UNIVERSIDAD CARLOS III DE MADRID

ESCUELA TÉCNICA SUPERIOR

UNIVERSITY OF PIRAEUS

DEPARTMENT OF INFORMATICS

EVALUATION AND IMPROVEMENT OF A FACE DETECTION METHOD ON THE BASIS OF SKIN COLOR

FINAL THESIS REPORT

TELECOMMUNICATIONS TECHNICAL ENGINEERING

SPECIALTY IN SOUND AND IMAGE

AUTHOR: ELENA LUCAS SIMARRO

ATHENS, SEPTEMBER 2011

Project: Evaluation and improvement of a face detection method on the basis of skin color

Author: Elena Lucas Simarro

Tutors: Dr. Christos-Nikolaos Anagnostopoulos

Dr. Dimitrios D. Vergados

ACKNOWLEDGMENTS

I would like to thank, not only this project, but all these university years, to all those who I have felt somehow, supporting and motivating me during this great period of my life.

The first, of course, are my family, every one of them, but mainly those who have lived close to me, the significance of this stage, my parents and my brothers. My brothers, as a reference in all decisions of my life, they are the best examples to follow, and my parents for helping me to find the strength in every moment.

I have to thank Luisvi all the years by my side, years of loving and understanding, I feel him near, even though we're so far away.

I would like to thank to my friends of Cuenca, all life together, they know they are essential for me, and my friends from Madrid, because live with them every day has made us to share extraordinary experiences.

Thanks to my partners and university friends with whom I spent many good times, times of laughter and study, now, it has been worth.

Specially I´m so gratefull with my Erasmus friends, people that in only 5 months have become very important for me. Share this experience with you is what has made it so special. My little erasmus family...Thanks!

Finally, I would like to thank the people who have helped me academically. Dimitrios Vergos, my Erasmus coordinator of Athens, for all his help, and to make a little bit easier to adapt me to this great change. Christos Anagnostopoulos, the project coordinator, because with this work I have explored a field that has surprised me very pleasantly. And my Erasmus coordinator Madrid, Albert Banchs, for their disposal at any time I needed.

THANK YOU.

AGRADECIMIENTOS

Me gustaría agradecer, no solo este proyecto, sino todos estos años de universidad, a todas aquellas personas que de alguna forma he sentido apoyándome y motivándome durante este gran periodo de mi vida.

Los primeros, por supuesto, son mi familia, todos y cada uno de ellos, pero principalmente los que han vivido de cerca el significado de esta etapa para mí, mis padres y mis hermanos. Mis hermanos, por ser una referencia en todas las decisiones de mi vida, y mis padres, por ayudarme a conseguir sacar la fuerza necesaria en cada momento.

A Luisvi, por tantos años a mi lado llenos de amor y comprensión, por hacerme sentirle cerca, aunque estemos tan lejos.

A mis amigas de Cuenca, las de toda la vida, que saben lo imprescindibles que son para mí, y a mis amigas de la Madrid, porque convivir con ellas día a día nos ha hecho compartir experiencias únicas.

A mis compañeros y amigos de la universidad, con los que tantos buenos momentos he pasado, horas de risas y estudio que, ahora sí, han merecido la pena.

Especialmente a mis amigos Erasmus, que solo en 5 meses han llegado a ser personas muy importantes para mí. Compartir esta experiencia con vosotros es lo que la ha hecho tan especial. Gracias!

Por último, me gustaría agradecer a las personas que me han ayudado académicamente. Dimitrios Vergados, mi coordinador Erasmus de Atenas, por toda su ayuda, y por hacer un poco más fácil la adaptación al este gran cambio. Christos Anagnostopoulos, coordinador del proyecto, por animarme con este trabajo a explorar un campo que me ha sorprendido muy gratamente. Y a mi coordinador Erasmus de Madrid, Albert Banchs, por su disposición en cualquier momento que he necesitado.

A todos, GRACIAS.

SUMARIO

Este proyecto aborda el análisis y evaluación de un método de detección de rostros basado en el color de la piel, es decir, teniendo en cuenta principalmente las características de color de la imagen para llevar a cabo la detección de rostros. Este proceso se logra a través de tres pasos fundamentales: la transformación del espacio de color, la generación de imágenes en escala de grises y la estimación del valor umbral óptimo que nos lleva al último paso que es la segmentación.

La detección de rostros es un paso crítico en cualquier aplicación de análisis facial, tal como el reconocimiento de rostros, la codificación de vídeo para videoconferencia, inteligencia artificial, etc. En general, este tipo de detección suele implicar problemas muy complejos, ya que los rostros tienen diferentes colores, expresiones, poses y tamaños relativos o se encuentran en diferentes condiciones de iluminación.

MOTIVACIÓN, OBJETIVOS Y APLICACIONES

Un sistema centrado en la detección de rostros humanos nos ayudará a determinar la presencia y la posición de un rostro humano en una imagen. Esta es una tarea importante en el campo de reconocimiento de patrones, de hecho, la detección del rostro humano es el primer paso en un sistema de reconocimiento de patrones. Los estudios en este campo se basan en encontrar métodos para detectar e identificar las caras en una escena con poco esfuerzo, porque nuestro cerebro analiza y segmenta, clasifica, reconoce e interpreta una gran cantidad de datos capturados por los órganos sensoriales cuando nos fijamos en una escena. Estas tareas son realizadas por nuestro cerebro en fracciones de segundo con un bajo error.
Las aplicaciones y la dificultad de la detección en sistemas de detección de rostros es un problema interesante. En términos de aplicaciones, la detección de la cara es muy importante para el problema de reconocimiento rostros.

Hasta ahora, los investigadores se han centrado principalmente en el problema de reconocimiento de rostros, en el que la tarea de encontrar caras en un fondo arbitrario se suele sustituir por cualquier método de segmentación manual de la imagen de entrada, o mediante la captura de las caras contra en un fondo uniforme. En la última década, la detección de rostros ha atraído una gran atención, precisamente porque el sistema de reconocimiento facial requiere de la detección automática de caras como primer paso, sobre todo para las imágenes con fondo variante.
La detección de rostros es difícil debido a tres razones principales. En primer lugar, hay un gran componente de no-rigidez y diferencias de textura entre las caras. La apariencia del rostro se diferencia de uno a otro. En segundo lugar, la detección de rostros también se hace difícil debido a factores adicionales que aumentan la variabilidad de los patrones de la cara que un sistema de detección debe manejar. En tercer lugar, la presencia de condiciones impredecibles en la imagen, cuando esta se encuentra en un entorno sin restricciones aumenta la dificultad de la tarea. Un cambio en la distribución de fuente de luz puede causar un cambio significativo en la apariencia de la imagen de la cara.

El propósito de este proyecto es la detección de rostros humanos en imágenes de diferentes colores. Hay diferentes aplicaciones en las que se podría incluir un programa de detección de rostros. Una de ellas podría estar la localización de un usuario concreto dentro de una base de datos de imágenes. Es decir, primero identificar los rostros en todas las imágenes de la base de datos y una vez hecha la detección, aplicar un sistema de Reconocimiento de Patrones.

El objetivo de nuestro proyecto es sólo detectar la presencia de una cara en un conjunto de imágenes de varias bases de datos propuestas. Para ello, contamos con un código desarrollado en lenguaje de programación Matlab que tenemos que poner a prueba para comprobar los resultados. Estos resultados dependen del tipo de imágenes que elijamos. Las imágenes tienen que ser en color, porque el programa trabaja con las características de color de piel. Una vez que el funcionamiento del programa se ha entendido y probado, debemos hacer una evaluación objetiva, donde se deben identificar las ventajas y desventajas del método utilizado, y proponer, si fuera necesario, todo tipo de mejoras que puedan hacer que el algoritmo sea más eficiente.

La detección de rostros tiene muchas aplicaciones del mundo real, sobre todo como paso previo a sistemas de reconocimiento facial. Algunas de las aplicaciones pueden ser para la interfaz usuario/PC, la vigilancia, la autenticación y la indexación de vídeo. Sin embargo la investigación en este campo es todavía joven. La detección de rostros y el reconocimiento dependerá en gran medida de la elección particular de las características utilizadas. Por lo general comienza con un conjunto dado de características y luego intenta seleccionar un subconjunto más óptimo (basado en algunos criterios) de características.

Algunas de las aplicaciones de esta área de desarrollo (detección y posterior reconocimiento) son:

- Videoconferencia. Es posible el desarrollo de algoritmos para detectar de un rostro en una secuencia de vídeo y darle seguimiento, por lo que la cámara puede ajustar correctamente el campo de visión si el individuo se mueve, o desconectar de forma automática si el usuario ya no está.
- Vigilancia de la multitud. Estos algoritmos son perfectos en el caso de que alguien buscado por la ley esté tratando de camuflarse entre la multitud, por ejemplo, una manifestación o un estadio de fútbol. Sólo es necesario hacer un barrido de la gente y dejar que el ordenador haga el resto.
- Seguridad en casa. No es solo un sistema de monitorado en el hogar mediante cámaras de seguridad para alertar a la policía si alguien entra en la casa. Estos algoritmos también pueden determinar si el “intruso” es alguien conocido (en este caso no se daría parte de intrusión) o desconocido.

IMPLEMENTACIÓN Y MEJORAS

Tras un marco introductorio sobre el estado del arte relativo al procesamiento de imágenes y concretamente enfocado a la detección de rostros, pasamos a explicar la fase de desarrollo del proyecto.

En primer lugar hemos seleccionado dos bases de datos de imágenes con las que trabajaremos y comprobaremos el funcionamiento del algoritmo propuesto.

Dentro de la base de datos A, encontramos una serie de imágenes que clasificaremos como "imágenes con fondo controlado". Pertenece al "International Workshop on Visual Observation of Deictic Gestures of Cambridge, UK". Está formada por 2790 imágenes monoculares de rostros de 15 personas con variaciones de ángulos de giro e inclinación de - 90 a + 90 grados. Por cada persona hay 2 series de 93 imágenes (93 posturas diferentes). Las imágenes se toman en un entorno controlado, donde el fondo es blanco y los cambios de iluminación son muy pequeños en un alto porcentaje de imágenes. Las personas normalmente están a una distancia fija de la cámara (sin cambios significativos), y todas las son del mismo tamaño, 384 x 288 píxeles.

La base de datos B incorpora un conjunto de caras en posición frontal recogidas por Markus Weber en el "California Institute of Technology". Contiene 450 imágenes de rostros con el mismo tamaño, 896 x 592 píxeles y formato JPEG. En las imágenes aparecen alrededor de 27 personas con diferentes expresiones faciales, condiciones de iluminación y fondo.

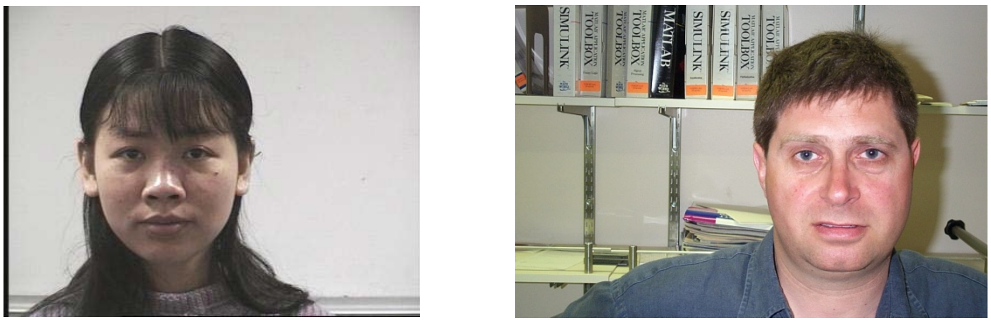


Figura A: *Ejemplos de imágenes de entradas*

En la fase de implementación estudiaremos al detalle el código de algoritmo utilizado con el fin de comprender su funcionamiento.

El primer paso ha sido aplicar una función para leer todas las imágenes de la base de datos desde un directorio (en nuestro código se llama "pathDatabase"), y tras su procesamiento guardar los resultados de salida en otro directorio (en nuestro caso "pathResults"). El algoritmo con este fin está implementado en Matlab, y se llama "readImages.m", como podemos ver en el Apéndice A. Para procesar las imágenes hemos utilizado el algoritmo propuesto llamado "faceDetection.m" (código Matlab en el Apéndice A). Podemos ejecutarlo a partir de la función readImages.m anterior.

El segundo paso en el desarrollo de este programa es modificar la imagen original en la que estamos interesados en realizar la detección de rostros. Para eliminar los efectos de la luz, es necesario tomar la imagen de color de entrada al sistema y convertirla a otro espacio de color. Esto es debido a que el espacio de color en el que se encuentra la imagen original no es fiable para la caracterización de los colores de la piel. El espacio de color en el que se encuentran las imágenes originales es el espacio RGB, que a través de sus tres componentes (r, g, b) representa no sólo un color, sino que también representan el brillo o luminosidad, que puede variar la cara de una persona debido a las condiciones ambientales de luz. Por tanto, no ofrece la posibilidad de establecer una medida fiable para la segmentación de la imagen en regiones según el color de la piel. La luminancia se puede eliminar de la representación transformando la imagen a un espacio de color cromático. Este método utiliza el espacio de color CIELAB, que es válido porque tiene la intensidad de la imagen como una de sus componentes. En primer lugar, utiliza funciones de transformación del color para convertir la imagen de RGB al espacio de color L * a * b * d. Posteriormente trabaja en la capa 'L * ', que es la capa de luminosidad de la imagen. La manipulación de luminosidad afecta a la intensidad de los píxeles, preservando al mismo tiempo los colores originales.

El tercer paso del programa crea, a partir de la imagen anterior (espacio de color Lab), dos imágenes en escala de grises con la particularidad de que la primera representa los niveles de intensidad de color verde de la imagen, y la segunda representa los niveles de intensidad de color azul. El objetivo de este proceso es, a partir de estas imágenes, establecer un nivel de umbral para la segmentación posterior.

Debido a que las regiones de la piel son más brillantes que otras partes de la imagen en escala de grises, estas regiones de piel pueden ser segmentadas del resto de la imagen a través de un proceso en el que establecemos un umbral variable para la segmentación. El umbral debe ser variable ya que en las imágenes de entrada hay determinadas características que usualmente cambian debido a los diferentes tipos de piel, por lo que un umbral fijo no es posible.

En el código del proyecto se utiliza el método de Otsu para obtener el valor umbral óptimo para cada imagen en escala de grises. El algoritmo de Otsu asume que la imagen que tiene que ser segmentada contiene dos clases de píxeles (por ejemplo, los pixeles de primer plano y los de fondo). A partir de esta premisa calcula el umbral óptimo que separa estas dos clases de modo que la varianza entre las clases es mínima.

Una vez que el valor umbral óptimo se determina (para cada imagen), transformamos la imagen en escala de grises en una imagen binaria. La imagen binaria tiene valores de 1 (blanco) para todos los píxeles de la imagen de entrada (en escala de grises) con niveles de intensidad mayor que el valor umbral óptimo, y 0 (negro) para todos los otros píxeles.

En este punto tenemos dos imágenes binarias creadas a partir de la intensidad de nivel de azul o verde de la imagen. El siguiente paso del proyecto consiste en multiplicar las dos imágenes, obteniendo de esta manera nuestra máscara final de la cual seleccionemos cuáles son los píxeles que pertenecen a la piel, y cuáles no lo son.

Una vez determinamos nuestra máscara, creamos una matriz que marque las propiedades que consideramos necesarias para encuadrar el contenido de la cara. En este caso, sólo hemos elegido la propiedad BoundingBox. Es una característica que calcula la posición y las dimensiones del rectángulo mínimo que rodea la región de interés.

El último paso del código del proyecto consiste en encontrar la posición en la que tenemos que dibujar el rectángulo con la posición de la cara. Para ello, el autor del código utiliza el bucle que recorre todos los cuadros delimitadores generados en el paso anterior, tratando de encontrar el que tenga una relación de aspecto de menos de 1,8. Una vez que tenemos la posición de la cara se puede dibujar el rectángulo para terminar el programa.

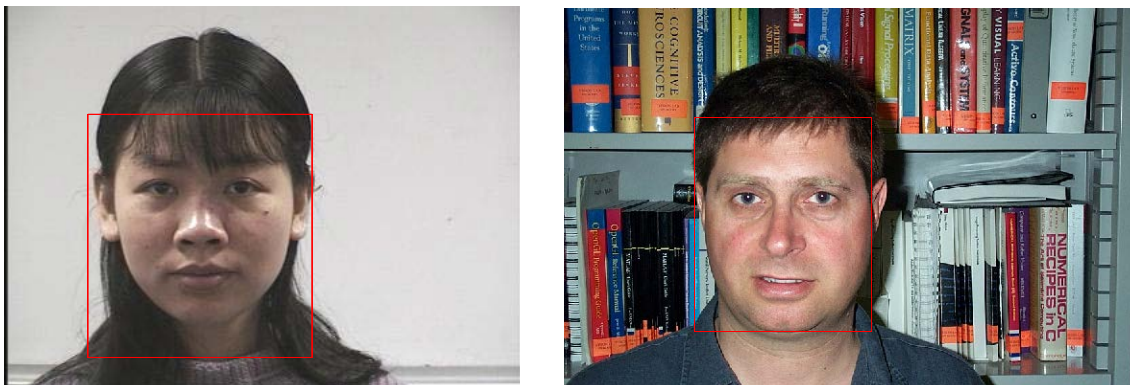


***Figura B:** Resultado final. Imagen original con el rostro detectado*

Como indicamos en el principio del desarrollo, se ha modificado el código para guardar en un directorio de sólo el rostro de la imagen original recortada.

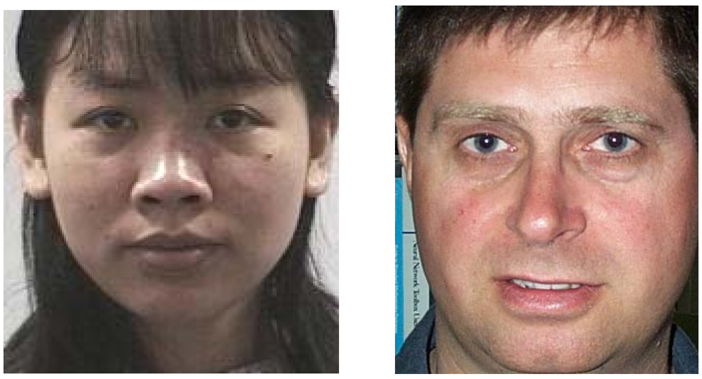


***Figura C**: Rostro detectado y recortado*

CONCLUSIONES Y TRABAJO FUTURO

Para terminar el análisis comentaremos brevemente algunas conclusiones del proyecto, así como posibles líneas para desarrollar en un trabajo futuro.
Tras aplicar el detector basado en las características de color de la piel, observamos que los primeros resultados muestran un número muy elevado de aciertos. Estos resultados los obtenemos a partir de imágenes que presentan unas condiciones muy favorables. Sin embargo, a medida que aumentamos ligeramente el grado de dificultad, se observa que el detector empieza a cometer errores en un alto porcentaje.

El detector de rostros podría ser utilizado a nivel de usuario, dependiendo de las características requeridas, pero nunca profesionalmente, ya que, según el estado de desarrollo de técnicas en este ámbito, existen en la actualidad métodos con mejores prestaciones.

Como posibles líneas para implementar en un trabajo futuro, específicamente para este estudio, además de mejorar las prestaciones que ofrece el sistema, se debería tener en cuenta que el algoritmo ha sido probado únicamente en caras en posición frontal o casi frontales. Las diferentes pruebas han demostrado que se pueden detectar caras con un ángulo de rotación de hasta aproximadamente 30° en profundidad sin una pérdida significativa en el rendimiento. Sin embargo, no se sabe la eficiencia del sistema con un mayor ángulo de rotación, lo que puede ser un buen proyecto el desarrollo de esta línea de investigación.

Por otra parte, se puede proponer como una línea diferente de trabajo futuro el diseño o adaptación del sistema a imágenes de varias caras. Analizando el algoritmo que tenemos, es evidente que si probáramos el análisis en bases de datos con varias caras los resultados no serían favorables, por ello puede ser interesante ampliar su funcionamiento en este campo, ya que se aproxima en gran medida a situaciones reales.

ABSTRACT

This project deals with the analysis and evaluation of a face detection method based on skin color, ie, taking into account mainly the color characteristics of the image to perform face detection. This process is accomplished through three fundamental steps: transformation of color space, generating grayscale images and establishing optimal thresholding value to reach the last step that is segmentation.

To evaluate the proposed algorithm we have decided to use two databases with different characteristics. The first works with images with a controlled background, specifically white and smooth, the only change are the characters of the images and the illumination. The second database works with a not controlled background, that is, each picture is taken in a different environment, with different luminosities, characters and objects, these changes hinder the detection process as discussed below.

After analyzing the performance of the face detector we must determine whether the results we have are good for each of the situations that arise

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Face detection is a critical step in any application where you do some type of facial analysis such as face recognition, video coding for videoconferencing, intelligent man-machine interfaces, etc.. In general, face detection is a very complex problem because the objects to be detected can be different colors, expressions, poses, relative sizes or have very different lighting conditions.

1.1. MOTIVATION

A system focused on the detection of human faces will help us to determinate the presence and the position of a human face in an image, which is an important task in the field of pattern recognition; in fact the human face detection is the first step in a pattern recognition system. Humans are working to detect and identify faces in a scene with little effort, because our brain analyzes and segments, classifies, recognizes and interprets a large amount of data captured by the sense organs when we look in a scene. These tasks are performed by our brain in fractions of a second with a low error.

The applications and the difficulty of face detection make face detection an interesting problem. In terms of applications, face detection is quite important for the face recognition problem, since it is the most important step for a face recognition system. So far, the researchers have mainly focused on the face recognition problem, in which the task of finding faces in an arbitrary background is usually avoided by either manual segmentation of the input image, or by capturing faces against a known uniform background. In the last decade, face detection has attracted great attention, as face recognition system requires automatic face detection as a first step, especially for images with cluttered background.

Face detection is difficult due to three main reasons. First, there is a large component of non-rigidity and textural differences among faces. Facial appearance differs from face to face. Second, face detection is also made difficult because of additional features. All these additional features increase the variability of the face patterns that a face detection system should handle. Third, the presence of unpredictable imaging conditions in an unconstrained environment increases the difficulty of the task. A change in light source distribution can cause a significant change in the appearance of the face image.

1.2. OBJECTIVES

The purpose of this project is to detect human faces in various color images. There are different applications in which a face detection program could be used. One of which could be using this system with a database containing images. For example, if you find someone special in a database that contains several images of human faces, the user could simply ask the program to find images containing specific people, which could be determined by detecting faces people in each image and then once you are detected, proceed to step Pattern Recognition

The objective of our project is only detecting the presence of a face in a set of images of several databases proposed. To do that, we have a code developed in Matlab programming language that we have to test to check the results. These results will depend on the kind of images we choose. These images have to be in color because the program works with color skin features.

Once the program operation has been understood and tested, we should make an objective evaluation, where we should identify the advantages and disadvantages of the used method, and proposing if it was necessary, all the kinds of improvements that can make the algorithm more efficient.

APPLICATIONS

Detect a face is the first step towards the recognition of it. That is, before knowing who is the person shown in an image, we must devise algorithms that automatically detect that in the photo is shown a face. It has many real-world applications like human/computer interface, surveillance, authentication and video indexing. However research in this field is still young. Face detection and recognition depend heavily on the particular choice of features used. It usually starts with a given set of features and then attempts to select a more optimal subset (under some criteria) of features.

Some of the applications of this development area (detection and subsequent recognition) are:

- Videoconferencing. It´s possible to develop algorithms to detect a face in a video stream and follow it up, so the camera could adjust properly the vision field if the individual moves, or hang up automatically if it goes.
- Surveillance crowd. These algorithms are perfect in case someone wanted by the law is trying to camouflage in a crowd, for example, a demonstration or a football stadium. It´s only necessary panning the crowd and let the computer do the rest.
- Security home. There is a home monitoring systems using security cameras, and alert the police if someone comes into the house. These algorithms may also determine if the inflow is known (so you do not have to give part) or unknown.

1.3. PROJECT STRUCTURE

This report, as we will see below, has been divided in five main chapters:

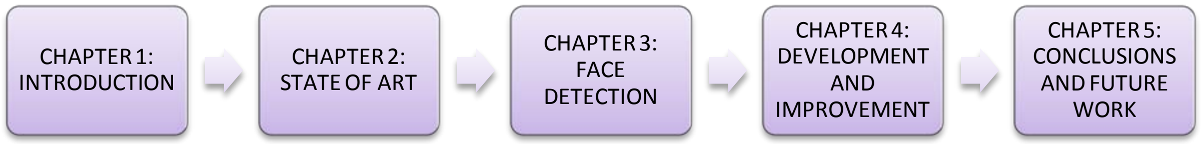

CHAPTER 1: INTRODUCTION
CHAPTER 2: STATE OF ART
CHAPTER 3: FACE DETECTION
CHAPTER 4: DEVELOPMENT AND IMPROVEMENT
CHAPTER 5: CONCLUSIONS AND FUTURE WORK

Figure 1: *Project Structure*

The first one describes like introduction the possibilities of the face detection field, as well as the utilities and applications that nowadays are using this kind of techniques. It has also detailed some reasons as motivations to make the final year project in this area, and we specify the objectives of our concrete work.

The second chapter briefly presents a theoretical background of the elemental concepts necessary to understand the development of the project. This theoretical presentation ranges from the concept of digital image, to the different kinds of processing the same.

In the third chapter face detection methods will be introduced with short overview to give any general ideal about the history of face detection and the future approaches of it.

The fourth chapter is the main section of the project. In this chapter we explain the face detection method proposed, and we analyze step by step the different function on the code. Finally we propose some improvements to increase the effectiveness of the program.

Finally, in the chapter 5, we are going to briefly detailed some conclusions of the project, and some future works that can be interesting to further develop face detection area.

CHAPTER 2: STATE OF ART

In this chapter are discussed some theoretical aspects necessaries for the progress of the project. It is divided into two main sections, the first deals with general concepts of digital image and its representation, and in the second one we will explain some image processing techniques in depth.

2.1. BASIC CONCEPTS

This section will introduce a number of general concepts of representation and processing that we must be taken into account when we work with images.

2.1.1. DIGITAL IMAGE

A digital image is the representation of a continuous image f (x,y) by a 2-d array of discrete samples f [x,y], each element of the 2-d array of samples is a pixel. The amplitude of each sample is quantized to be represented by a finite number of bits.

To properly understand the above definition, we must clarify the meaning of pixel and resolution:

- Pixel: Pictures are made up of little dots called pixels. Pixel stands for PICture ELement, is the minimum element of an image. If we put enough of them together, we will have a picture. They are arranged horizontally and vertically (rows and columns) as we can see below.

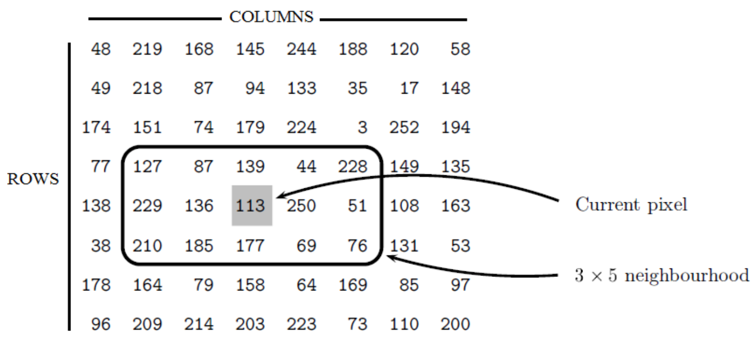


Figure 2: *Pixels with a neighborhood*

– Resolution: The image resolution or "dpi" ("dots per inch", also called "ppi" or "pixels per inch"), determines the number of pixels that a digital image can contain and as such the depth of quality and size of the file. The more pixels an image can contain, the better quality it can be, but also the larger the file size.

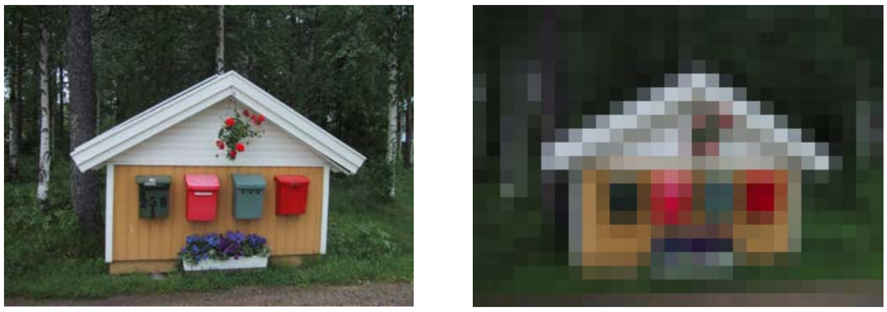


Figure 3: *Different resolutions a) Resolution 300x220 ppi b) Resolution30x22 ppi*

2.1.1.1. DIGITAL IMAGE CLASSIFICATION

Digital images can be classified according to number and nature of those samples in: binary images, grayscale images and color images (true color images).

➢ **Binary Images**

Binary images are images that have been quantified to two values, usually denoted 0 and 1, but often with pixel values 0 and 255, representing black and white. They are used in many applications since they are the simplest to process, but they are such an impoverished representation of the image information that their use is not always possible.

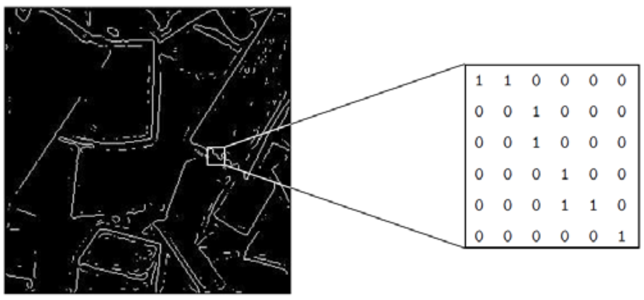
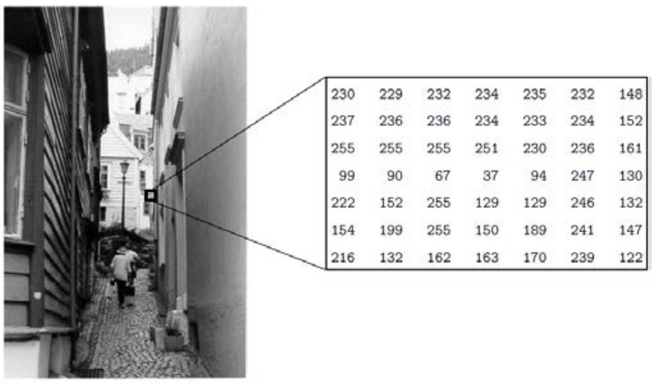


Figure 4: *Binary image*

➢ Grayscale images

A grayscale image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. Usually the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white.

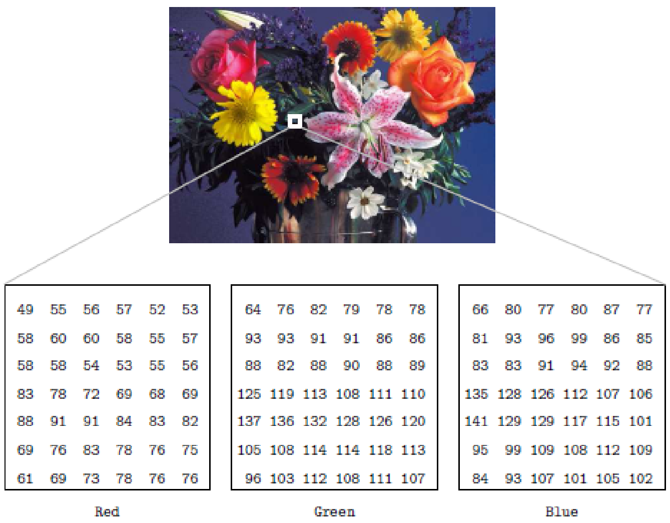


230	229	232	234	235	232	148
237	236	236	234	233	234	152
255	255	255	251	230	236	161
99	90	67	37	94	247	130
222	152	255	129	129	246	132
154	199	255	150	189	241	147
216	132	162	163	170	239	122

Figure 5: *Grayscale image*

➢ Color images

True color images are made up of pixel arrays (m-by-n-by-3 data array) containing values for red, green and blue for each individual pixel,that define the specific color in that particular area. The RGB color space is commonly used in computer displays.



Red

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Green

64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Blue

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Figure 6: *True color image*

2.1.2. COLOR SPACES

A color space is a method by which we can specify, create and visualize color. As humans, we may define a color by its attributes of brightness, hue and colorfulness:

- Brightness: the human sensation by which an area exhibits more or less light.
- Hue: the human sensation according to which an area appears to be similar to one, or to proportions of two, of the perceived colors red, yellow, green and blue.
- Colorfulness: the human sensation according to which an area appears to exhibit more or less of its hue.

A color is thus usually specified using three co-ordinates, or parameters. These parameters describe the position of the color within the color space being used. They do not tell us what the color is, that depends on what color space is being used. Some of the parameters that color spaces use are:

- Lightness: the sensation of an area's brightness relative to a reference white in the scene.
- Chroma: the colorfulness of an area relative to the brightness of a reference white.
- Saturation: the colorfulness of an area relative to its brightness.

The tri-chromatic theory describes the way three separate lights, red, green and blue, can match any visible color – based on the eye's use of three color sensitive sensors. This is the way that most computer color spaces operate, using three parameters to define a color.

2.1.2.1. RGB Color Space

The RGB color space consists of the three additive primaries: red, green and blue. Spectral components of these colors combine additively to produce a resultant color. The RGB model is represented by a 3-dimensional cube with red green and blue at the corners on each axis. Black is at the origin. White is at the opposite end of the cube. The gray scale follows the line from black to white. In a 24-bit color graphics system with 8 bits per color channel, red is (255, 0, 0). On the color cube, it is (1, 0, 0). The RGB model simplifies the design of computer graphics systems but is not ideal for all applications. The red, green and blue color components are highly correlated. This makes it difficult to execute some image processing algorithms. Many processing techniques, such as histogram equalization, work on the intensity component of an image only.

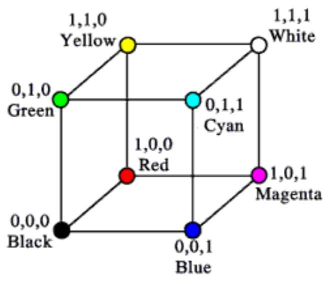


Figure 7: *RGB color cube*

2.1.2.2. YCbCr Color Space

YCbCr color space has been defined in response to increasing demands for digital algorithms in handling video information, and has since become a widely used model in a digital video.

It belongs to the family of television transmission color spaces. The family includes others such as YUV and YIQ. YCbCr is a digital color system, while YUV and YIQ are analog spaces for the respective PAL and NTSC systems. These color spaces separate RGB (Red-Green-Blue) into luminance and chrominance information and are useful in compression applications however the specification of colors is somewhat unintuitive.

$$[Y \quad Cb \quad Cr] = [R \quad G \quad B]\begin{bmatrix} 0.299 & -0.168935 & 0.499813 \\ 0.587 & -0.331665 & -0.418531 \\ 0.114 & 0.50059 & -0.081282 \end{bmatrix}$$

Figure 8: *The RGB to YCbCr conversion matrix*

2.1.2.3. HSI COLOR SPACE

Since hue, saturation and intensity are three properties used to describe color, it seems logical that there be a corresponding color model, HSI. When using the HSI color space, you don't need to know what percentage of blue or green is required to produce a color. You simply adjust the hue to get the color you wish. To change a deep red to pink, adjust the saturation. To make it darker or lighter, alter the intensity.

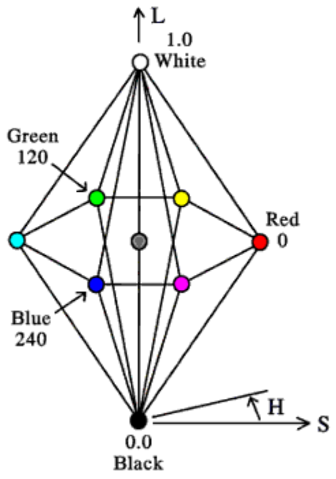


Figure 9: *Double cone model of HSI color space*

As we can see in the figure above HSI is modeled with cylindrical coordinates. The hue (*H*) is represented as the angle 0, varying from 0º to 360º. Saturation (*S*) corresponds to the radius, varying from 0 to 1. Intensity (*L*) varies along the *z* axis with 0 being black and 1 being white.

When *S* = 0, color is a gray value of intensity 1. When *S* = 1, color is on the boundary of top cone base. The greater the saturation, the farther the color is from white/gray/black (depending on the intensity). Adjusting the hue will vary the color from red at 0º, through green at 120º, blue at 240º, and back to red at 360º. When *L* = 0, the color is black and therefore *H* is undefined. When

S = 0, the color is grayscale. *H* is also undefined in this case. By adjusting *L*, a color can be made darker.

2.1.2.4. OTHER COLOR SPACES

– **HSV Color Space**

The HSV space defines color as Hue, Saturation and Brightness. The transformation of RGB to HSV is invariant to high intensity at white lights, ambient light and surface orientations relative to the light source and hence, can form a very good choice for skin detection methods.

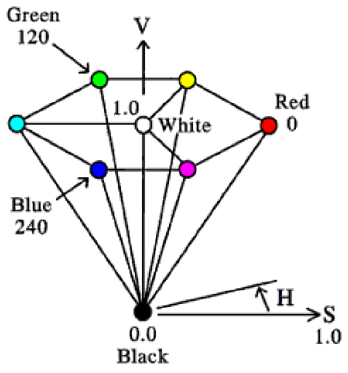


Figure 10: *The six sided pyramid HSV color space*

– **CIELab and CIELuv**

Perceptual uniformity represents how two colors differ in appearance to a human observer and hence uniform color spaces (UCS) were defined such that all the colors are arranged by the perceptual difference of the colors. However, the perceptual uniformity in these color spaces is obtained at the expense of heavy computational transformations. In these color spaces, the computation of the luminance (L) and the chroma (ab or uv) is obtained through a non-linear mapping of the XYZ coordinates.

They are nearly linear with visual perception, or at least as close as any color space is expected to sensibly get. Since they are based on the CIE system of color measurement, which is itself based on human vision, CIELab and CIELuv are device independent but suffer from being quite unintuitive despite the L parameter having a good correlation with perceived lightness.

Our face detection method has based in CIELab color space, so can we find more information about it in *the Appendix B1.*

2.2. IMAGE PROCESSING

Image processing is the processing in which the image is passed through a processor where a new shape is given to image. For the image processing to work, it is important to take into account that it must be passed through a three stage process of input, process and output. In this particular situation, the input is basically the image that is to be changed and in the image processing document the particular image is changed with the help of implemented functions from the process. After the process stage had been completed the output is produced.

Image processing can be used to feature different items such as the face detection of the person, the microscope image, the optical imaging process and many other types of application can also be used. It is found to be easily used as in it only the individual had to insert the image that needs to be treated with certain changes and after that the desired changes can be implemented. Some of the techniques usually used are explained below.

2.2.1. HISTOGRAM MODELING TECHNIQUES

These techniques are mainly aimed at improving the image display. The histogram of an image is a chart that provides a comprehensive description of the appearance thereof, the axis of abscissa represents the range of pixel values, while the y-axis represents the range of values that can take those pixels.

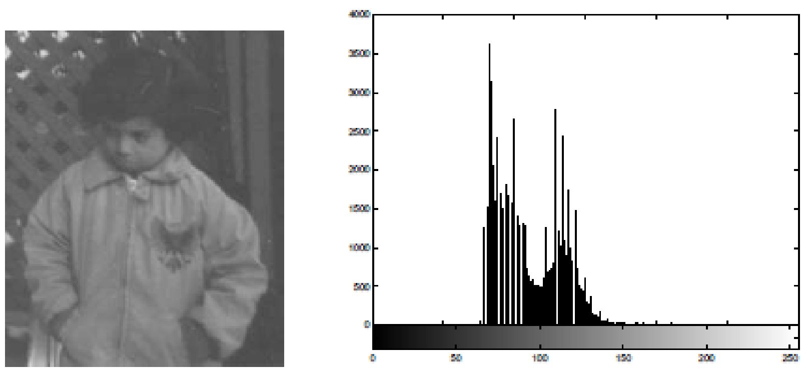


Figure 11: *Original image and its histogram*

2.2.1.1. HISTOGRAM EQUALIZATION

Histogram equalization is one of the most widely used techniques for improving the contrast of the original image. This technique enhances the original image by a given modification or alteration of the histogram called EQ or equalization of the histogram.

What we seek is to find a function F (g) that enhances the overall contrast in the original image to expand the distribution of gray levels, or levels R, G and B in the case of an RGB color image.

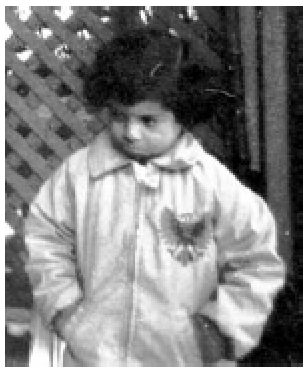
This expansion should be smooth in the sense that ideally should have the same number of pixels per gray level. The purpose of histogram equalization is to distribute the levels of gray (or color for RGB) uniformly over the entire range of values of gray levels.

Considering the values of maximum and minimum intensity in the range of gray levels of the given image, g max and g min, respectively, and considering a probability distribution function of uniform.

$$F(g) = [g_{max} - g_{min}]\, P_g(g) + g_{min}$$

$$P_g(g) = \sum_{g=0}^{s} p(g)$$

Overall and in summary when we apply a histogram equalization on an original image, the effect that occurs with a uniform density distribution is an enhanced image.



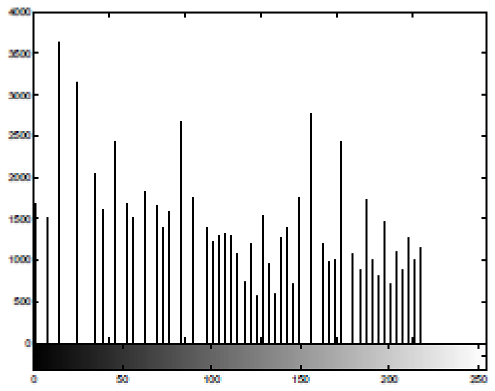


Figure 12: *Image of Figure 11 after equalization and its histogram*

There are other histogram modeling techniques like Histogram Shifting and Histogram Expansion:

- Histogram Shifting: The displacement of the histogram is used to lighten or darken an image but maintaining the relationship between the values of gray levels or color.
- Histogram Expansion: Histogram expansion is the operation that takes an input image f expands the histogram along the full range of values of gray levels or color. As such it has the effect of increasing the contrast of an image of low contrast.

2.2.2. SPATIAL FILTERING

Spatial filtering is a procedure where we apply a function to a neighborhood of each pixel. The idea is to move a mask: a rectangle (usually with sides of odd length) or other shape over the given image. As we do this, we create a new image whose pixels have grey values calculated from the grey values under the mask.

The combination of mask and function is called filter. If the function by which the new grey value is calculated is a linear function of all the grey values in the mask, then the filter is called a linear filter.

2.2.2.1. LOW PASS FILTERING

A low pass filter allows low spatial frequencies to pass unchanged, but suppress high frequencies. The low pass filter smoothes or blurs the image. This tends to reduce noise, but also obscures fine detail.

The following shows a 3 x 3 kernel for performing a low-pass filter operation, where M is total number of pixels in the neighborhood N

$$h = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad g(x,y) = \frac{1}{M} \sum_{j,k \in N} f(j,k)$$

Figure 13: *Kernel filter 3x3*

In regions of low spatial frequency (where a neighborhood's pixel values are about the same), the output pixel is nearly identical to the input pixels. Hence the name low-pass, implying that low-frequency areas are unchanged. High-frequency regions, though, will experience the same averaging of pixels, which tends to eliminate the rapid changes from dark to light.

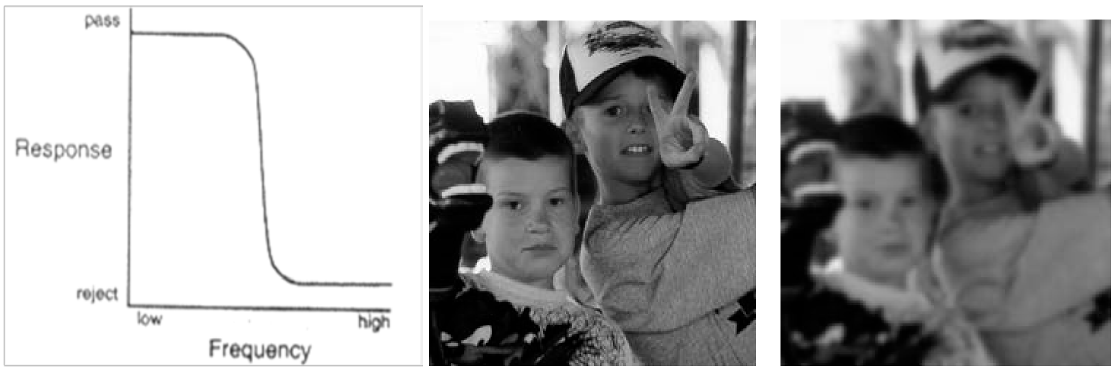


Figure 14: *Low pass filter a) Curve Response b) Original image c) Filtered image*

The *median filter* is a simple edge-preserving smoothing filter. It may be applied prior to segmentation in order to reduce the amount of noise in a stack of 2D images. The filter works by sorting pixels covered by a N x N mask according to their gray value. The center pixel is then replaced by the median of these pixels, i.e., the middle entry of the sorted list.

2.2.2.2. HIGH PASS FILTERING

Since low-pass filtering can be accomplished using convolution, it follows that high-pass filters also exist. The classic 3 x 3 implementation is $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

The sum of the coefficients in this kernel is zero. This means that, when the kernel is over an area of constant or slowly varying grey level, the result of convolution is zero or some very small number. However, when gray level is varying rapidly within the neighborhood the result of convolution can be a large number. The number can be positive or negative and we need to choose an output image presentation that supports negative numbers.

The effect of these filters is that he high frequencies, or edges of the image are highlighted, while the low frequencies are diminished. The visual impact of this is to make the image appear sharpened.

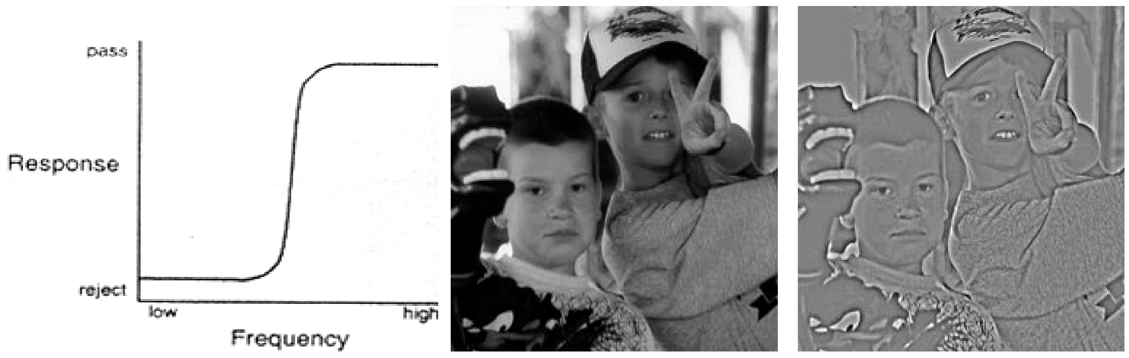


Figure 15: *High pass filter a) Curve Response b) Original image c) Filtered image*

In high pass filtering the objective is to get rid of the low frequency or slowly changing areas of the image and to bring out the high frequency or fast changing details in the image. This means that if we were to high pass filter the box image we would only see and outline of the box. The edge of the box is the only place where the neighboring pixels are different from one another.

2.2.2.3. EDGE DETECTION FILTERS

They perform other kind of operations with data, but always with the result of emphasizing the borders around an object in an image to make it easier to analyze. These filters typically create an image with gray background, and black and white lines around the edges of objects and features of the image.

➢ **Sobel Operator**

The operator consists of a pair of 3×3 convolution kernels as we can see in *Figure 16*. One kernel is simply the other rotated by 90°.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 16: *Masks used by Sobel Operator*

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient

➢ **Robert's cross operator**

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point. The operator consists of a pair of 2×2 convolution kernels as shown in *Figure17*. One kernel is simply the other rotated by 90º]. This is very similar to the Sobel operator.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Figure 17: *Masks used for Robert Operator*

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient.

➢ Prewitt's operator

Prewitt operator is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images.

-1	0	+1
-1	0	+1
-1	0	+1

Gx

+1	+1	+1
0	0	0
-1	-1	-1

Gy

Figure 18: *Masks for the Prewitt gradient edge detector*

➢ Laplacian of Gaussian

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian Smoothing filter in order to reduce its sensitivity to noise. The operator normally takes a single gray level image as input and produces another gray level image as output.

The Laplacian *L(x,y)* of an image with pixel intensity values *I(x,y)* is given by:

$$L(x,y)= \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian.

2.2.3. FREQUENCY FILTERING

Frequency filters process an image in the frequency domain. The image is Fourier transformed, multiplied with the filter function and then re-transformed into the spatial domain. Attenuating high frequencies results in a smoother image in the spatial domain, attenuating low frequencies enhances the edges.

All frequency filters can also be implemented in the spatial domain and, if there exists a simple kernel for the desired filter effect, it is computationally less expensive to perform the filtering in the spatial domain. Frequency filtering is more appropriate if no straightforward kernel can be found in the spatial domain, and may also be more efficient.

Frequency filtering is based on the Fourier Transform. (For the following discussion we assume some knowledge about the Fourier Transform, therefore it is advantageous if you have already read the corresponding worksheet.) The operator usually takes an image and a filter function in the Fourier domain. This image is then multiplied with the filter function in a pixel-by-pixel fashion:

$$G(k,l) = F(k,l)H(k,l)$$

where $F(k,l)$ is the input image in the Fourier domain, $H(k,l)$ the filter function and $G(k,l)$ is the filtered image. To obtain the resulting image in the spatial domain, $G(k,l)$ has to be re-transformed using the inverse Fourier Transform.

Since the multiplication in the Fourier space is identical to convolution in the spatial domain, all frequency filters can in theory be implemented as a spatial filter. However, in practice, the Fourier domain filter function can only be approximated by the filtering kernel in spatial domain.

The form of the filter function determines the effects of the operator. There are basically three different kinds of filters: lowpass, highpass and bandpass filters.

A low-pass filter attenuates high frequencies and retains low frequencies unchanged. The result in the spatial domain is equivalent to that of a smoothing filter; as the blocked high frequencies correspond to sharp intensity changes, *i.e.* to the fine-scale details and noise in the spatial domain image.

A highpass filter, on the other hand, yields edge enhancement or edge detection in the spatial domain, because edges contain many high frequencies. Areas of rather constant graylevel consist of mainly low frequencies and are therefore suppressed.

A bandpass attenuates very low and very high frequencies, but retains a middle range band of frequencies. Bandpass filtering can be used to enhance edges (suppressing low frequencies) while reducing the noise at the same time (attenuating high frequencies).

2.2.4. MORPHOLOGICAL FILTERS

Morphological operators aim at extracting relevant structures of the image. This can be achieved by probing the image with another set of given shape - the structuring element (SE). *Dilation* and e*rosion* are the two fundamental morphological operators because all other operators are based on their combinations

2.2.4.1. DILATION

Dilation is an operation that grows or thickens objects in a binary image. The specific manner and extent of this thickening is controlled by a shape referred to as a structure element *(SE)*. We will define the operation of dilation mathematically and algorithmically.

First let us consider the mathematical definition. The dilation of *A* by *B*, denoted $A \oplus B$, is defined as

$$A \oplus B = \{z \mid (\widehat{B})_z \cap A \neq \Phi\}$$

where Φ is the empty set and *B* is the structure element. In words, the dilation of *A* by *B* is the set consisting of all the structure element origin locations where the reflected and translated *B* overlaps at least some portion of *A*.

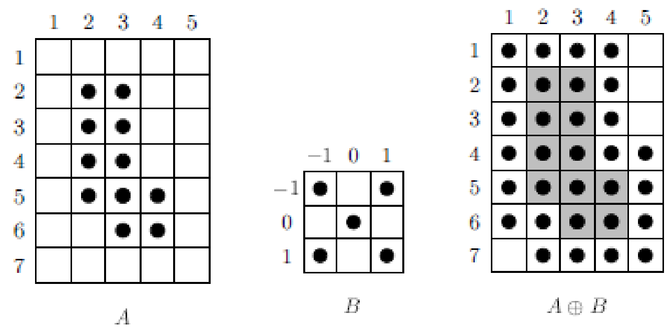


Figure 19: *Dilation of A with the Structure Element B*

2.2.4.2. EROSION

Erosion shrinks or thins objects in an image. The mathematical definition of erosion is similar to that of dilation. The erosion of *A* by *B*, denoted *A B*, is defined as:

$$A \ominus B = \{z \mid (B)_z \cap A^c \neq \Phi \}$$

In other words, erosion of *A* by *B* is the set of all structure element origin locations where the translated B has no overlap with the background of *A*.

Algorithmically we can define erosion as: the output image $A \ominus B$ is set to zero. *B* is place at every black point in *A*. If A contains *B* (that is, if *A AND B* is not equal to zero) then *B* is placed in the output image. The output image is the set of all elements for which *B* translated to every point in *A* is contained in *A*.

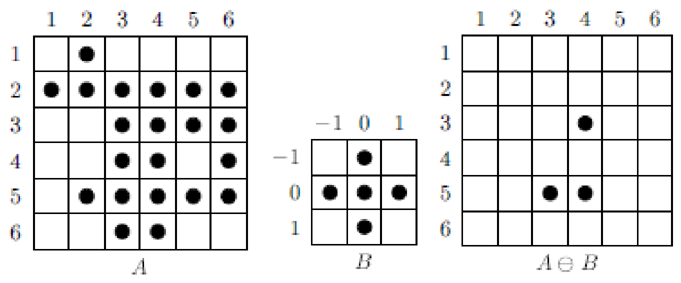


Figure 20: *Erosion of A with the Structure Element B*

2.2.4.3. OPENING

Once an image has been eroded, there exists in general no inverse transformation to get the original image back. The idea behind the morphological opening is to dilate the eroded image to recover as much as possible the original image.

The process of erosion followed by dilation is called *opening*. The opening of *A* by *B*, denoted *A B* is defined as:

$$A \circ B = (A \ominus B) \oplus B$$

The geometric interpretation for this formulation is: *A o B* is the union of all translations of *B* that fit entirely within *A*. Morphological opening removes completely regions of an object that cannot contain the structure element, generally smoothes the boundaries of larger objects without significantly changing their area, breaks objects at thin points, and eliminates small and thin protrusions.

2.2.4.4. Closing

The process of dilation followed by erosion is called *closing*. The closing of *A* by *B*, denoted *A B*, is defined as:

$$A \circledast B = (A \oplus B) \ominus B$$

Geometrically, the closing $A \circledast B$ is the complement of the union of all translations of *B* that do not overlap *A*. It has the effect of filling small and thin holes in objects, connecting nearby objects, and generally smoothing the boundaries of objects without significantly changing their area.

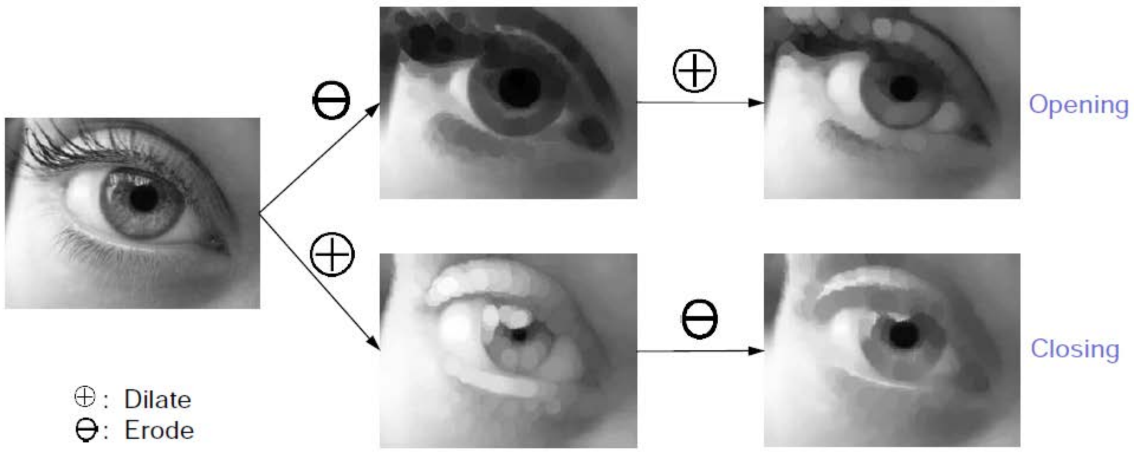


Figure 21: *Opening and closing of a gray-scale image with an 8 × 8 disk Structure Element*

CHAPTER 3: FACE DETECTION

Face Detection is a technology to determine human face in videos and arbitrary digital images. It can be regarded as a specific case of object-class detection. For the detection is to locate the face in the digital images/video stream, no matter what the pose, scale, facial expressions. It task to identify a given images to decides it has face or not.

Sometimes there is much confusion with the concepts of "face detection" and "face recognition". It is important not to confuse because face recognition and face detection are two distinct processes. The main difference is that face recognition is a technique to detect faces and search through a dataset in order to find an exact match but on the other hand face detection is looking for any match and as soon as a match is found the search stops.

3.1. HISTORY OF FACE DETECTION

In the early stage, face detection algorithms mainly focused to detect the frontal human face. However, newer algorithms try to consider the different view of face as a core of face detection.

The first of face detection system has been developed since in early 1970's. Due to the limitation of computation, system can't be satisfied the requirement of users, which is identify passport photograph real time.

At the beginning of 1990's, techniques are proposed focused on the face recognition on and increase the need of face detection. Many systems were constructed to deal with video streaming. In the past few years, lots of methods are developed at least more than 150 methods.

3.2. FACE DETECTION TECHNIQUES CLASSIFICATION

According to their main image processing techniques, we can classify face detection methods into two main approaches: Features-based approaches and

Image-based approaches. Also, these two approaches have several sub face detection methods. Following figure show main two approaches and their sub-methods.

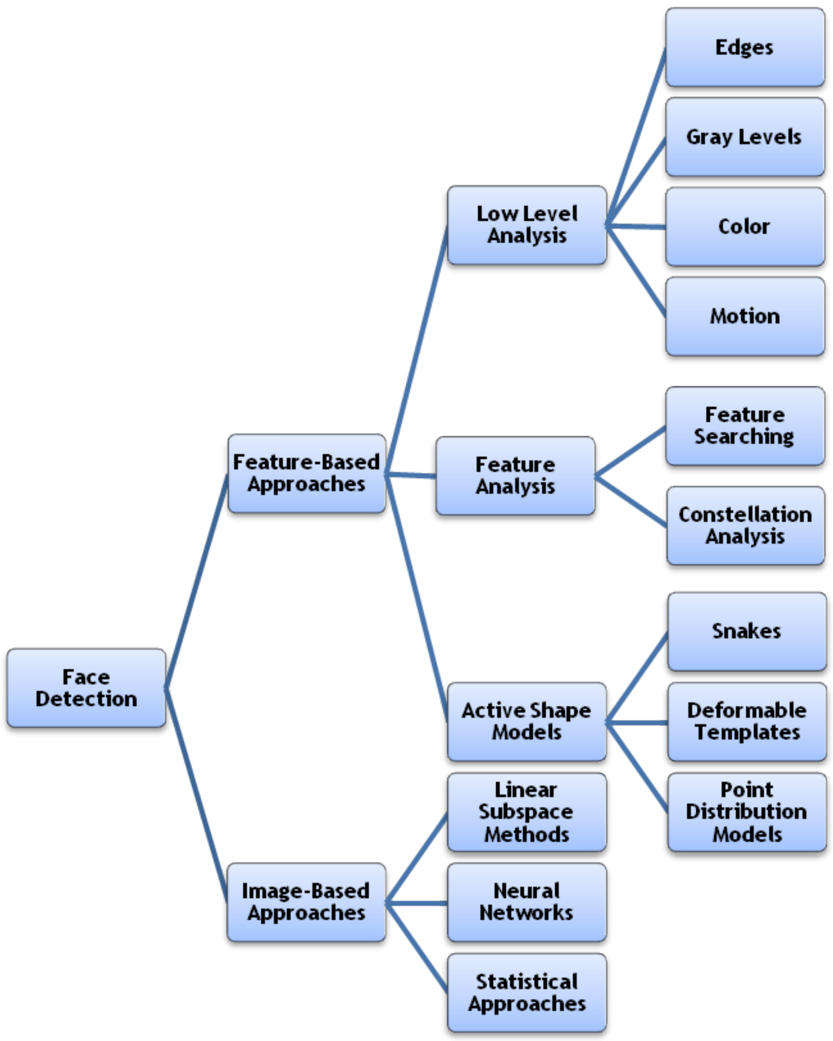


***Figure 22:** Face detection approaches diagram*

3.2.1. Feature-Based Face Detection Approaches

Feature-based face detection approach is divided into three areas. These are low-level analysis, feature analysis, and active shape models. These sub-approaches are methods to find some facial features. To design robust face detection system, instead of using just one approach, using hybrids of these approaches gives us more success. Here, there is more information about sub-approaches of Feature-Based face detection.

3.2.1.1. LOW LEVEL ANALYSIS

Low-level analysis deals with the segmentation of basic visual features by using pixel properties like intensity levels, edges, and color properties.

➢ **Edges**

The idea is based on analyzing the lines that form the edges of a face and use them to detect the facial features. The algorithm follows these steps:

- Detect the edges of the image.
- After obtaining the edges, it proceeds to a thinning to obtain for each edge, one line of a pixel wide to represent it
- Filtering components. The algorithm is only with the components that are most likely to be part of a face. For example, looking for lines which together resemble an ellipse of certain proportions of width and height.
- Labelling. Once obtained these components are labeled as right side of the face, left side, hairline, etc.

➢ **Gray Levels**

Gray levels in face can be used as features. Facial features like eyebrows, pupils, and lips are usually darker than their surrounding regions. By this property, we can differentiate various facial parts. Several face detection algorithm were developed based on this differentiation. This kind of algorithms consist of the following parts:

- Increase the contrast of the image to highlight again the difference in brightness between the different parts of the face.
- Thresholding. The algorithm keeps only areas of the image whose gray value exceeds a threshold.
- Face detection using weighted templates.

➢ **Color**

While gray level information gives us basic features about face representation, color information can provide us more face features by extra dimensions of pixel representation. For example, same features in intensity space can be very different in color space. On the other hand, skin color estimations can help us about finding possible face regions in the image. Actually, this is complex task when faces of different races are considered.

The most common color model is RGB representation in which colors are defined by combinations of red, green, blue color components. Lighting conditions can dramatically change RGB variations of images. Because of this reason, normalized RGB values are preferred in color-based feature detection.

On the other hand, other color representation models are used for face detection like HIS, HSV, YUV, YCrCb, as we will see in the next chapter, the author of our code performs the transformation to the CIE Lab color space (more information in *Appendix B1*)

Color based face detection is usually performed by using skin color thresholds according to pre-calculated skin color models. More complex methods use statistical measures of large training sets (adaptive learning). These methods' implementations can be improved by new face examples, and they become more robust against environmental factors' changes like illumination conditions or camera characteristics.

- **Motion**

When use of video sequence is available, motion information can be used to locate moving objects. Moving silhouettes like face and body parts can be extracted by simply thresholding accumulated frame differences. Besides face regions, facial features can be located by frame differences.

3.2.1.2. FEATURE ANALYSIS

Low-level analysis features are not very effective and robust. By these features, possible face regions can be found, but also false regions are found. For example, in skin color model face detection, background objects of similar color can be also detected as a face region. To solve this problem, higher level feature analysis can be used. In feature analysis, visual features are organized due to global concept of face by using face geometry information. Feature analysis approach is divided into two sub-approaches. These are feature searching and constellation analysis. Feature searching strategies are based on relative positions of simple facial features. Constellation strategies use flexible features of various face models.

- **Feature Searching**

In the feature searching strategies, firstly important facial features are determined. These features are generally biometrics measurements like eyebrows' lines, eyes' circles, etc. In the literature, most commonly used facial feature is distinct side by side appearance of a pair of eyes. Also, main face axis, outline (top of the head), and body (below the head) are searched to detect face regions. After determining features which we focus on, these features are searched. According to found features' orientations and geometric their geometric ratios, face regions are detected.

- **Constellation Analysis**

Feature models of low-level analysis and feature searching strategies are more rigid. Because of this reason, locating faces of various poses in complex backgrounds can fail. To solve this problem, feature constellation analysis strategies are developed. In these face-like constellation strategies, more robust modeling methods like statistical analysis are used.

3.2.1.3. ACTIVE SHAPE MODELS

Active shape models focus on complex non-rigid features like actual physical and higher level appearance of features. Active shape models use local features (edges, brightness) to find shape of feature models. Active shape models are divided into three groups: These are snakes, deformable templates, and point distribution models.

- **Snakes**

In this approach, active contours (snakes) are used to locate head boundary. Also features' boundaries can be found by these contours.

- **Deformable Templates**

Locating facial features' boundaries by using active contours is not easy task. Finding and locating facial edges is difficult. Sometimes there can be edge detection problems because of bad lighting or bad contrast of image. So, we need more flexible methods. Deformable templates approaches are developed to solve this problem. Deformation is based on local valley, edge, peak, and brightness.

- **Point Distribution Models**

These models are compact parameterized descriptions of the shapes based on statistics. By using principal components analysis, variations of features in the training set develop point distribution models. So, point distribution models become linear flexible models.

3.2.2. IMAGE-BASED FACE DETECTION APPROACHES

In feature-based face detection approaches, feature models have troubles with unpredictability of face appearance and environmental conditions. Some of feature based methods are improved against unpredictability, but there is still needs for improvements against unpredictability. Image-based face detection approaches are developed for these needs. In image based approaches, window scanning techniques are applied to original images.

According to remodeled and trained face information, multiple faces are detected. Image-based face detection approaches can be divided into three main sub-approaches. These are linear subspace methods, neural network approaches, and statistical approaches.

3.2.2.1. LINEAR SUBSPACE METHODS

Human face images lie in a subspace of overall image space. By using this subspace concept, several analysis methods are developed. In image processing world, the most important three methods are principal component analysis (PCA), linear discriminant analysis, and factor analysis (FA).

Principal component analysis (PCA) involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components*. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. According to this analysis method, firstly principal components are expressed. In face detection examples, eigenvectors of face images can be used. Each individual face in the face set can be approximately represented by a linear combination of largest eigenvectors (eigenfaces). PCA method is used in a lot of face detection methods.

In linear discriminant analysis method, dimensionality reduction techniques are used like PCA method, but LDA produce transformation matrix which maximize between class variance, and minimize within class variance. This analysis method is not common like PDA, but it is used in a lot of face detection system.

3.2.2.2. NEURAL NETWORKS

Neural networks can be applied successfully in face detection systems. The advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class conditional density of face images. However, one drawback is that the network architecture has to be extensively tuned (number of layers, number of nodes, learning rates, etc.) to get exceptional performance. Several Neural network models are used in face detection projects.

3.2.2.3. STATISTICAL APPROACHES

Apart from linear subspace methods and neural networks, there are several other statistical approaches to face detection like systems based on information theory, a support vector machine, and Bayes' decision rule.

CHAPTER 4: DEVELOPMENT AND IMPROVEMENT

The face detection system we use is to locate human faces in a given image, where the output of this system is the input image but with a rectangle located on the faces detected. This are the different steps of the code proposed:

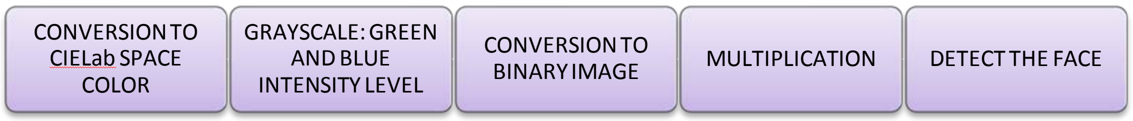

CONVERSION TO CIELab SPACE COLOR
GRAYSCALE: GREEN AND BLUE INTENSITY LEVEL
CONVERSION TO BINARY IMAGE
MULTIPLICATION
DETECT THE FACE

Figure 23: *Structure of the algorithm*

The software used for the program implementation is MATLAB:

- Matlab is a data analysis and visualization tool which has been designed with powerful support for matrices and matrix operations. As well as this, Matlab has excellent graphics capabilities, and its own powerful programming language. One of the reasons that Matlab has become such an important tool is through the use of sets of Matlab programs designed to support a particular task. These sets of programs are called toolboxes, and the particular toolbox of interest to us is the image processing toolbox.

  Image Processing Toolbox™ provides a comprehensive set of reference-standard algorithms and graphical tools for image processing, analysis, visualization, and algorithm development. You can perform image enhancement, image deblurring, feature detection, noise reduction, image segmentation, spatial transformations, and image registration. Many functions in the toolbox are multithreaded to take advantage of multicore and multiprocessor computers.

4.1. DATABASES

For the development of the project we have chosen two databases *[web]*, to check the operation of the method. We have tried to choose these databases with different characteristics to compare both situations and form good conclusions.

4.1.1. DATABASE A: PICTURES WITH CONTROLLED BACKGROUND

This database belongs to "International Workshop on Visual Observation of Deictic Gestures of Cambridge, UK". The head pose database is a benchmark of 2790 monocular face images of 15 persons with variations of pan and tilt angles from -90 to +90 degrees. For every person, 2 series of 93 images (93 different poses) are available.

The images are taken in a controlled environment, the background is white, where changes in lighting are very small in a high percentage of the images, people usually are at a fixed distance from the camera (no significant changes), and all images are the same size, 384 x 288 pixels.

Filenames are constructed according the following grammar:

person[Id][Serie][Number][Tilt][Pan].jpg

Number of the person	Id = {01, ..., 15}
Number of the serie	Serie = {1, 2}
Number of the file in the directory	Number = {00, 01, ..., 92}
Vertical angle	Tilt = {-90, -60, -30, -15, 0, +15, +30, +60, +90}
Horizontal angle	Pan = {-90, -60, -45, -30, -15, 0, +15, +30, +45, +60, +90}

Table 1: *Database A. Grammar of the filename*

We have considered only 105 images with the more proximate angles to the frontal position. Due to our method is not interested in position recognition, is estimated that this number of images is sufficient to check the quality of the code in controlled backgrounds pictures.

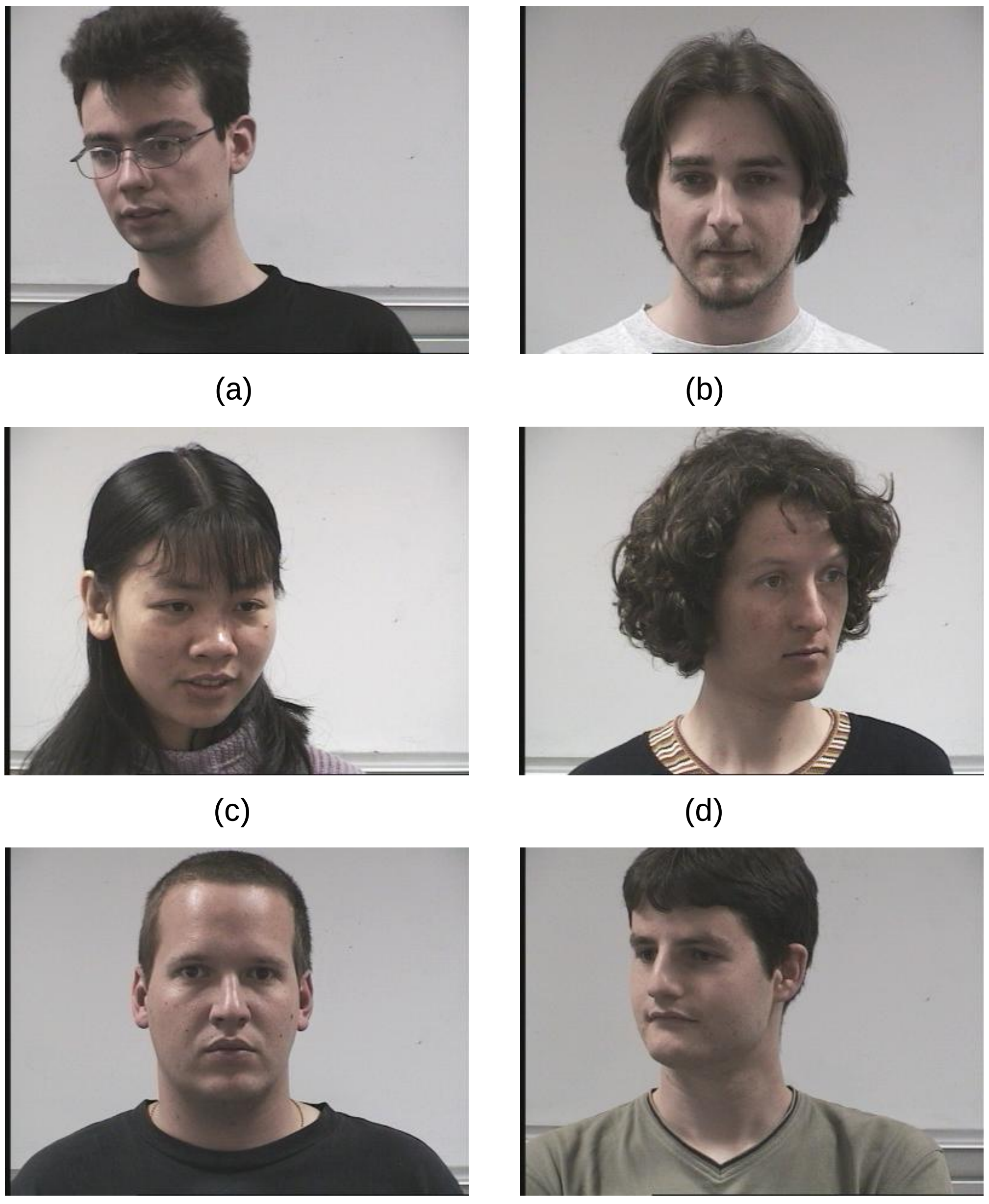

(a) (b) (c) (d) (e) (f)

Figure 24: *Examples of Database A: Controlled background*

4.1.2. DATABASE B: PICTURES WITHOUT CONTROLLED BACKGROUND

Our second database is a frontal face dataset collected by Markus Weber at “California Institute of Technology”. It contains 450 face images with the same size, 896 x 592 pixels, and Jpeg format. There are 27 or so unique people under with different facial expressions, illumination conditions and backgrounds.

Filenames are constructed according the following grammar, where Id is the unique identifier of the image, each image has a different identifier.

person_[Id]

We only have considered 135 images to test the code in non controlled environment photos.

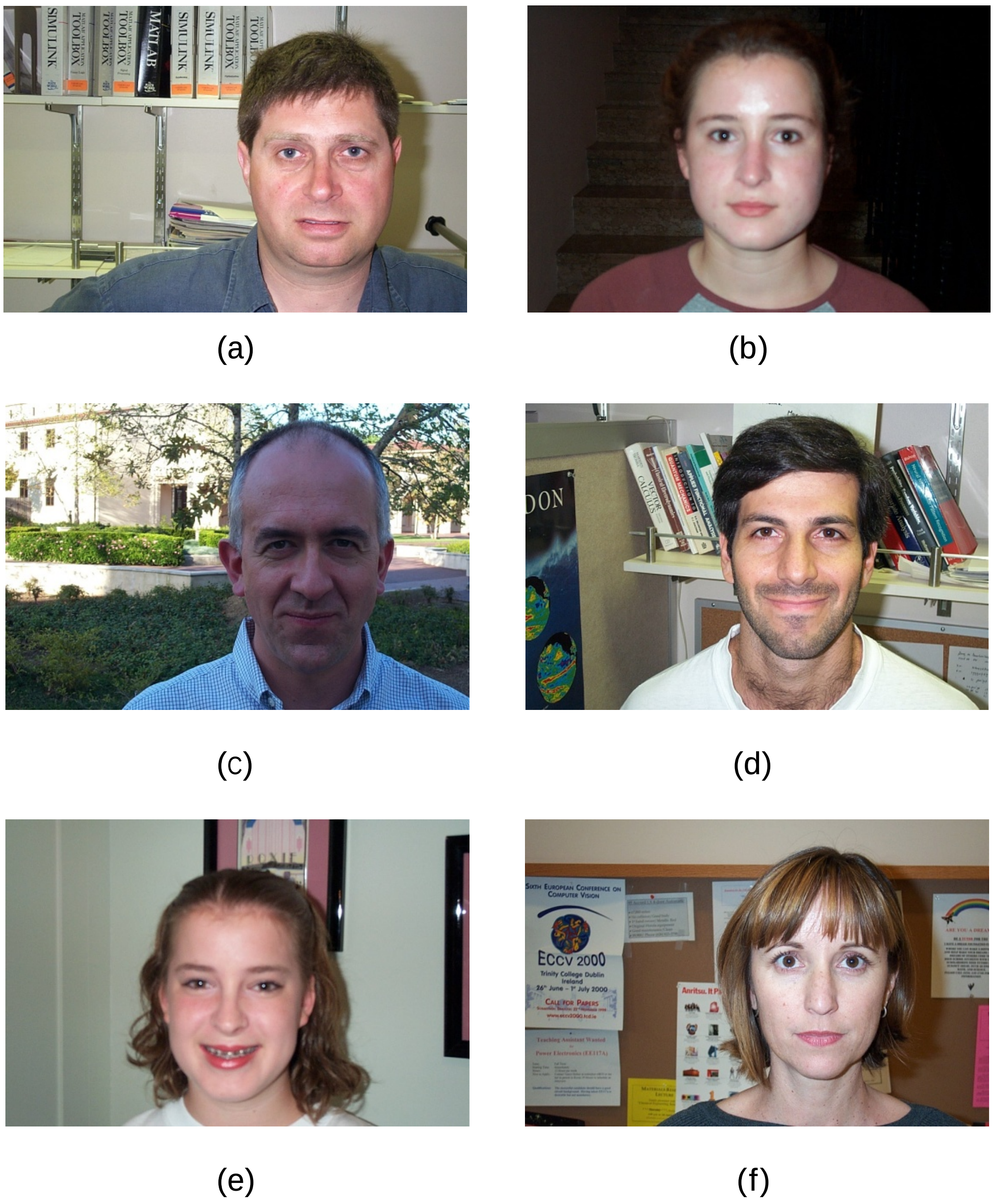


Figure 25: *Examples of Database B: Non controlled background*

4.2. DEVELOPMENT

This section describes in detail the algorithm used in the proposed code *[web],* in order to understand its operation. We will only explain the most significant functions and statements.

The first step we have done is to implement an algorithm to read all the images of the database from a directory (in our code it´s called "pathDatabase"), then we process them, and we save the output results in other directory (in our case "pathResults"). This algorithm is implemented in Matlab too, and it´s called "readImages.m", as we can see in Appendix A.

The most important operations of *readImages.m* are implemented by the following functions:

Name	Function	Input	Output
imread	Read an image (color image in our case) from the file specified	String that specifies the file name	Array containing the image data
imwrite	Writes an image to the specified file	Array containing the image data	——

Table 2: *Matlab functions to implement the input/output of images*

We have chosen two images, one of each database as examples, to see the different processing steps of this technique. These images are shown in *Figure 26.*

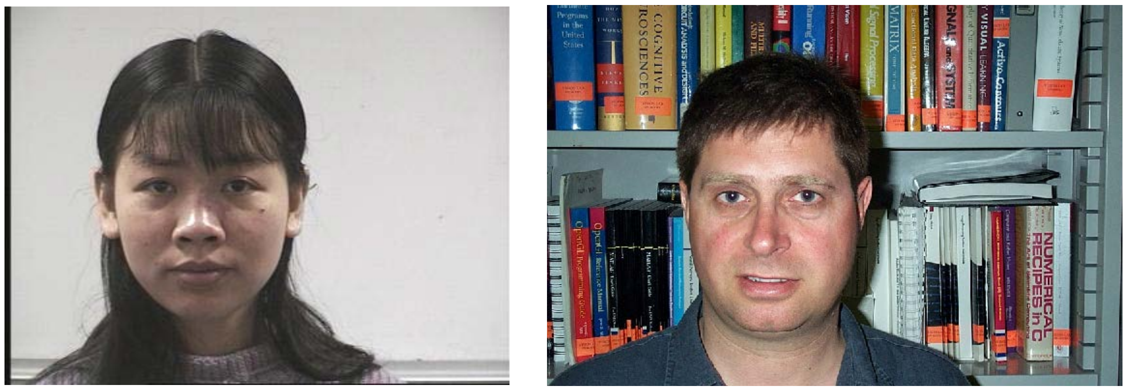


Figure 26: *Examples of input images a) Database A, b) Database B*

To process the images, we have to use the algorithm proposed called “faceDetection.m” (Matlab code in *Apendix A*). We can use it since our previous function *readImages.m.*

The second step in the development of this program is to modify the original image, in which we are interested in detecting faces. To eliminate the effects of light, it is necessary to take the input color image to the system and convert to another color space because the space in which is the original color image is not reliable for the characterization of skin colors, because the three components (r, g, b) represent not only color, but also represent the brightness or luminance, which can vary the face of a person due to environmental conditions of light, thus not a reliable measure in separating skin regions contain which do not contain. Luminance can be removed from the color representation, and putting it to chromatic color space.

This method uses CIELab color space, which is valid because it has image intensity as one of its components. First it uses color transform functions to convert the image from RGB to L*a*b* color space, and then work on the luminosity layer 'L*' of the image. Manipulating luminosity affects the intensity of the pixels, while preserving the original colors.

To implement this color space transformation, we use the following Matlab functions:

Name	**Function**	**Input**	**Output**
makecform	Transform the color space specified	String that defines the color space conversion(1)	The color transformation structure
applycform	Apply the color space transformation specified in the input arguments	Array with the image data to transform and the color transformation structure	Array containing the transformed image data

Table 3: *Matlab functions to implement the color space transformation*

(1) In the proposed code the input string is 'srgb2lab', that Converts from the sRGB to the CIE L*a*b* color space.

With the previous functions we obtain these results:

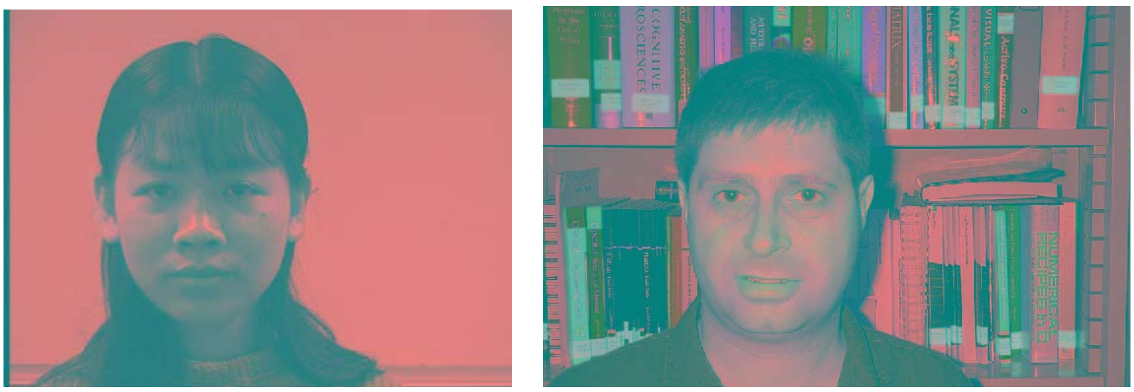


Figure 27: *Examples of conversion to CIELab space color a) Database A b) Database B*

In color skin distribution of different races people, it found that it was clustered in a small area of chromatic color space. Although the skin color of different people appears to vary over a wide range, it is not, they differ much less in color than in brightness. In other words, the skin color of different people is clustered, but differs mainly in intensity [4].

The Chroma Graph obtained by [4] Cai and Goshtasby is the result of processing of 2300 samples of skin obtained from a wide range of skin colors from 80 test images and it was found that they all fall within a small region within the CIE LAB space 1976, *a* Є [-10, 60] and *b* Є [-10, 40]. Where the two components represent the values of *a* and *b* in CIE LAB 1976 Space.

The color chart in *Figure 28* was obtained from 2,300 samples of skin taken from a wide range of skin colors in 80 images. The horizontal axis shows the component a, and the vertical axis shows the color component b of the graph.

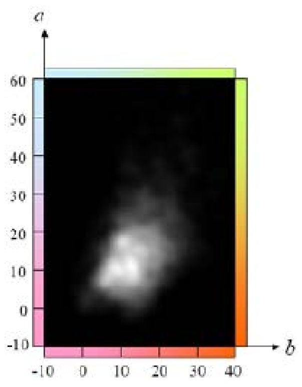


Figure 28: *Chroma Chart*

The third step of the program creates, from the previous image (Lab color space), two grayscale images with the peculiarity that the first represents the intensity levels of green in the image, and the second one represents the intensity levels of blue. The aim with this process is, from these images, set a level for subsequent thresholding segmentation.

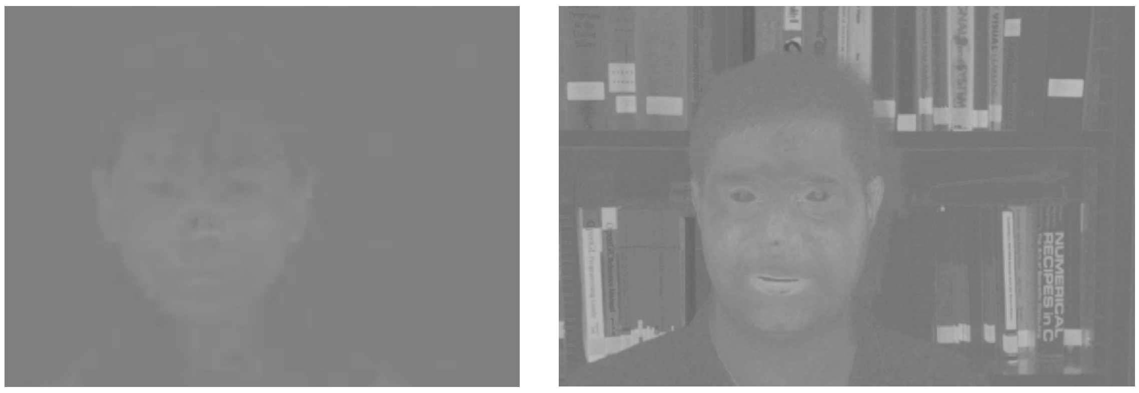


Figure 29: *Examples of grayscale images representing the intensity level of green in them a) Database A, b) Database B*

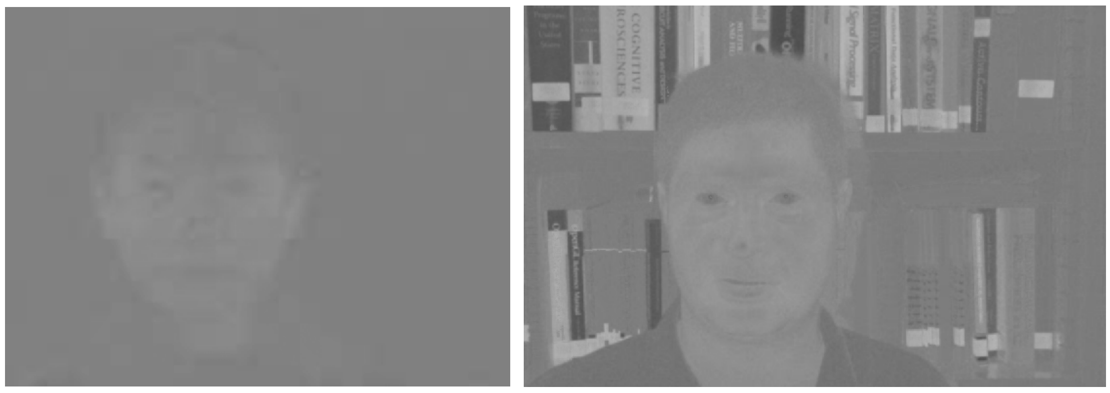


Figure 30: *Examples of grayscale images representing the intensity level of blue in them a) Database A, b) Database B*

Because the skin regions are brighter than elsewhere in the grayscale image, skin regions can be segmented from the rest of the image through a process where the threshold for segmentation is not be a fixed value for all input images, since the images contain some people with different skin types, so a fixed threshold is not possible.

In the proposed code it´s used Otsu's method to obtain the optimal threshold value for each grayscale image. Otsu´s algorithm assumes that the image that has to be thresholded contains two classes of pixels (e.g. foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal. To learn more about the operation of Otsu's method see *Appendix B*.

Once the optimal threshold value is determined (for each image), the grayscale images are converted to a binary image. The binary image has values of 1 (white) for all pixels in the input image (grayscale image) with intensity levels greater than optimum thresholding value, and 0 (black) for all other pixels.

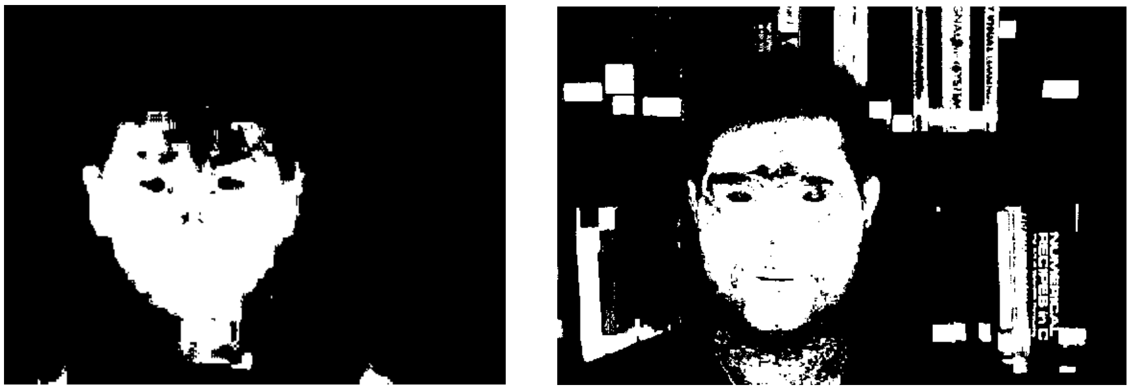


Figure 31 *Examples of binary created from the intensity level of green*
a) Database A, b) Database B

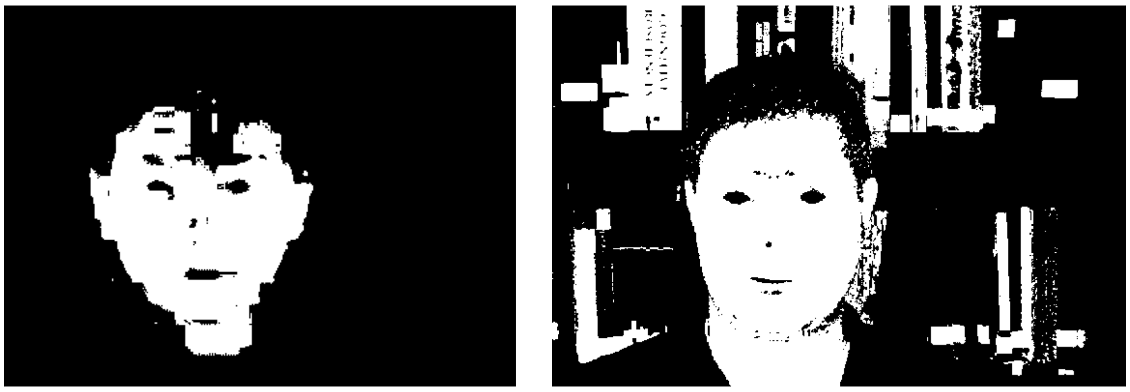


Figure 32: *Examples of binary created from the intensity level of blue*
a) Database A, b) Database B

To implement these operations in Matlab we use the following functions:

Name	Function	Input	Output
graythresh	Compute a global threshold image that can be used to convert an intensity image to a binary image[(2)]	Array containing the image data	Double scalar that represents a normalized intensity value
im2bw	Convert an image to a binary image by thresholding	Image (grayscale in our case), and double scalar with an intensity level	Binary image

Table 4: *Matlab functions to implement the thresholding and segmentation process*

In this point we have two binary images created from the intensity level o blue and green. The next step of the project is to multiply both images, obtaining by this way our final mask from which we select what are the pixels that belong to skin, and which ones are not. These are the results of this process:

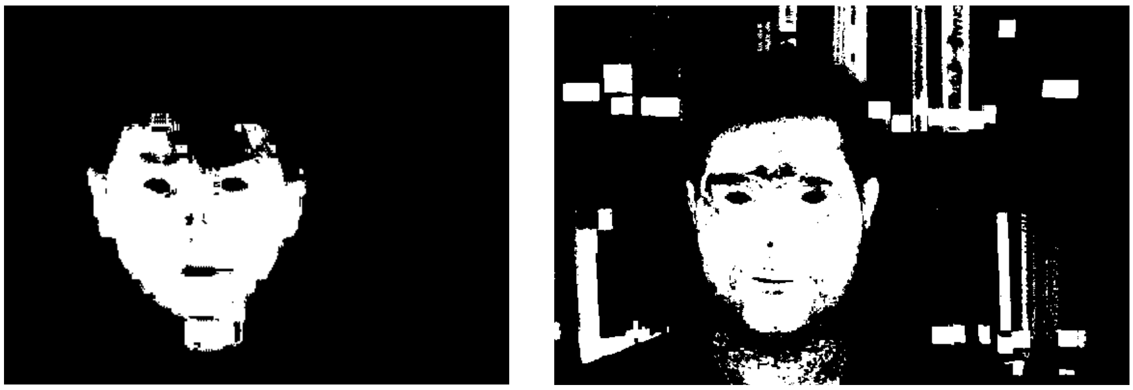


Figure 33: *Multiplication of the binary images shown in Figure 31 and Figure 32*
a) Database A, b) Database B

The next step consist in create a matrix that has to label the properties we consider necessaries. In this case, we have only chosen the BoundingBox property. It is a property which calculates the position and dimensions of the minimum rectangle that surrounds the region. The following steps that are performed are necessary to ensure that these data are displayed in matrix

(2) Graythresh uses Otsu's method to choose the threshold value

form. At first it is created a structure array, which becomes a cell array that finally becomes a matrix. In Matlab we can carry out this process with these set of functions:

Name	Function	Input	Output
bwlabel	Produce a label matrix with connected components in a binary image	Matrix containing the image data, and number of connected objects	Matrix containing labels
regionprops	Measure a set of properties of different regions	Matrix of labels that represent each labeled region, and properties$^{(3)}$	Structure array with the different properties for each region
struct2cell	Convert structure array to cell array	Structure array	Cell array
cell2mat	Convert the contents of a cell array into a single matrix	Cell array	Matrix

Table 5: *Matlab functions to implement the process to generate the bounding boxes matrix*

The last step of this proposed code is to find the position where we have to draw the rectangle with the position of the face. To do this, the author of the code uses the loop that loops through all the bounding boxes generated before, trying to find the bounding box that has the largest aspect ratio less than 1,8. Once we have the position of the face we can draw the rectangle to finish the program.

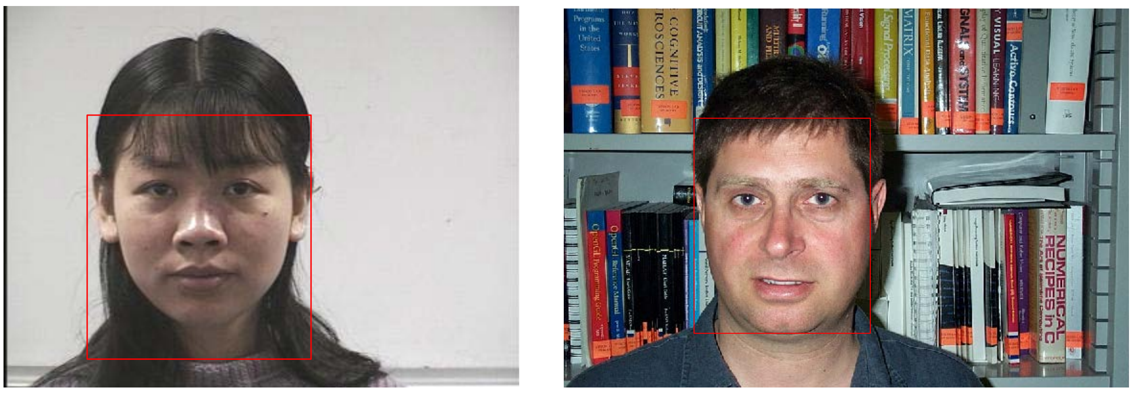


Figure 34: *Final result. Original image with the face detected*

a) Database A, b) Database B

(3) The proposed code uses the property 'BoundingBox', that computes the position and dimensions of the minimum rectangle that surrounds the region.

As we said at the beginning of the chapter, we have modified the code to save in a directory only the face of the original image cropped as we can see in *Figure 35.*

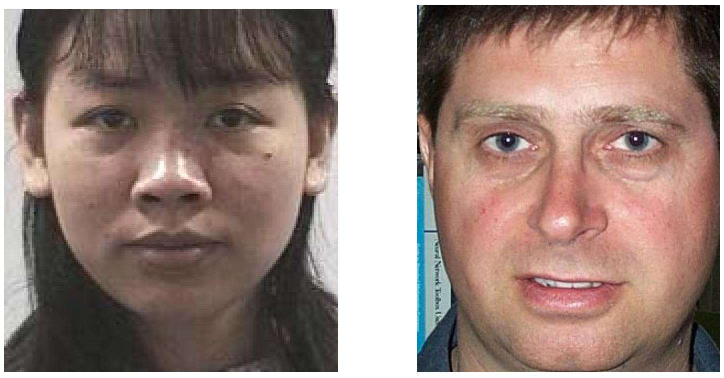


Figure 35: *Face detected cropped*
a) Database A, b) Database B

4.1.1. DIAGRAM OF THE PROCESS

In this section we only include a big diagram of the process in order to see all the steps together and understand better the algorithm operation.
The meaning of the following imager is:

A. Original image.
B. Transformation to CIELab Space Color.
C. Grayscale image. Green intensity level.
D. Grayscale image. Blue intensity level.
E. Binary image. Green intensity level.
F. Binary image. Blue intensity level.
G. Multiplication of binary images
H. Image with face detected.
I. Face detected cropped.

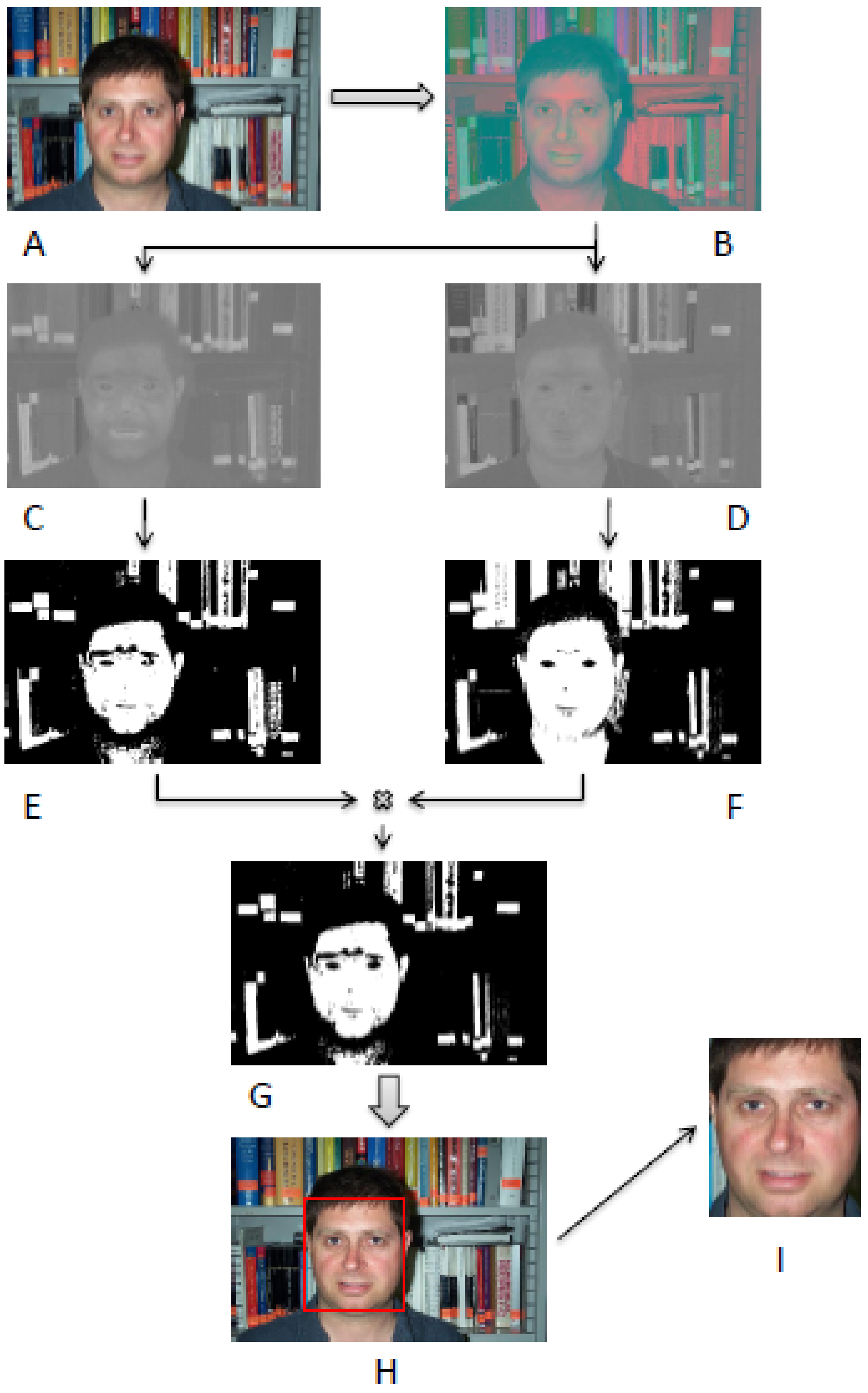


Figure 36: *Diagram of the process. Example with Database B*

4.3. EVALUATION

Using the algorithm described in the previous section has produced more than reasonable results when applied to the Database A, however, the results of Database B are not good as we can see in *Table 6.*

	NUMBER OF IMAGES	**SUCCESSFULLY DETECTED FACES**	**ERROR RATE**	***SUCCESS***
DATABASE A	*105*	*105*	*0 %*	*100 %*
DATABASE B	*135*	*71*	*47,4 %*	*52,6 %*

Table 6: *Evaluation of the algorithm*

Images with controlled background get all the hits, regardless of the angle of the face. In this case, the test conditions are very simple because this is a plain white background, just change the brightness.

Images with non controlled background have a very high percentage of error. This is unacceptable, because even though conditions are harder than Database A because of the background changes, all the faces are in frontal position and are the main focus of the image.

It is true that one of the situations that can confuse the algorithm is when we find a picture with the background color very similar to the color of the skin, but it is not the main problem. The code is simple and fast but is incomplete and has not a pre-defined number of steps, which has been evaluated in numerous studies and are critical in the face detection process.

The images below show the results obtained with this algorithm:

- DATABASE A:

(a)

(b)

(c)

(d)

(e)

(f)

Figure 37: *Results for Database A*

- DATABASE B:

(a)

(b)

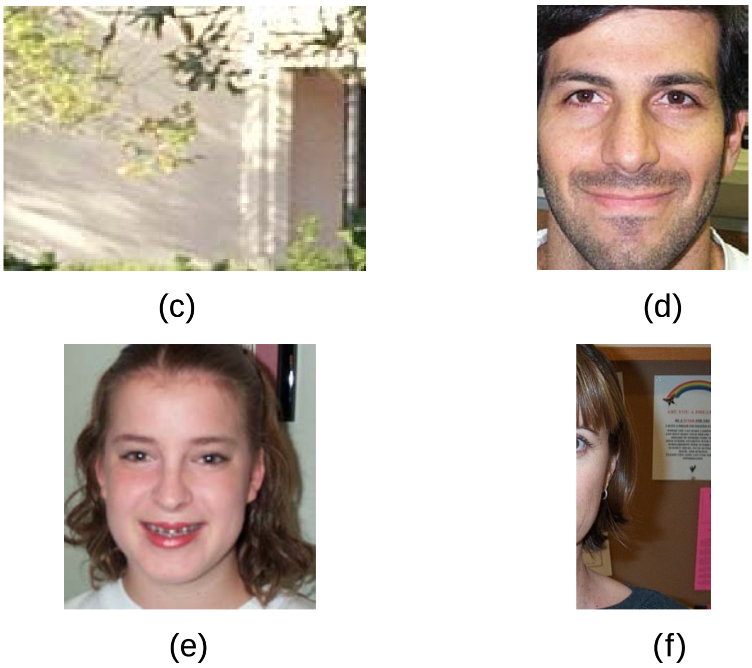


Figure 38: *Results for Database B*

Otherwise our algorithm is reasonably fast, but this point is not enough to compensate for errors produced, the method certainly does not work accurately enough.

4.4. IMPROVEMENTS

As we have seen at the beginning of the chapter, the used method presents some problems depending on the picture we use. To improve this fails, we are going to propose some changes in order to obtain better results. First of all, we are going to add a pre-processing stage that intends to improve or enhance the properties of the image to facilitate the following operations.

Preprocessing techniques can be divided generally in the following steps:

- **Contrast enhancement**: increases the total contrast of an image by making light colors lighter and dark colors darker at the same time. It does this by setting all color components below a specified lower bound to zero, and all color components above a specified upper bound to the maximum intensity (that is, 255). Color components between the upper and lower bounds are set to a linear ramp of values between 0 and 255. Because the upper bound must be greater than the lower bound, the lower bound must be between 0 and 254, and the upper bound must be between 1 and 255.

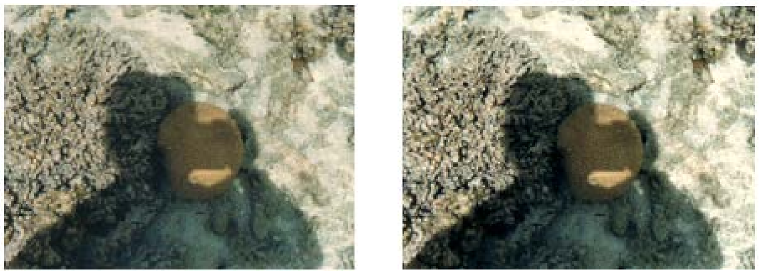


Figure 39: *Contrast enhancement. a) Original image, b) Filtered image*

- **Denoising**: Digital images are prone to a variety of types of noise. Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. There are several ways that noise can be introduced into an image, depending on how the image is created. For example:

  - If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself.
  - If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise.
  - Electronic transmission of image data can introduce noise.

  To remove the noise of digital images, we can use different techniques as:

  - **Linear Filtering**: You can use linear filtering to remove certain types of noise. Certain filters, such as averaging or Gaussian filters, are appropriate for this purpose. For example, an averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by grain are reduced.
  - **Median filtering**: is similar to using an averaging filter, in that each output pixel is set to an average of the pixel values in the neighborhood of the corresponding input pixel. However, with

median filtering, the value of an output pixel is determined by the *median* of the neighborhood pixels, rather than the mean. The median is much less sensitive than the mean to extreme values (called *outliers*). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image.

- **Adaptive Filtering:** This approach often produces better results than linear filtering. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image

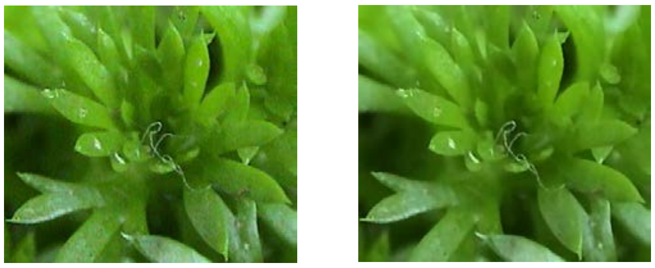


Figure 40: *Denoising a) Original image, b) Filtered image*

➢ **Edge detection**: is one of the most commonly used operations in image analysis, and there are probably more algorithms in the literature for enhancing and detecting edges than any other single subject. The reason for this is that edges form the outline of an object. An edge is the boundary between an object and the background, and indicates the boundary between overlapping objects. This means that if the edges in an image can be identified accurately, all of the objects can be located and basic properties such as area, perimeter, and shape can be measured. Since computer vision involves the identification and classification of objects in an image, edge detections is an essential tool.

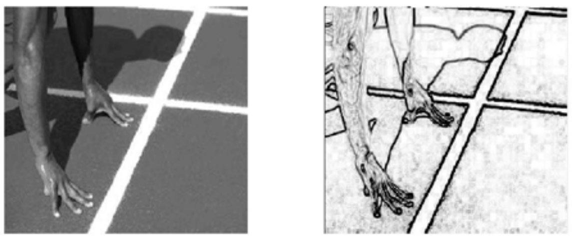


Figure 40: *Edge detection a) Original image, b) Filtered image*

In our case there are two changes that could improve the results significantly

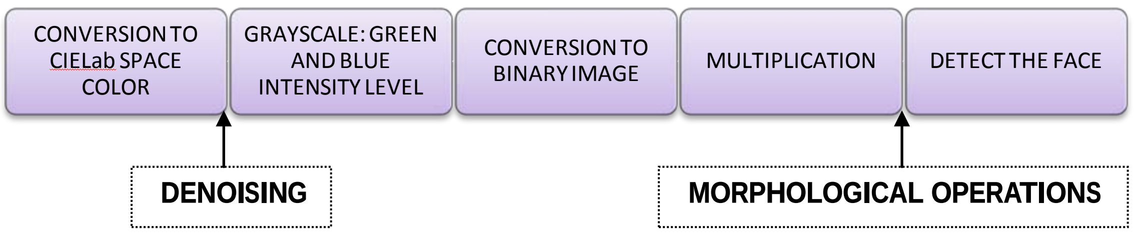

CONVERSION TO CIELab SPACE COLOR
GRAYSCALE: GREEN AND BLUE INTENSITY LEVEL
CONVERSION TO BINARY IMAGE
MULTIPLICATION
DETECT THE FACE
DENOISING
MORPHOLOGICAL OPERATIONS

Figure 41: *Improvements of the algorithm*

The first change is to introduce a denoising filter after the color space conversion. With a low pass filter it can be enough in this case.

The second improvement is the use of morphological operations. It is important to apply this process at the point indicated, when we have the binary image with pixels representing 1's skin, and 0's representing non-skin pixels. Then there is the need for such operations, as filler, erosion and dilation to separate areas of skin that are connected to areas of non-skin that survived the segmentation. The procedure is as follows: First fill the gaps in the binary image, for which use the "*imfill*" of Matlab. Then erosion is applied to reduce those areas, together with a dilation to make bigger skin areas that may have been lost due to erosion applied in the previous step, in Matlab we have "imerode" and "imdilate" which executed processes of erosion and dilation, respectively. Finally the binary image should be multiplied with dilated binary image of segmentation process output to maintain the gaps.

CHAPTER 5: CONCLUSIONS AND FUTURE WORKS

To finish the assignment, in this chapter it is briefly commented some conclusions of the project, and some proposals of future work.

5.1. CONCLUSIONS

This project has introduced a new face detector, based on color skin characteristics. The first results show a very high number of hits, these results are from images that are in an environment that no great difficulty. However when we increase slightly the degree of difficulty we observe that the detector begins to make mistakes in a high percentage.

The face detection could be used at the user level, depending on the features required, but never professionally, since, according to the state of development of such techniques, there are many methods currently developed with more satisfactory results.

Moreover, with regard to the development of the project, I have to emphasize that because of this work I got familiar with the context of image processing, and specifically with face detection, learning concepts, processing techniques and above all, the state in which this area actually developed, area that I find useful as well, very interesting.

5.2. FUTURE WORK

There are many interesting problems that remain in the area of face detection, but specifically for this project, the first propose we have to keep in mind is to improve the system in order to have better results in the conditions tested in the project.

In the other hand, the previously developed system has been only tested on frontal or almost frontal faces. Experiments have shown that it can detect faces rotated up to about 30º in depth without significant loss in performance. However, we don´t know the efficiency of the system with a greater angle of rotation, so it can be a good project the development this investigation line

Finally there is another possible future work to implement, because although we haven´t tested the system with several faces images, because of the design of the algorithm we can guess it won´t work, so it can be interesting trying to develop this area, and use it in such situations

APPENDIX

APPENDIX A: MATLAB CODES

A1. FACE DETECTION ALGORITHM PROPOSED

```
%Detection of Face on the basis of skin color

clear all;
close all;
clc

I=imread('face1.jpg');
imshow(I)

cform = makecform('srgb2lab');
J = applycform(I,cform);
figure;
imshow(J);

K=J(:,:,2);
figure;
imshow(K);

L=graythresh(J(:,:,2));
BW1=im2bw(J(:,:,2),L);
figure;
imshow(BW1);

M=graythresh(J(:,:,3));
figure;
imshow(J(:,:,3));

BW2=im2bw(J(:,:,3),M);
figure;imshow(BW2);
O=BW1.*BW2;

% Bounding box

P=bwlabel(O,8);
BB=regionprops(P,'Boundingbox');
BB1=struct2cell(BB);
BB2=cell2mat(BB1);

[s1 s2]=size(BB2);

mx=0;
for k=3:4:s2-1
    p=BB2(1,k)*BB2(1,k+1);
    if p>mx & (BB2(1,k)/BB2(1,k+1))<1.8
        mx=p;
        j=k;
    end
end

figure, imshow(I);
hold on;

rectangle('Position',[BB2(1,j-2),BB2(1,j-1),BB2(1,j),BB2(1,j+1)],
'EdgeColor','r')
```

A2. ‘readImages.m’

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                    %
%  Function ‘readImages.m’:  Read images from the specified          %
%  directory, process with ‘faceDetection.m’, and save the results %
%  in the appropriate directory                                      %
%                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function X = readImages

%Path of the directory that contains the databases

pathDatabase ='C:\Users\ELENA\Desktop\PROYECTO 1\Codigo
matlab\DataBase1\';

%Path of the directory that contains the results

pathResults ='C:\Users\ELENA\Desktop\PROYECTO 1\Codigo
matlab\ResultsDataBase1\';

%Loop that reads the images, processes them and saves the results

list=dir(pathDatabase);
list=list(3:end);

    for i=1:length(list);

        name=list(i).name;
        final=strcat(pathResults,name);

        I=imread([pathDatabase '/' list(i).name]);

        b=faceDetection (I);

        imwrite(b,final);

    end

end
```

A3. ‘faceDetection.m’(modified proposed code)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                    %
%  Function ‘faceDetection.m’: processes the images read by          %
%  ‘readImages.m                                                     %
%                                                                    %
%  Input: Original image                                             %
%  Output: Face detected image                                       %
%                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function b = faceDetection (I)

%Color space transformation (RGB to L*a*b*)

cform = makecform('srgb2lab');
J = applycform(I,cform);

%Grayscale image representing green and blue intensity level

 K=J(:,:,2); Q=J(:,:,3);

%Optimum threshold by Otsu´s Method

L=graythresh(K);
M=graythresh(Q);

%Binary images

BW1=im2bw(J(:,:,2),L);
BW2=im2bw(J(:,:,3),M);

%Multiplication of binary images

O=BW1.*BW2;

%Looking for all possible bounding box, and convert to matrix form

P=bwlabel(O,8);
BB=regionprops(P,'Boundingbox');
BB1=struct2cell(BB);
BB2=cell2mat(BB1);

%Loop tgo find face position

[s1 s2]=size(BB2);

mx=0;

for k=3:4:s2-1
    p=BB2(1,k)*BB2(1,k+1);
    if p>mx && (BB2(1,k)/BB2(1,k+1))<1.8
        mx=p;
        j=k;
    end
end
```

```
hold on;

%Draw rectangle
rectangle('Position',[BB2(1,j-2),BB2(1,j-1),
BB2(1,j),BB2(1,j+1)],'EdgeColor','r' )

%Cropped face detected
b = imcrop(I,[BB2(1,j-2),BB2(1,j-1), BB2(1,j),BB2(1,j+1)]);

end
```

APPENDIX B: COMPLEMENTARY DOCUMENTS

B1. CIELAB COLOR SPACE

A Lab color space is a color-opponent space with dimension L for lightness and a and b for the color-opponent dimensions, based on nonlinearly compressed CIE XYZ color space coordinates.

The coordinates of the Hunter 1948 L, a, b color space are L, a, and b.[1][2] However, Lab is now more often used as an informal abbreviation for the CIE 1976 (L*, a*, b*) color space (also called CIELAB, whose coordinates are actually L*, a*, and b*). Thus the initials Lab by themselves are somewhat ambiguous. The color spaces are related in purpose, but differ in implementation.

Both spaces are derived from the "master" space CIE 1931 XYZ color space, which can predict which spectral power distributions will be perceived as the same color, but which is not particularly perceptually uniform. Strongly influenced by the Munsell color system, the intention of both "Lab" color spaces is to create a space which can be computed via simple formulas from the XYZ space, but is more perceptually uniform than XYZ Perceptually uniform means that a change of the same amount in a color value should produce a change of about the same visual importance. When storing colors in limited precision values, this can improve the reproduction of tones. Both Lab spaces are relative to the white point of the XYZ data they were converted from. Lab values do not define absolute colors unless the white point is also specified. Often, in practice, the white point is assumed to follow a standard and is not explicitly stated (e.g., for "absolute colorimetric" rendering intent ICC L*a*b* values are relative to CIE standard illuminant D50, while they are relative to the unprinted substrate for other rendering intents).

The lightness correlate in CIELAB is calculated using the cube root of the relative luminance.

ADVANTAGES OF LAB

Unlike the RGB and CMYK color models, Lab color is designed to approximate human vision. It aspires to perceptual uniformity, and its L component closely matches human perception of lightness. It can thus be used to make accurate color balance corrections by modifying output curves in the a and b components, or to adjust the lightness contrast using the L component. In RGB or CMYK spaces, which model the output of physical devices rather than human visual perception, these transformations can only be done with the help of appropriate blend modes in the editing application.

Because Lab space is much larger than the gamut of computer displays, printers, or even human vision, a bitmap image represented as Lab requires more data per pixel to obtain the same precision as an RGB or CMYK bitmap. In the 1990s, when computer hardware and software was mostly limited to storing and manipulating 8 bit/channel bitmaps, converting an RGB image to Lab and back was a lossy operation. With 16 bit/channel support now common, this is no longer such a problem.

Additionally, many of the "colors" within Lab space fall outside the gamut of human vision, and are therefore purely imaginary; these "colors" cannot be reproduced in the physical world. Though color management software, such as that built in to image editing applications, will pick the closest in-gamut approximation, changing lightness, colorfulness, and sometimes hue in the process, author Dan Margulis claims that this access to imaginary colors is useful, going between several steps in the manipulation of a

CIE 1976 (L*, A*, B*) COLOR SPACE (CIELAB)

CIE L*a*b* (CIELAB) is the most complete color space specified by the International Commission on Illumination (French Commission internationale de l'éclairage, hence its CIE initialism). It describes all the colors visible to the human eye and was created to serve as a device independent model to be used as a reference.

The three coordinates of CIELAB represent the lightness of the color (L* = 0 yields black and L* = 100 indicates diffuse white; specular white may be higher), its position between red/magenta and green (a*, negative values indicate green while positive values indicate magenta) and its position between yellow and blue (b*, negative values indicate blue and positive values indicate yellow). The asterisk (*) after L, a and b are part of the full name, since they represent L*, a* and b*, to distinguish them from Hunter's L, a, and b, described below.

Since the L*a*b* model is a three-dimensional model, it can only be represented properly in a three-dimensional space. Two-dimensional depictions are chromaticity diagrams: sections of the color solid with a fixed lightness. It is crucial to realize that the visual representations of the full gamut of colors in this model are never accurate; they are there just to help in understanding the concept.

Because the red/green and yellow/blue opponent channels are computed as differences of lightness transformations of (putative) cone responses, CIELAB is a chromatic value color space.

A related color space, the CIE 1976 (L*, u*, v*) color space (a.k.a. CIELUV), preserves the same L* as L*a*b* but has a different representation of the chromaticity components. CIELUV can also be expressed in cylindrical form (CIELCH), with the chromaticity components replaced by correlates of chroma and hue.

Since CIELAB and CIELUV, the CIE has been incorporating an increasing number of color appearance phenomena into their models, to better model color vision. These color appearance models, of which CIELAB, although not designed as can be seen as a simple example, culminated with CIECAM02.

B1. OTSU´S METHOD

In computer vision and image processing, Otsu's method is used to automatically perform histogram shape-based image thresholding, or, the reduction of a graylevel image to a binary image. The algorithm assumes that the image to be thresholded contains two classes of pixels (e.g. foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal. The extension of the original method to multi-level thresholding is referred to as the Multi Otsu method. Otsu's method is named after Nobuyuki Otsu .

METHOD

In Otsu's method we exhaustively search for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes:

$$\sigma^2_w(t) = \omega_1(t)\sigma^2_1(t) + \omega_2(t)\sigma^2_2(t)$$

Weights ω_i are the probabilities of the two classes separated by a threshold *t* and σ^2_i variances of these classes. Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance:

$$\sigma^2_b(t) = \sigma^2 - \sigma^2_w(t) = \omega_1(t)\omega_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2$$

which is expressed in terms of class probabilities ω_i and class means μ_i which in turn can be updated iteratively. This idea yields an effective algorithm.

ALGORITHM

1. Compute histogram and probabilities of each intensity level
2. Set up initial $\omega_i(0)$ and $\mu_i(0)$
3. Step through all possible thresholds $t = 1 \ldots$ maximum intensity
    1. Update ω_i and μ_i
    2. Compute $\sigma^2_b(t)$
4. Desired threshold corresponds to the maximum $\sigma^2_b(t)$

BIBLIOGRAPHY

- *R. C. Gonzalez and R. E. Woods. "Digital Image Processing". 2nd ed. Prentice Hall, 2002..*

- *Digital Image Processing – Part Two. Huiyu Zhou, Jiahua Wu, Jianguo Zhang & Ventus.*

- *Cai and A. Goshtasby. Detecting in human faces in color images. Image and Vision Computing, 1999.*

- *K.-K. Sung: Learning and Example Selection for Object and Pattern Recognition. PhD thesis, MIT, Artificial. Intelligence Laboratory and Center of Biological and Computational Learning, Cambridge, 1995.*

- *E. Hjelmas and B. K. Low. Face detection: A survey. Computer Vision and Image Understanding, 2001.*

- *E. Argyle. "Techniques for edge detection," Proc. IEEE, vol. 59, pp. 285-286, 1971*

- *M.H. Yang and N. Ahuja. Detecting human faces in color images. Proceedings of the IEEE International Conference on Image Processing,*

- *B. D. Zarit. Skin detection in video images. Technical report VISLAB-99-01, University of Illinois at Chicago, 1999.*

- *E. R. Davies. "Constraints on the design of template masks for edge detection". Partern Recognition Lett., vol. 4.*

- *C. Garcia and G. Tziritas, "Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis", IEEE transactions on multimedia, vol. 1, Sept 1999*

- *Jie Yang and Alex Waibel, "A Real-Time Face Tracker", CMU CS Technical Report.*

- *Zarit, B. D.; Super, B. J. and Quek, F. K. H. "Comparison of five color models in skin pixel classification". Recognition, Analysis and Tracking of faces and gestures in Real-Time systems, 1999.*

- *Brand, J.; Mason, J. S. "A comparative assessment of three approaches to pixel-level human skin-detection". 15th International Conference on Pattern Recognition; 2000*

- *Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers, 1997.*

- *H. A. Rowley, **S.** Baluja, and T. Kanade, "Neural network-based face detection," IEEE Transactions on PAMI, vol. 20, no. 1, January 1998.*

- *H. Wu, Q. Chen, and M. Yachida, "Face detection from color images using a fuzzy pattern matching method," IEEE Transactions on PAMI, vol. 21, no. 6, June 1999.*

- *O. H. Jensen, "Implementing the viola-jones face detection algorithm,"2008.*

- *I.Craw, d.Tock, and a.Bennett, "Finding face features",Proc.second European conf. Computer vision,1992.*

- *Sanjay singh et.al, "A robust skin color based face detection algorithm", Tamkang Journal of Science and Engineering vol.6, 2003*

- *M. Elad et al., Rejection based classifier for face detection, Pattern Recognition Letters, 23, 2002.*

- *E. Gose, R. Johnsonbaugh, S. Jost, "Pattern Recognition and Image Analysis", Prentice Hall, Inc., Upper Saddle River, NJ, 1996*

- *F. Samaria and A. Harter, "Parameterisation of a stochastic model for human faceidentification," 2nd IEEE Workshop on Applications of Computer Vision, Sarasota (Florida),December 1994*

❖ *WEBS*

– *http://www.facedetection.com/facedetection/datasets.htm*

– *http://www.mathworks.com/*

– *http://www.syseng.anu.edu.au/~luke/cvcourse.htm*

– *www.wikipedia.org*

– *"Deteccion de caras y analisis de expresiones faciales". http://alojamientos.us.es/gtocoma/pid/pid10/deteccioncaras .htm#_Toc32300901*

– *Computer Vision Homepage ttp://www-2.cs.cmu.edu/~cil/vision.html*

– *Face Recognition Homepage http://www.face-rec.org/*

– *Resources for Face Detection http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html*

– *CIELab Color Space http://www.fho-emden.de/~hoffmann/cielab03022003.pdf*