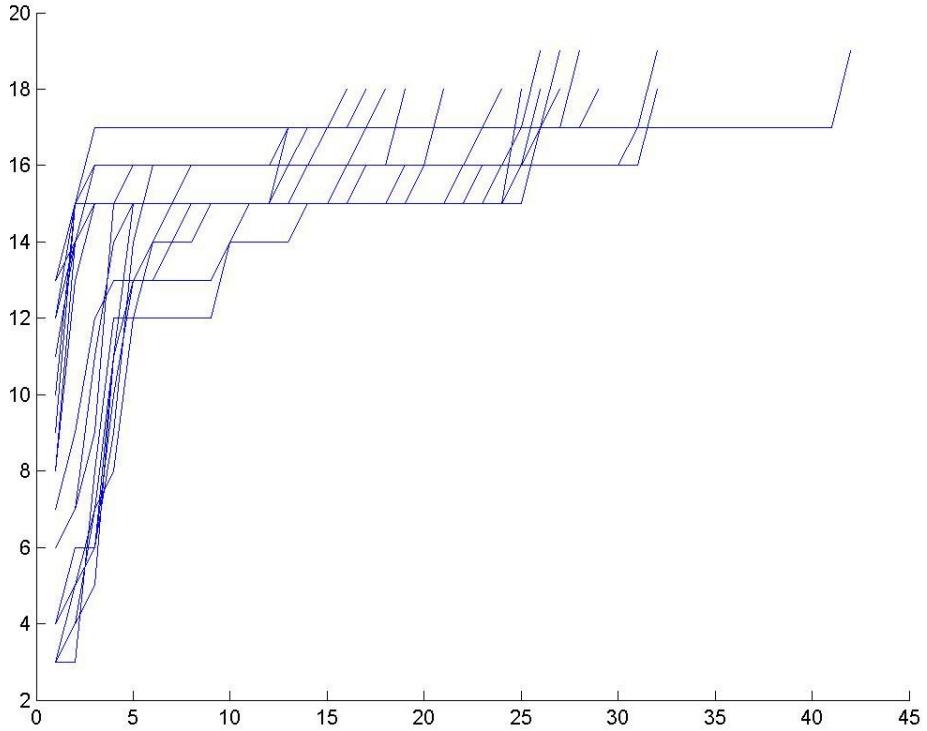
Label 8
Textures



Label 9
Fractals



Label 10
Buildings



Appendix B: Self evaluations

Alper
I learned to analyze images on their content and extract features out of them using Matlab. The implementation part was not very difficult; it was more difficult to come up with suitable and discriminating features.
The work I did on the relevance-feedback engine was difficult and drawn out over a long period of time. After several failed attempts an idea sparked by me was implemented by Oliver providing the final result.
The writing of the testing framework was fairly straightforward. This probably was because of the experience we had already gained using Matlab in other projects and during this one.

Leon
The teamwork during this project wend fine. We where all present during the scheduled hours (on some exceptions left aside) and that made it possible to immediately discuss certain problems we bumped up to, to solve them with each other. When we where working on some problems you could see how enthusiastic we all where, cause we even fantasized about our solutions during our way back to our homes, to come back the next day with new results and ideas.
The only thing noticeable was that you could see that Alper, Melle, Oliver and me where used to work with each other, and Pascal didn't give as much feedback on what he was working on as we did to each other. This may have lead to some misassumptions on his work, but I think that after all everything worked out fine.

During this course I've learned a lot about how to create and use datasets for solving problems. And how useful and powerful PCA and clustering techniques are to analyze and simplify large datasets and get more information out of them.

Melle
Overall the project was a success. We developed a system that works as requested, there was no real time pressure (apart from writing the final report, perhaps), and there where no implementational problems that couldn't be solved in time. The only thing that could have been better was the project management. This, however, was a choice we made in the beginning. When we started this project some of us (including myself) felt it was not necessary to have regular meetings. After all, we knew each other pretty well and had worked together on different projects. Having regular meetings, meeting minutes, a chairman and a secretary felt a bit like shooting a bird with a bazooka. As said earlier, the project was a success, and generally teamwork was quite good, but not having meetings sometimes created confusion about who was doing what. If the project had been any bigger or more complex, this might have become a problem.

Oliver
I have learned that Java classes can be very easily executed from Matlab.
I have learned the global idea of representing objects (images in this case) as points in an n-dimensional space using the features as their coordinates, and how to calculate the distance between them. I have played with some image filters and learned how they can be summarized to one or more numbers, so you can use them as image features.

Teamwork went ok, although it must have been a bit harder for Pascal, because the four of us (Alper, Melle, Leon and I) already knew each other, also from previous projects.

We did not have regular meetings; everybody was just doing his thing and we felt we had enough time to finish this project.

Pascal

In this project I learned to perform some basic Data-mining techniques on a relative small amount, but very specific data. This data was generated by us by using various image processing techniques. Due to this data gathering, I learned more about applying these image processing techniques as well as being able select which features might be important and/or useful to know for the end result.

There was some confusion for me in the beginning of the project about the nature of the assignment, in other words I didn't have a clear view of the assignment. I had mixed both the different assignments up in my mind. And because of this I went about my part in the wrong way at first, but after seeing the data generated by other team members I knew where I had taken a wrong turn and corrected my errors.

The team work went quite well in my opinion, there were no real communication errors as far as I know and everyone has contributed to the end result.