TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology


# Children and Computer Vision

## BACHELOR THESIS

Performed with the purpose of obtaining the academic degree of
Bachelor of Science (BSc)

Under the direction of

Dipl.-Ing. Lara Lammer

Submitted to

Technischen Universität Wien

Fakultät für Elektrotechnik und Informationstechnik
Institut für Automatisierungs- und Regelungstechnik

by
Carlos Arregui
Matrikelnummer: 1327899

<Vienna, September 2014>

Gruppe Vision for Robotics

A-1040 Wien, Gusshausstr. 27, Internet: http://www.acin.tuwien.ac.at

# ABSTRACT

The present Bachelor thesis deals with the development of a education system, where children are the main objective. This system intends to transmit the concept of computer vision as well as basic concepts and ideas of this complex term. It is also explained how at some places this process of teaching computers, robotics or how to code has already begun, and it tries to go a step further at this kind of education.

The method chosen has not been going directly to the concepts but explaining these concepts through a story, in order to capture the interest of the children and after that going with the explanations. In this work several possibilities at each algorithm are presented, but only one has been chosen, due to several reasons. Finally, an evaluation has taken place, putting the method in practice with two students, proving this method as an efficient solution.

# INDEX

*1. Introduction……....7*

*2. State of the Art……..8*

*2.1 Computer Vision……....8*

*2.2 Robotics and Children……13*

*3. Method……17*

*4. Story……18*

*5. Algorithms……....22*

*5.1 Background Subtraction……..22*

*5.2 SIFT and image alignment……..23*

*5.3 Object recognition……....24*

*6. Evaluation……....26*

*7. Conclusions……27*

*8. References..……....28*

*9. Annexes……...30*

# TABLE OF CONTENTS

2.1 RGB colours generation……..……10
2.2 Scratch blocks interface……14
2.3 App Inventor blocks interface……...15
2.4 Raspberry Pi……...15

4.1 Superhero look (with glasses)……18
4.2 Trophy subtracted detection……19
4.3 Panoramic picture……20
4.4 Object recognition……20
4.5 Trophy back in the cabinet……21

5.1 Background subtraction window algorithm……...23
5.2 Keypoints matched (SIFT)……..24
5.3 Object detection algorithm (by colours)……25

# 1. Introduction

Nowadays technology has bigger and bigger importance in our daily lives, and this field evolves with big steps. It is important to know how to use this technology relevance, and computer vision applications are really impressive and useful.

Besides, children coexist with this technology since they are born, so they usually touch, use and live with this; and would be better for them if they receive an education where technology has an important role within it.

That is the reason of doing this Project, which main goal is to teach some children what computer vision is and why is so important for us to keep developing it. This goal would be really profitable for them because if they have this kind of education, they will be conscious of how big this field is and will be easier for them to study about it at the University or at any other place.

Several methods have been thought, but finally it has been decided to do this difficult task by telling to the children a little story and let them draw their own conclusions after it. But the technical methods used are several and different, not only in terms of results but also in terms of computational cost and complexity of the algorithms. These methods are, among others, Background Subtraction, Sift and more concretely panoramic photos applications, and Object Recognition through K-means and Image Processing techniques.

In this memory is also explained from more a theoretical point of view the amount of applications that computer vision has and practical examples were children are taught to code since really early ages. Attached at the end of it, an annex with the codes can be read.

# 2. State of the Art

## 2.1 Computer Vision

The first question should be wondered is:

What is computer vision? Computer vision is a field, with lots of subfields within itself and with the main goal of getting information from pictures or videos from the real world. It is wanted the computer to acquire this information we easily see with our eyes. This information the computer absorbs is in the shape of numbers, so computer vision methods are based on acquiring, processing, analyzing and understanding images from the real world in order to produce numerical information. Several science fields, such as geometry, physics, statistics and mathematics are essential and constitute the very important theoretical base.

As humans, we perceive easily the world around us. However, for computers this is much more difficult. We sometimes do not appreciate the complexity of our brain, and when we try to imitate it, then we realize how complex it is. Computers have already achieved several goals with success like reconstructing a 3D image, tracking a person moving, even recognize people by face recognition. Whilst that, they are not even close to a 3 years old child vision.

It is a very interesting field where investigate due to the big amount of applications that computer vision have [1]. Some of these are:

- Optical character recognition
- Machine inspection, controlling processes
- 3D models reconstruction since 2D images
- Medical applications
- Automotive safety
- Motion capture
- Interaction (computer-human interaction, robot-human interaction)
- Fingerprint recognition
- Organizing information

In fact, these are actually current applications, but when we visualise the future applications computer vision may have we realize why we should not stop trying it. Some of these future applications could be to place robots in factories to inspect items and detect defective material, we also could rely on computers for medical purpose in hospitals, for instance. Besides, computers could aid blind people in their daily tasks, and oil and other resources could be detected by examining the surface of location.

One of the points for computer vision is that there is not only one technique to solve a problem but several of them. The initial problem has to be identified, after that possible solutions are thought and finally these solutions are compared each other. Then the best one is chosen once the performance of each one has been seen. It is also important to keep in mind is the fact of analyzing noise sensitivity after analyzing the algorithm correctness, with more complex backgrounds or images. Thus, a statistical approach to an algorithm is important to evaluate it. To conclude this aspect, an algorithm has to fulfill mainly two requisites: being robust to noise and being efficient.

There are several subfields inside computer vision like image formation, image processing, features, rendering, recognition… All of them are important and cover different parts of computer vision, some of them help to understand the essence of processes and others already built efficient and robust algorithms ready to be used. These aspects are going to be briefly explained one by one.

### 2.1.1 Image processing

First of all, the image has to be taken, and then the computer will process it. This image can be taken by several methods. A normal digital camera which is connected to the computer by USB, the computer webcam itself is used or a camera which takes pictures due to sensors, for instance those in charge of the traffic control are some of the possibilities. In case the camera which takes the picture is analogic, the image has to be subjected to a process called digitalization.

Another possibility is that the input is a video instead of a picture. Videos are compound of lots of frames. Each of this frames contains the same information than a normal picture, so they can be taken for the subsequent analysis.

On one hand, a digital image is a two-dimensional matrix of HxW (HeightxWidth) elements. Each element is known as pixel, and each pixel is bound to a value, usually between 0 and 255, which means the light intensity of that pixel. The reason of these numbers is due to the high graphic capacity that nowadays computers have. So they use the ‘true color’ technology, also called RGB (Red, Green, Blue). Three different channels are used, one for each colour, and the resultant image is a combination of the three primary colours. Each pixel is defined by

3 bytes, one byte for each colour. These bytes are compound of 8 bits, so they can take a value between 0 and 255, as said before [2]

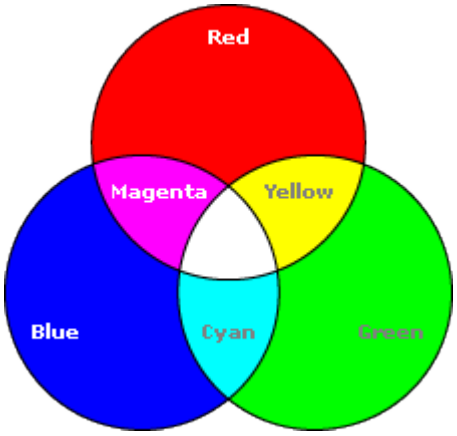


***Figure 2.1**: RGB colours generation*

On the other hand, before a computer vision method can be applied to image data in order to extract some piece of information, it is usually necessary to process the digital image, mainly due to the high computational cost it would have to work with this unspoilt image. This kind of processing leads us to two easier and lighter images:

The grayscale image is out of colours, it only represents gray tonalities. These kind of images requires less information than original digital images, because it is only needed one matrix instead of three. In other words, it is only specified the gray intensity by pixel. The value 255 would be completely black and 0 would be completely white. That is the reason why they are also called intensity images.

There is still another kind of image, the simplest one, the binary image. This binary image only shows two values: 0 and 1. They are commonly called as black and white images.

## 2.1.2 Feature extraction

Image features at several levels of complexity are extracted from the image data. The most usual features needed to be detected are:

- Specific locations in the images, such as mountain peaks, buildings corners, doorways… These are called keypoint features. They are often described by the appearance of patches of pixels surrounding the point location.

- Edges, ridges and lines. These features are a bit more complex than key-

points, but usually they also reveal more information.

Here the keypoints have been used because they are easier to use and to work with. There are two main and different approaches to finding feature points: features can be accurately tracked using a local search technique such as correlation or least squares, or can be detected independently in all the images and then match features based on the local appearance. The first approach is more suitable in frames of videos, while the second one is usually used when there are more changes. [3]

Those points with a high contrast are much easier to work with. If, in addition, the contrast exists in more than one direction it results much better. It is important to be careful if too much keypoints are detected in the same area. The solution to this problem is to measure this repeatability, measuring the frequency of keypoints found within 3 pixels, for example.

Once these keypoints are detected, feature descriptors are used in order to match them. Keypoints locations change in each image, and scale may also change between them. That is the reason why invariant image descriptors have been developed. This is the origin of the SIFT (Scale Invariant Feature Transform), which I will use later to develop one code. There are more complex variations such as PCA-SIFT or SURF.

### 2.1.3 Segmentation

Segmentation, when related to computer vision, is a process that consists on dividing a digital image in several parts or objects. Segmentation is used in both locating objects and finding their boundaries. More concretely, segmentation is a process where each pixel is assigned to a label, so that the pixels with the same label will have some similar visual characteristics.

Basically, there are two types of segmentation methods: those based on borders and those based on clusters. The first one consists on detecting the object borders. Once these borders are detected, the following process is quite easy, but the fact of finding these borders is the difficult task. On the other hand, in those methods based on clusters what is needed to be done is to highlight the clusters with similar characteristics. Usually it is easier if characteristics are easily recognizable, such as different colours, brightness or gray levels. [4]

Actually there are many types of segmentation (clustering, based on the histogram, boundaries detection, regions growing methods, watershed, threshold methods...) but this one would be a easy way to classify them. Later on those methods based on clusters will be explained a bit deeper, since they were the ones we worked with here.

## 2.1.5 Recognition

This section deals with recognition, which is an application itself in most occasions. For instance, is really useful when our digital camera recognize human faces before the photo is made or in some security service. For this being possible it is needed a previous face detection.

There are mainly three different valid systems for the face detection, but the main one is features based; it was the first method being developed, and it simply seeks some characteristics such as eyes, nose or mouth. Currently in use is the one introduced by Viola and Jones (2004), with the particularity of being the first method to introduce the concept of boosting. Once inside face recognition, face recognizers only work with the entire image, a high contrast and a uniform background. It is still being improved through time and it will be better in some years. [5]

## 2.2 Robotics and children

Despite the fact that it may seem a bit strange to teach robotics and things related to computer vision or coding to children, there are many reasons why we should really think about it. In a world where everything is on the Internet, everything is based on technology, what better than teaching children how to understand this complex concept? And which better way to teach this that with concrete concepts?

Many studies have been made and it has been conclude that teaching robotics to children is usually positive. There are several reasons, for instance, if they learn robotics since early ages, is easier to acquire and apprehend complex concepts. If children grow up with robotics at their education, it will be a familiar concept for them when they will be older. Not only robotics, but also everything related to it. If these concepts are solidly learnt, then at university it will be quite easier to go in depth for them.

Another remarkable reason is the fact that, if these things, as coding for example, are well-taught, children may conceive it like a game. They would learn something really important while they think they are playing. If all this is led the right way, coding may be something really interesting and enjoyable for them. This characteristic will open children mind about technology, and what it is even more important, it will stimulate their curiosity.

Curiosity is one of the most important concept when we talk about education. If it is achieved to arouse their curiosity, most of the work done is worth it. Because arousing this curiosity is the door they will use to go in depth into this topic. Besides, if this happens, they will not fell bored with the topic and they will keep thinking about it as a game and will want to keep learning it.

Due to all these reasons, education has realized all the profits it has and it has already started to develop several methods or systems to make this possible. Consequently, several special code languages, smartphones applications or single-boards computers have been invented. The list of services provided below are some concrete examples that are already developed and working at some areas. [6] [7]

### 2.2.1 Scratch

Scratch is a tool, created by ‘Kindergarten’, at the MIT (Massachusetts Institute of Technology) Media Lab. The team was led by Mitchel Resnick, and the first appearance it was in 2007. This special tool allows to code to beginners or people who have never code before. It is possible to achieve concrete results without writ-

ing in the right way. Therefore, Scratch is a tool which allows an independent learning. The adjective we would use to describe it would be intuitive. It includes simulations of experiments, the possibility of creating animated stories, and interactive art and music. As their slogan says: 'Imagine, Program, Share'.

This intuitive code language allows children to learn how to code through different stories, challenges. Step by step, with different colours, forms and filling gaps, they learn the importance of how to put in different blocks both their thoughts and then the orders they want to transmit. In other words, they learn to think in a more effective way.

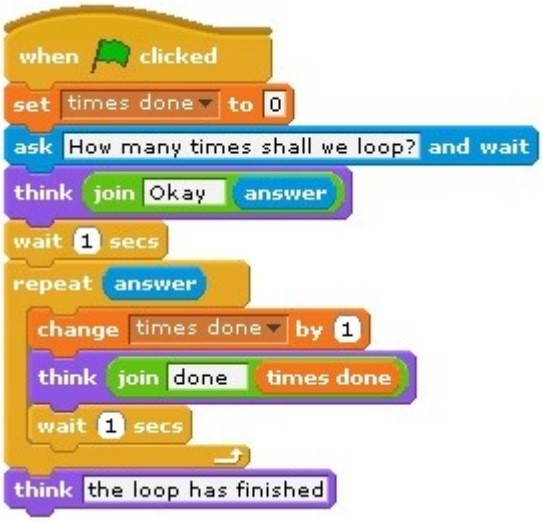


***Figure 2.2:*** *Typical Scratch blocks interface*

Scratch is a real alternative which has been working since several years ago, where it exists a community of users and more than six million of projects have already been done and shared. It is thought to be used for ages between 8 and 16 , but older people use it too. It is widely used all around the world, in more than 150 countries and it is available in more than 40 languages. Scratch is being used at schools, at primary school and at university as well. [8]

Besides Scratch, there are also more current initiatives going on, as I explain in the following paragraph.

## 2.2.2 Concrete robotics in education

Some MIT students created App Inventor, which is a great application for those ones who are interested in creating android applications but they do not have the knowledge enough for knowing how to do it. The possibilities offered are incredible. App Inventor is simple, but also powerful. With a visual form and with a group

of basic tools, the user can put some blocks together in order to create an android application. App Inventor was created in 2009 and is been already used by 12 year-old children and users of several web educative centers. [9]

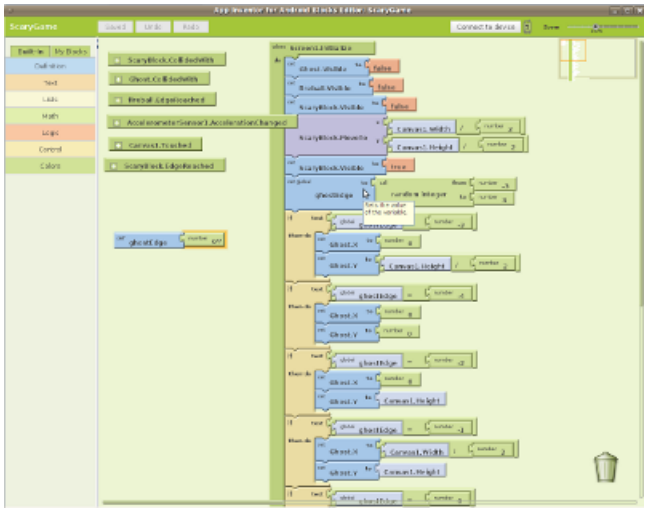

***Figure 2.3:*** *App Inventor blocks interface*

Another example would be the Raspberry Pi, which is a single-board computer with the size of a credit card, developed in the UK with the intention of promoting the teaching of basic computer science in schools. This has been already used in both public and private schools. This credit-card sized computer is done in order to plugging it into a computer monitor or TV. This little device enables people of all ages to explore computing and to learn how to program in languages like Scratch or Python. It is capable of doing everything we would expect a desktop computer to do. The purpose Raspberry Pi was created with was that children all around the world were able to use it, to learn to program and understand how computers work. [10]

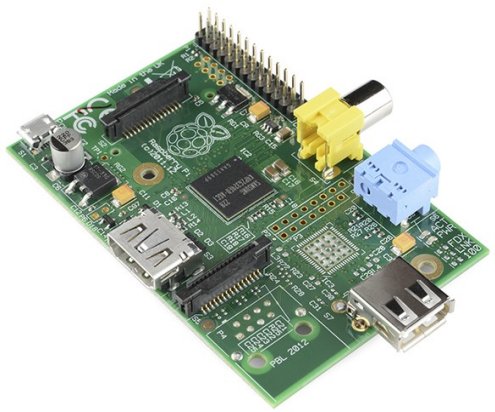

***Figure 2.4****: With this little device everything we do in our desktop computer can also be done*

On the other hand, apart from these diverse platforms, there is an issue that is being controversial at some countries: teaching children how to code since primary school. There are lots of debates about this, because education is usually a traditional field where radical changes are difficult. But little by little many countries have realized how important and how beneficial this could be and they are introducing it at their educational system. Several ones could be examples here, but the first one who did it was Estonia, where Skype was created. At Estonia, schools were connected by Internet since late 90's, and several years ago they started working with teaching how to code at secondary school. The novelty now is the same teaching but at primary school, when children already have the capacity of acquiring the main concepts. The governmental program responsible of this initiative is called 'Tiger Leap', which is going on since 1996. Another similar efforts are being carried out by another countries like England, where code teaching at primary school will be compulsory from next year on. Another remarkable country at this situation could be Canada, where exists 'Code for Kids', founded by the Ottawa University, and is currently active in Toronto, Montreal or Kingston. However, to foment and teach technology to children, educating them with coding is not all. Getting used to electronic devices since early ages and showing them how much they can make, learning and creating with them is also inside the process. This is done at some Chicago schools, where 5 or 6 years-old children interact with computers with special software like 'Reading Eggs'. [11]

To sum up, technologically education is little by little making inroads in these education area at a primary and secondary school, although is still far away to be at the same level as artistic or scientific education. It is a relatively new topic and it is also a bit complex in a educational way. But soon it will occupy a prominent place at education, for sure.

# 3. Method

Once this little research was done, it was time to contribute to this technology education in a special way. Keeping in mind how hard it is to introduce this technology field, or more concretely, computer vision field, into children's (11-12 years old) mind, it was decided to work scientifically in a method. For considering scientific a method, this investigation method has to be based on empirics and measurement. The two main pillars over scientific method is based are:

- To let the reality speak for itself, which means that an experiment can be repeated, wherever and by whoever.
- It has to be refutable, that is to say, the theory is supported when predictions are confirmed but the theory is challenged when its predictions prove false.

So, with all this in mind, the story-method I spoke at the beginning of this memory was created. The intention was to transmit knowledge through a story, in order to make enjoyable the topic I am talking about. After the story was in their heads, they would have to draw their own conclusions. The measurement was thought to be statistically, but due to the difficulty of this kind of measurement, it was thought to check the method with real persons to see if worked.

Therefore the story was told to two university students distant to the technology field. They were told to think like a 12 years old, and in reality it was similar because the initial knowledge about the topic was the same, absent. After that, they were asked about the different impressions they had and what they had understood. The different results these interviews had will be told later, when the evaluation itself will be also done.

At first the story will be written with the same images and a similar explanation that were used while the story was told to these two students. After that, we will go in depth about the different concepts and algorithms used during the work.

# 4. Story

Kyle was a normal boy. He was at the school, despite the fact he did not like to go, with a funny group of friends and he belonged to the basketball team. He was 14 years old, and when he turned 15, tired of seeing a sad world, he decided to be a kind of superhero. He wore a green costume designed by himself, and with an old engine that belonged once to his father, he devised some equipment, with which it was possible to fly some meters.

He tried to do good deeds at his environment: at school, in the city… Little by little he was achieving to turn everything around him into better. But, one unlucky day, something disastrous happened. At chemistry practices in the lab, while an experiment took place, a little quantity of acid went into his eyes. However, he did not lose completely the sight, but he did not see clear and all the images he perceived were a little blurry. He did not want to stop what he had started, and as he had received some classes about informatics, he invented a special kind of glasses. These glasses had a camera on one side, and a little processor inside them. The camera captured all kind of images and after that, the processor was in charge of process and understand that image. In this way, with the help of his 'magic glasses', he was still able to do everything he proposed to himself, and was also able to keep helping the rest of the people.

***Figure 4.1:*** *This was the complete look Kyle had once the magic glasses were invented.*

He always stored many images in the glasses memory. That glasses were really useful, and had lots and lots of applications. For example, with them it was possible to detect all the objects with the same colour, it was able to know exactly who was who due to the application of face recognition, or to make panoramic pictures

taking 5 pictures of the same background. It was possible to count how many objects there were on a table, and to construct a 3D model through the 2D pictures they captured.

Due to the great amount of images stored in the glasses memory, while he was hanging out with some friends at school, as they passed basketball glass cabinet side, Kyle realized the trophy they had won last year was missing! (*background subtraction)*

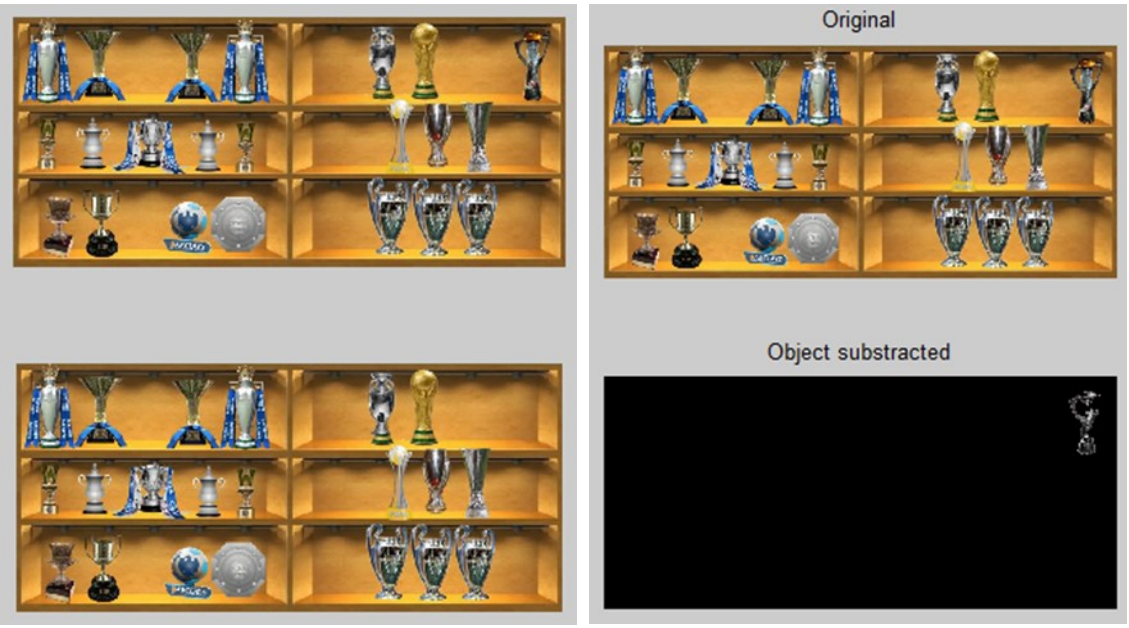


***Figure 4.2:*** *On the left side can be seen the original image, and below the current image, with the trophy missing. On the other hand, on the right side can be seen how Kyle's glasses detected the trophy that was missing, on the corner of the glass cabinet.*

Of course, Kyle first objective was to find out who had stolen it and he wanted to recover it too. He already supposed who had done it, and one day in the school library, through his special glasses, which gave him the ability of a panoramic vision (*sift alignment and panoramic construction*), he found the one who did it.

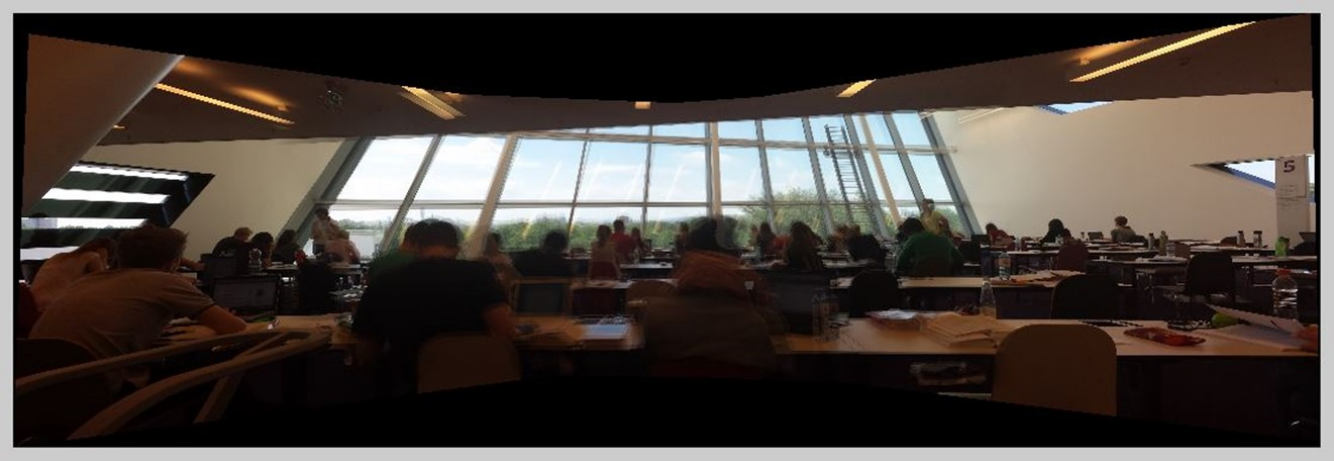

***Figure 4.3:*** *Here it is seen a panoramic picture, as we can see them when smartphones do them. It is done with five pictures taken from the same pont of view.*

He waited patiently, and in the moment this guy went to the toilet, Kyle took advantage of the situation and came quickly to get a picture of the objects placed on his table. The picture taken is the following one:

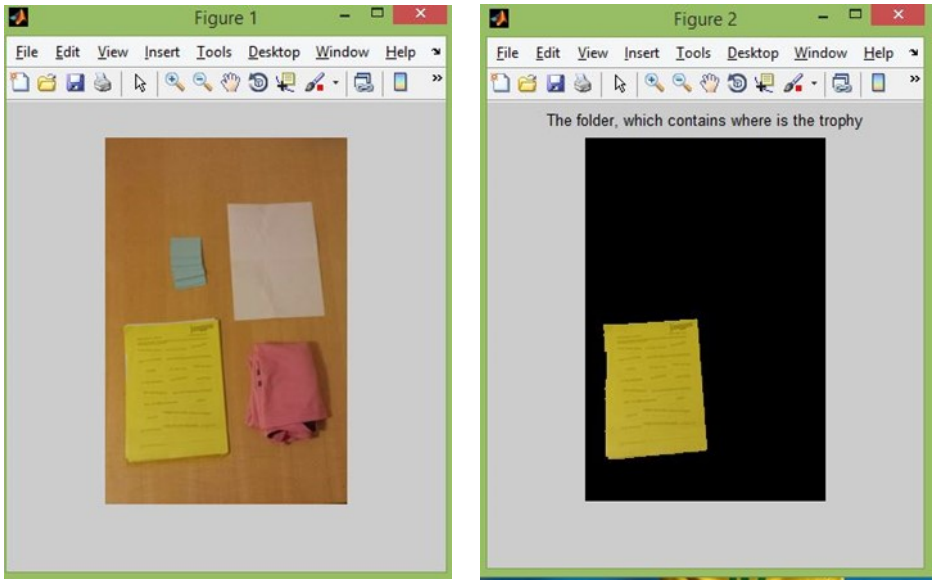


***Figure 4.4:*** *At the image on the left side can be seen the picture Kyle took of the objects placed on the table, and at the other image once the object recognition, through the different colours of the image, was done.*

Again, through his magic glasses, he could even recognized the different objects there were on the table! (*object recognition*) Do you imagine something similar could have happened if Kyle had not invented his glasses? With the information there were on the papers he knew where the trophy was. Finally, he went where the trophy was hidden and gave it back to the school glass cabinet, as shining as usual.

***Figure 4.5:*** *The trophy back in the glass cabinet*

Once with the trophy back in the glass cabinet, Kyle was really happy that his invent had work so well, and everybody at the school congratulated him personally. Through his glasses he had been able to overcome his accident. Without them he had been a visually disabled boy and of course this little story had not taken place. It had been possible to notice the trophy missing, simulate a panoramic vision only with 5 pictures, or to recognize several objects placed on a table. Now that this little story has been told, imagine by yourselves all the possibilities that this, computer vision, offers.

# 5. Algorithms

For making this possible, a previous Matlab work took place. The three main algorithms with which I have worked are:

- Background subtraction
- SIFT and Image alignment (panoramic picture)
- Object recognition. At this task two algorithms were used: segmentation and color association.

These algorithms were chosen due to they were useful and quite illustrative for the commitment we had. Below it is going to be briefly explained the fundaments of each algorithm, and how they were done.

## 5.1 Background subtraction

Background subtraction is a technique used in computer vision where an image's foreground is extracted for further processing. On one hand, the background is what is fix in a group of images, or in every frame of a video. On the other hand, foreground are those things which are in movement, or these things that appear at different locations.

The background subtraction is usually used with videos, and the algorithm itself is quite simple. As it has been explained before, a digital image consists in a matrix with different values in every pixel. Well, this algorithm lies in comparing pixel by pixel two different images. So what is being compared is the intensity of each pixel. When two images are very similar, when you subtract one from the other one, the resultant matrix will have a value of approximately 0. If these two images present some changes, when the subtraction is done, the resultant matrix would be 0 at the similar parts and higher values (here we work with absolute values) at these parts that change. Here is where the threshold takes value. The choice of the threshold is made in function of the sensibility we want for our background subtraction. If this threshold is really low, the sensibility would be very high, and vice versa. The values of the resultant matrix which are higher than the threshold are considered foreground. [12]

```
% ------------------------------------------------------------------------

fr_diff = abs(double(i2_bw) - double(i1_bw));

imshow(uint8(fr_diff))

for j=3:width-2
    for l=3:height-2
        if (mean(mean(fr_diff(l-2:l+2,j-2:j+2))) > th)
            fg(l,j) = i1_bw(l,j);
            bfg(l,j)= 1;
        else
            fg(l,j) = 0;
            bfg(l,j)= 0;
        end
    end
end

 bg_bw = i2_bw;

% ------------------------------------------------------------------------
```

***Figure 5.1:*** *The main part of the background subtraction algorithm.*

As can be seen in the picture above, the algorithm consists in deducting one image from the another one, once the images have been read, transformed into grayscale images, the threshold have been established and the size (height, width) of the images measured.

Once the resultant image is obtained, is compared with the threshold. However, comparing pixel by pixel could lead to some mistakes if the camera would have shaken at the moment of taking the picture. That is the reason why a pixels window is made in order to be a more precise process. Then, the average of the window is done, and if it is higher than the threshold, is considered foreground. This is the background subtraction, and in the story is the process that Kyle glasses make to realize the trophy is missing.

This matlab code can be seen at the annex 9.1

## 5.2 SIFT and image alignment

SIFT (Scale-invariant feature transform), is one of the most popular algorithms used in computer vision. It is based on detecting at first the keypoints of the images, that is to say, the most ‘important’ features of the images, in order to recognize objects, motion capture or pictures alignment, which particularly is the case used at Kyle’s story.

The algorithm used is the following one: first, features in all images are detected. After that, pairs of corresponding features between adjacent images are found. The wrong matches have to be eliminated, by homography between images. These homographies are applied to every image, and finally, colours of overlapped images are blended. [13] [14]

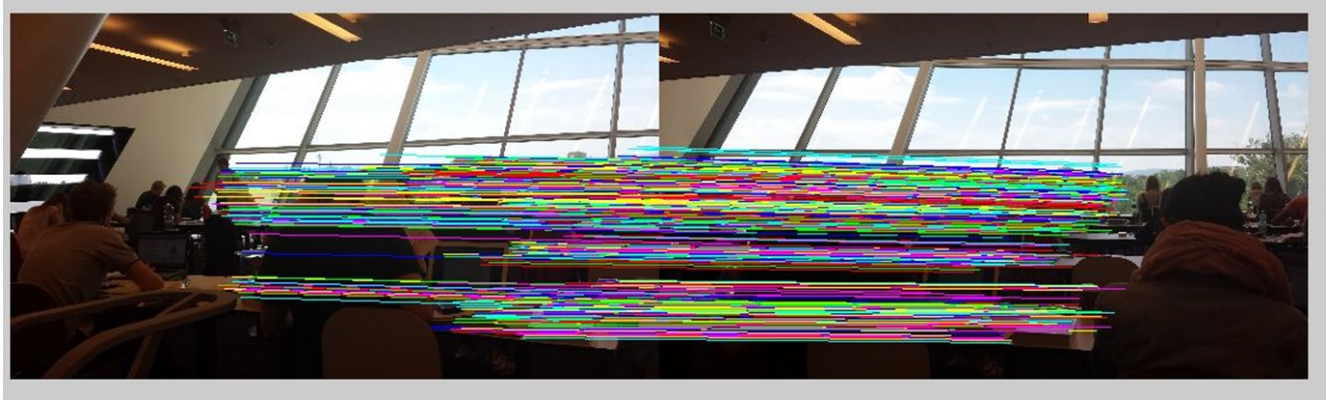

***Figure 5.2:*** *Two of the images once the sift has been done and the keypoints matched. After that the wrong matches have been corrected and this is how it remains.*

The matlab code can be seen at the annex 9.2

## 5.3 Object Recognition

Object recognition is not properly an algorithm but an aim, and there are several algorithms to achieve it. In my particular case was chosen at first doing a segmentation, and once the segmentation was done, characterizing each object by its colour. The segmentation was done with a method based on clusters, as said in the section '2.1.3 Segmentation'. A segmentation based on the edges would be also possible, but is usually more complex and it requires a really good contrast at the outlines. Personally segmentation by clusters was chosen because it is easier and simpler to implement and then to explain the children a segmentation where colours took an important value in it. [15]

More concretely the technic used was k-means, which is a method of clustering. This method consists in:

1. A number of k clusters is chosen. This choice can be done randomly, by a heuristic method or manually, as was done here due to the easiness of knowing the number of objects on a clear and tidy table.

2. Assign every pixel of the image to the cluster that minimizes the variance between the pixel and the cluster center.

3. Recalculate cluster centers making the average of every cluster pixels.

4. Repeat second and third steps until the algorithm converges.

The variance named at the second step of the algorithm consists in the absolute difference between a pixel and the cluster center. This difference can be calculated with the colour, the intensity, the texture, the pixel location or the mean of all of them. Here the colour was used as the distinguishing factor. [16]

```
[i3_r , i3_c] = find (i3gray > th);
m31 = mean(mean(i3(i3_r, i3_c, 1)))
m32 = mean(mean(i3(i3_r, i3_c, 2)))
m33 = mean(mean(i3(i3_r, i3_c, 3)))

[i4_r , i4_c] = find (i4gray > th);
m41 = mean(mean(i4(i4_r, i4_c, 1)))
m42 = mean(mean(i4(i4_r, i4_c, 2)))
m43 = mean(mean(i4(i4_r, i4_c, 3)))

if((m11>100) & (m12>100) & (m13<50))
    figure (2)
    imshow (i1), title ('The folder, which contains where is the trophy')
%     write ('This object is the folder, here is said where is the trophy')
end

if(m21>100) & (m22>100) & (m23<50)
    figure(2)
    imshow (i2), title ('The folder, which contains where is the trophy')
%     write ('This object is the folder, here is said where is the trophy')
end
```

***Figure 5.3:*** *As is explained below, this is part of the final algorithm of object recognition*

Once the segmentation was achieved, the initial image is converted in four different ones, within them there were only one of the objects. Now the objective is to know which one is the one which contains the folder. The algorithm used is the following: four images are converted into grayscale ones, and once they are grayscale images, and with a 10 threshold, is detected where (width and height of the pixels) the image remains. At those pixels, an average of the three RGB colours is done, and the one that fulfills the yellow colour conditions (approximately >100, >100, <50), has to be the folder.

The code can be seen in more detail in the annex 9.3

# 6. Evaluation

With the whole work done, an evaluation of the method talked before was needed. This evaluation took place. However, it was not in a statistically way, but just put in practice. The story was told to two persons at the same time. Both of them were far away of the technology/computers field, and they were notified in advance the normal conditions this test would have, being told to children of 11-12 years old. The first person is 22, economics student at the University of Valencia, male. The second person is 20, student of biotechnology at University of Tarragona.

At first, they did not even know what computer vision was. Kyle story was told to them and they listened to it carefully and with much interest. When the story was finished they were totally curious about computer vision and the wide possibilities it may have. The most positive point was this curiosity showed by them, during and after the story. One of the main goals of the project was done, create curiosity and a great attitude to computer vision. They understood the applications shown in the story, but obviously they were not able to imagine all the possibilities existent. Nevertheless, they connected this computer vision field with other daily applications, like face detection at digital cameras or panoramic pictures at smartphones.

Once this was done, the next step was going in depth into the 'coding'. Matlab interface may scare at the beginning, even more if you have never seen it before. But the explanation was not line by line but it was with the algorithms, explaining them by blocks what had been done and which was the idea of every step, of every block. It was a bit difficult at the beginning because they focused on every single word, and sincerely is quite difficult to understand at first sight. But when they listened attentively, they were little by little catching the main ideas and the thinking of working by blocks.

So, evaluation of the method here was over, and the results were clear. As said before, the curiosity and interest was created, and this was showed due to the questions they made during and after the story. Some computer vision applications were identified, and the general concept was correctly transmitted. On the other hand, more in the code writing thing, concepts were finally transmitted too but with a little bit of difficulty of understanding.

# 7. Conclusions

After the evaluation finished, it was time to draw conclusions of how the project had worked. The idea of transmitting the computer vision concept and getting the children attention through a story worked, so this can be thought as a wise choice. Perhaps with a higher code level, more examples could have been given. Main goals set at the beginning have been achieved.

One alternative, having watched the evaluation, would have been to show the code to the interviewee in a different way. For instance, and as it is possible to see in the *'2.2 Robotics and children'*, there are several code languages that have been done specially to children, like Scratch.

So this alternative would consist in telling a similar story, but when showing the code and giving an explanation of how it has been done, instead doing it with the Matlab code, doing it with a Scratch interface or something similar. Divided by colours, by blocks, with a visual structure. This would help and results would be quite better.

Despite this, and knowing that perhaps more efficient methods could also have been worked, in my opinion results have been generally positive, ant the project has been developed with an acceptable success.

# 8. References


[1] Szeliski, R. *Computer Vision: Algorithms and Applications.* Springer, London, 2011. pp. 1-20.

[2] Mendieta, D. *Reconocimiento de Objetos Bidimensionales en Imágenes mediante la Transformada de Distancia utilizando Matlab.* December 2003. [Checked on May 2014] http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/mendieta_d_d/capitulo2.pdf

[3] Szeliski, R. *Computer Vision: Algorithms and Applications.* Springer, London 2011. pp. 181-209.

*[4]* Garcia Santillan, E. *Detección y clasificación de objetos dentro de un salón de clases empleando técnicas de procesamiento digital de imágenes.* May 2008. [Checked on June 2014] http://newton.azc.uam.mx/mcc/01_esp/11_tesis/tesis/terminada/080513_garcia_santillan_elias.pdf

[5] Szeliski, R. *Computer Vision: Algorithms and Applications.* Springer, London 2011. pp. 575-601.

[6] Naughton, J. *Why all our kids should be taught how to code.* The Observer (The Guardian). March 2013. [Checked on June 2014] http://www.theguardian.com/education/2012/mar/31/why-kids-should-be-taught-code

[7] Evans, J. *Computer Classes for Kids: Why Programming Is (and should be) Taught Earlier.* Education.com. April 2013. [Checked on June 2014] http://www.education.com/magazine/article/computer-classes-for-kids/

[8] MIT project Scratch. [Checked on March 2014] http://scratch.mit.edu/about/

[9] MIT project App Inventor. [Checked on March 2014] http://appinventor.mit.edu/explore/about-us.html

[10] Raspberry Pi Foundation. [Checked on March 2013] http://www.raspberrypi.org/

[11] Villasana, J. *Acciones públicas para enseñar tecnología en los colegios.* May 2014 [Checked on July 2014]. http://mobileworldcapital.com/es/articulo/4367

[12] Gallardo, A. S*imulador de algoritmos de estimación de background con aplicaciones docentes.* January 2014, pp. 21-34. [Checked on May 2014] https://upcommons.upc.edu/pfc/bitstream/2099.1/20779/4/memoria.pdf

[13] Naveen, C. *SIFT (Scale invariant Feature Transform) Algorithm.* October 2013. [Checked on April 2014] http://www.mathworks.com/matlabcentral/fileexchange/43723-sift--scale-invariant-feature-transform--algorithm

[14] Image Alignment Toolbox. *Non-rigid alignment with SIFT-flow.* 2014. [Checked on June 2014] http://iatool.net/tutorials/siftflow/

[15] Electronic, Automatic and Informatics Department. *Prácticas de Segmentación de Imágenes.* University of Madrid. [Checked on June 2014] http://www.elai.upm.es/webantigua/spain/Asignaturas/Robotica/PracticasROVA/prROVA5Segmentacion.pdf

[16] Documentation Center. *Color-Based Segmentation Using K-Means Clustering.* Mathworks. [Checked on June 2014] http://www.mathworks.es/es/help/images/examples/color-based-segmentation-using-k-means-clustering.html

# 9. Annexes

## 9.1 Background subtraction

```
clc;
clear all;
disp('Running BSubstraction1.m...');
i1 = imread('esc2.jpg');
i2 = imread('esc3.jpg');
% subplot(2, 2, 1);
% imshow (i1);
% set(gcf, 'Position', get(0, 'ScreenSize'));
% drawnow;

th = 30;
i1_bw = rgb2gray(i1);
i2_bw = rgb2gray(i2);

% --------- Defining variables ----------------------------------------

fr_size = size(i1);
height = fr_size(1);
width = fr_size(2);
fg = zeros(height, width);
bfg = zeros(height, width);

% ------------------------------------------------------------------------

fr_diff = abs(double(i2_bw) - double(i1_bw));

imshow(uint8(fr_diff))

for j=3:width-2
    for l=3:height-2
        if (mean(mean(fr_diff(l-2:l+2,j-2:j+2))) > th)
            fg(l,j) = i1_bw(l,j);
            bfg(l,j)= 1;
        else
            fg(l,j) = 0;
            bfg(l,j)= 0;
        end
    end

end

 bg_bw = i2_bw;
 % ------------------------------------------------------------------------
```

```
figure(1);
subplot (2, 1, 1), imshow(i1)
subplot (2, 1, 2), imshow(i2)

figure(2);
subplot(2, 1, 1), imshow (i1),            title('Original');
subplot(2, 1, 2), imshow (uint8(fg)),            title('Object substrac-
ted');

figure(3)
subplot(2,2,1),imshow (i1),                 title('Original');
subplot(2,2,2),imshow(uint8(bg_bw)),        title('Background');
subplot(2,2,3),imshow(uint8(fg)),           title('Foreground
(Grayscale)');
subplot(2,2,4),imshow(double(bfg)),         title('Foreground (Binary)');
```

# 9.2 SIFT and image alignment

```
function imageStitchingMain()

global showFigures;
showFigures = true;

img1 = imread('b1.jpg');
img2 = imread('b2.jpg');
img3 = imread('b3.jpg');
img4 = imread('b4.jpg');
img5 = imread('b5.jpg');

 %SIFT
 % Conversion to single is recommended in the documentation
img1S = single(rgb2gray(img1));
img2S = single(rgb2gray(img2));
img3S = single(rgb2gray(img3));
img4S = single(rgb2gray(img4));
img5S = single(rgb2gray(img5));


% detect features with SIFT
[F1 D1] = vl_sift(img1S);
[F2 D2] = vl_sift(img2S);
[F3 D3] = vl_sift(img3S);
[F4 D4] = vl_sift(img4S);
[F5 D5] = vl_sift(img5S);
```

```
% match SIFT features
[matches_12 score_12] = vl_ubcmatch(D1,D2);
[matches_23 score_23] = vl_ubcmatch(D2,D3);
[matches_43 score_43] = vl_ubcmatch(D4,D3);
[matches_54 score_54] = vl_ubcmatch(D5,D4);


% plot features onto image
% plot matched features
if(showFigures)
    points1 = horzcat(F1(1,matches_12(1,:))', F1(2,matches_12(1,:))');
    points2 = horzcat(F2(1,matches_12(2,:))', F2(2,matches_12(2,:))');
    figure;
    imshow(img1);
    hold on;
    vl_plotframe(F1);
    hold off;
    match_plot(img1, img2, points1, points2);
end

T = 5;
N = 1000;

H_12 = determineHomography(F1, F2, matches_12, N, T);
H_23 = determineHomography(F2, F3, matches_23, N, T);
H_43 = determineHomography(F4, F3, matches_43, N, T);
H_54 = determineHomography(F5, F4, matches_54, N, T);

if(showFigures)
    points1Transformed = tformfwd(H_12, points1(:,1), points1(:,2));
    % euclidean distance
    diff = points2 - points1Transformed;
    dist = diff.*diff;
    dist = bsxfun(@plus,dist(:,1),dist(:,2));
    dist = sqrt(dist);
    inliers = dist < 5;


    % homography with maximum number of inliers
    pp = points1(inliers, :);
    pp2 = points2(inliers, :);

    match_plot(img1, img2, pp, pp2);

    [m n d] = size(img2);
    img1Tr = imtransform(img1, H_12, 'XData',[1 size(img1,2)], 'YData',[1
size(img1,1)], 'Size', [m n] );
```

```
% Apart from small errors, the images should now be
% aligned. Demonstrate that by showing the absolute differences between
the two images.
    diffImg = imabsdiff(img2,img1Tr);
    figure;
    imshow(diffImg);
end

% img 3 is reference image
% compute composite homographies to map other images to reference image
H_13 = H_12;
H_13.tdata.T = H_23.tdata.T * H_12.tdata.T;
H_13.tdata.Tinv = inv(H_13.tdata.T);
H_53 = H_54;
H_53.tdata.T = H_43.tdata.T * H_54.tdata.T;
H_53.tdata.Tinv = inv(H_53.tdata.T);

cornersA = [ ...
    1               1
    1               size(img1, 1)
    size(img1, 2)   1
    size(img1, 2)   size(img1, 1)];

out_pointsA = tformfwd(H_13, cornersA);

cornersB = [ ...
    1               1
    1               size(img5, 1)
    size(img5, 2)   1
    size(img5, 2)   size(img5, 1)];

out_pointsB = tformfwd(H_53, cornersB);

corners = vertcat(out_pointsA, out_pointsB);
minX = ceil(min(corners(:,1)));
maxX = ceil(max(corners(:,1)));
minY = ceil(min(corners(:,2)));
maxY = ceil(max(corners(:,2)));

% transform all images to the plane defined by the reference image
output1 = imtransform(img1, H_13, 'XData',[minX maxX], 'YData',[minY
maxY]);
output2 = imtransform(img2, H_23, 'XData',[minX maxX], 'YData',[minY
maxY]);%, 'Size', [yDim xDim] );
output3 = zeros(size(output1), 'uint8');
output3(abs(minY):abs(minY)+size(img3,1)-1, abs(minX):abs(minX)+size
(img3,2)-1,:) = img3;
output4 = imtransform(img4, H_43, 'XData',[minX maxX], 'YData',[minY
maxY]);%, 'Size', [yDim xDim] );
output5 = imtransform(img5, H_53, 'XData',[minX maxX], 'YData',[minY
```

```
maxY]);%, 'Size', [yDim xDim] );
% figure;
% imshow(output1);
% figure;
% imshow(output2);
% figure;
% imshow(output3);
% figure;
% imshow(output4);
% figure;
% imshow(output5);

% blend overlapping pixel color values

[m n d] = size(img1);
distImg = zeros(m,n);
distImg(1:m, 1) = 1;
distImg(1:m, n) = 1;
distImg(1, 1:n) = 1;
distImg(m, 1:n) = 1;
alphaImg = bwdist(distImg);
alphaImg = alphaImg./max(max(alphaImg));

alphaImg1 = imtransform(alphaImg, H_13, 'XData',[minX maxX], 'YData',[minY
maxY]);
alphaImg2 = imtransform(alphaImg, H_23, 'XData',[minX maxX], 'YData',[minY
maxY]);
alphaImg3 = zeros(size(alphaImg1), 'single');
alphaImg3(abs(minY):abs(minY)+size(distImg,1)-1, abs(minX):abs(minX)+size
(distImg,2)-1) = alphaImg;
alphaImg4 = imtransform(alphaImg, H_43, 'XData',[minX maxX], 'YData',[minY
maxY]);
alphaImg5 = imtransform(alphaImg, H_53, 'XData',[minX maxX], 'YData',[minY
maxY]);

alphaComplete = alphaImg1 + alphaImg2 + alphaImg3 + alphaImg4 + alphaImg5;



output1Alpha = single(output1);
output1Alpha(:,:,1) = output1Alpha(:,:,1).*alphaImg1;
output1Alpha(:,:,2) = output1Alpha(:,:,2).*alphaImg1;
output1Alpha(:,:,3) = output1Alpha(:,:,3).*alphaImg1;
% output1Alpha = uint8(output1Alpha);

output2Alpha = single(output2);
output2Alpha(:,:,1) = output2Alpha(:,:,1).*alphaImg2;
output2Alpha(:,:,2) = output2Alpha(:,:,2).*alphaImg2;
output2Alpha(:,:,3) = output2Alpha(:,:,3).*alphaImg2;
% output2Alpha = uint8(output2Alpha);
```

```
output3Alpha = single(output3);
output3Alpha(:,:,1) = output3Alpha(:,:,1).*alphaImg3;
output3Alpha(:,:,2) = output3Alpha(:,:,2).*alphaImg3;
output3Alpha(:,:,3) = output3Alpha(:,:,3).*alphaImg3;
% output3Alpha = uint8(output3Alpha);

output4Alpha = single(output4);
output4Alpha(:,:,1) = output4Alpha(:,:,1).*alphaImg4;
output4Alpha(:,:,2) = output4Alpha(:,:,2).*alphaImg4;
output4Alpha(:,:,3) = output4Alpha(:,:,3).*alphaImg4;
% output4Alpha = uint8(output4Alpha);

output5Alpha = single(output5);
output5Alpha(:,:,1) = output5Alpha(:,:,1).*alphaImg5;
output5Alpha(:,:,2) = output5Alpha(:,:,2).*alphaImg5;
output5Alpha(:,:,3) = output5Alpha(:,:,3).*alphaImg5;
% output5Alpha = uint8(output5Alpha);


%out = output1 + output2 + output3 + output4 + output5;
out = output1Alpha + output2Alpha + output3Alpha + output4Alpha + out-
put5Alpha;
out(:,:,1) = out(:,:,1)./alphaComplete;
out(:,:,2) = out(:,:,2)./alphaComplete;
out(:,:,3) = out(:,:,3)./alphaComplete;
out = uint8(out);

figure;
imshow(out);

end
```

## 9.2.2 Added functions

```
function [ homography ] = determineHomography( featuresA, featuresB, mat-
ches, N, T )

nrMatches = size(matches, 2);
maxInliers = 0;

matchedFtsA = horzcat(featuresA(1,matches(1,:))', featuresA(2,matches
(1,:))');
matchedFtsB = horzcat(featuresB(1,matches(2,:))', featuresB(2,matches
(2,:))');


for i = 1:N
```

```
    randMatches = randsample(nrMatches, 4);

    rndfeaturesA = matchedFtsA(randMatches,:);
    rndfeaturesB = matchedFtsB(randMatches,:);

    try
        % homography H
        H = cp2tform(rndfeaturesA, rndfeaturesB, 'projective');
        points1Transformed = tformfwd(H, matchedFtsA(:,1), matchedFtsA
(:,2));

        % euclidean distance
        diff = matchedFtsB - points1Transformed;
        dist = diff.*diff;
        dist = bsxfun(@plus,dist(:,1),dist(:,2));
        dist = sqrt(dist);

        nrOfInliers = sum(dist < T);

        if nrOfInliers > maxInliers
            maxInliers = nrOfInliers;
            homography = H;
        end

    catch
        errmsg = lasterr;
    end
end
End
```

```
function h = match_plot(img1,img2,points1,points2)
h = figure;
colormap = {'b','r','m','y','g','c'};
height = max(size(img1,1),size(img2,1));
img1_ratio = height/size(img1,1);
img2_ratio = height/size(img2,1);
img1 = imresize(img1,img1_ratio);
img2 = imresize(img2,img2_ratio);
points1 = points1 * img1_ratio;
points2 = points2 * img2_ratio;
points1 = [points1(:,2) points1(:,1)];
points2 = [points2(:,2) points2(:,1)];
match_img = zeros(height, size(img1,2)+size(img2,2), size(img2,3));
match_img(1:size(img1,1),1:size(img1,2),:) = img1;
match_img(1:size(img2,1),size(img1,2)+1:end,:) = img2;
imshow(uint8(match_img));
hold on;
```

```
for i=1:size(points1,1)
    plot([points1(i,2) points2(i,2)+size(img1,2)],[points1(i,1) points2
(i,1)],colormap{mod(i,6)+1});
end

hold off;

end
```

# 9.3 Object Recognition

```
clear all;
close all;
clc;

he = imresize( imread ('esc2.jpg'), 0.2);

cform = makecform('srgb2lab');
lab_he = applycform(he,cform);

ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);

nColors = 4;
% repeat the clustering 3 times to avoid local minima
[cluster_idx, cluster_center] = kmeans
(ab,nColors,'distance','sqEuclidean', 'Replicates',3);

pixel_labels = reshape(cluster_idx,nrows,ncols);
imshow(pixel_labels,[]), title('image labeled by cluster index');

segmented_images = cell(1,3);
rgb_label = repmat(pixel_labels,[1 1 3]);

for k = 1:nColors
    color = he;
    color(rgb_label ~= k) = 0;
    segmented_images{k} = color;
end

subplot(2,2,1) , imshow(segmented_images{1}), title('objects in cluster
1');
subplot(2,2,2) , imshow(segmented_images{2}), title('objects in cluster
```

```
2');
subplot(2,2,3) , imshow(segmented_images{3}), title('objects in cluster
3');
subplot(2,2,4) , imshow(segmented_images{4}), title('objects in cluster
4');
% subplot(3,2,5) , imshow(segmented_images{5}), title('objects in cluster
5');

% figure (2);
% imshow(segmented_images{2}), title('objects in cluster 2');
%
% figure (3);
% imshow(segmented_images{3}), title('objects in cluster 3');
%
% figure (4);
% imshow(segmented_images{4}), title('objects in cluster 4');
%
% figure (5);
% imshow(segmented_images{4}), title('objects in cluster 5');

i1 = segmented_images{1};
i2 = segmented_images{2};
i3 = segmented_images{3};
i4 = segmented_images{4};

i1gray = rgb2gray (i1);
i2gray = rgb2gray (i2);
i3gray = rgb2gray (i3);
i4gray = rgb2gray (i4);

th = 10;

[i1_r , i1_c] = find (i1gray > th);
m11 = mean(mean(i1(i1_r, i1_c, 1)))
m12 = mean(mean(i1(i1_r, i1_c, 2)))
m13 = mean(mean(i1(i1_r, i1_c, 3)))

[i2_r , i2_c] = find (i2gray > th);
m21 = mean(mean(i2(i2_r, i2_c, 1)))
m22 = mean(mean(i2(i2_r, i2_c, 2)))
m23 = mean(mean(i2(i2_r, i2_c, 3)))

[i3_r , i3_c] = find (i3gray > th);
m31 = mean(mean(i3(i3_r, i3_c, 1)))
m32 = mean(mean(i3(i3_r, i3_c, 2)))
m33 = mean(mean(i3(i3_r, i3_c, 3)))

[i4_r , i4_c] = find (i4gray > th);
m41 = mean(mean(i4(i4_r, i4_c, 1)))
```

```
m42 = mean(mean(i4(i4_r, i4_c, 2)))
m43 = mean(mean(i4(i4_r, i4_c, 3)))

if((m11>100) & (m12>100) & (m13<50))
    figure (2)
    imshow (i1), title ('The folder, which contains where is the trophy')
%     write ('This object is the folder, here is said where is the trop-
hy')
end

if(m21>100) & (m22>100) & (m23<50)
    figure(2)
    imshow (i2), title ('The folder, which contains where is the trophy')
%     write ('This object is the folder, here is said where is the trop-
hy')
end

if(m31>100) & (m32>100) & (m33<50)
    figure(2)
    imshow (i3), title ('The folder, which contains where is the trophy')
%     write ('This object is the folder, here is said where is the trop-
hy')
end

if(m41>100) & (m42>100) & (m43<50)
    figure(2)
    imshow (i4), title ('The folder, which contains where is the trophy')
%     write ('This object is the folder, here is said where is the trop-
hy')
end
```