



**DUBLIN CITY UNIVERSITY
SCHOOL OF ELECTRONIC ENGINEERING**

**An investigation of eye gaze tracking utilities in
image object recognition**

Sergi Imedio Pereira
August 2014

**BACHELOR OF ENGINEERING
IN
ELECTRONIC SYSTEMS**

**Supervised by Xavier Giró, Noel O'Connor
and Rami Albatal**

Acknowledgements

I would like to thank my supervisor Xavier Giró for his guidance in this project. Thanks are also due to Noel O'Connor and Rami Albatal in DCU for supporting this work to date and attending my queries about this project while I was in Dublin and also back in Barcelona. Eva Mohedano and Irene Gris also deserve my kindest thanks for their support during the development of this work. Finally I would like to express my appreciation to Dr. Derek Molloy for preparing the original Transfer Report template, from which this document is hacked.

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the referencing guidelines found at <http://www.library.dcu.ie/citing&refguide08.pdf> and/or recommended in the assignment guidelines.

Name: Sergi Imedio Pereira

Date: 17/08/14

Abstract

Computer vision has been one of the most revolutionary technologies of the last few decades. This project investigates how to improve an image recognition system (image classifier) using a not very exploded technology; eye gaze tracking. The aim of this project is to explore the benefits that this technology can bring to an image classifier. The experiment that is set in this project is to build a dataset with an eye tracking device and (using different sized cropped parts of the image based on the eye tracking data) see how the performance of an image classifier is affected with these images. The results are interesting. Since smaller images have to be processed by using this method, the system is more efficient. Regarding the performance, it is very similar to the one obtained without using any eye tracking data, so it is arguable to state that it presents an improvement, and opens new directions of investigation for future works.

Table of Contents

ACKNOWLEDGEMENTS	II
DECLARATION	II
ABSTRACT	III
CHAPTER 1 - INTRODUCTION	1
1.1 THE PROJECT.....	2
1.2 EQUIPMENT AND SOFTWARE.....	3
1.2.1 The Tobii Glasses	3
1.2.2 Tobii Studio Software	3
1.2.3 Languages used and Pyxel Project.....	4
1.2.4 Support basic equipment	5
CHAPTER 2 - TECHNICAL BACKGROUND (CHECK APPENDED SURVEY)	6
CHAPTER 3- DESIGN OF THE SYSTEM	7
3.1 DATASET RECORDING.....	7
3.2 FRAME EXTRACTION	7
3.3 FIXATIONS DETECTION	9
3.3.1 Eye tracking data conversion	9
3.3.2 The approach.....	9
3.4 DATA ANNOTATION.....	11
3.5 IMAGE CROPPING BASED ON GAZE.....	12
3.5.1 Crop Models	12
3.5.2 Issues with Image Borders.....	13
3.6 FEATURE EXTRACTION, AGGREGATION, CLASSIFICATION AND EVALUATION	13
CHAPTER 4- IMPLEMENTATION AND TESTING OF	14
4.1 THE DATASET.....	14
4.1.1 Data acquisition	14
4.1.2 The Ontology	16
4.1.3 Data partitioning	17
4.2 THE CLASSIFIER.....	19
4.2.1 Pyxel Overview.....	19
4.2.2 The Evaluation Module	21

4.2.3 <i>The Eye Tracking Modules</i>	23
4.3 THE METRICS	24
CHAPTER 5 - RESULTS AND DISCUSSION	25
5.1 RESULTS TABLES AND INTERCLASS PERFORMANCE	25
5.2 EFFICIENCY VERSUS PERFORMANCE RESULTS	30
CHAPTER 6 - CONCLUSIONS AND FURTHER RESEARCH	33
6.1 CONCLUSIONS.....	33
6.2 FUTURE WORK AND RESEARCH.....	35
REFERENCES	37
APPENDIX 1	38

Table of Figures

FIGURE 1.1. AN IMAGE FROM A VIDEO SEQUENCE WITH THE GAZE POINT PLOTTED ON ..	1
FIGURE 1.2. TOBII GLASSES.....	3
FIGURE 1.3. TOBII STUDIO ENVIRONMENT.....	4
FIGURE 3.1. HIGH LEVEL BLOCK DIAGRAM OF THE PROJECT.....	8
FIGURE 3.2. SCHEME OF THE FIXATION DETECTION PROCESS.	10
FIGURE 3.3. CAPTURE OF THE ANNOTATION PROCESS WITH QADIA TOOL.	11
FIGURE 3.4. THE DIFFERENT CROP MODELS USED.....	12
FIGURE 4.1. STORES RECORDED TO BUILD THE DATASET.	15
FIGURE 4.2. DETAILS OF THE RECORDINGS.....	15
FIGURE 4.3. ONTOLOGY OF THE SYSTEM.	16
FIGURE 4.4. DATA PARTITIONING GRAPHIC.	18
FIGURE 4.5. IMPLEMENTATION OF THE PARTS IN THE IMAGE CLASSIFIER.	19
FIGURE 4.6. COHERENT AND WRONG DISPOSITION EXAMPLES FOR EVALUATING.	21
FIGURE 5.1. PRECISION RESULTS TABLE.....	27
FIGURE 5.2. INTERCLASS PRECISION RESULTS GRAPH	27
FIGURE 5.3. RECALL RESULTS TABLE.....	28
FIGURE 5.4. INTERCLASS RECALL RESULTS GRAPH.	28
FIGURE 5.5. F1-SCORE RESULTS TABLE.....	29
FIGURE 5.6. INTERCLASS F1-SCORE RESULTS GRAPH.	29
FIGURE 5.7. AVERAGED METRICS FOR EACH WINDOW SIZE.	31
FIGURE 5.8. TIME OF COMPUTATION FOR EACH WINDOW SIZE.....	31
FIGURE 5.9. AVERAGED PRECISION VS. TIME.....	32
FIGURE 5.10. AVERAGED RECALL VS. TIME.....	32
FIGURE 5.11. AVERAGED F1-SCORE VS TIME.	33

Chapter 1 - Introduction

One of the most important branches of computer vision is Image Recognition [1] (be it object, faces, events, etc.), since it provides all kinds of interesting and useful multimedia applications as surveillance features or automatic image annotation, which with the current amount of data that there is in the net becomes really important. It is a constant challenge to improve the algorithms contained in the image recognition process, both in efficiency (speed) and accuracy. In this 4th year electronic engineering project this challenge is aimed using a specific technology; eye gaze tracking (EGT).

Eye Gaze Trackers [2] are devices that are capable to measure and track the gaze of the human visual system (HVS). Although there have always been many different applications with this technology (marketing investigations for example [3]), the fact that it has improved dramatically over the last years has made possible to use it for even more things. An example of an EGT device is the Tobii Glasses [4] (which are later further described). This glasses record video while some user is wearing them and tracks his or her gaze.

Using this tool we investigate if this technology can provide some improvement in an Image Recognition system.



Figure 0.1. An Image of a video sequence with the Gaze point plotted on

1.1 The project

***Important note:** A literature survey was made a month after beginning this project. This survey is appended to this document and it contains a section that deals with the project goals and outline. These goals have changed and evolved during the development of the work, however it is interesting to see the section and it contains some useful information as well.*

This 4th year Electronic Engineering project aims to investigate if eye tracking data can be useful to improve the performance and/or efficiency of an image object recognition system (classifier). To do so, a data collection must be done first with an eye tracking device (in this case a video data collection due to our specific device).

The context of the data in this specific project is retail, since it is an interesting one to build applications on, as well as easily accessible to collect data containing it. The idea is to do the video recordings in several stores, and make sure that a small set of different products are recorded in each of these.

Once the collection of this data is finished, a multiclass object detection system must be build according to the products contained in the recorded data.

At this point, the main characteristic goals of the project are confronted; there are two main tasks implemented using the eye tracking data that the device provides along the recordings:

- 1- See how the performance and/or efficiency of the system change when using small cropped parts of the images (centred in the eye tracking points) with different sizes instead of the complete images.***
- 2- Implement an algorithm to detect fixations in time using the variations of this gaze data, so that the images that contain a fixation are filtered as images of interest from the video stream, achieving smaller amounts of data.***

In the next chapters it is explained what is the approach to reach these goals.

1.2 Equipment and Software

1.2.1 The Tobii Glasses

The Tobii Glasses (figure 1.2) are a wearable device that records video and shows exactly what the user is looking at in real time [5]. They consist on the glasses and a recording assistant to handle the data and the processes.

The main features of this device are the following:

- It records *MJPEG2000* video data at *30fps*.
- The resolution of the video is 640x480.
- It works alongside private software called *Tobii Studio*.



Figure 0.2. Tobii Glasses

1.2.2 Tobii Studio Software

The Tobii Studio Software is used to handle eye tracking data once it has been recorded by an eye tracking device (the glasses in this case). Among many other things, it allows exporting the videos with and without the plotted gaze data, as well as the gaze data by itself which is later accessible to read with any file editor.

This gaze data basically consists on all the parameters and details of the specific recording and device (resolution, date and time, etc), and then the coordinates for each of the points and its corresponding timestamp. With this information it's possible to match each of the frames extracted in the videos with its corresponding eye tracking data (both of the coordinates in the image). The timestamp is given in milliseconds, and the coordinates go from 0 to 640 and from 0 to 480 for dimensions X and Y of the picture, respectively

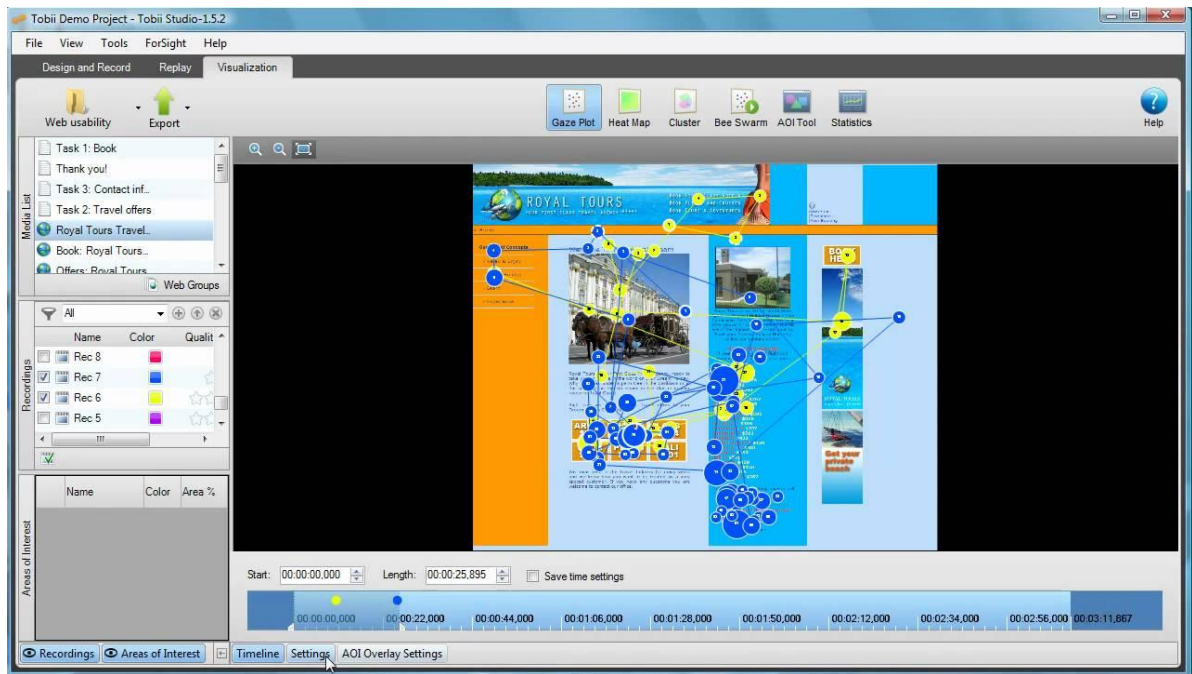


Figure 0.3. Tobii Studio Environment

1.2.3 Languages used and Pyxel Project

The programming language used to implement this project is mainly Python due to the previous experience of some of the supervisors in this language as well as some previous work related with computer vision programming. Shell scripts have also been implemented to deal with some issues not directly related to the core of the project as it is explained later.

Since there were several people in our research group working in image recognition matters, the option of creating a common project to share the work with everyone in the group was pointed. That is how the *Pyxel* project started, and it is a project that belongs to the image processing group at UPC in Barcelona. In this *Pyxel* project, there was an unfinished machine learning (classifier) system that belonged to another project called MediaEval. Although a very important part of this project has been to adapt this system to the requirements for this specific data and finishing the parts that were not implemented yet, there have been parts of the project that directly use the work that belongs to some other projects in the *Pyxel* group, that is the reason why it must be included in this section as part of the used software and equipment.

1.2.4 Support basic equipment

This project has been developed from two different localities (Dublin for February – May period and Barcelona for June-August period). The machines used to work in this project have been a PC with Ubuntu in Dublin and a laptop (Windows) in Barcelona. When in Dublin, all the needed data was in the same local computer, but in Barcelona we connected to a remote server using the Nomachine software, hence all the data was stored there and operated remotely from a personal laptop.

The Nomachine software connects to a remote server and provides a visual interface so it is possible to navigate freely through it as if it was a local machine. It was not only used to access the data but also to execute the computational processes related to this work.

Chapter 2 - Technical Background

About this chapter:

This chapter is fully covered by the work made through the literature survey appended in this document. In it, it can be found an exhaustive research on the state of the art, as well as other details about the project.

Chapter 3- Design of the system

Since this project is not an application but a research work, and consequently there are no interfaces kind of elements to design, in this chapter it is analyzed the design of the system in terms of the approach that has been carried out in a high level, to investigate and aim the goals that are set in the introduction chapter.

To understand the complete process, the project is illustrated in the diagram in figure 3.1. The different tasks are separated into blocks and ordered in the complete scheme. In the following chapters, all the blocks are individually explained.

3.1 Dataset Recording

This block provides the raw data to work with; the different video streams and the attached eye gaze tracking metadata. To complete it, the number of videos to be recorded was first defined (based on the number of different stores close to the campus area) and, while recording the videos, it was intended that the same products appeared as much as possible in each store. Before starting this process, the Tobii Glasses had to be calibrated as stated in the instruction manual.

The details of how this block is completed are totally explained in the chapter 4.

3.2 Frame Extraction

After importing the videos recorded with Tobii Studio software to the working personal computer, the frame extraction was done. To do so, the ffmpeg tool was used. Ffmpeg tool is available on the basic Linux package and it offers different usage options; the most important being the number of frames extracted per second in the process. In this case, all the frames were extracted from the video streams (30 frames per second) although not all of them were eventually used for the process of training and testing the classifier. The filename of each extracted frame was "*VideoX – Y.jpeg*", X being the number of the video and Y the sequential number of the specific frame in that video.

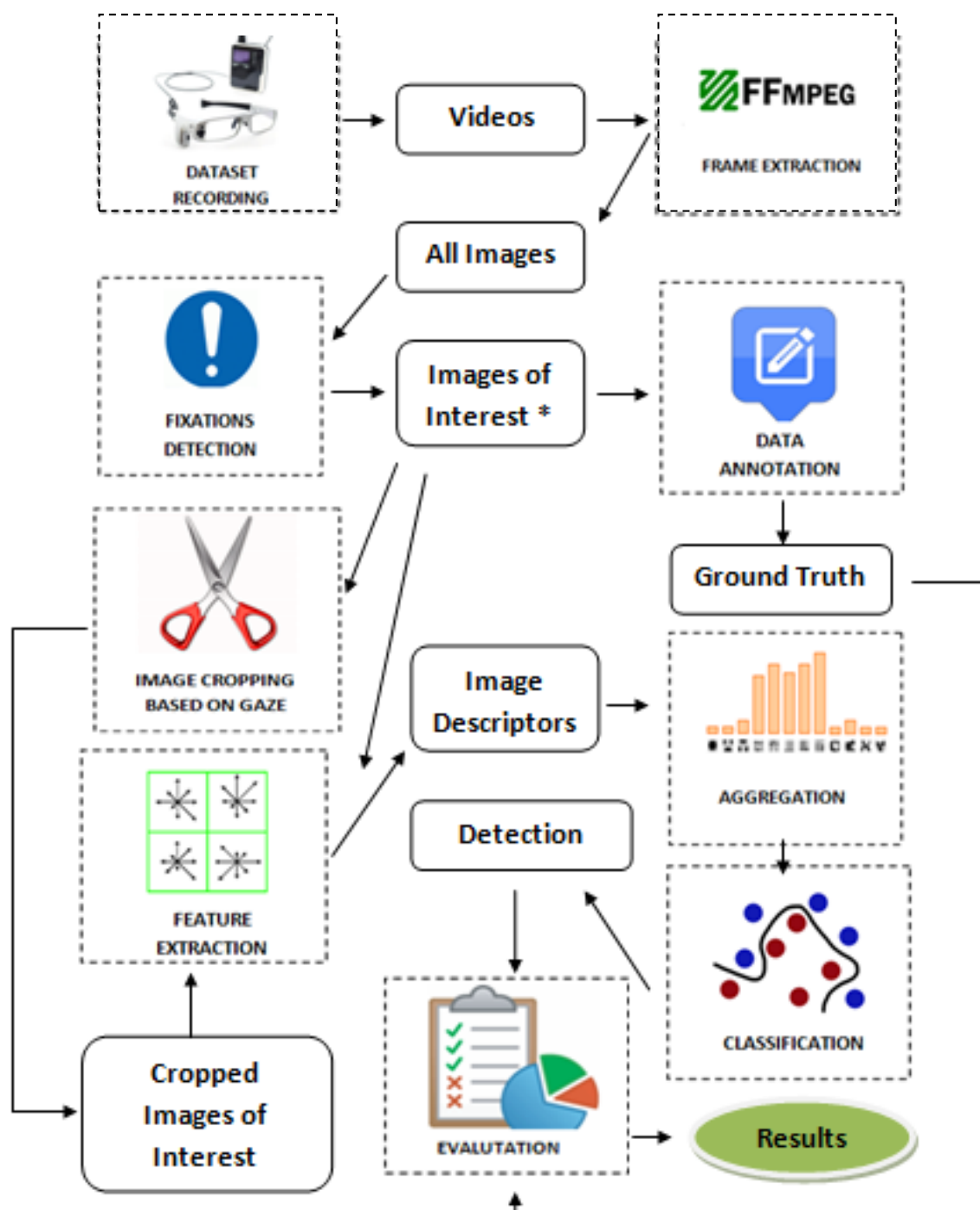


Figure 3.1 High Level Block Diagram of the Project

**Images of Interest from the Fixation Detection block were not used in the end*

3.3 Fixations Detection

This block represents one of the initial goals that were stated in the project. It was first necessary to clean the gaze data and then design the approach to implement the algorithm.

3.3.1 Eye tracking data conversion

The eye tracking data that was attached to each of the videos recorded was readable by any regular file editor. As it is explained in chapter 1, there is a lot of information in it that is not relevant for implementing the desired algorithms. The basic core of this data was a timestamp and two coordinates that represent the exact instant on the video stream and the point estimated by the gaze tracker, respectively. It was necessary to build a system that generated a new regular text file containing just the chronological number of each frame in the stream (thus meaning, converting the timestamp into a N frame position) followed by the two gaze coordinates. To convert the timestamp into a frame number, the frame rate (30 fps) had to be multiplied by the timestamp in seconds, and then rounded to an integer.

3.3.2 The approach

The approach used to detect fixations once the data has been filtered and aligned has been totally extracted from *Museum Guide's* paper [5], which is discussed in the literature survey and deals with the same problem. However, the code implementation has been made as an own work and beginning from scratch. The main idea of the approach for this problem is shown in figure 3.2. There are two adjustable important parameters; the noise margin and the duration. The noise margin works as a threshold in terms of distance, while the duration determines for how long the consecutive coordinates have to remain similar in order to be considered as a fixation. The first gaze coordinates are read and stored. Once there is a reference pair of coordinates, the next ones are sequentially read from the gaze cleaned file. For both of the coordinates, x and y, if the difference (in distance) between the reference ones and the new ones is below the noise margin parameter a counter begins. If this counter keeps on being incremented for the next new frames and reaches the duration parameter, it is considered a fixation and the filename of the frame is written in a separate text file.

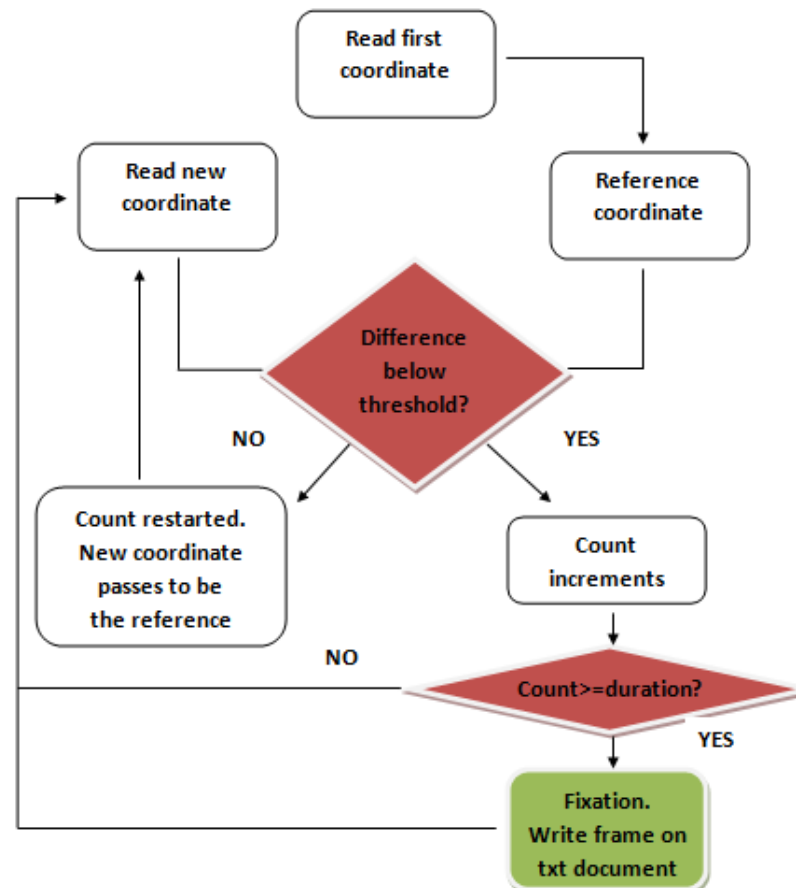


Figure 3.2. Scheme of the fixation detection process

It is important to remark that this works separately for each of the coordinates. However the noise threshold condition must be accomplished for both coordinates in order to increment the counting. With this process, a text file that contains only images with fixations is achieved. These images with fixations can be considered as images of interest, since it is sensible to assume that the user was firmly staring at some element, and thus the images that contain that element will be useful for the process of the classifier development.

3.4 Data Annotation

This task was also necessary in order to use the collected data for the classifier. With annotating the data the process of building a complete dataset was finished, hence it can be used for any other image classifying system.

To do this data annotation Qadia tool was used, which has been developed by Brian Moynagh at the Insight lab in DCU. For each class in the ontology of the system (which is explained in chapter 4) all positive appearances were manually annotated with this tool.

The resulting output of this process were several text files (one per class in the ontology) with the filenames of the images in it that were annotated as belonging to that particular class.



Figure 3.3. Capture of the annotation process with Qadia tool

3.5 Image Cropping Based on Gaze

This was the most important block in terms of design to achieve the goals of this project. The most important things regarding the design of this part were firstly the model of the crops used and on the other how to deal with problems of image borders.

3.5.1 Crop Models

The crops were first of all always centred on the gaze point (the one formed by the coordinates in the gaze data file) for that specific frame, and also were designed as square crops and with different sizes. For each of these sizes, the system would give different results, so when comparing them it would be possible to see which variation provided the best results. It is important to clarify that also one experiment (called ground) is done without cropping the images so it is also appreciable how the classifier performs with normal images and compare the results with the other variations. There were a lot of different sizes to test at the beginning. However, when extracting the features of these cropped images some issues appeared with the small sizes, hence they had to be taken out of the process. In the end, the selected sizes (in pixels) were; 400, 350, 300, 250 and 200 square crop models. It is good to insist that the original size of the images is 640x480.



Figure 3.4. The different crop models used

3.5.2 Issues with Image Borders.

When the gaze estimated points were positioned close enough to the borders, part of the cropped images surpassed these and included black zones where the image reached its ends. Since it was not desired to have these black parts in the images that were going to be used for the classifier, the solution was to estimate first if the crop was going to surpass the borders and in that case get the immediate part of the image next to the border with the specific crop size, even if the gaze point was not centred on it.

3.6 Feature Extraction, Aggregation, Classification and Evaluation

These last blocks are discussed together because there is not much design on them. The role they play in terms of implementation of the system is discussed in the next chapter. The quick important points to touch on regarding these blocks are the following:

- **Sparse SIFT** was used for the block of feature extraction. Although there was the initial idea of using other kinds of varieties like colour descriptors or opponent SIFT, in the end the choice was regular sparse SIFT.
- For the aggregation block, BoF (**Bag of Features**) was the method used.
- **SVC** was the support vector machine variation used to build the system. The reason for this decision is that it supports multiclass schemes like the one required for this specific project.
- **The evaluation block** compares the ground truth with the labels given as an output of the classifier. At the output of this block the performance results are achieved.
- The **visual vocabulary** used in this process has not been considered a separate block in this diagram. However, it is important to mention that it was generated with **1200 images** and **1024 different visual words**

Chapter 4- Implementation and Testing of ...

4.1 The Dataset

In this work, the dataset has an especially important role. Although there exist some previous datasets recorded with an eye tracking device, the fact of having the eye tracking Tobii Glasses was an important factor to consider as well, hence the dataset used ended up being recorded as part of this work. Being able to record the dataset to use it later for this project made the whole experience more complete, and also allowed to discover how the eye tracking recording device actually works

In this section, the details of the procedure of recording the dataset are explained.

4.1.1 Data acquisition

First of all, it is important to clarify that the resulting material of recording video with an eye tracking device depends on the particular user that carries the Tobii Glasses. This dataset has all been recorded with the same one user due to the lack of time and voluntary users. Although this does not mean that the data is not good (because there is the same ‘gaze behaviour’ in every video), it does not provide the averaged and balanced advantages of having recorded it with, for instance, six different users (one per store).

In order to achieve a solid dataset in a retail context, several stores have had to be included. Doing so, images from the same products are sure to be taken from different perspectives, distances and illumination conditions; hence the resulting dataset is more reliable and realistic. As well as featuring video content from several different stores, the recordings in each of these have different lengths due to the high diversity of dimensions and products.

An important constrain that has been confronted when recording the data has been to carry an expensive device on the street, thus meaning the stores have been chosen all close from the research lab where the Tobii Glasses are kept (as visible in figure 4.1). However the chosen places offer different conditions regarding illumination, product settling, and dimensions.

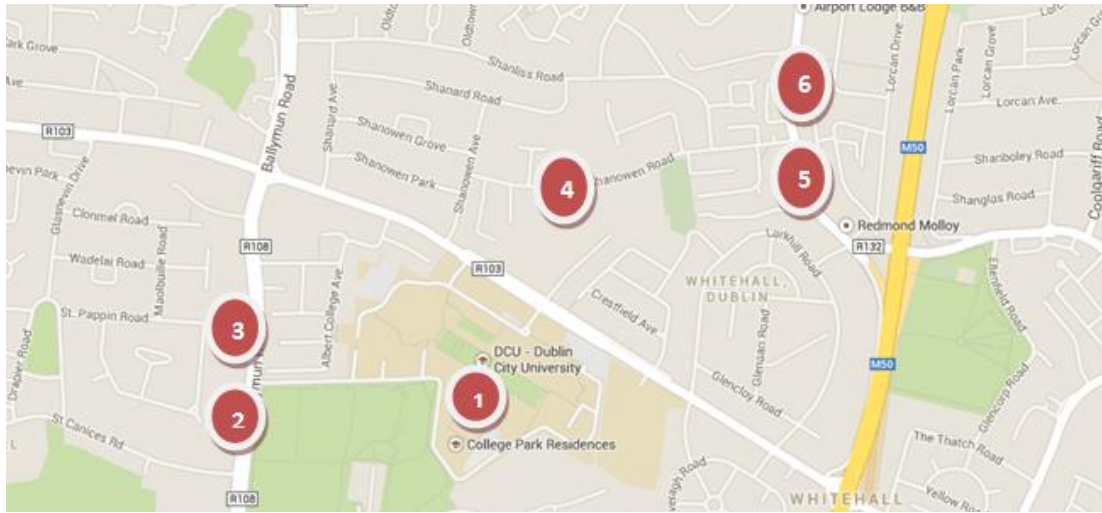


Figure 4.1. Stores recorded to build the dataset

In essence, the recorded dataset contains 6 videos, each of them made in a different store, making an overall of 42 minutes of eye tracking video data.

The more specific details of each of these recordings are visible on the table in figure 4.2.

The recordings try to focus always on finding the same products in each of the stores (these products are afterwards discussed and analyzed in another section) so that the dataset offers a bounded and solid range of products to train and test the classifier.

Video Id	Place	Duration (minutes)
1	DCU Campus Spar	06:16
2	Glennon's Eurospar	06:55
3	Ballymun Centra	05:37
4	Shanowen Road Spar	05:57
5	Maxol Station Swords Road	04:14
6	Swords Road Tesco	13:08

Figure 4.2. Details of the recordings

Important Note: Although video number 6 (Tesco) was recorded and exported with Tobii devices and software, its images are not used afterwards for training or testing the classifier because there were problems with the annotation tool for such an amount of frames in the same video.

4.1.2 The Ontology

As it is explained in previous chapters, the system this project aims to implement is a multiclass image classifier in a retail context. In other words, the desired goal is to get a system that receives an image as an input and gets a label that identifies a product as output. To do so, first of all it must be decided how many and which products are the ones that are going to build this system, in essence, the ontology. The ontology used is the one pictured in figure 3.1.

There are five different products in this ontology:

- Plastic 2 litter bottle of coke (Coca-Cola brand)
- Potato chips bag (Hunky Dorys brand)
- Cereal box (Special-K brand)
- Potato chips tub (Pringles brand)
- Bananas



Figure 4.3. Ontology of the system

As visible in figure 3.1, the structure and justification of the classes are the following:

- Five different products and a ‘trash’ category, ending up with an overall of 6 different classes.
- Between the products there is a considerable variety in shapes and sizes, hence it can be considered a balanced ontology.
- With this number and type of products, the performance results are very likely to be different depending on the product, as well as easy to analyze and discuss.

It is important to mention that not all the videos of the dataset contained each of these products. However the products chosen are still the ones that appear most through all the dataset.

To sum up this section, the system that is implemented in this project is set to classify an input image as any of these 5 products or as trash (none of them). Although this ontology was not chosen until the recordings were finished (more details in the *data set collection* section), it was planned to have ontology of less than 10 products that offered differences in shape and size.

4.1.3 Data partitioning

Once the dataset is acquired and the ontology of the system is defined, the next decision is to decide the amount of data images to use for both stages of training and testing the classifier. From the 5 videos from the dataset used (containing exactly an overall of 51.960 frames), a balanced selection of images has been picked to use for the classifier.

From the recorded dataset, video number 2 is used for testing the classifier, while all other videos (1, 2, 3 and 5) are used for the training stage.

Therefore, for each class contained in the ontology, 200 images of that product evenly spread over videos 1, 2, 3 and 5 are used for the training stage of the classifier, in other words, to define the models of each of the classes. On the other hand other 85 images of each class in the ontology from video number 2 are used for the testing stage.

The train/test ratio for our classifier is then 70% / 30%.

In overall, the amount of images used for the classifier process is:

$[6 \text{ categories} * (200 \text{ training} + 85 \text{ test})] = 1710 \text{ Images}$. (1200 for training, 510 for testing)

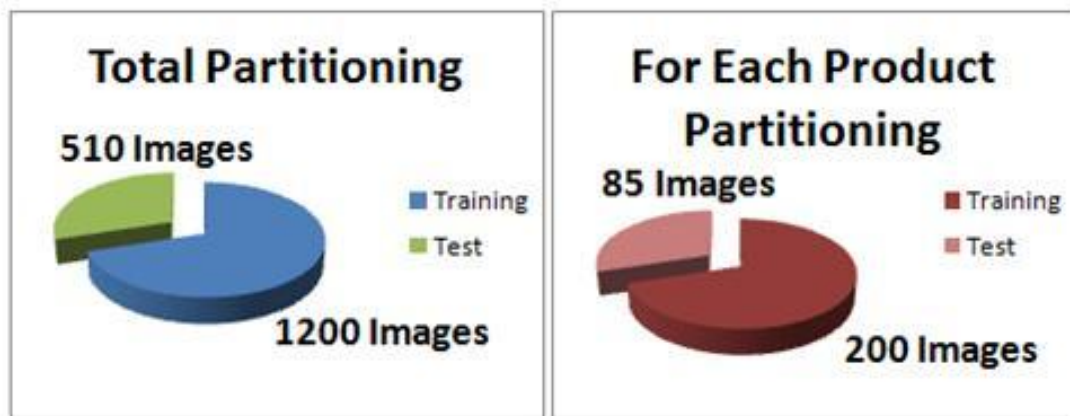


Figure 4.4. Data Partitioning Graphic

4.2 The Classifier

In this chapter, the implementation of the classifier algorithm is explained. This algorithm is composed of several parts and modules. Not all of these have been implemented within this project; only a few have been newly created or briefly modified/improved. The rest of them are part of a shared work called Pyxel and have been implemented in the works of other researchers. Hence, an overview of Pyxel and the image classifier is given, and finally the parts of it that have been implemented in this project are more in detail explained.

4.2.1 Pyxel Overview

The image classifier system that was part of Pyxel before this project started its development was not yet completed. In figure 4.5 is shown a diagram of the different modules which this image classifier is composed of, as well as the ones that have been fully or partially implemented in this project. The modules are separated in levels; the modules of a given level cannot be executed without having executed one or all the modules of the previous level (for instance, it is not possible to generate the models of the classes without first having extracted the visual descriptors of the images).

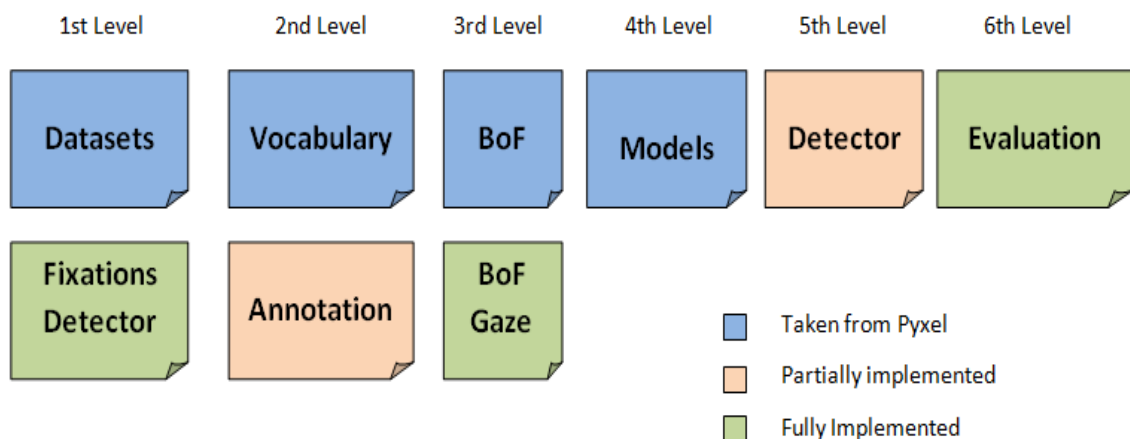


Figure 4.5. Implementation of the Parts in the Image Classifier

As it is visible, only the evaluation module was totally missing to complete the chain of the image classifier. Furthermore the contributions done on the annotation and detector modules were related to the evaluation process as well. Besides that, the other modules that were totally implemented as work of this project are the ones that are related to the contribution of eye tracking data in the system.

Here is an overview of the modules belonging to Pyxel that were used for this project:

- Once the images of the dataset have been partitioned and put into different directories (train and test), the **Datasets** module writes in a text file each of the train and test sets of data. In essence, these text files are named as the dataset that they contain (train or test most commonly) and they contain the list filenames of the images in the corresponding datasets. These text files are needed for almost the rest of the modules in the system.
- The **Vocabulary** module generates the visual vocabulary that is going to be used to quantify the descriptors of the train and test images and generate the Bag of Features.
- The **BoF** module generates the visual signature of each of the train and test images based on the visual vocabulary previously generated.
- The **Models** module generates a model for each of the classes of the corresponding ontology. In other words, it trains the classifier to afterwards perform the automatic annotation of images.

Since the implementation of the rest of the modules has been part of the work of this project, it is explained more in detail in the next section.

4.2.2 The Evaluation Module

In this section the implementation of the evaluation process of the image classifier is explained in detail, since it has been fully implemented within this project.

The evaluation was done using basically one function of `sklearn.metrics` library called `precision_recall_curve`. Although this classifier does not work with thresholds and therefore there's no point in plotting a curve of precision and recall, this function also provided an easy way to easily compute the precision and recall performances.

The parameters it had to receive were just two lists (corresponding to the ground truth and the generated results of the classifier) containing a stream of ones and zeroes. The meaning of this stream of ones and zeroes is nothing else than the labels (positive {1} or negative {0}) assigned to the corresponding image documents. It was necessary that the order of these one and zero labels was coherent for both ground truth and results lists. In other words, the labels had to correspond to the same sequence of documents. Figure 4.6 shows a graphic example of this process.

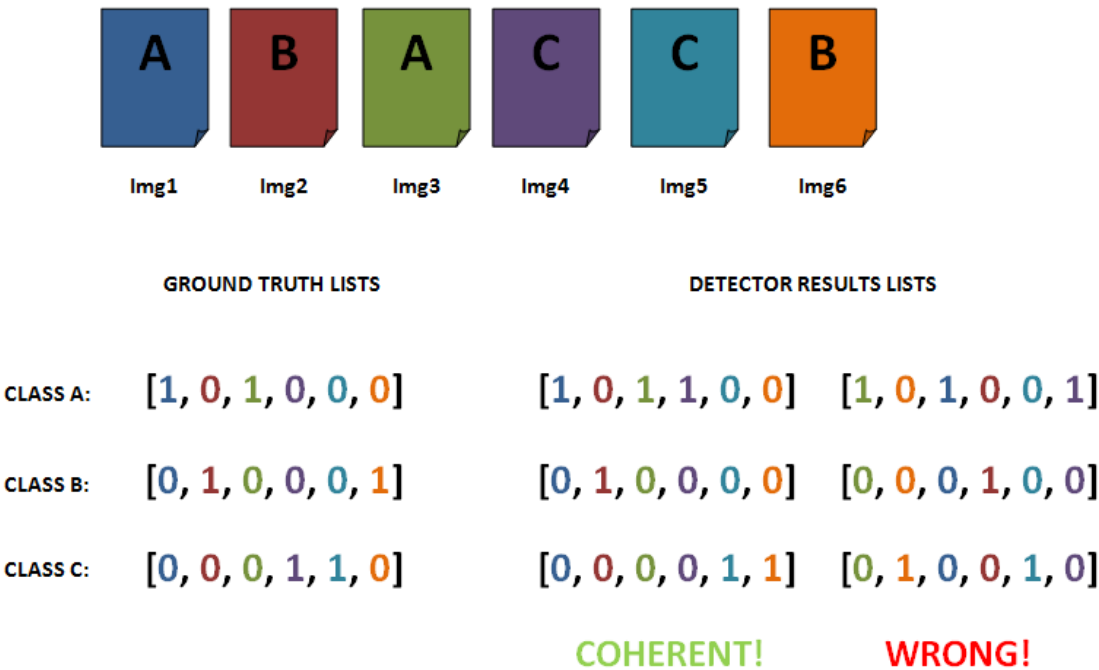


Figure 4.6 Coherent and Wrong Disposition Examples for Evaluating

It shows a set of 6 images to be classified from 3 different classes. An example of a coherent and a wrong result is given for the detector results lists. It is visible than, although the results are the same in both examples, one set of lists is coherent to the order of the ground truth's lists while the other is not.

Although in this example it seems pretty straightforward, images are not ordered that simple when creating the lists.

Therefore, to compute the performances and evaluate the system, both the ground truth and the classifier results had to be converted into this type of list and they had to be coherent to the same order.

Since it is a multiclass system, this process worked separately for each class in the ontology, hence the performances were different for each class, and there was a different list as well for each of the classes in the system.

To explain how these lists were created and made coherent between each other, it is necessary to briefly explain how the annotation and detector modules work.

The **annotation module** reads all the files generated with the manual annotation of the data (more details in chapter 3) and creates a data structure that contains this annotated ground truth. There are two python classes that this annotation module uses called `AnnotatedSemanticClass` and `Annotation`.

The ground truth annotations of each of the classes in the ontology of the system are put in a separate `AnnotatedSemanticClass` structure, while the `Annotation` class gathers each of the generated 'AnnotatedSemanticClasses' into a mother data structure. For each annotated image of each class, a new instance was added. This instance contained three elements; the id of the document (i.e. the filename of the image), the label (positive or negative) and the corresponding class.

To generate coherent lists for evaluation, every time that a new instance was added the corresponding file document was written on a text file. This text file was called `order.txt` and was afterwards used to coherently build a second list of results.

For each positive instance of a class, a ‘1’ was appended to the list of that class, and a ‘0’ to the lists of the rest of the classes in the dataset’s ontology.

Regarding the **Detector** module, the images were classified following the same order than the ground truth thanks to the *order.txt* file; hence it was guaranteed that the lists would always be coherent. Once that was done, an Annotation data structure was created and the lists were stored for each semantic class, the same way as in the annotation module process.

In the end, the evaluation module only had to load the stored Annotation classes generated for both the annotation and the detector modules (that had been previously saved using python’s **pickle** library), and call the `precision_recall_curve` function for each product in the ontology, giving these lists as parameters.

4.2.3 The Eye Tracking Modules

The remaining two modules are the ones that are strictly corresponding to the eye tracking work of this project. Regarding the fixation detection algorithm its implementation comes directly following the idea of the design (chapter 3). The implementation of the **BoF Gaze** module is next explained.

First of all a function called `Crop_Gaze` was created. This function received as parameters the image to crop, the two coordinates of the gaze point and the size of the window. The crop function from Python’s **Image library** was used. The only issue that came up is that with this crop function in the library, the points that were passed as parameters were taken as the ones on the top left corner of the desired cropping. To solve this, an offset was added to the Gaze Points to obtain the desired crops around the gaze points.

Once this function was implemented, BoF Gaze module used it to extract the signature of the cropped images. The file containing the gaze data of all images in the dataset was used to find the points that corresponded to every image in the dataset. Once these were found, BoF Gaze made use of `Crop_Gaze` before starting the process of feature extraction for that image. These cropped images were overwritten for the next ones once the process of feature extraction was finished and the BoF signature for that image had been saved.

4.3 The Metrics

Terminology in this section:

TP → True Positives

FP → False Positives

FN → False Negatives

In order to evaluate the outputs of the classifier and, in other words, the experiments that motivate this project, there are 4 metrics of interest:

- **Precision:** This classic metric in evaluating image classifiers refers to how accurate is the system when it detects a positive in a particular class (how often it's correct positive). This metric is computed for each product and cropping variation.

$$\text{PRECISION} = \text{TP} / (\text{TP} + \text{FP})$$

- **Recall:** This metric is complementary to the precision metric. It refers to how many of the total relevant images of a specific class were correctly retrieved for the system (detected as positive for that class). This metric is also computed for each product and cropping variation.

$$\text{PRECISION} = \text{TP} / (\text{TP} + \text{FN})$$

- **F1-Score:** This metric is the harmonic mean of precision and recall metrics. It is often useful to summarize the performance of a system in just one metric. As well as the previous ones, it is computed for each product and cropping variation.

$$\text{F1} = 2 \cdot \text{PRECISION} \cdot \text{RECALL} / (\text{PRECISION} + \text{RECALL})$$

- **Time:** This metric is useful for our specific project. Since different cropping variations are used, this metric will determine how a specific cropping changes the efficiency of the system. This metric is not computed for each product but for the overall of every product, for each cropping variation.

Chapter 5 - Results and Discussion

The results of this project come directly from the tool that has been developed to evaluate the labels that the detector gives at the output. These results have been split into two main factors to analyze. Firstly the impact that the fact of using different crop windows has on both efficiency (time required for computation) and performance, and on the other hand the performances of each of the classes separately have depending also on the crop window's size. The performance will be measured with the metrics that are explained in chapter 4 (precision, recall and F1-score).

To basic approach for discussing the results is to compare the performance of the system without using cropped images (ground results) against the performance resulting from cropping the images with different sizes.

5.1 Results Tables and Interclass Performance

In this section the results in numbers are shown in tables and depicted graphs. The results are given for each different class in the ontology and for each variation on the crop window size and ground test. Hence, this section is useful to see which products are classified with best results generally, and for each specific product which is the crop window size that works better.

The tables that contain the precision, recall and f1-score results for each product and crop window size are shown in the following figures 5.1, 5.3, and 5.5, respectively.

For each product, the best result is highlighted in bold, as it is interesting to appreciate if there is any cropping variation that improves the performance from the ground results.

The graphs that respectively depict the results in the tables are visible in figures 5.2, 5.4 and 5.6, just below their corresponding tables.

Since the differences between the three metrics are not very significant, the discussion of these results is done about all three sections together after showing the tables and graphs, instead of separately.

	Special-K	Bananas	Bad Images	Pringles	Hunky Dorys	Coca-Cola
Ground Results	0.87	0.46	0.37	0.76	0.69	0.72
400x400 crop	0.96	0.50	0.34	0.56	0.82	0.55
350x350 crop	0.88	0.51	0.34	0.61	0.86	0.52
300x300 crop	0.82	0.40	0.38	0.58	0.72	0.54
250x250 crop	0.85	0.30	0.32	0.54	0.58	0.41
200x200 crop	0.72	0.27	0.30	0.34	0.58	0.54

Figure 5.1. Precision Results Table

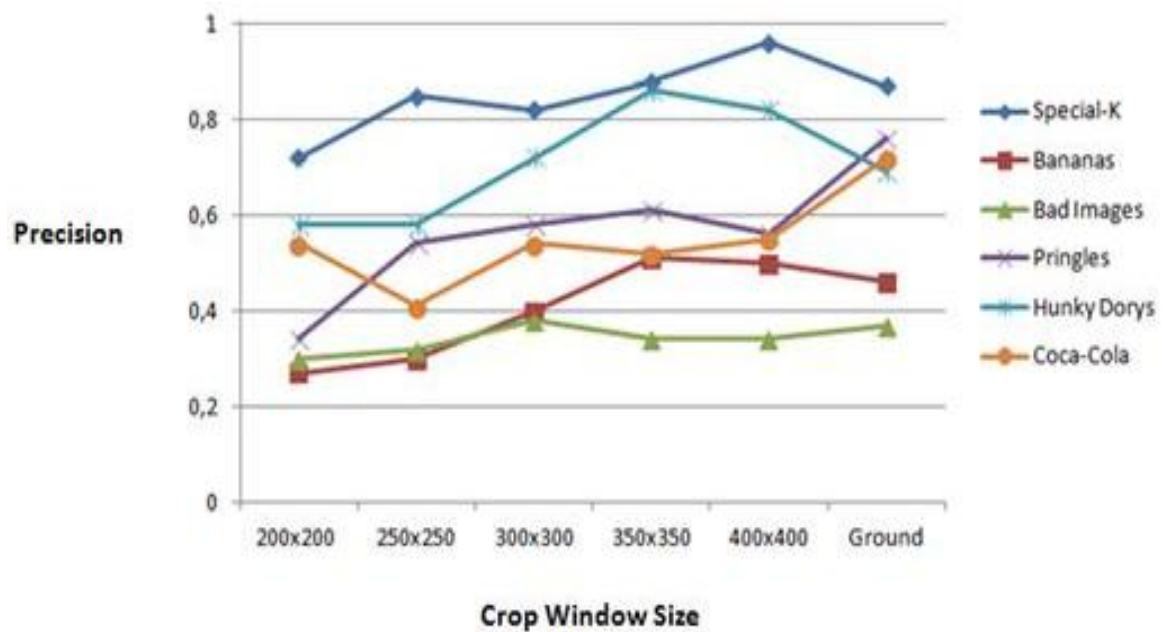


Figure 5.2 Interclass Precision Results

	Special-K	Bananas	Bad Images	Pringles	Hunky Dorys	Coca-Cola
Ground Results	0.92	0.42	0.61	0.73	0.44	0.52
400x400 crop	0.83	0.56	0.48	0.64	0.73	0.31
350x350 crop	0.84	0.53	0.46	0.70	0.64	0.40
300x300 crop	0.75	0.37	0.45	0.75	0.68	0.39
250x250 crop	0.70	0.29	0.38	0.52	0.57	0.43
200x200 crop	0.59	0.29	0.36	0.50	0.57	0.21

Figure 5.3 Recall Results Table

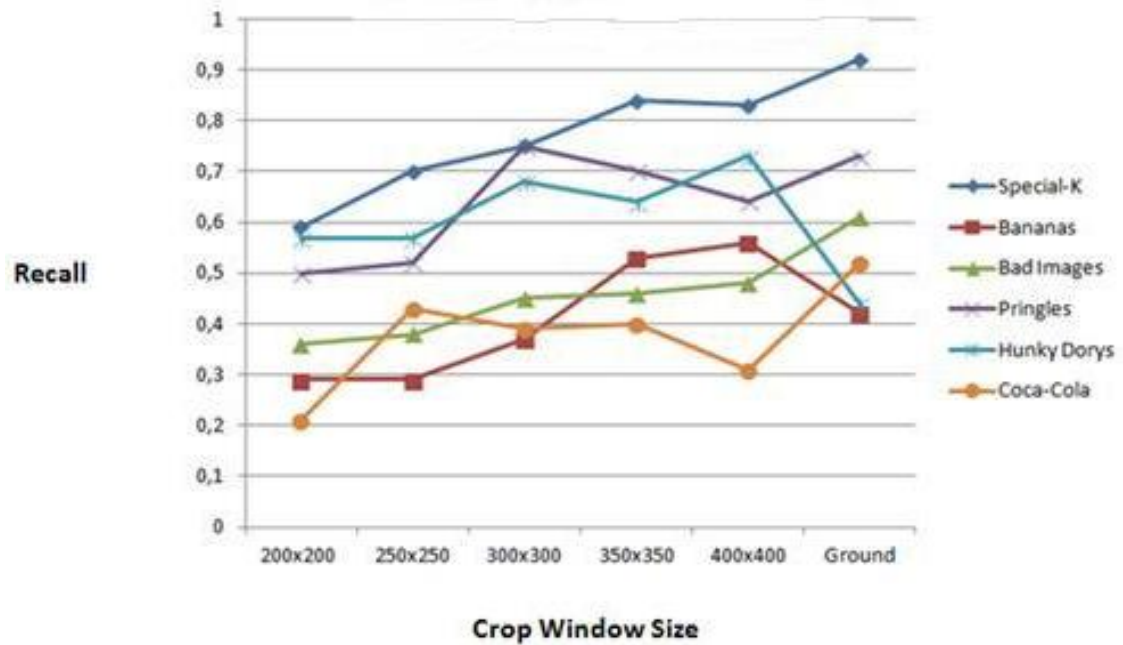


Figure 5.4 Interclass Recall Results

	Special-K	Bananas	Bad Images	Pringles	Hunky Dorys	Coca-Cola
Ground Results	0.89	0.43	0.45	0.74	0.53	0.60
400x400 crop	0.89	0.53	0.40	0.60	0.78	0.40
350x350 crop	0.86	0.52	0.39	0.65	0.74	0.45
300x300 crop	0.79	0.39	0.42	0.66	0.70	0.46
250x250 crop	0.77	0.30	0.35	0.53	0.58	0.42
200x200 crop	0.65	0.28	0.33	0.41	0.57	0.29

Figure 5.5 F1-Score Results Table

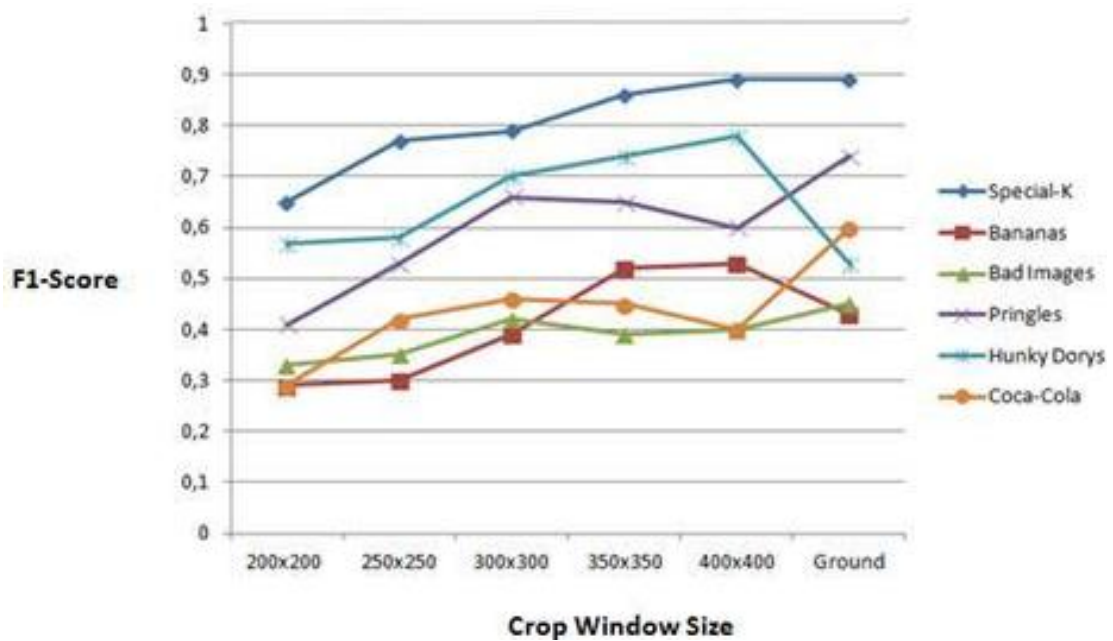


Figure 5.6 Interclass F1-Score Results

Regarding the results shown in all these graphs and tables above, the first significant observation is that **the results are very different depending on the product**. This is sensible, since the products have been chosen very different in all senses on purpose. It is possible that **the fact of using sparse sift for feature extraction** makes the classifier more effective for products with a particular visual signature (for example Special-K cereals), and more vulnerable to miss with products like bananas, that are mainly characterized for other features like the colour. Hence, it is exactly the three classes with a very specific and visually particular logo the ones that get by far the best performances.

Another interesting point to mention is the impact on the classifier's performance depending on the size of the crop window. As in the previous comment, the results are very different for each product. However, although it is true that **the general performance tends to get worse as the crop window gets smaller**, the performances resulting from using images without cropping based on gaze (**ground results**) are in several occasions not the best for a given product. This is very important to consider in order to build the conclusions of this project.

It is also interesting to see that the cropping windows of sizes 200x200 and 250x250, although do not always achieve the worse performances, neither they present any improvement in any case for a given class. All the other window sizes present at least one best performance for a specific metric and product. Nonetheless and as said previously, it is generally true that the smaller the window gets, the worse the performances tend to be

Speaking of performances of particular classes however, it is remarkably noticeable the case of the *Hunky Dorys*' class. Although this is an uncommon behaviour, this class presents the worst performance with the ground test. For a particular reason, there must be an element in the background that worsens the performance dramatically, while when cropping the image around the gaze points it disappears to boost the results positively.

The fact that the behaviour is in any case constantly getting worse for every decreasing of the size of the window, it is important to conclude that using less image data can lead to better results (if the remaining data it is well selected).

5.2 Efficiency versus Performance Results

The results in this section are the most important ones in order to conclude if eye tracking technology can be helpful to improve an image classifier. Since efficiency is together with performance, the factor that determines whether an image classifier is better or worse, the following results focus on how efficiency and performance vary for the different cropping window sizes.

In this case, the metrics of all products are averaged for every cropping variation into one single score, because the job of detecting images is not separated for classes but done in a single process.

Hence, first of all, the results of averaged metrics of interest (fig 5.7) for each crop window size as well as their time of execution (for all the classifying process) (fig 5.8) are shown.

	Average Precision	Average Recall	Average F1-Score
Ground Results	0,65	0,61	0,63
400x400 crop	0,62	0,60	0,61
350x350 crop	0,62	0,60	0,61
300x300 crop	0,58	0,56	0,57
250x250 crop	0,50	0,49	0,50
200x200 crop	0,46	0,42	0,44

Figure 5.7 Averaged metrics for each window size

	Time (seconds)
Ground Results	833
400x400 crop	645
350x350 crop	566
300x300 crop	481
250x250 crop	410
200x200 crop	340

Figure 5.8 Time of computation for each crop window size

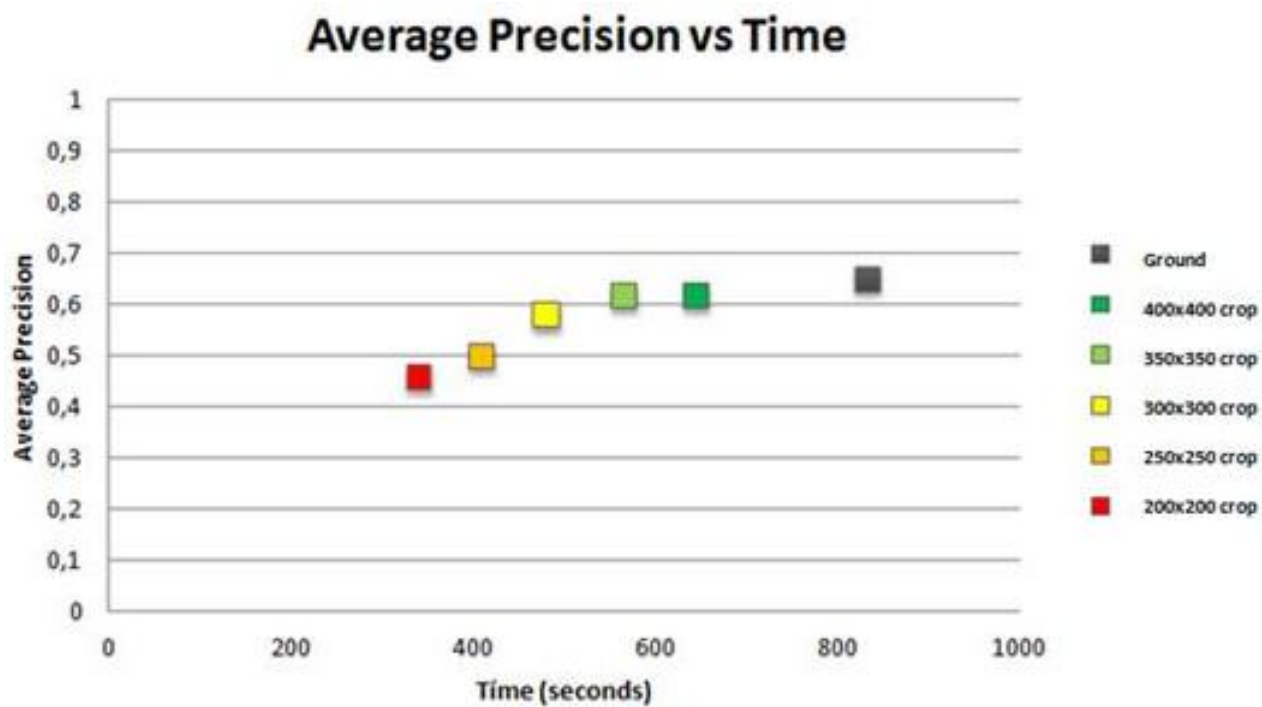


Figure 5.9 Averaged precision vs efficiency

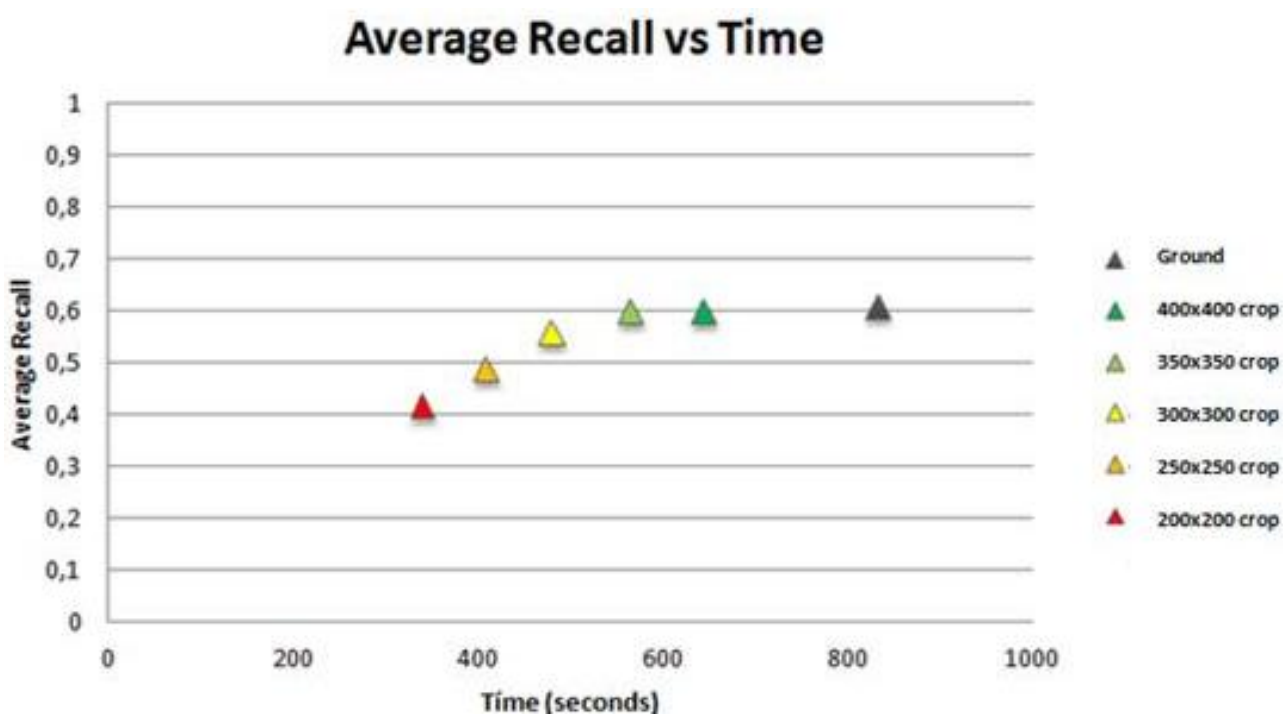


Figure 5.10 Averaged Recall vs. Time

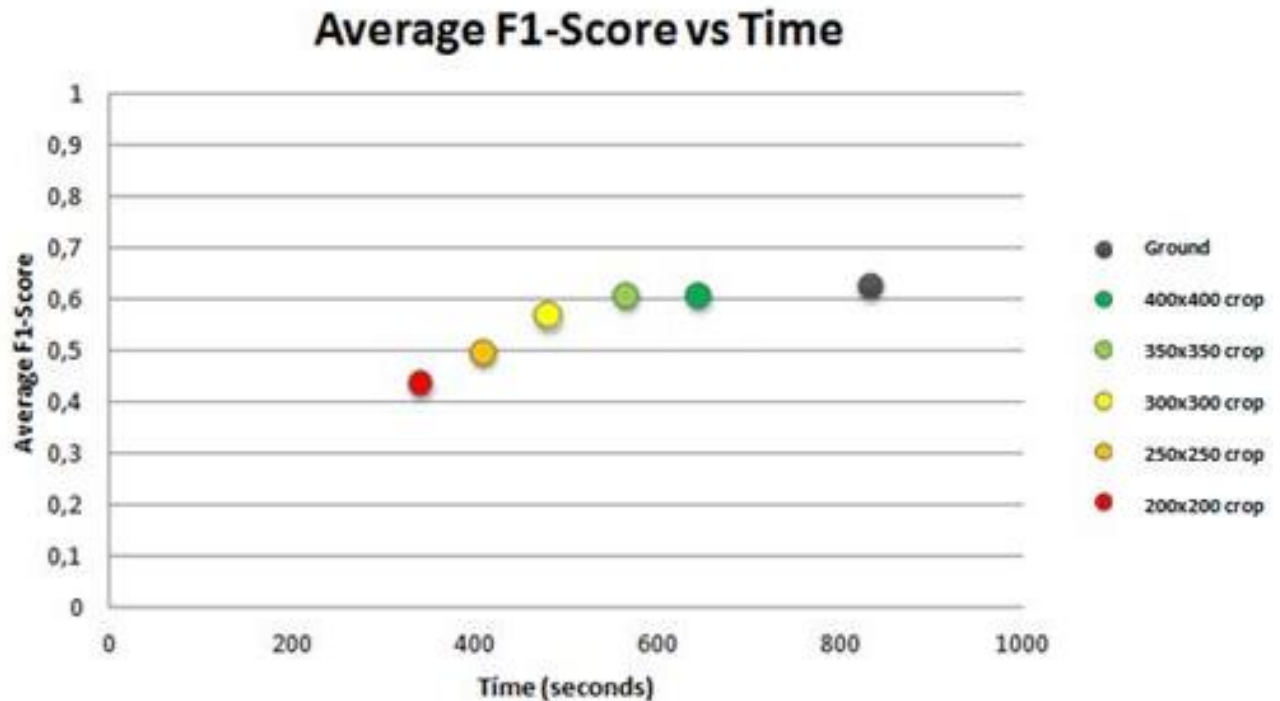


Figure 5.11 Averaged F1-Score vs. Time

First of all and to reassert the ideas that are discussed in the previous section of results, it is possible to notice in the graphs that **ground test (using non-cropped images) gives generally the best results**. All three of the averaged metrics present its best score on the ground test and that is important to touch on. While perhaps precision metric presents more margin of difference between ground results and the other ones, all three metrics present few and not very significant variations between them (f1-score, obviously neither, because it depends on the other two).

However, it is remarkable how just **using the 400x400 gaze window (the biggest) improves the efficiency respecting the ground results** (the time of computation of the results is reduced in a 22.57%). The differences in efficiency are more or less constant between the cropped models, and of course, always improving the efficiency as the crop window gets smaller (less image data to process). It is especially important to observe that performances of the two bigger crop windows are almost as good as the ground performance, while the time it takes to compute the classifying process is remarkably reduced

Chapter 6 - Conclusions and Further Research

The goal of investigating if eye tracking data was useful to improve an image classifier has been achieved within this work. However, there are a lot of directions to expand this work in the future that could lead to very interesting further investigations on the usage of eye tracking data. In this chapter it is analyzed the whole work and conclusions of this project as well as the most interesting ideas to expand it.

6.1 Conclusions

The initial goals of this project are now analyzed and discussed according to the achieved results:

- 1- See how the performance and/or efficiency of the system change when using small cropped parts of the images (centred in the eye tracking points) with different sizes instead of the complete images.*

About this first goal, an experiment has been successfully completed that allows analyzing at some level the impact that using cropped images based on gaze data has on an image classifier.

The results are interesting. Although, when using cropping based on gaze, performance has not improved from ground results, the bigger window sizes performances have been really close. Furthermore, the efficiency of the whole process is significantly improved when using this cropped images. Objectively, the losses in performance are so insignificant that it can be fairly considered worth in the context of this experiment to use a crop model instead of the complete images for what it would mean in terms of time saving and efficiency.

Besides of having results to this experiment on exploding eye tracking data, an image classifying chain has been partially build in this project, as well as modules that feature uses of eye tracking data have been implemented, and this remains as well as a tool to work with and improve in the future.

Having analyzed and concluded this first goal of the project, let us bring back the second one:

2- Implement an algorithm to detect fixations in time using the variations of this gaze data, so that the images that contain a fixation are filtered as images of interest from the video stream, achieving smaller amounts of data.

This second goal was also interesting because it exploded the eye tracking data that this project works with, but in this case it has been only partially achieved. The algorithm to detect fixations has been implemented and works, but it has not been improved enough to use it in the process.

However, although the parameters of this algorithm (movement margin and duration) needed to be very well adjusted to provide a good filtering of frames as well as a good detection in fixations, a lack of time for achieving the first goal (which was considered more important) in this project has leaded to not having been able to finish this algorithm in its totality.

On a personal level, I believe this project has been interesting to learn dealing with new technologies and its usage like this case of eye tracking, and also to design an experiment that has allowed discussing if using eye tracking for this system was genuinely a better choice, because the achieved results are more than decent.

6.2 Future Work and Research

This project represents an initial investigation on the potential utilities of eye tracking data for image object recognition. In this sense, there are many possible directions to expand this work in the future and all of them are interesting in some way.

Here there are some good ideas for future works:

- **Further investigate and improve the algorithm to detect fixations with eye tracking data.** This was one of the initial goals of the project and has not been fully achieved. This is an interesting part to investigate since it deals with using eye tracking data in a temporal level. With a well developed and tested algorithm to detect fixations, a lot of unnecessary images wouldn't be processed for the system, so it would be much more efficient.
- **Redo the experiment of cropping with a different dataset, number of users, ontology and amount of data.** The results that have been obtained in the experiment developed in this project present a high bias. There can be a lot of factors that change considerably the results. It would be interesting to build a different dataset recorded by several users and using different products in the ontology, to average the results and make them more reliable.
- **Implement an algorithm that makes use of temporal information for this kind of classifier.** This idea represents a short extension of this work. The main idea would be to put the algorithm of this image product detector in the context of a video signal. In other words, it does not make much sense to have three different products in three consecutive frames in a video stream. If a product has been detected in a given frame, it is likely that the next frame also contains it. This idea can lead to a very useful algorithm in the context of the idea that is next exposed for future work.

- **Implement a real time detection application that works within an eye tracking device.** This is probably the ultimate immediate goal that can be developed in the context of eye tracking data. Perhaps it would be a good idea to first succeed in the previous tasks above before trying to implement an application of this magnitude. If it is possible to integrate an algorithm that detects products on real time in an eye tracking device, all of these proposed works can be used and tested.

References

Important note on references:

There are few references in this section due to the fact that the chapter that provides more references (technical background) is covered in the appended literature survey. In it, there is a more exhaustive list of references.

- [1] “Computer Vision”, 2014. [Online]. Available:
http://en.wikipedia.org/wiki/Computer_vision

- [2] “Eye Tracking”, 2013. [Online]. Available:
http://en.wikipedia.org/wiki/Eye_tracking

- [3] Michel Wedel, Rik Pieters (2008), A Review of Eye-Tracking Research in Marketing, in Naresh K. Malhotra(ed.) *Review of Marketing Research (Review of Marketing Research, Volume 4)* Emerald Group Publishing Limited, pp.123 - 147

- [4] Tobii Glasses User Manual [Online] Available:
http://www.tobii.com/Global/Analysis/Downloads/User_Manuals_and_Guides/Tobii_Glasses_UserManual.pdf

- [5] Toyama, T., Kieninger, T., Shafait, F., & Dengel, A. (2012, March). Gaze guided object recognition using a head-mounted eye tracker. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (pp. 91-98). ACM.

Appendix 1



**DUBLIN CITY UNIVERSITY
SCHOOL OF ELECTRONIC ENGINEERING**

Object Recognition Using Eye-Tracking
Literature Survey

Sergi Imedio Pereira
April 2014

**FINAL PROJECT DEGREE
OF
ELECTRONIC ENGINEERING**

Supervised by Rami Albatal, Xavier Giro and Noel
O'Connor

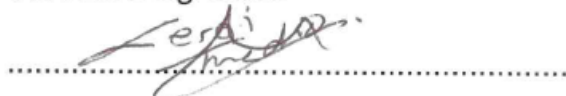
Acknowledgements

I would like to thank my supervisor Dr. Xavi Giró for his guidance, enthusiasm and commitment to this project. Thanks are also due to Dr. Rami Albatal for supporting this work to date, etc. Finally I would like to express my deep appreciation to Dr. Derek Molloy for preparing the original Transfer Report template, from which this document is hacked.

Declaration

I hereby declare that, except where otherwise indicated, this document is entirely my own work and has not been submitted in whole or in part to any other university.

Student's signature

A handwritten signature in black ink, appearing to read 'Sergi Imedio', is written over a horizontal dotted line.

Date: ...21/04/14.....

Abstract

This is a short Literature Survey for a 4th year Electronic Engineering Degree Project. In this paper we define a project about using Eye Gaze Trackers to investigate if they can be useful to improve image object recognition using the provided gaze metadata.

To do that the plan is to first build a typical machine learning system, and afterwards, improve the performance of the system by using the gaze data to filter in time (detect fixations and process only certain frames) and in space (process only cropped parts of the image around the gaze points).

Table of Contents

ACKNOWLEDGEMENTS	8
DECLARATION	9
ABSTRACT	10
TABLE OF CONTENTS	11
TABLE OF FIGURES	12
CHAPTER 1 - INTRODUCTION	13
1.1 PROJECT MOTIVATION	13
1.1.1 <i>Human Gaze Behaviour</i>	13
1.1.2 <i>Feature Detection for Recognition in Images</i>	14
1.1.3 <i>Approaching Interactive Video Scenarios</i>	14
1.2 THE PROJECT	15
1.2.1 <i>Goals</i>	15
1.2.2 <i>Development</i>	16
CHAPTER 2 - EYE GAZE TRACKERS	17
2.1 INTRODUCTION TO THE TECHNOLOGY	17
2.2 THE TOBII GLASSES.	18
2.3 SUMMARY.....	19
CHAPTER 3 – PREVIOUS WORKS ON OBJECT RECOGNITION USING EGT ..	20
3.1 GAZE GUIDED OBJECT RECOGNITION BY TOYAMA ET AL (2012)	20
3.2 AUTOMATIC ANALYSIS OF EYE TRACKING DATA BY DE BEUGHER ET AL (2012).	26
3.3 SUMMARY.....	27
REFERENCES	28

Table of Figures

FIGURE 1.1. SALIENCY MAP EXTRACTED FROM GAZE TRACKING. 1

FIGURE 1.2 GOOGLE GLASS DEVICE..... 6

FIGURE 1.3 SCHEME OF THE BASIC CORE OF THE PROJECT TASKS 6

FIGURE 1.4 PROJECT PLAN TABLE..... 6

FIGURE 2.1. TYPES OF EGT APPLICATIONS AND TECHNIQUES. 1

FIGURE 2.2. TOBII GLASSES..... 1

FIGURE 2.3. GAZE DATA FILE EXPORTED TO EXCEL. 1

FIGURE 3.1. MUSEUM GUIDE 2.0 SYSTEM SCHEME. 1

FIGURE 3.2. GAZE FIXATION DETECTION SYSTEM..... 1

Chapter 2 – Introduction

This first chapter introduces the project motivation and its goals and structure, defining a solid outline for the objectives and the matters we aim to investigate.

1.1 Project Motivation

The technology of Eye Gaze Trackers (EGT) has been a tool used in many investigations during the last years. The possibility of studying the human eye gaze behaviour led to investigating how people are hardwired to use their sight when they were assigned a certain task with an image or scenario.

However, as multimedia technologies started improving more and more, not only the technology of EGT has become more efficient, but also a whole universe of possibilities for its potential applications have also become possible to implement.

In this project we aim to investigate new possibilities in the nowadays computer vision panorama, specifically in object recognition matters.

1.1.1 Human Gaze Behaviour

When visualizing a scene or an image, the Human Visual System (HVS) does not focus on all the elements. Instead, it takes some specific regions as “important” in the sense that they will provide more useful information. In other words, it prioritizes some regions in order to simplify the task of understanding a whole image [1].

Furthermore, the human often does not only prioritize the regions he believes they will be more useful to him, but also draws specific patterns, returning repeatedly to the same elements instead of exploring new regions of the image or scene [2].

By tracking these points, we can extract information from them and see if it is useful.

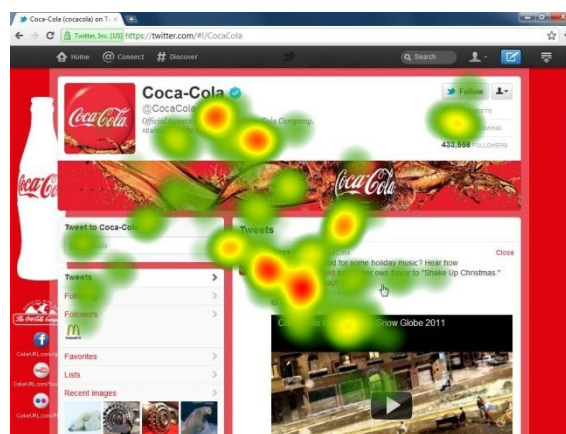


Figure 2.1. Saliency Map Extracted From Gaze Tracking.

1.1.2 Feature Detection for Recognition in Images

There are many techniques that can be used to approach the issue of object recognition in images. Many of them consist (simplifying) on detecting a set of key points (features) from the image that are believed to be representative. These features can be extracted by means of many techniques such as dense sampling[3] (sampling the image and taking many points straight away as features) or Harris-Laplace [4](performing a series of filtering to the image in order to find edges and corners). Once these features have been detected, a representation of them is done using descriptors such as SIFT[5], that can be already compared between each other to compute similarity [6].

Hence, if a EGT device is used it could be a matter of investigation to optimize the feature detection stage and assume the points of the image that are closer to the gaze point as more representative features, and therefore being able to overlook some parts of the image, making the process faster.

1.1.3 Approaching Interactive Video Scenarios

For the last few decades, the issue of dealing with video data to solve problems as object recognition or other interactive purposes has been a matter of investigation. We can see lots of devices and new emerging technologies that are based on dealing with recorded video data such as Google Glass (figure 1.2)

The fact of operating with video adds a temporal correlation between frames (or images), that has to be taken into account, as well as a huge amount of data.

EGT systems have also been adapted to video data over the years, by adding text information (timestamps) that link the gaze data (x position, y position, left or right eye...) with the exact instant (frame) of the video data. Therefore, a wide range of useful possibilities are yet to be investigated and studied with these devices.



Figure 2.2 Google Glass Device.

1.2 The Project

This is a 4th year Electronic Engineering Project, that suits a MSc proposal. In this section, the project work that will be developed in the next months will be analyzed with its goals and its initially supposed development.

1.2.1 Goals

With the Tobii Eye Gaze Tracker device as the main tool for the investigation, from now until August we will try to answer the following questions with this work:

- 1 - *Is the EGT data useful for improving image object detection and recognition?*
- 2 - *If so, can we implement algorithms to optimize the process using this data?*

We will try to answer the first question by comparing the recognition success of using common feature detection methods versus using the points tracked by the EGT device already as features, using the same feature representation (SIFT descriptors) method in both cases. The initial idea is to use dense sampling and Harris-Laplace as feature detection methods to compare with the results of using the features extracted by means of EGT.

The second question is actually an extension of the first one. There is so much terrain to explore regarding the uses of this technology that we can focus on several things to investigate once we have seen some results concerning our first question. The first main idea would be to crop the image circularly centred on the gaze points, so the system could work faster by not having to process the whole image. Another extension to do would be to build an algorithm that allowed filtering the frames where there is no fixation (e.g. the user is walking and there is not a product or is not distinguishable due to the movement). This would keep the system from having to process all the frames in the video stream.

Ultimately, this optimization could lead to build a real time application that processed video and detected products. We have chosen a retail environment since we think it is a matter of general interest and it is quite sure that it would be useful. Therefore, we shall record a dataset that contains footage in stores.

1.2.2 Development

This project will be carried on between the months of February and August of 2014. We will split the project development into three stages (introduction, main body and conclusions), these are further explained below:

Introduction [February-March]

Tasks belonging to this stage:

- Learn the basics of Python.
- Learn how the Tobii EGT device works and how to deal with its data.
- Learn how the required libraries for recognition work and start doing some trials with them.
- Define the highlights of the required dataset do the corresponding recordings.

Main body [April-May]

Tasks belonging to this stage:

- Implement a system to extract the frames from the video.
- Implement a system to match the EGT data with its corresponding frames.
- Annotate the data to create a ground truth
- Extract SIFT descriptors from the different methods of feature detection and create a visual Bag of Words (BoW) for each of them.
- Train the classifiers for each defined object of the dataset using SVM.
- Test the classifiers with this initial version and evaluate the results.

Conclusions [June-August]

Tasks belonging to this stage:

- Implement an algorithm to use just a circle around the gaze points and see if the results improve.
- Implement an algorithm to process only certain key frames in the video stream.
- Keep on the optimization and find the best parameters in terms of accuracy and speed.
- Final Report and Oral Defence

Chapter 3 – Eye Gaze Trackers

We believe that is useful to do some brief review of the EGT technology since it's the main tool in our project. In this chapter we will make an overview of the EGT technology, which is the main highlight in our project as well as introducing the basics and features of the exact device we will use: the Tobii glasses.

2.1 Introduction to the technology

Eye Gaze Trackers (EGTs) are devices that can estimate the gaze direction of a person. They are classified concerning two aspects: Its application or purpose and its technique to estimate the eye gaze.

Concerning the purpose of the application there are two possible types [7]:

- *Diagnostic Applications*. These applications are addressed to take some conclusions regarding how the humans scan different situations instead of offering a service.
- *Interactive Applications*. These applications on the other hand, are addressed to detect a certain pattern or behaviour on the gaze of a person to have a response that is useful to the user and coherent to that gaze pattern/behaviour. The application we would try to implement would be an example of an interactive type of EGT application.

Concerning the gaze estimation technique of the device there are two possible types [8]:

- *Intrusive Devices*. These Devices are “carried” by the user, i.e. they are in physical contact with him. Within this type of devices we find different types (contact lenses [9], electrode related techniques [10], etc). Although this kind of devices can bother the user, they are also more accurate. The Tobii device we will use for our experiment is also an example of an intrusive EGT device.
- *Non-Intrusive Devices*. These are remote devices that are able to track the gaze of a person by means of, for instance, an infrared (IR) light source. A remote camera sends a ray of IR light and detects the contrast of the refraction difference between the pupil and the iris, and is able to estimate the movement of the eye and therefore the gaze of that person.

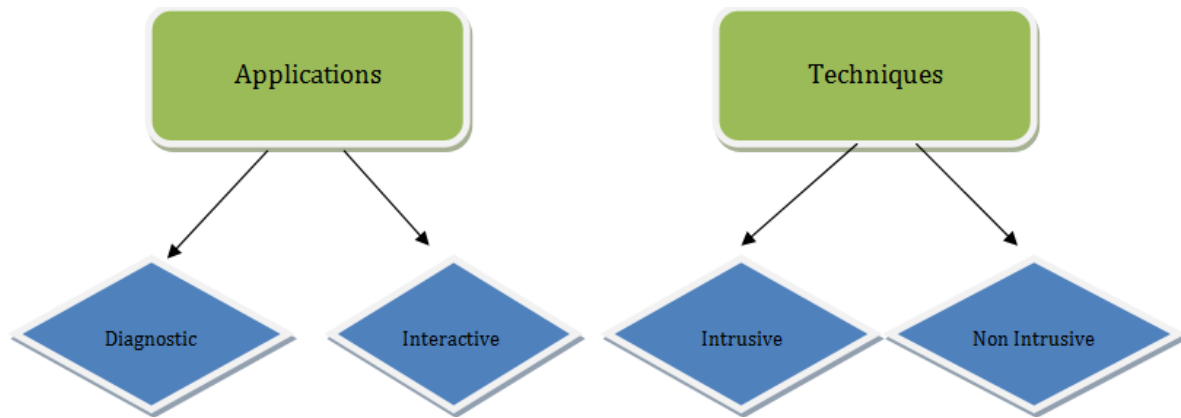


Figure 3.1 Types of EGTs applications and techniques

EGTs have a long history already, since the first studies concerning the problem of tracking human's gaze go back to 1850. These technologies have been improving more and more until nowadays, but it is yet a technology with so many possibilities to be exploited.

2.2 The Tobii Glasses

The Tobii Glasses are an intrusive EGT device that record *MJPEG2000* video data at *30fps* [11]. As visible in the picture (Fig 2.2) they work with a recording assistant device to handle the recorded data. This device can work whether with or without a provided IR markers to communicate more accurately the eye movements and hence the gaze of the person wearing the glasses to the recording assistant. With this information, the gaze data is built in a text file that contains timestamps attached to each single gaze data point, besides extra data regarding information of the user or specific events in the recording (Fig 2.3).

On the other hand, there's also an specific software to work with the recorded data that is *Tobii Studio*, which we will try to learn its state of the art to work properly.

The pack also provides storage memory cards, with enough capacity to handle the dataset that our project would require. Hence, the device should be appropriate for our project.



Figure 2.2 Tobii Glasses.

Exported data in Microsoft Excel

	A	B	C	D	E	F	G
16	Participant:	Shampoo 043D					
17	Participant properties:						
18							
19	Gender:	Female					
20	Age:	33-40					
21	Product Offer:	Not relevant					
22	Test:	Shampoo					
23							
24	Filter settings:						
25							
26	Eye: Right						
27	Validity: Normal						
28	Fixation filter: Raw data						
29							
30	Timestamp	Number	PupilRigh	Event	EventKey	Data1	Data2
31	0	0	0				
32	0			GlassesRecordingStarted	300	0	0
33	33	1	0				
34	67	2	0				
35	100	3	0				
59	900	27	0				
60	933	28	0				
61	967	29	0				
62	1000	30	0				
63	1029			LogData	0 Found Shampoo	Scanning middle section first	
64	1033	31	0				
65	1067	32	0				
66	1100	33	0				
67	1133	34	0				
68	1167	35	0				
69	1200	36	0				
70	1233	37	0				

Rec 43-All-Data

Figure 2.3 Gaze Data File Exported to Excel.

2.3 Summary

EGTs are devices that can measure the position of a person's gaze. There exist intrusive (in physical contact with the user) and non intrusive (using remote cameras). These latter are more comfortable to the user but less accurate. EGTs can be used whether for diagnostic or interactive applications.

We will intend to use the Tobii EGT intrusive device to develop an interactive retail application. Tobii glasses record MJPG2000 video at 30 fps and are suitable for our project requirements.

Chapter 4 – Previous Works on Object Recognition Using EGTs

To find ways and ideas to approach the problems we will be facing, it is very useful to do a review of some previous image recognition related papers using the EGT technology. In this chapter we will break down two articles that merge recognition algorithms with EGT data.

3.1 Gaze Guided Object Recognition Using a Head Mounted Eye Tracker by Toyama et al (2012)

Since people usually rather using easy intuitive technologies instead of complicated or incomprehensible ones, eye tracking is a very remarkable technology because of its closeness to human intuition.

Using eye tracking with Augmented Reality (AR) is a way to provide a lot of opportunities to create EGT applications for the users to interact with the environment around them in several ways such as embedded signs or sounds.

On the other hand we have image based object recognition. This research field has also been of remarkable interest during the last years. Its objective is to recognize the objects present in an image or video in the same way as humans do. Methods based on local features (extracted from small patches of an image) took over global features based methods since these latter ones are not robust enough to illumination changes or occlusions.

A very interesting article to work on merging EGTs and object recognition is the one we will analyze now [12]. In this paper, the authors develop an Augmented Reality application called *Museum Guide 2.0*. The application uses an EGT device called *iView X HED*, and consists on using the raw eye gaze data to detect when users (wearing a head mounted eye tracker device) in a museum are staring at a given exhibit during a certain duration, so then the system automatically presents AR meta-information regarding the corresponding exhibit.

Therefore, there's a limited and known selection of possible exhibits to be recognized. The captions of these images containing the exhibits are used to perform a feature extraction and consequently a database of features. This database will later be queried with new features, when the EGT device detects a fixation, and the exhibit that corresponds to the majority of matched features will be the one displayed by the AR system. In this project, only a pre recorded audio is used for each exhibit match, but the authors say there is much room left for many other complex possibilities. This process is represented in the picture below (Figure 3.1).

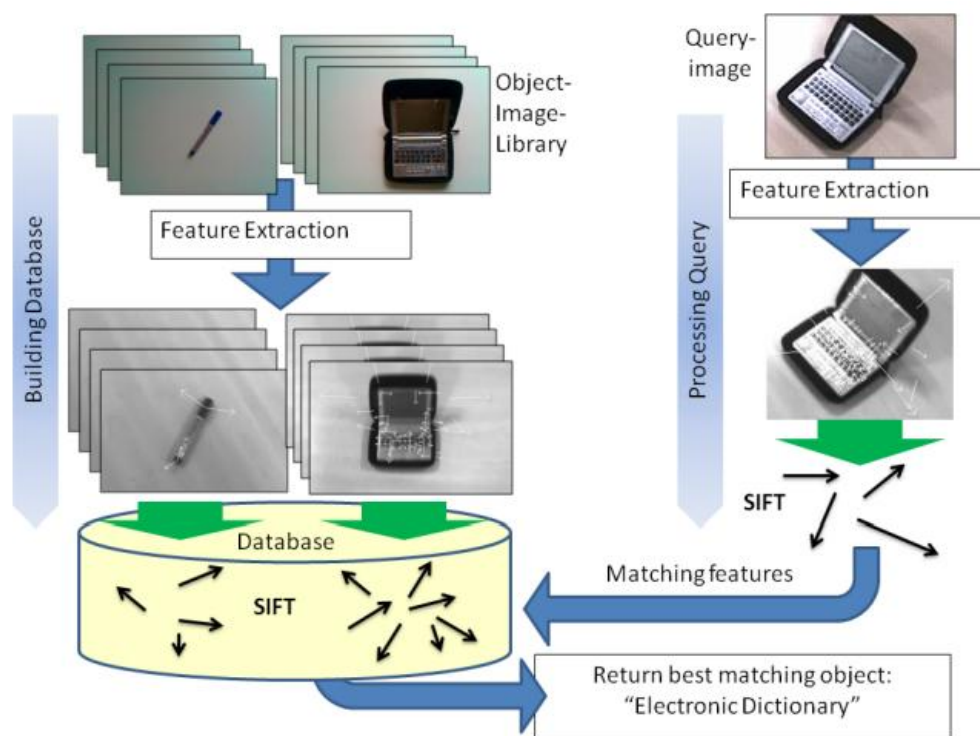


Figure 3.1 Museum Guide 2.0 System Scheme

The method used to detect interesting points is SIFT. These interesting points (features) will correspond to the maxima/minima of the Difference of Gaussian (DoG) [13], that are subtractions between same images filtered by Gaussian Kernel with multiple scales (providing scale invariation). To match the features Euclidean distance is used. A histogram is built representing how frequently a particular identity was returned.

To build the database, for each exhibit the process is the following:

- The object is placed on a museum table (same as all the other objects).
- A person wearing the EGT device walks around the table directing the scene camera to the object and recording video data of it.
- SIFT features are extracted from selected images of the video stream, showing different viewpoints of the object.
- Label these images with the name of the exhibit.

Since not all the objects produce the same number of detected SIFT features, there are more images of the objects with fewer SIFT descriptors, to avoid disproportion in the number of features for each object.

One of the keypoints in this system is that it takes advantage on the fixation points provided by the user. Whilst a typical object recognition system has to deal with complete images and perform an image analysis to discern the object of interest from the rest of the image, this system uses the gaze data to know where the object of interest is located, hence it simplifies this task. A rectangular region centred on the gaze points is “cut” from the image and feature points are extracted from it. To be precise, a parameter n is defined as the number of features to be taken close to the gaze point. Hence, if $n=10$, the 10 closest features to the gaze point will be extracted. This speeds up the recognition process as well as working locally on the region of interest. The features are also weighted regarding its closeness to the gaze point (the closest it is to the gaze point the greater the weight given to it will be).

To define a solid gaze-based ground truth (so to have a criterion for the evaluation of the system results), two important parameters are defined. The first one is the duration threshold T_{dur} and the second one is the noise threshold T_n . The system works as following. When a frame is not detected as any object of the database, it is categorized as undefined. When otherwise an object ‘X’ is recognized in a given frame, the algorithm starts counting the number (duration) of the frames that have that label ‘X’. Once this counting is underway, when the number of frames not corresponding to the object X (considered as noise) reach the value of T_n , the counting stops and the sequence is dropped. On the other hand, when

the counter reaches the value of T_{dur} , a gaze to the object X is recognized. The scheme below (Figure 3.2) represents how this system works with an example.

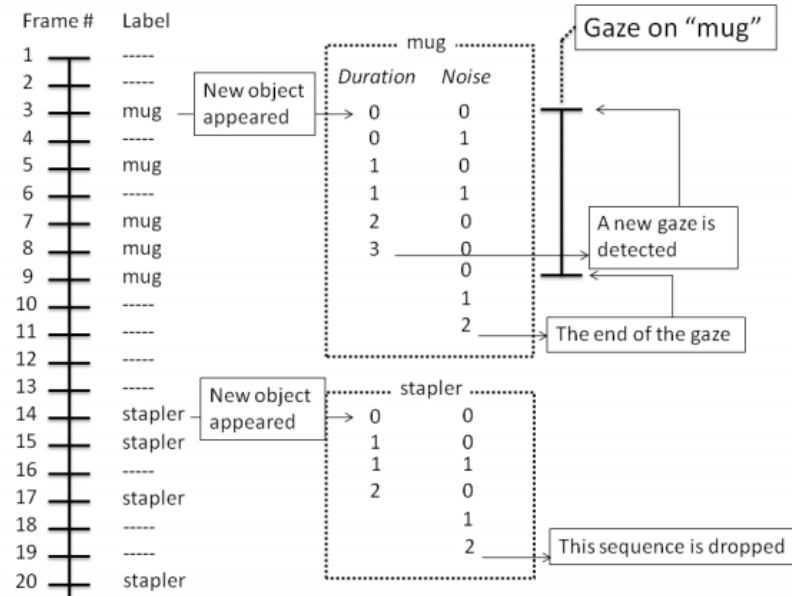


Figure 3.2 Gaze Fixation Detection System

Once a ground truth has been defined, the authors talk about how to detect the existence of gaze from results of fixation guided object recognition framework. A first assumption is done regarding using only frames in a first instance instead of a real time sequence. As every frame has a label denoting the object that appears in them, a method has to be defined to make sure the user is really looking at that object and it's not because of an involuntary glance or noise. Hence, to compute existence of gaze from a sequence of labels (results), three methods are proposed.

A first method is just called *Plain Method*. In this case the results from fixation based object recognition are directly returned. A second more sophisticated method accumulates the weighted histograms of best SIFT matches. This is done for n frames. In the end, the object that has the highest value on the histogram is returned as the result. A third and last method (*Pseudo Method*), uses the same procedure that the one explained above to post process the manually labelled ground truth data, so it works with the threshold parameters mentioned above (noise and duration thresholds).

These are, therefore, the precision-recall results of the offline (not in a real time case, but frame per frame instead) simulation for each of these three methods:

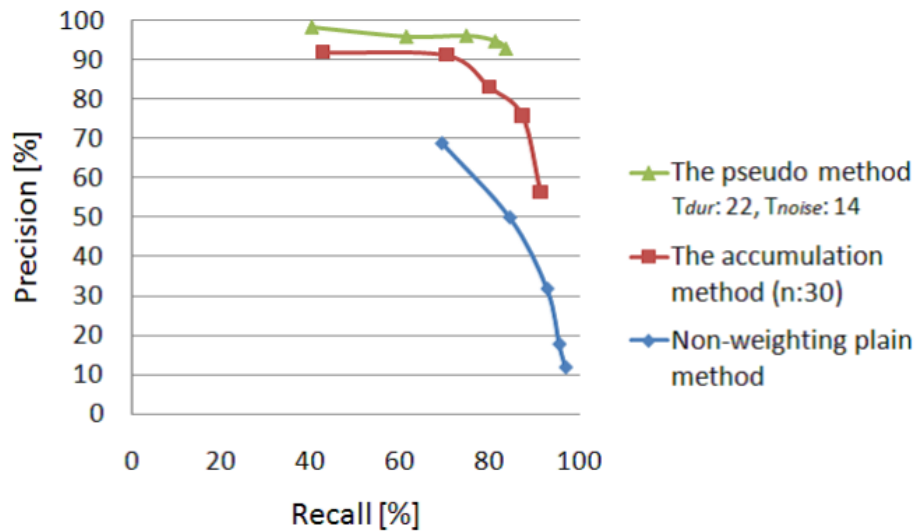


Figure 3.3 Offline Results for Each Gaze Detection Method

Notice that in the pseudo method, the threshold parameters have been given an optimized value. As it is visible in the results, the pseudo method out performs the two other methods, specifically in terms of precision although the accumulation method is much better than the plain method. By seeing these results, it is in our best interest to implement a little bit of a complex gaze detection method, since the improvements on the results are dramatic.

To approach the real time scenario, there is a problem to deal with. The SIFT process computational time per frame is too high to work at the rate of the recording device (25 fps), hence the authors propose a solution. It consists on storing a ‘stand by’ image when a new fixation is detected, to catch up with the real time environment. To be direct, it makes the recognition unit of the system to be kept as busy as possible whilst the system always analyzes the newest fixation image.

In the graph below (Figure 3.4) a precision/recall results are shown for each experiment (the off-line, and the two real time experiments, with and without the compensation computed).

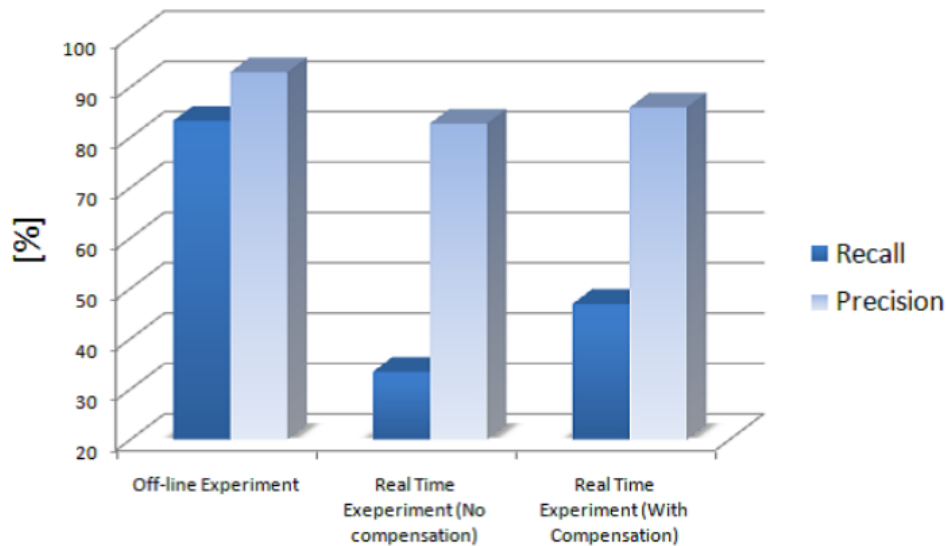


Figure 3.4 Comparison of Off-line and Real Time Results

Whilst the precision results seem to be pretty regular, recall results are clearly affected when the system is taken to the real time experiment. However, the time compensation helps remarkably to improve the recall results, which is interesting on the other hand.

To sum up with the review of this article, it is also interesting to talk about the feedback the users gave to this gaze detection system. Most of the users found more comfortable and preferred a gaze detection system rather than a simple audio player to learn about the exhibits. It is sensible, since the main advantage of this technology is that it can provide a direct and intuitive interaction to the user because of being linked to such a simple action as staring a certain exhibit. Therefore, if it is a thing that most users find useful it is worth investigating it and keep working on it.

3.2 Automatic Analysis of Eye Tracking Data by De Breugher et al (2012)

The large amount of data (video and metadata) produced by mobile EGT devices requires an analysis and a corresponding annotation. This annotation is a tough task to do manually, since it carries such a large set of elements to be taken care of. On the other hand, there have been investigations that try to deal with this issue by means of building an automatic annotation system. The paper we are going to analyze next [14] approaches that issue and suggests an implementation method of this system. Furthermore, it works with retail video data, making it interesting for its similarity with our intended project in this aspect.

The main idea of the paper consists on comparing the image region around the gaze data points looking for a closest match within a database of images using object recognition. This provides an automatic generation of statistics showing the time span of gaze fixation on relevant objects.

For the feature detection matter, the authors compare several techniques and its performances. The comparison of these techniques is shown in the graph below (Figure 3.5).

Algorithm	Rotation Invariant	Scale Invariant	Affine Invariant	Efficiency	Speed	#Keypoints	Correctness	Resistance to noise
SIFT	X	X		+++	++	++	+++	++
ASIFT	X	X	X	+++	+	+++	++	++
SURF	X	X		+++	+++	+	+++	+++

Figure 3.5 Features of several feature extraction techniques.

The chosen method for feature extraction was eventually the well known SIFT. Whilst SURF provides the fastest processing, it doesn't provide enough features. On the other hand ASIFT provides the best performance in terms of number of keypoints detected, plus it is as well Affine Invariant.. However, this method is dropped in the end, because of its low speed, taking SIFT as the best choice in this performance/speed trade off comparison.

Once the method used for feature extraction is chosen, the approach of how the system will work can be analyzed. It suits the typical training/experiment process. From a certain set of training data, features are extracted and manually annotated. These annotated features will form a database that later will be used for the test experiment part to query which is the best match for a certain new feature extracted from test. The scheme of this system is show in the following figure (Figure 3.6).

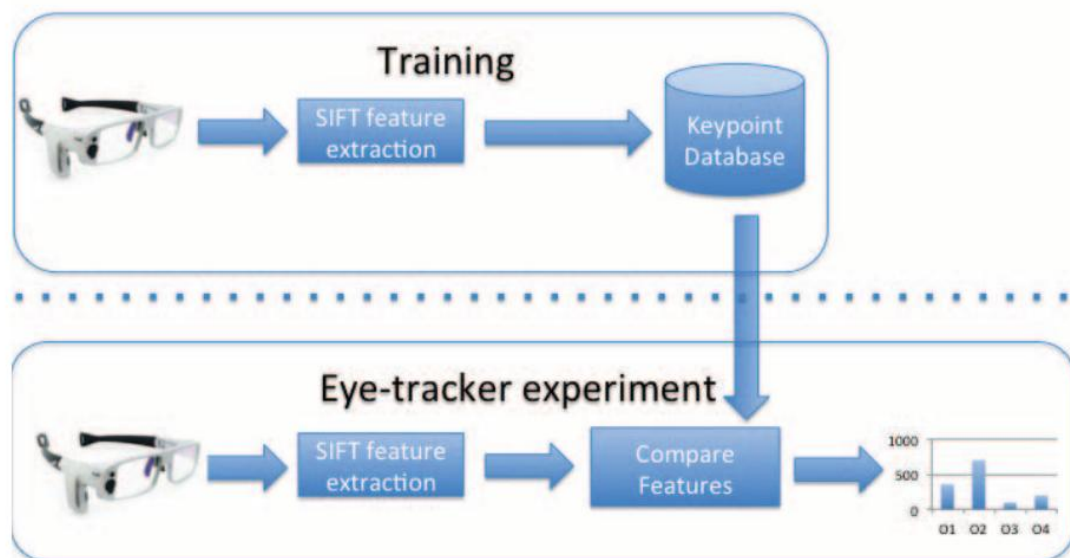


Figure 3.5 Scheme of the Approach.

To select a proper size of each region of interest, a 41x41 pixel size window is cropped around the gaze point for each frame. Features are extracted by means of SIFT in this cropped image that represents the ROI.

To perform the feature comparison on the other hand, a computing of distances between the new test features and all database features is done building an histogram, and an object is only recognized if it has the highest number of matches and exceeds a certain threshold.

3.3 Summary

In this chapter we have been able to review some previous works using EGT, thus giving us ideas to solve the problems we will have to face during our project. It is specially interesting the idea of cropping the image and take the N closer features to the gaze point for each frame to process. The part of weighting is also interesting and we will try to implement it as well to see if it works too for our dataset.

Another interesting idea that is explained in this chapter is the detection of a fixation. As we have explained previously, it is another focus of interest for us to filter the frames to process somehow. By only processing the frames where a fixation has been detected we get a coherent and faster system, hence we will try to approach this problem inspired by this idea.

References

- [1] Yarbus, Alfred L. *Eye movements and vision*. Ed. Lorrin A. Riggs. Vol. 2. No. 5.10. New York: Plenum press, 1967.
- [2] “Eye Tracking”, 2013. [Online]. Available: http://en.wikipedia.org/wiki/Eye_tracking
- [3] Jurie, F., & Triggs, B. (2005, October). Creating efficient codebooks for visual recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* (Vol. 1, pp. 604-610). IEEE.
- [4] “Harris Affine Region Detector”, 2013. [Online]. Available: http://en.wikipedia.org/wiki/Harris_affine_region_detector
- [5] “Scale-Invariant Feature Transform”, 2013. [Online]. Available: http://en.wikipedia.org/wiki/Scale-invariant_feature_transform
- [6] Xian-Sheng Hua and Hong-Jiang Zhang (2008), Scholarpedia [Online] Available: http://www.scholarpedia.org/article/Media_Content_Analysis - “Object detection”
- [7] A.T. Duchowski, A breadth-first survey of eye tracking applications, *Behav. Res. Methods Instrum. Comput.* (2002) 1–16.
- [8] Morimoto, C. H., & Mimica, M. R. (2005). Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding*, 98(1), 4-24
- [9] D.A. Robinson, A method of measuring eye movements using a scleral search coil in a magnetic field, *IEEE Trans. Biomed. Eng.* 10 (1963) 137–145.
- [10] A. Kaufman, A. Bandopadhyay, B. Shaviv, An eye tracking computer user interface, in: *Proc. of the Research Frontier in Virtual Reality Workshop*, IEEE Computer Society Press, 1993, pp. 78–84.
- [11] Tobii Glasses User Manual [Online] Available: http://www.tobii.com/Global/Analysis/Downloads/User_Manuals_and_Guides/Tobii_Glasses_UserManual.pdf
- [12] Toyama, T., Kieninger, T., Shafait, F., & Dengel, A. (2012, March). Gaze guided object recognition using a head-mounted eye tracker. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (pp. 91-98). ACM.
- [13] Difference of Gaussians (2013). [Online] Available: http://en.wikipedia.org/wiki/Difference_of_Gaussians.

- [14] De Beugher, S., Ichiche, Y., Brône, G., & Goedemé, T. (2012, September). Automatic analysis of eye-tracking data using object detection algorithms. *In Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (pp. 677-680). ACM.