King Saud University
College of Computer and Information Sciences
Computer Sciences Department

Automatic Face Detection for Count Estimates

Submitted By:
Abeer S. Al-Humaimeedy

Supervisor
Prof. Mona F. M. Mursi

Project Submitted in Partial Fulfillment for the
Degree of Master in Computer Sciences

Thoul Ki'dah 1424 H. (January 2004)
First Semester
Riyadh, Saudi Arabia

ALL GRATITUDE IS DUE TO ALLAH

It is my pleasure to express my appreciation to my supervisor,
Prof. Mona Mursi
for her invaluable guidance and help, and most of all for her encouragement and patience during the preparation of this project.

I would also like to thank
My family for their support

Abstract

This project presents an application of computers in the area of computer vision, whereby, an image of people in a confined area is analyzed and the people in the image are detected by locating their faces, and a total count for the number of faces in the images tallied. Since color is a powerful fundamental cue that can be used as a first step in the face detection process, color segmentation of an input image in perceptually plausible hue-saturation color space where the effects of the variability of chrominance on changes in illumination are reduced. This is followed by grouping likely face regions into clusters of connected pixels. Median filtering is then performed to eliminate the small clusters. Finally, the resulting blobs are each matched with a face pattern (an ellipse) and subjected to certain criteria to reject non-face blobs. The system was implemented and tested and the results demonstrate the efficiency of the technique.

Contents

List of Figures

Chapter 1

Introduction

The current evolution of computer technologies has envisaged an advanced machinery world, where human life is enhanced by artificial intelligence. Indeed, this trend has already prompted an active development in machine intelligence. Computer vision, for example, aims to duplicate human vision. Traditionally, computer vision systems have been used in specific tasks such as performing tedious and repetitive visual tasks of assembly line inspection. Current development in this area is moving toward more generalized vision applications such as face recognition, automatic people-counting, video surveillance, and image database management.

In the last years, the effective development of automatic people-counting systems aroused a considerable interest of research in electronics and information science. Such interest is evident by the considerable amount of related R&D activities recently carried out at the European level (e.g., PEDMON, DIMUS, and CREW projects, etc.), and by the availability of patented products put on the market. Automatic people counting is a problem already addressed in the literature for various actual applications ranging from railway transport security or pedestrian traffic management, monitoring the attendance of students in a classroom of a school or university, detecting the number of persons in a room of a building etc. The basic principle of operation of all automatic people-counting systems lies in the processing of images coming from some camera installed within viewing distance from the people. The main differences in such systems are: the kind of camera employed, and the kind of image processing algorithms to be implemented in order to derive the number of people. These algorithms vary according to the estimated number of people to count. Most automatic people-counting systems that count small range of people in a classroom for example is based on detecting the faces present in the image.

Detecting the faces in a digital image has gained a great deal of interest in the last decade, with the applications in various fields such as surveillance systems, law enforcement and security systems. Although facial detection is an extremely simple task for the human eye, automating the process to a computer requires the use of various image processing techniques in the field of computer vision.

Many different algorithms have been implemented for facial detection, including the use of color information, template matching, neural network, edge detection and Hough transforms [26,7,25,20,42].

The approach in this project follows one similar to [42,1,43]; which use the skin color as a first cue to remove most of the non-facial information then use some image processing filters and morphological operations to prepare the remaining information for further processing. The final step was to use one of the template matching techniques as a final decision for considering the region as face or not.

The project aims to develop software for automatic face counting in an image taken from a camera installed in rooms using the algorithm explained briefly before, to detect faces as the first and main step followed by counting detected faces. Experimental results produced by the project demonstrate successful detection over 14 set of images of accuracy ranging from 70 % to 100% and with 1.7 sec processing time on the average.

The rest of this report is organized as follows:

Chapter 1: (Introduction)

This chapter introduces the subject and briefly mentions the algorithm used and the results produced by the software.

Chapter 2: (related work)

This Chapter discusses previous works done in this area with some explanation of the general algorithms that have been used in face detection.

Chapter 3: (system overview)

Chapter 5 presents an abstract description of the system and the high-level architecture of the software.

Chapter 4: (system design)

Describes the steps that I followed in solving the problem. This chapter also discusses the difficulties that were faced while developing the software, and the solutions that have been undertaken to solve these problems. It also describes the implementation of the software.

Chapter 5: (interface design)

This chapter presents the user interface and the graphical user interface (GUI) design issues and principles.

Chapter 6: (experimental results)

This chapter discusses the experimental results that are produced by the software.

Chapter 7: (conclusion and remarks)

This chapter discusses future work that can be carried out to expand the system and gives a brief summary of the work done and conclusion.

Appendix A:

Explains the ToolKit that was used in the development of the graphical user interface (GUI) (wxWindows).

Appendix B:

Presents the user guide of the developed software.

Chapter 2

Related Work

The earliest automatic people-counting software assumes the availability of vertical frontal faces of similar sizes. In reality, this assumption may not hold due to the varied nature of face appearance and environment conditions. The exclusion of the background in these images is necessary for reliable face classification techniques. However, in realistic application scenarios, a face could occur in a complex background and in many different positions. Counting systems that are based on standard face images are likely to mistake some areas of the background as a face. In order to rectify the problem, a visual front-end processor is needed to localize and extract the face region from the background.

Face detection is one of the visual tasks which humans can do effortlessly. However, in computer vision terms, this task is not easy. A general statement of the problem can be defined as follows: *Given a still or video image, detect and localize an unknown number (if any) of faces. The solution to the problem involves segmentation, extraction, and verification of faces and possibly facial features from an uncontrolled background. As a visual frontend processor, a face detection system should also be able to achieve the task regardless of illumination, orientation, and camera distance.*

Early efforts in face detection have dated back as early as the beginning of the 1970s, where simple heuristic and anthropometric techniques were used. These techniques are largely rigid due to various assumptions such as plain background, frontal face atypical passport photograph scenario. To these systems, any change of image conditions would mean fine-tuning, if not a complete redesign. Despite these problems the growth of research interest remained stagnant until the 1990s [10], when practical face recognition and video surveillance systems started to become a reality. Over the past decade there has been a great deal of research interest spanning several important aspects of face detection. More robust segmentation schemes have been presented, particularly those using motion, color, and generalized information. The use of statistics and neural networks has also enabled faces to be detected from cluttered scenes at different distances from the camera[7,14,25].

Because face detection techniques requires *a priori* information of the face, they can be effectively organized into two broad categories feature-based approach and image-based approach distinguished by their different approach to utilizing face knowledge.

2.1. Feature-based approach:

This category make explicit use of face knowledge and follow the classical detection methodology in which low level features are derived prior to knowledge-based analysis [6]. The apparent properties of the face such as skin color and face geometry are exploited at different system levels. Typically, in these techniques face detection tasks are accomplished by manipulating distance, angles, and area measurements of the visual features derived from the scene. Since features are the main ingredients, these techniques are termed the feature-based approach. These approaches have embodied the majority of interest in face detection research starting as early as the 1970s.

The development of the feature-based approach can be further divided into two areas: low-level analysis and feature analysis.

2.1.1. Low-Level analysis:

2.1.1.1. Edges: As the most primitive feature in computer vision applications, edge representation was applied in the earliest face detection work by Sakai *et al.* [28]. The work was based on analyzing line drawings of the faces from photographs, aiming to locate facial features. Craw *et al.* [11] later proposed a hierarchical framework based on Sakai *et al.*'s work to trace a human head outline. The work includes a line-follower implemented with curvature constraint to prevent it from being distracted by noisy edges. Edge features within the head outline are then subjected to feature analysis using shape and position information of the face.

Edge detection is the foremost step in deriving edge representation. So far, many different types of edge operators have been applied. The Sobel operator was the most common filter.

In an edge-detection-based approach to face detection, edges need to be labeled and matched to a face model in order to verify correct detections. Govindaraju [15] accomplishes this by labeling edges as the left side, hairline, or right side of a front view face and matches these edges against a face model by using the golden ratio3 for an ideal face:

$$\frac{\text{height}}{\text{width}} \equiv \frac{1+\sqrt{5}}{2}. \qquad (2.1)$$

Govindaraju's edge-based feature extraction works in the following steps:
Edge detection: an edge operator.
Thinning: a classical thinning algorithm.
Spur removal: each connected component is reduced to its central branch.
Filtering: the components with non-face-like properties are removed.
Corner detection: the components are split based on corner detection.
Labeling: the final components are labeled as belonging to the left side, hairline, or right side of a face.

The labeled components are combined to form possible face candidate locations based on a cost function (which uses the golden ratio defined above). On a test set of 60 images with complex backgrounds containing 90 faces, the system correctly detected 76% of the faces with an average of two false alarms per image.

2.1.1.2. Gray information: Besides edge details, the gray information within a face can also be used as features. Facial features such as eyebrows, pupils, and lips appear generally darker than their surrounding facial regions. This property can be exploited to differentiate various facial parts. Several recent facial feature extraction algorithms [5,16] search for local gray minima within segmented facial regions. In these algorithms, the input images are first enhanced by contrast-stretching and gray-scale morphological routines to improve the quality of local dark patches and thereby make detection easier.

The extraction of dark patches is achieved by low-level gray-scale thresholding. On the application side, Wong *et al.* [35]

Yang and Huang [36], on the other hand, explore the gray-scale behaviour of faces in mosaic (pyramid) images. When the resolution of a face image is reduced gradually either by subsampling or averaging, macroscopic features of the face will disappear. At low resolution, face region will become uniform. Based on this observation, Yang proposed a hierarchical face detection framework. Starting at low resolution images, face candidates are established by a set of rules that search for uniform regions. The face candidates are then verified by the existence of prominent facial features using local minima at higher resolutions.

2.1.1.3. Color. Whilst gray information provides the basic representation for image features, color is a more powerful means of discerning object appearance. Due to the extra dimensions that color has, two shapes of similar gray information might appear very differently in a color space. It was found that different human skin color gives rise to a tight cluster in color spaces even when faces of difference races are considered [37]. This means color composition of human skin differs little across individuals.

One of the most widely used color models is RGB representation in which different colors are defined by combinations of red, green, and blue primary color components. Since the main variation in skin appearance is largely due to luminance change (brightness) [37], normalized RGB colors are generally preferred [16,35,37], so that the effect of luminance can be filtered out. The normalized colors can be derived from the original RGB components as follows:

$$r = \frac{R}{R+G+B}$$
$$g = \frac{G}{R+G+B} \qquad (2.2)$$
$$b = \frac{B}{R+G+B}.$$

From equations in (2.2), it can been seen that $r + g + b = 1$. The normalized colors can be effectively represented using only r and g values as b can be obtained by noting $b = 1 - r - g$. In skin color analysis, a color histogram based on r and g shows that face color occupies a small cluster in the histogram. By comparing color information of a pixel with respect to the r and g values of the face cluster, the likelihood of the pixel belonging to a flesh region of the face can be deduced.

Besides RGB color models, there are several other alternative models currently being used in the face detection research. In [1] HSI color representation has been shown to have advantages over other models in giving large variance among facial feature color clusters. Hence this model is used to extract facial features such as lips, eyes, and eyebrows. Since the representation strongly relates to human perception of color , it is also widely used in face segmentation schemes [39,40].

The YIQ color model has been applied to face detection in [12]. By converting RGB colors into YIQ representation, it was found that the I-component, which includes color's ranging from orange to cyan, manages to enhance the skin region of Asians [12]. The conversion also effectively suppresses the background of other colors and allows the detection of small faces in a natural environment. Other color models applied to face detection include HSV, YCrCb , CIE-xyz , L*a*b* , L*u*v* [1,2,18].

Color segmentation can basically be performed using appropriate skin color thresholds where skin color is modeled through histograms or charts [18,40]. More complex methods make use of statistical measures that model face variation within a wide user spectrum [37,39]. For instance, Oliver *et al.* [24] and Yang and Waibel [37] employ a Gaussian distribution to represent a skin color cluster of thousands of skin color samples taken from difference races. The Gaussian distribution is characterized by its mean (μ) and covariance matrix ($\sum$).

2.1.1.4. Motion. If the use of a video sequence is available, motion information is a convenient means of locating moving objects. A straightforward way to achieve motion segmentation is by frame difference analysis. This approach, whilst simple, is able to discern a moving foreground efficiently regardless of the background content. In [5,16], moving silhouettes that include face and body parts are extracted by thresholding accumulated frame difference.

2.1.2. Feature Analysis:

Features generated from low-level analysis are likely to be ambiguous. For instance, in locating facial regions using a skin color model, background objects of similar color can also be detected. This is a classical many to one mapping problem which can be solved by higher level feature analysis. In many face detection techniques, the knowledge of face geometry has been employed to characterize and subsequently verify various features from their ambiguous state. There are two approaches in the application of face geometry among this literature review. The first approach involves sequential *feature searching* strategies based on the relative positioning of individual facial features. The confidence of a feature existence is enhanced by the detection of

nearby features. The techniques in the second approach group features as flexible *constellations* using various face models.

2.1.2.1. Feature searching. Feature searching techniques begin with the determination of prominent facial features. The detection of the prominent features then allows for the existence of other less prominent features to be hypothesized using anthropometric measurements of face geometry. For instance, a small area on top of a larger area in a head and shoulder sequence implies a "face on top of shoulder" scenario, and a pair of dark regions found in the face area increase the confidence of a face existence. Among the literature survey, a pair of eyes is the most commonly applied reference feature [5,35] due to its distinct side-by-side appearance.

The facial feature extraction algorithm by De Silva *et al.* [13] is a good example of feature searching. The algorithm starts by hypothesizing the top of a head and then a searching algorithm scans downward to find an eye-plane which appears to have a sudden increase in edge densities (measured by the ratio of black to white along the horizontal planes).

The length between the top and the eye-plane is then used as a reference length. Using this reference length, a flexible facial template covering features such as the eyes and the mouth is initialized on the input image. The initial shape of the template is obtained by using anthropometric length with respect to the reference length ,obtained from the modeling of 42 frontal faces in a database. The flexible template is then adjusted to the final feature positions according to a fine-tuning algorithm that employs an edge-based cost function. The algorithm is reported to have an 82% accuracy (out of 30 images in the same database) in detecting all facial features from quasi-frontal (<=30±) head and shoulder faces on a plain background. Although the algorithm manages to detect features of various races since it does not rely on

gray and color information, it fails to detect features correctly if the face image contains eyeglasses and hair covering the forehead.

Jeng *et al.* [19] propose a system for face and facial feature detection which is also based on anthropometric measures. In his system, he initially try to establish possible locations of the eyes in binarized pre-processed images. For each possible eye pair the algorithm goes on to search for a nose, a mouth, and eyebrows. Each facial feature has an associated evaluation function, which is used to determine the final most likely face candidate. They report a 86% detection rate on a dataset of 114 test images taken under controlled imaging conditions, but with subjects positioned in various directions with a cluttered background.

2.1.2.2. Constellation analysis. Some of the algorithms mentioned in the last section rely extensively on heuristic information taken from various face images modeled under fixed conditions. If given a more general task such as locating the face(s) of various poses in complex backgrounds, many such algorithms will fail because of their rigid nature. Later efforts in face detection research address this problem by grouping facial features in face-like constellations using more robust modeling methods such as statistical analysis.

Various types of face constellations have been proposed [8,32,38]. Burl *et al.* [8] make use of statistical shape theory on the features detected from a multi-scale Gaussian derivative filter. A probabilistic model for the spatial arrangement of facial features enables higher detection flexibility. The algorithm is able to handle missing features and problems due to translation, rotation, and scale to a certain extent. A successful rate of 84% out of 150 images taken from a lab-scene sequence, is obtained. Most detection failures are caused by significant rotation of the subject's head.

Probabilistic face models based on multiple face appearance have also been proposed in [32,38]. In Yow and Cipolla's model [38], faces are classified into several partial facial appearance groups that are common under different viewpoints. These partial facial groups are classified further into feature components. After facial features are obtained from a low-level edge based processing, active grouping then allows various facial groups to be derived hierarchically from the bottom end of the partial face classification. The grouping effectively reduced erroneous features arising from cluttered scenes. A Bayesian network then enables a gross combination of detection confidence of all the partial groups and thereby ensures the hypothesis of true faces to be reinforced with a high confidence level. A 92% detection rate was reported by Yow and Cipolla in 100 lab scene images. The algorithm is able to cope with small variations in scale, orientation, and viewpoint. The presence of eyeglasses and missing features are also handled by the algorithm.

In the system of Maio and Maltoni [22], the input images are converted to a directional image using a gradient-type operator over local windows (7 X 7 pixels). From this directional image they apply a two stage face detection method consisting of a generalized Hough transform and a set of 12 binary templates representing face constellations. The generalized Hough transform is used to generate face candidates by searching for ellipses. The face candidates are then fed to the template matching stage which accepts or rejects the face candidate as shown in Fig. 5. By efficient implementations and design

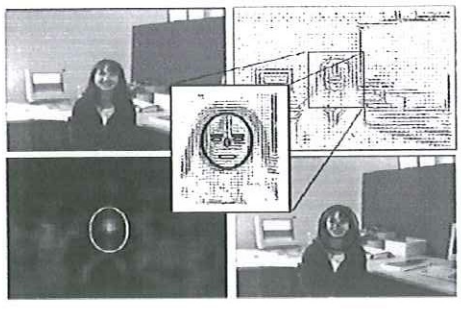


FIG. 2.1. The face detection system of Maio and Maltoni.

considerations, the system functions in real-time. Maio and Maltoni report correct detection in 69 out of 70 test images with no false alarms, where the test images consist of single faces of varying sizes with complex backgrounds.

2.2. Image-based approach:

It has been shown in the previous section that face detection by explicit modeling of facial features has been troubled by the unpredictability of face appearance and environmental conditions. Although some of the recent feature-based attempts have improved the ability to cope with the unpredictability, most are still limited to head and shoulder and quasi-frontal faces (or are included as one of the techniques in a combined system). There is still a need for techniques that can perform in more hostile scenarios such as detecting multiple faces with clutter-intensive backgrounds. This requirement has inspired a new research area in which face detection is treated as a pattern recognition problem. By formulating the problem as one of learning to recognize a face pattern from examples, the specific application of face knowledge is avoided. This eliminates the potential of modeling error due to incomplete or inaccurate face knowledge. The basic approach in recognizing face patterns is via a training procedure which classifies examples into face and non-face prototype classes. Comparison between these classes and a 2Dintensity array (hence the name image-based) extracted from an input image allows the decision of face existence to be made. The simplest image-based approaches rely on template matching [17], but these approaches do not perform as well as the more complex techniques presented in the following sections.

Most of the image-based approaches apply a window scanning technique for detecting faces. The window scanning algorithm is in essence

just an exhaustive search of the input image for possible face locations at all scales, but there are variations in the implementation of this algorithm for almost all the image-based systems. Typically, the size of the scanning window, the subsampling rate, the step size, and the number of iterations vary depending on the method proposed and the need for a computationally efficient system.

In the following three sections we have roughly divided the image-based approaches into linear subspace methods, neural networks, and statistical approaches.

2.2.1. Linear Subspace Methods:

Images of human faces lie in a subspace of the overall image space. To represent this subspace, one can use neural approaches (as described in the next section), but there are also several methods more closely related to standard multivariate statistical analysis which can be applied. In this section we describe and present results from one of these techniques, principal component analysis (PCA).

In the late 1980s, Sirovich and Kirby [31] developed a technique using PCA to efficiently represent human faces. Given an ensemble of different face images, the technique first finds the principal components of the distribution of faces, expressed in terms of eigenvectors (of the covariance matrix of the distribution). Each individual face in the face set can then be approximated by a linear combination of the largest eigenvectors, more commonly referred to as eigenfaces, using appropriate weights.

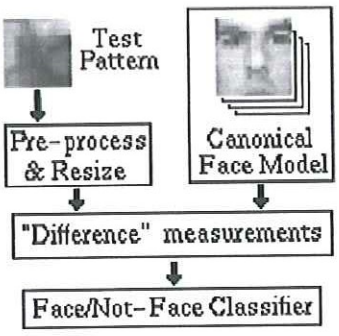


FIG. 2.2. Classification in Sung and Poggio's system

PCA is an intuitive and appropriate way of constructing a subspace for representing an object class in many cases. However, for modeling the manifold of face images, PCA is not necessarily optimal. Face space might be better represented by dividing it into subclasses, and several methods have been proposed for this, of which most are based on some mixture of multidimensional Gaussians. This technique was first applied for face detection by Sung and Poggio [33]. Their method consists mainly of four steps (Fig. 2.2):

1. The input sub-image is pre-processed by re-scaling to 19 X 19 pixels, applying a mask for eliminating near-boundary pixels, subtracting a best-fit brightness plane from the unmasked window pixels, and finally applying histogram equalization.

2. A distribution-based model of canonical face- and nonface-patterns is constructed. The model consists of 12 multi-dimensional Gaussian clusters with a centroid location and a covariance matrix, where six represent face- and six represent no-face pattern prototypes. The clusters are constructed by an elliptical *k*-means clustering algorithm which uses an adaptively changing normalized Mahalanobis distance metric.

3. A set of image measurements is computed for new images relative to the canonical face model. For each of the clusters, two values are computed.

One is a Mahalanobis-like distance between the new image and the prototype centroid, defined within the subspace spanned by the 75 largest eigenvectors of the prototype cluster, while the other is the Euclidean distance from the new image to its projection in the subspace.

4. A multi-layer perceptron (MLP) is trained for face–nonface classification from the 24-dimensional image measurement vector. The MLP is not fully connected, but exploits some prior knowledge of the domain. The training set consists of 47,316 image measurements vectors, where 4150 are examples of face patterns.

When a new image is to be classified, steps 1 and 3 are applied and the MLP provides the classification.

One issue which arises when training pattern recognition systems for face–nonface classification is how to collect a representable set of training samples for nonface images. The set of positive training samples is easily defined as all kinds of face images, but to define the complementary set is harder. Sung and Poggio suggested a training algorithm, known as "boot-strap training," to partially deal with this problem (a more precise strategy than the one proposed in [7]). The algorithm consists of the following steps:

1. create the initial set of nonface images simply by generating images of random pixels,

2. train the system,

3. run the system on scenes not containing faces and extract the false positives, and

4. pre-process the false positives and add them to the training set of nonfaces; go to step 2.

2.2.2. Neural Networks:

Neural networks have become a popular technique for pattern recognition problems, including face detection. Neural networks today are much more than just the simple MLP. Modular architectures, committee–ensemble classification, complex learning algorithms, autoassociative and compression networks, and networks evolved or pruned with genetic algorithms are all examples of the widespread use of neural networks in pattern recognition.

The first neural approaches to face detection were based on MLPs [7], where promising results where reported on fairly simple datasets. The first advanced neural approach which reported results on a large, difficult dataset was by Rowley *et al.* [26]. Their system incorporates face knowledge in a retinally connected neural network shown in (Fig. 2.3). The neural network is designed to look at windows of 20 X 20 pixels (thus 400 input units). There is one hidden layer with 26 units, where 4 units look at 10 X 10 pixel subregions, 16 look at 5 X 5 subregions, and 6 look at 20 X 5 pixels overlapping horizontal stripes.

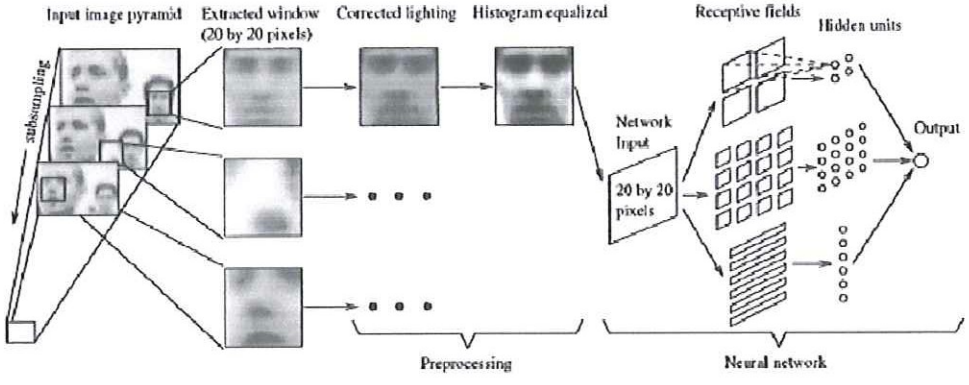


FIG. 2.3. The system of Rowley *et al.*

The input window is pre-processed through lighting correction (a best fit linear function is subtracted) and histogram equalization. This pre-processing method was adopted from Sung and Poggio's system mentioned earlier and is shown in Fig.2.4.

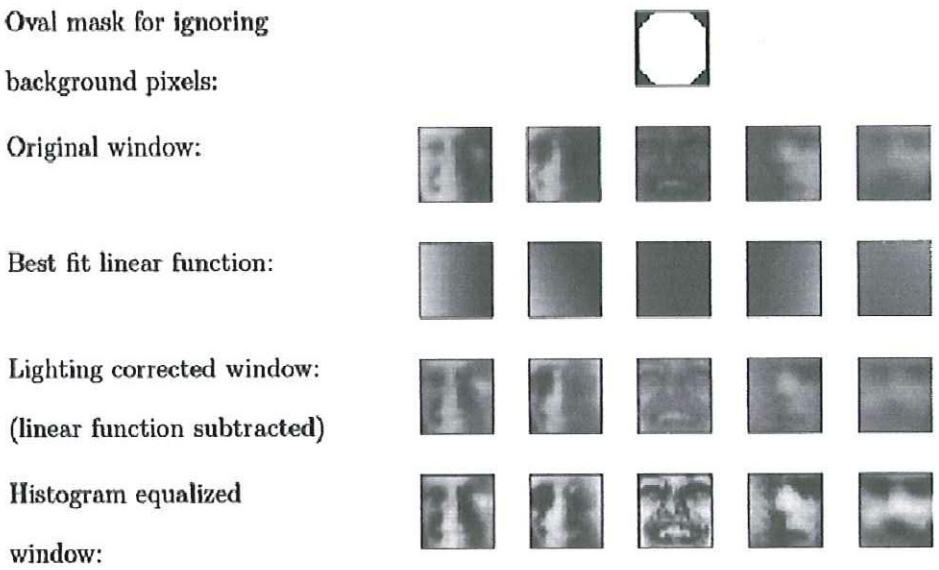


FIG. 2.4. The preprocessing method applied by Rowley *et al*

A problem that arises with window scanning techniques is overlapping detections. Rowley *et al.* deals with this problem through two heuristics:

1. Thresholding: the number of detections in a small neighborhood surrounding the current location is counted, and if it is above a certain threshold, a face is present at this location.

2. Overlap elimination: when a region is classified as a face according to thresholding, then overlapping detections are likely to be false positives and thus are rejected.

Recently, Rowley *et al.* [27] combined this system with a router neural network to detect faces at all angles in the image plane. They use a fully

connected MLP with one hidden layer and 36 output units (one unit for each 10± angle) to decide the angle of the face. The system detects 79.6% of the faces in two large datasets with a small number of false positives.

2.2.3. Statistical Approaches:

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

In Osuna *et al.* [25], a support vector machine (SVM) is applied to face detection. The proposed system follows the same framework as the one developed by Sung and Poggio [33], described in Section 2.2.1 (scanning input images with a 19 X 19 window). A SVM with a 2nd-degree polynomial as a kernel function is trained with a decomposition algorithm which guarantees global optimality. Kumar and Poggio [20] recently incorporated Osuna *et al.*'s SVM algorithm in a system for real-time tracking and analysis of faces. They apply the SVM algorithm on segmented skin regions in the input images to avoid exhaustive scanning. SVMs have also been used for multiview face detection by constructing separate SVMs for different parts of the view sphere [23]. In Terrillon *et al.* [34], SVMs improved the performance of the face detector compared to the earlier use of a multi-layer perceptron.

Schneiderman and Kanade [29,30] describe two face detectors based on Bayes' decision rule (presented as a likelihood ratio test in equation (2.3), where italics indicate a random variable).

$$\frac{P(image \mid \text{object})}{P(image \mid \text{non-object})} > \frac{P(\text{non-object})}{P(\text{object})} \tag{2.3}$$

If the likelihood ratio (left side) of equation (2.3) is greater than the right side, then it is decided that an object (a face) is present at the current location. The advantage with this approach is that if the representations for *P*(*image* | object) and *P*(*image* | non-object) are accurate, the Bayes decision rule is proven to be optimal [14]. In the first proposed face detection system of Schneiderman and Kanade [29], the posterior probability function is derived based on a set of modifications and simplifications (some of which are mentioned here):

- the resolution of a face image is normalized to 64 X 64 pixels
- the face images are decomposed into 16 X 16 subregions and there is no modelling of statistical dependency among the subregions
- the subregions are projected onto a 12-dimensional subspace (constructed by PCA)
- the entire face region is normalized to have zero mean and unit variance.

In the second proposed system [30], the visual attributes of the image are not represented by local eigenvector coefficients (as in the first approach), but instead by a locally sampled wavelet transform. A wavelet transform can capture information regarding visual attributes in space, frequency, and orientation and thus should be well suited for describing the characteristics of the human face. The wavelet transform applied in [30] is a three-level decomposition using a 5/3 linear phase filter-bank. This transform decomposes the image into 10 subbands. From these subbands 17 visual attributes (each

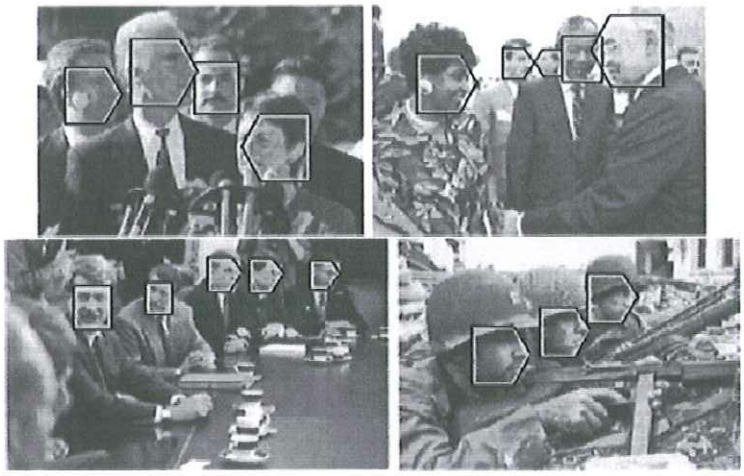


FIG. 2.5. Face detection examples from Schneiderman and Kanade

consisting of 8 coefficients) are extracted and treated as statistical independent random variables. The coefficients are requantized to three levels and the visual attributes are represented using histograms. With this approach, a view-based detector is developed with a frontal view detector and a right profile detector (to detect left profile images, the right profile detector is applied to mirror reversed images). The best results from the two systems described here are obtained with the eigenvector system, but this is due to the dataset consisting of mostly frontal view faces. In a separate experiment on a dataset consisting mostly of profile view faces, the wavelet detector outperformed the eigenvector detected (which of course had been modified to detect profile views also). Some examples of processed images with the wavelet system are shown in Fig.2.5.

Chapter 3

System Overview

The development of automatic people-counting software is not an easy task. The current software is varying according to the kind of camera employed, and the kind of image processing algorithms to be implemented in order to derive the number of people. Algorithms used in this software also, seem to be very different from each other and that depends on the software objectives, like the estimated number of people to count, accuracy level, camera location, and type of the image, etc. Most automatic people-counting systems that count small range of people in a classroom for example depends on detecting the faces inside the image.

3.1. System objectives:

The goal of this project is to count frontal faces existing in a color image of different types (bitmap, GIF, JPEG, PNG, PCX, PNM, TIFF, ,XPM, Windows icon, Windows cursor, and Windows animated cursor). The objective was to design and implement a people counter software that will detect frontal human faces in an image and then count them.

The problem of face detection has been studied extensively. A wide spectrum of techniques have been used as illustrated in the previous chapter. However, it is difficult to design algorithms that work for all illuminations, face colors, sizes and geometries, and image backgrounds. As a result, face detection remains as much an art as science.

3.2. System architecture:

The development of this software was dividing in two parts: (1) design the graphical user interface of the software and, (2) design the counter system itself.

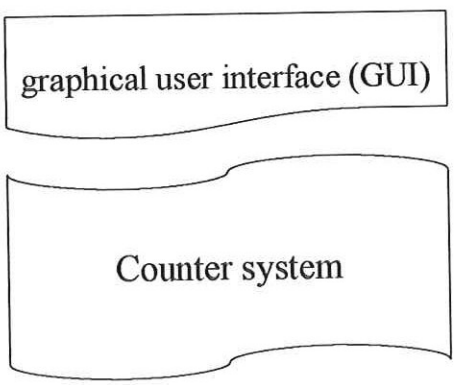


FIG 3.1: the general architecture of the software

3.2.1 Graphical user interface (GUI):

Graphical user interfaces revolutionized human computer interaction in the late 80's. More than a decade later, they still continue to be the dominant user interface mechanism. This can be attributed to their intutiveness and ease of use. Creating GUI's is different from programming simple command line applications. Creating a well designed GUI requires lot of care and attention to visual appeal, functionality and usability. Cultural contexts, localisation, online help, perceived responsiveness all play important parts in the end user experience. A systematic study of these topics is essentially for the modern software engineer.

User interfaces are the applications window to the world. User interfaces often decide the success of the applications in the real world. User interfaces are everywhere. Software products ranging from simple web-sites to complex molecular visualisation tools, all need to provide usable, intuitive user interfaces. Contrary to popular belief, good user interfaces are hard to build. Even so, user interface courses and books are hard to come by. Learning through real world experience is a long drawn and painful process. Therefore, as a software professional you will need to be conversant with principles of good user interface design.

So we tried our best to design an acceptable graphical user interface (GUI) using wxWindows toolkit.

3.2.2 Counter system:

The purpose of this project has been to try to replicate on a computer that which human beings are able to do effortlessly every moment of their lives, count people by detect the presence or absence of faces in their field of

vision. While it is something that to a layman appears trivial, to implement the necessary steps leading to the successful execution of this in an algorithm is difficult and still an unsolved problem in computer vision.

In deriving an algorithm of the system, we initially began by reviewing various articles and books on the topic. And then endeavoured to combined the useful ideas and methods mentioned in these articles. Some steps have been added to the algorithm which we think is important for the flow of the program. The project was implemented using visual C++ environment.

Chapter 4

System Design

Aiming to design a face counting system with the most optimal solution to ensure that the system will count people with a high accuracy rate with minimum false alarms, seemed to be a far-reached goal and its realisation a dream come true. After a long time of research, a system architecture was derived which combines some ideas from both feature-based approach and the image-based approach.

4.1. System flowchart:

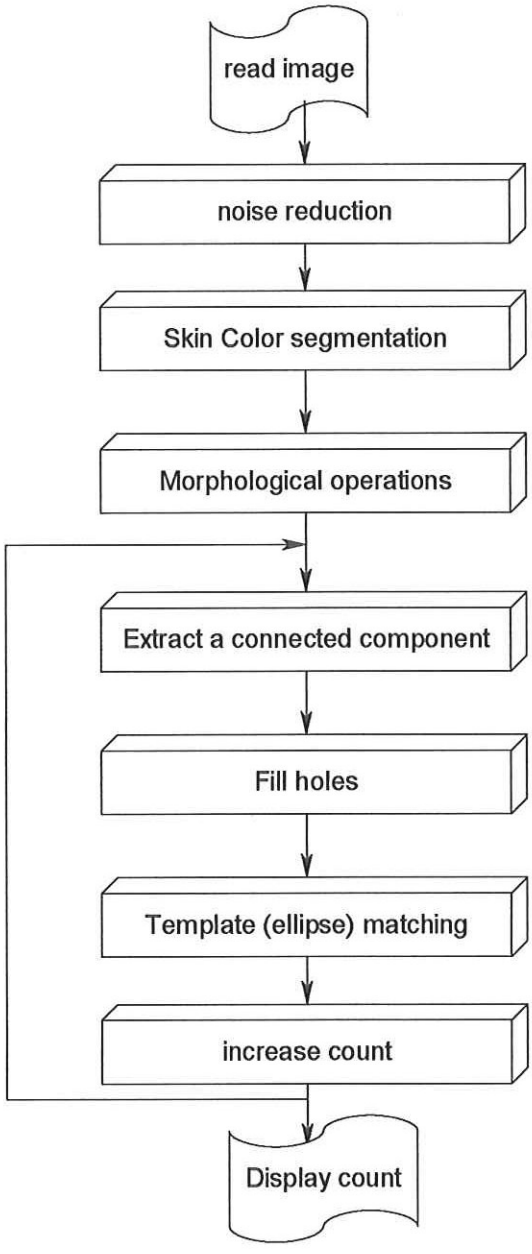


FIG 4.1: System flowchart

4.1.1 Image manipulation:

Any project involving images involving some image manipulating tasks to prepare it for low level pixels operations.

wxWindows toolkit which is known to have good support for a number of functions to deal with images and extract the pixel information and store it in a suitable data structure.

Functions of wxWindows need to be used in a certain program hierarchy with some environmental settings which affect the rest of the processing to satisfy these requirements. Once the pixels information is stored in the suitable data structure, wxWindows has no more role to play.

4.1.2 Noise reduction:

This step is quite important even if it was not mentioned in most of the different algorithms proposed for face detection. Its importance lies in reducing the number of non-facial blobs exists in the image.

In pursuing this goal, two algorithms were considered for noise removal, median filter done in the spatial domain of the image, and band-pass filter done in the frequency domain of the image.

The results from the two algorithms seem to have the same rate of noise reduction. (See FIG 4.2)

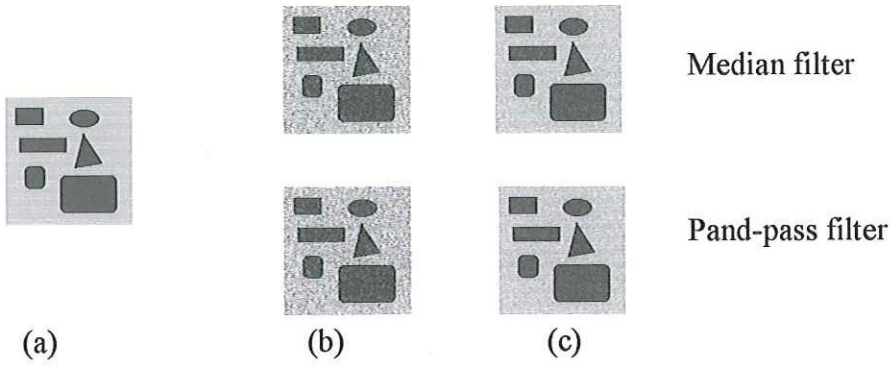


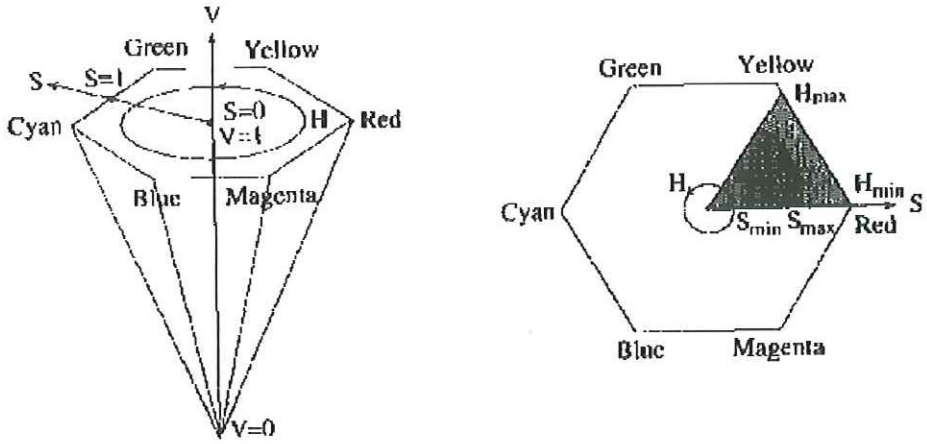
FIG 4.2: (a) ideal image (b) image with noise (c) result comes from applying the two algorithms

The second method, the band-pass filter method, takes more time to make the transformation to the frequency domain, and to do the filtering process and return the image to the original spatial domain for further detection processing which is done in the spatial domain. Hence the median filter approach was chosen for noise reduction to save time.

4.1.3 Skin Color Segmentation:

Assuming that a person framed in any random photograph is not an attendee at the Renaissance Fair or Mardi Gras, it can be assumed that the face is not white, green, red, or any unnatural color of that nature. While different ethnic groups have different levels of melanin and pigmentation, the range of colors that human facial skin takes on is clearly a subspace of the total color space. With the assumption of a typical photographic scenario, it would be clearly wise to take advantage of face-color correlations to limit our face search to areas of an input image that have at least the correct color components.

In pursuing this goal, two color spaces were considered, that have been reported to be useful in the literature, HSV and YCrCb spaces. YCrCb gave unacceptable results in comparison to HSV. As interesting color space that was considered is the hue-saturation-value (HSV) color space, because it is compatible with the human color perception. Alternativey, similar color spaces (e.g. HSI, HLS) can be used as well. The HSV color space has a hexcone shape as illustrated (in FIG 4.3 (a))



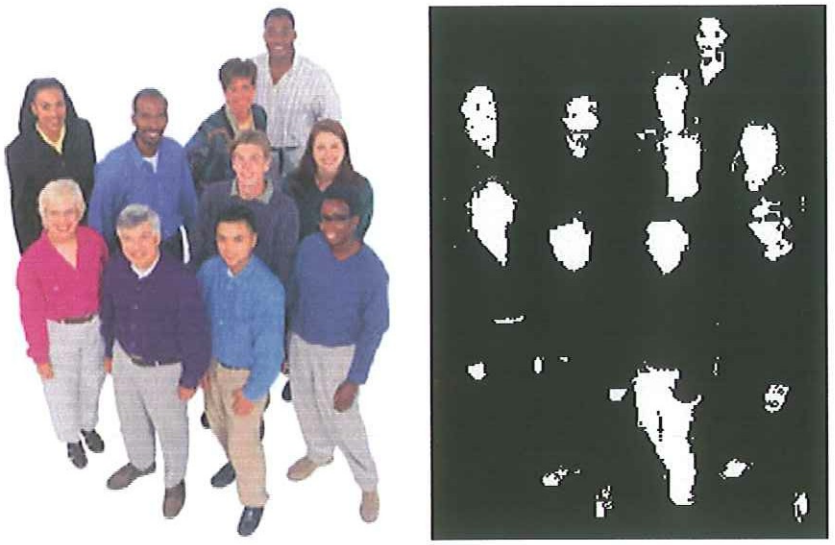
(a) (b)

FIG 4.3 (a) Hue- saturation-value color space (b) skin color segmentation in HS space

Hue (H) is represented as angle. The purity of colors is defined by the saturation (S), which varies from 0 to 1. The darkness of a color is specified by the value component (V), which varies also from 0 (root) to 1 (top level).

It is sufficient to consider hue and saturation as discriminating color information for the segmentation of skin-like regions. The hue and saturation domains, which describe the human skin color, can be defined or estimated a priori and used subsequently as reference for any skin color. After extensive research and experimentation the following parameters were chosen as follows: Smin =0.23, Smax =0.68, Hmin =0° and Hmax =50°.

listed in [42] where theses parameters were it was stated that these parameters when tested on their image database showed that they are appropriate to segment the white skin as well as the yellow skin of human beings. How far a segmentation of black skin is possible with these parameters, they have not tested yet.



(a) (b)

FIG 4.4 (a) original image and (b) is the skin color image result

4.1.4 Morphological processing:

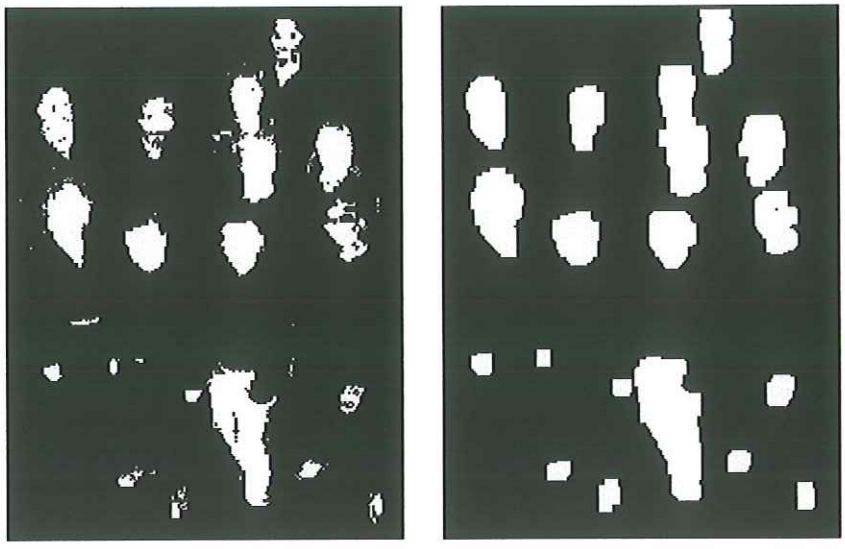
Based on the HSV thresholding, a resulting black and white "mask" is obtained with all the faces in addition to some artifacts (body parts, background). This mask is then refined through binary morphological operations to reduce the background contribution and remove holes within faces.

The image is first eroded with an 8-connected kernel of size equivalent to 1% of the image size, to eliminate regions occupied by 1% of the total size of the image or less. Also, some but not all overlapped skin regions which are connected by areas less or equivalent to the size of the kernel are separated.

One of the important effects of the erosion is eliminating the small undesired knobs of the region which may affect the template matching done later and cause it to give a false alarm.

This eroded image is then dilated with an 8-connected kernel of size equivalent to 1% of the image size -1 to fixed some of the undesired effects of the erosion like enlarging of the holes occupying more than 1% of the total size of the image and, refilling the holes which have a size equivalent to or less than the size of the kernel.

The last preparation done before starting the template matching is applying the median filter on the dilated image for removal of the small areas which appear as noise in the skin color image processing output, the source of which comes from the false consideration of some areas from the background as a face region due to the similarity of its color to the skin color space.



(a) (b)

FIG 4.5 (a) skin color image result and (b) is result of the morphological processing

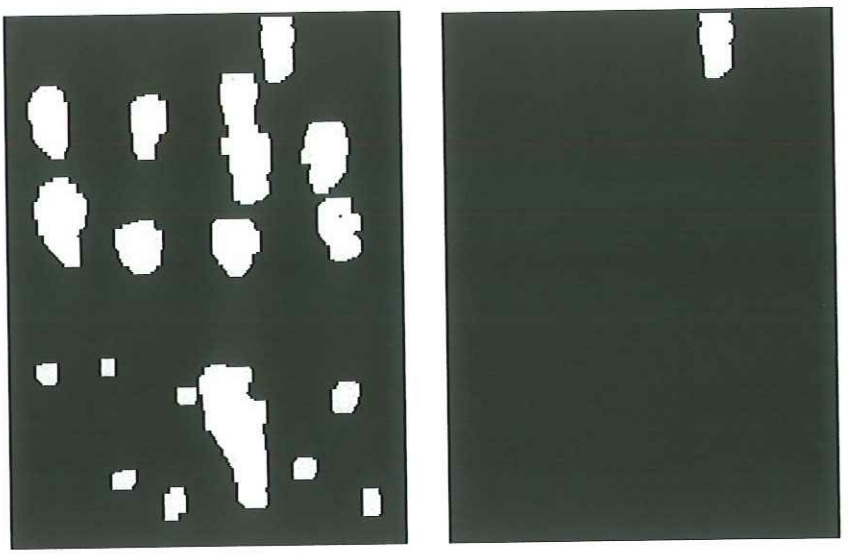
4.1.5. Face pattern Information:

The oval shape of a face can be approximated by an ellipse. Therefore, face detection in an image can be performed by detecting objects with elliptical shape. This can be done based on edges or, as we will be shown here, based on regions. The advantage of considering regions is that they are more robust against noise and changes in illumination.

As a first step, the connected components are located and extracted. Following that, each connected component is checked whether its shape is nearly elliptical or not.

The connected component is located by applying a region-growing algorithm described in [42,43] which depends on finding the 8-connected neighborhoods to a starting pixel which is considered as the first white pixel in

the image. This process is iterated until there is no pixels 8-connected to any pixel in the resulting region (see FIG 4.6).



(a) (b)

FIG 4.6 (a) result of the morphological processing and (b) is the first connected component

By processing one connected component at a time, a hole-filling algorithm is used to refill the gaps inside the region which is not filled by dilation. In the same time, the area of the connected component is determined in pixels.

For this solid known area, connected component which can be called C, the first step to start the template (ellipse) matching is by calculating its dimensions based on moments . The connected component is precisely defined by its center $(\bar{x}, \bar{y})$, its orientation θ and the length a and b of its minor and major axis.

To determine the center $(\bar{x}, \bar{y})$ of the connected component, one of the efficient ways is to compute the center of mass (i.e., centroid) of the region.

The center of the area in binary images is the same as the center of the mass and it is computed as shown below:

$$\bar{x} = \frac{\sum x * b(x,y)}{\sum_{(x,y)\in C} 1} \qquad \bar{y} = \frac{\sum y * b(x,y)}{\sum_{(x,y)\in C} 1} \tag{4.1}$$

with

$$b(x,y) = \begin{cases} 1 & if (x,y) \in C \\ 0 & othewise \end{cases}$$

The orientation θ can be computed by elongating the connected component. The orientation of the axis of elongation will determine the orientation of the region. In this axis we will find that the inertia should be the minimum (which is called the least moment of inertia). The axis will be computed by finding the line for which the sum of the squared distances between region points and the line is minimum. In other words, we compute the least-squares of a line to the region points in the image (central of moments). At the end of the process, the angle of inclination (theta) is given by:

$$\theta = \frac{1}{2} arac\tan\left(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right),$$

where

$$\mu_{1,1} = \sum x'y' * b(x,y)$$
$$\mu_{2,0} = \sum (x')^2 * b(x,y) \tag{4.2}$$
$$\mu_{0,2} = \sum (y')^2 * b(x,y)$$

and

$$x' = x - \bar{x}$$
$$y' = y - \bar{y}$$

Before completing the process we pass the value of the orientation through a filter that checks whether if it is in the interval between +45 and -45 degrees, with respect to the vertical axis or not. If outside, the region is eliminated and the process of the template matching is discontinued.

If the region passes from the previous filter the length of the major and minor axis of the connected component can then, be computed by evaluating the moments of inertia. If I_{min}, I_{max} are the least and greatest moment of inertia respectively, of the connected component with orientation θ:

$$I_{\min} = \sum_{(x,y)\in c}\left[(x-\bar{x})\cos\theta-(y-\bar{y})\sin\theta\right]^2 \tag{4.3}$$

$$I_{\max} = \sum_{(x,y)\in c}\left[(x-\bar{x})\sin\theta-(y-\bar{y})\cos\theta\right]^2 \tag{4.4}$$

the length a of the major axis and the length b of the minor axis are given by:

$$a=\left(\frac{4}{\pi}\right)^{1/4}\left[\frac{(I_{\max})^3}{I_{\min}}\right]^{1/8} \quad , \quad b=\left(\frac{4}{\pi}\right)^{1/4}\left[\frac{(I_{\min})^3}{I_{\max}}\right]^{1/8} \tag{4.5}$$

On the basis of the ratio between major and minor axis, a reduction of the number of potential face candidates is possible. We assume that the ratio of the major to minor is satisfying the golden ratio[15]:

$$\frac{height}{width}=\frac{1+\sqrt{5}}{2}$$

with an error range from (-0.5, +0.5).

For the remaining candidates we assess how well the connected component is approximated by its best-fit ellipse generated with the same dimension and orientation of the connected region.

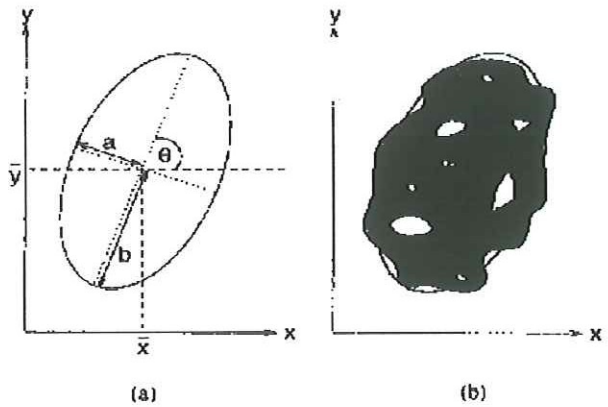


FIG 4.7: (a)Parameter of an ellipse (b) approximation of a connected component

For that purpose the following measure V is evaluated:

$$v = \frac{\sum_{(x,y)\in E}(1-b(x,y))+\sum_{(x,y)\in C\setminus E}b(x,y)}{\sum_{(x,y)\in E}1} \tag{4.6}$$

with

E denoting the best-fit ellipse

V determines the distance between the connected component and the best-fit ellipse by counting the 'holes' inside of the ellipse and the points of the connected component that are outside the ellipse (FIG. 4.7(b)).

The ratio of the number of false points to the number of points of the interior of the ellipse is calculated. Based on a threshold on this ratio, ellipses

that are good approximations of connected components are selected and considered as face candidates.

If the connected component fits completely inside the ellipse, there will be no false points so the threshold used was 0 with acceptable error range from (-1 , +1).

As s final step, if the region passes from the thresholding, the counter is increase by one indicating that there is a person in the image.

Chapter 5

Interface Design

Graphical User Interfaces (GUIs) have become one of the most widely used, and preferred interfaces in the past few years. From its earliest beginnings at Xerox PARC to its commercialisation by Apple Inc. with the Macintosh system and its mass market application by Microsoft Corp. with Microsoft Windows, GUIs have been immensely attractive to the ordinary user. The popularity of the interface can be attributed to three main factors.

1. GUI interfaces provide an interaction mechanism that is more intuitive than the other interface styles.

2. GUI interfaces create an environment in which on screen elements (controls) give the illusion of real world objects with which the user can interact.
3. GUI interfaces can be made more attractive than textual interfaces through the use of pictures, sound, colour, video, and so on.

From a programmer's perspective, however, GUIs are significantly more complex than other interfaces.

The first graphical user interface was designed by Xerox Corporation's Palo Alto Research Center in the 1970s, but it was not until the 1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and a high-quality monitor, which until recently were prohibitively expensive.

Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.

5.1 Interface design:

The first step in the interaction system design is task analysis. We must investigate the user needs in terms of interactivity with system. The interface is to be designed to satisfy the user needs and to satisfy the usability criteria.

5.1.1 Task analysis:

The purpose of the task analysis is to collect the required tasks needed by the user and draw the sequence of their execution need.

The user of the counting software needs to load image with different formats in the canvas of a window obtain the count of the people existing in the image and display the number in clarity, and perhaps save the image in same or different format

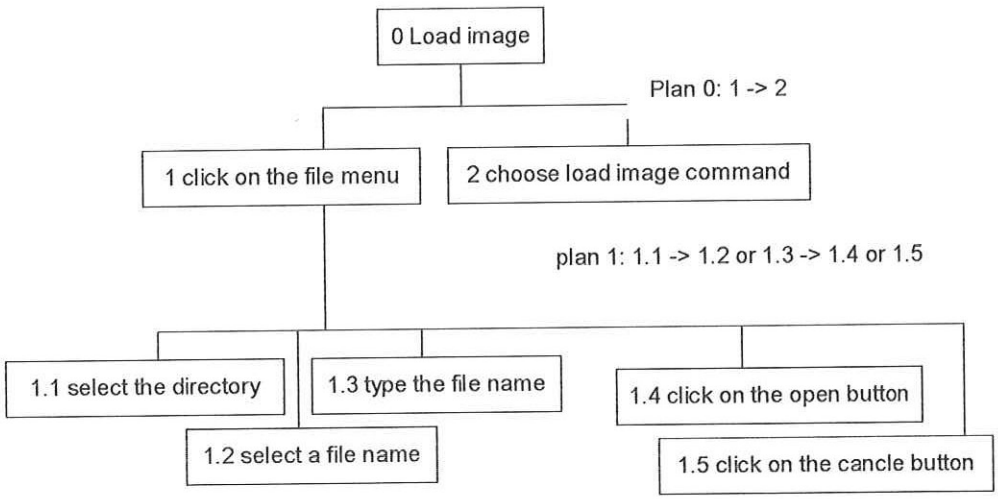
The user may also wish to see the results of the skin color segmentation and the morphological processing and may also like to save it as an image.

So the set of tasks that the user may wish to do in the software is:

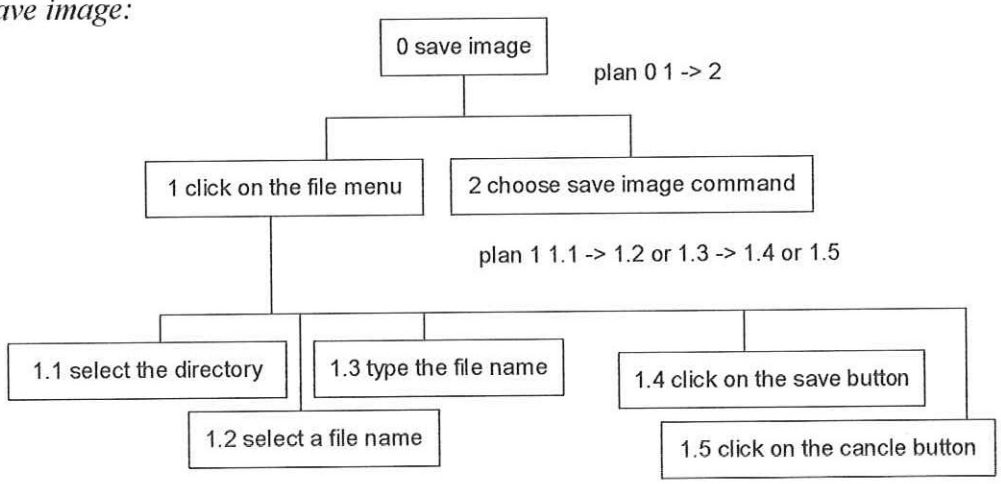
- Load image.
- Display the number of people in the displayed image
- Save image
- Display the skin color segmentation result of the image
- Display the morphological processing result of the image

In the rest of this section, the hierarchical task analysis is given for each task from the previous list. And its associated action plan.

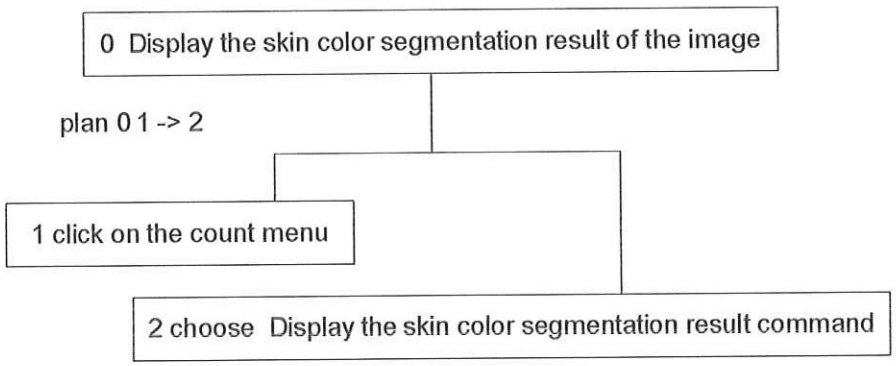
Load image:



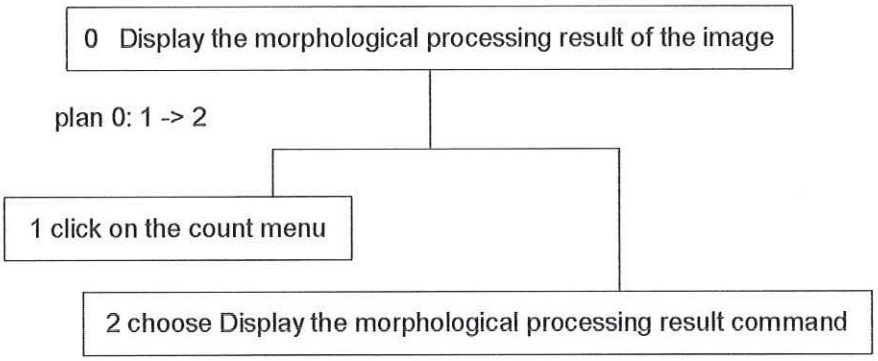
Save image:



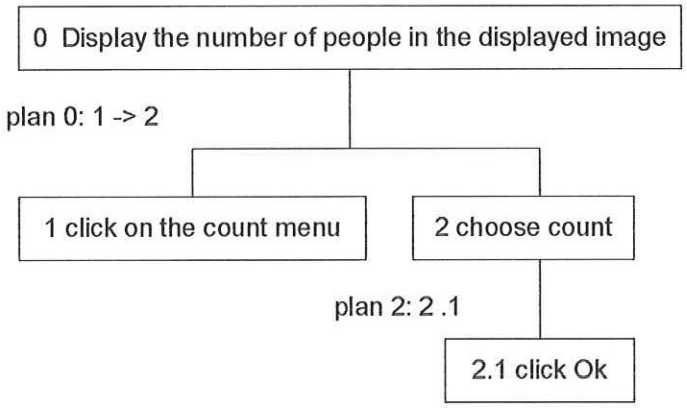
Display the skin color segmentation result of the image:



Display the morphological processing result of the image:



Display the number of people in the displayed image:



5.1.2 Usability testing:

The aim of this project is to design an automatic people counting system to replace humans in the mundane task of tallying the number of people present on a continuous basis.

The design tries to simulate the process of perceptual counting by using humans action sequence of people locating and counting, with the difference that in this project a person is located by identifying the face region only.

The people counting software has the same interface and the same input/output methods so the user can generalize her /his previous knowledge of the computer software to interact with the new system. And the system tries to support consistency by using the same methods and the same interface for controlling the command listed above. The system appears to be familiar so the user can guess the action to be done in a simple way. When some tasks is not available for any reason his action menu command will be grayed to inform the user that some action is not available

The system does not support multithreading because the user can't perform more than one task at he same time. The dialogs in the system are often a user pre-emptive, where the user initiates the dialog by choosing the task option. However, there are some dialogs which are system pre-emptive and these dialogs are the error boxes.

The control of a process in the system is done by the computer. And there is no customizability because the interface can't be modified by neither the system nor the user. The system tries to associate a cancel option to recover from the user action with all tasks. The response time depends on the computer characteristics in which the system is installed and the image size and information.

The basic tasks that the user may need have been covered.

5.1.3 Storyboard:

The final step in the interface design is present the storyboard for the system and how we think that the final system look like.

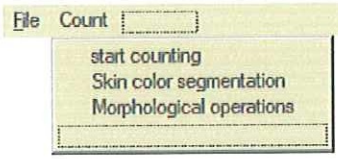
Storyboard Number *p1*	Project Name *Automatic people counting* Designer *Abeer S.* Date *January, 2004*

Comments
Click file menu -> jump to p2
Click count menu -> jump to p3

Storyboard Number *p2*	Project Name	*Automatic people counting*
	Designer	*Abeer S.*
	Date	*January, 2004*

Comments
Click load image command -> jump to p4
Click save image command -> jump to p5

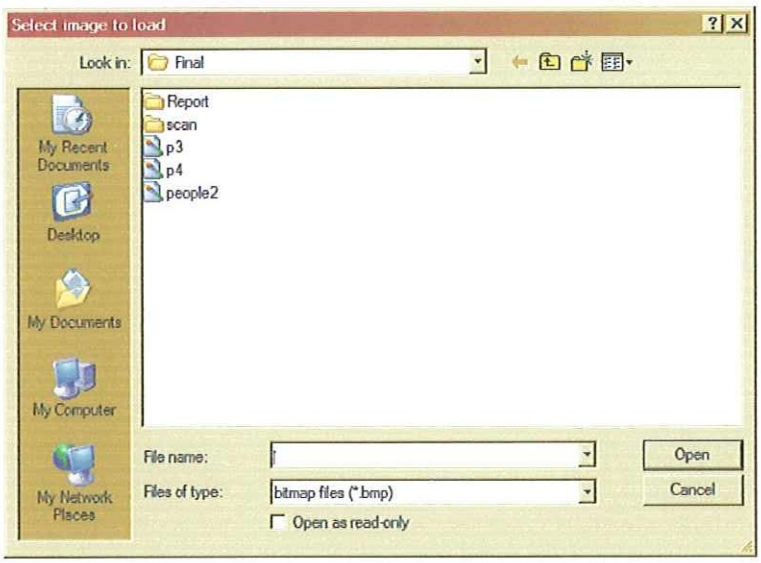
Storyboard Number *p3*	Project Name	*Automatic people counting*
	Designer	*Abeer S.*
	Date	*January, 2004*



Comments
Click start counting command -> jump to p6
Click skin image segmentation command -> jump to p7
Click morphological operations -> jump to p8

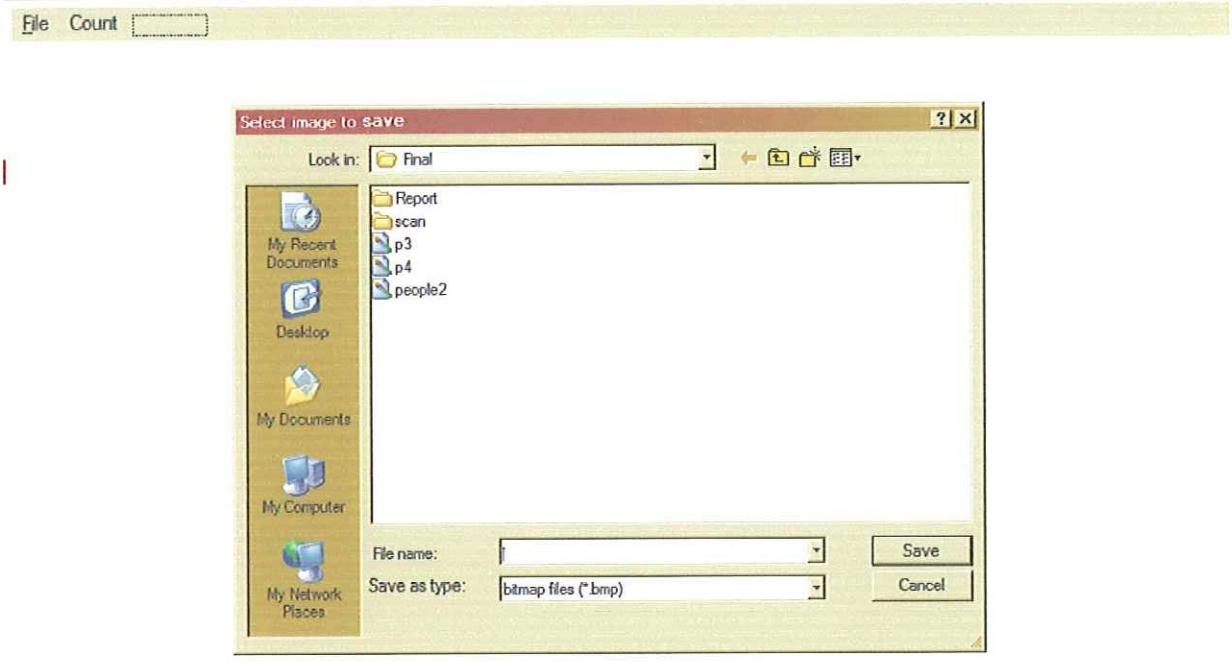
Storyboard Number *p4*	Project Name	*Automatic people counting*
	Designer	*Abeer S.*
	Date	*January, 2004*

File Count



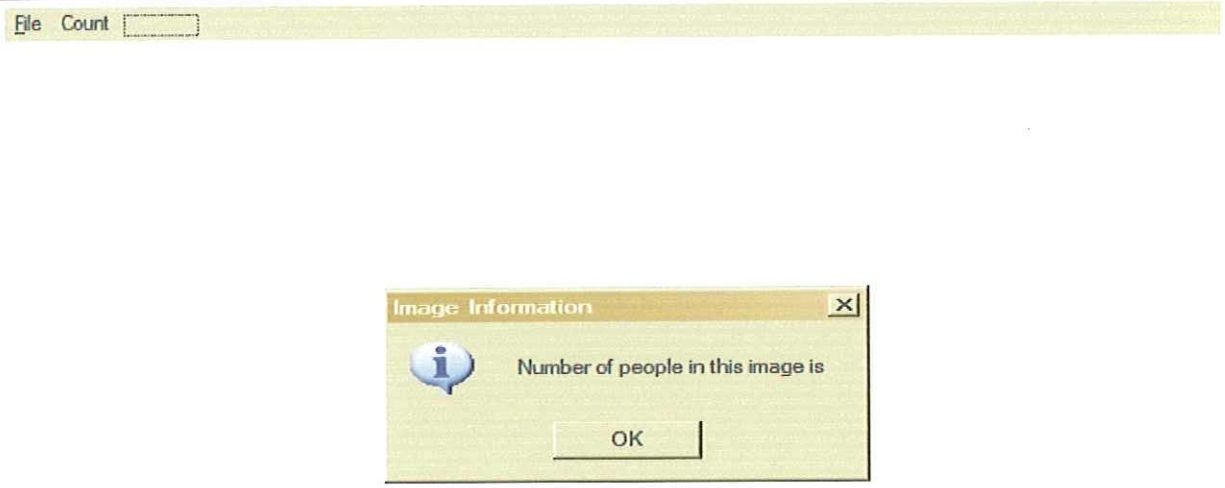
Comments
Click look in list of values -> to locate directory
Click file name list of values -> to choose a previous name
Click file type list of values -> to choose the image format
Click open button -> to open image
Click cancel button -> cancle the action

Storyboard Number *p5*	Project Name	*Automatic people counting*
	Designer	*Abeer S.*
	Date	*January, 2004*



Comments
Click look in list of values -> to locate directory
Click file name list of values -> to choose a previous name
Click file type list of values -> to choose the image format
Click save button -> to save image
Click cancel button -> cancle the action

Storyboard Number *p6*	Project Name	*Automatic people counting*
	Designer	*Abeer S.*
	Date	*January, 2004*

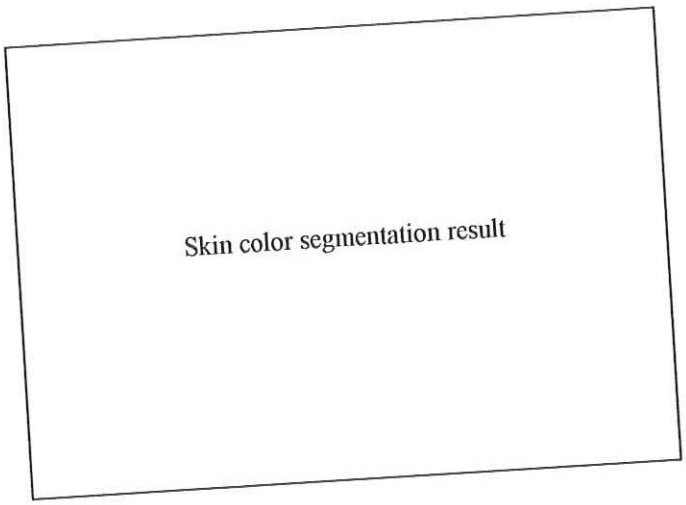


Comments

Click ok button -> to confirm action

Storyboard Number *p7*	Project Name	*Automatic people counting*
	Designer	*Abeer S.*
	Date	*January, 2004*

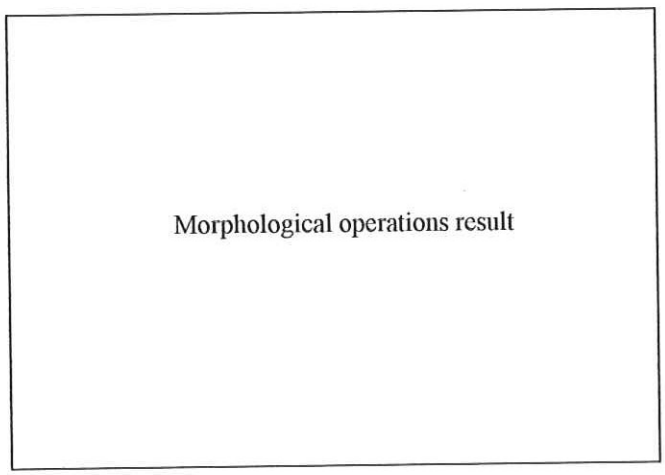
File Count



Comments

Storyboard Number *p8*	Project Name	*Automatic people counting*
	Designer	*Abeer S.*
	Date	*January, 2004*

File Count



Comments

Chapter 6

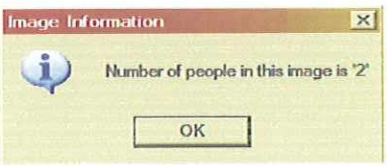
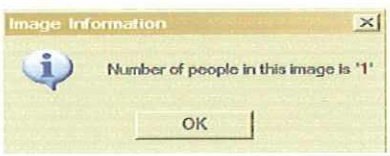
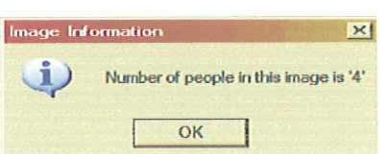
Experimental Results

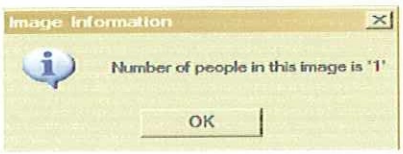
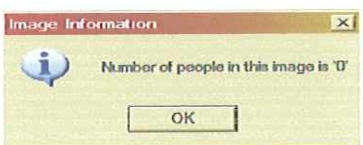
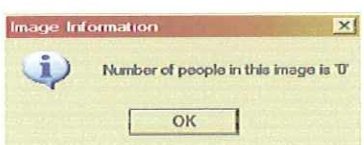
The system was tested on a number of images with different sizes and formats. The quality of images is varying also from simple to complex background and with different number of people in the images.

The system recorded an acceptable counting accuracy compared to similar software which was aimed to count people in different environments with different lighting and background conditions.

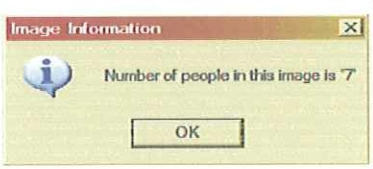
6.1. System results:

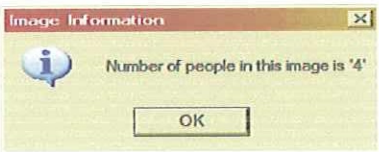
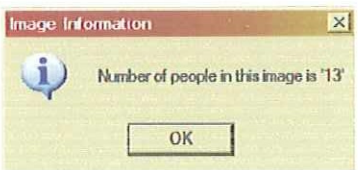
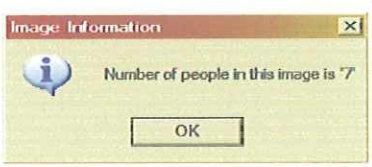
The system correctly detects the following images:

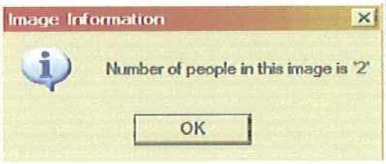
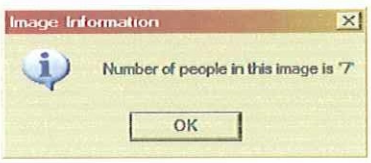
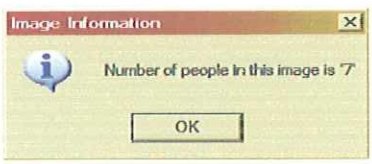
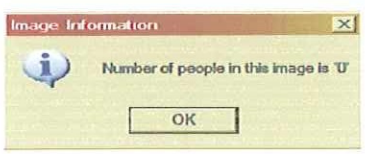
 633x640 JPEG Image 22.6 KB	# of people =2  Error =0	3.4 seconds
 200x267 JPEG Image 6.93 KB	# of people =1  Error =0	1 second
 155x260 JPEG Image 11.2 KB	# of people =4  Error =0	1 second

 367x380 JPEG Image 13.3 KB	# of people =1  Error =0	0.9 seconds
 270x453 JPEG Image 34.1 KB	# of people =0  Error =0	1 second
 100x100 GIF Image 6.62 KB	# of people =0  Error =0	0.4 seconds

The system has some error rate in detecting the following test images:

 306x154 Bitmap Image 92.1 KB	# of people =6  Error =+1	0.8 seconds

 155x260 JPEG Image 11.2 KB	# of people =5  Error =-1	0.6 seconds
 264x188 GIF Image 36.4 KB	# of people =14  Error =-1	1.6 seconds
 706x481 JPEG Image 55.1 KB	# of people =10  Error =-3	5.4 seconds

 176x186 Bitmap Image 61.2 KB	# of people =3  Error =-1	0.6 seconds
 525x333 JPEG Image 31.4 KB	# of people =4  Error =+3	2.5 seconds
 391x271 JPEG Image 93.1 KB	# of people =8  Error =-1	2.3 seconds
 640x454 JPEG Image 27.2 KB	# of people =1  Error =-1	1.1 second

The system was tested on a set of 14 images with 3 different formats (JEPG, GIF, Bitmap) and with different sizes and with different number of people in the image. The background is varying from simple white background to complex background with different colors. The system recoded time averaged 1.7 seconds on a Pentium®*4* 2.0 GHZ. Images with complex background or of large size take more time to process as expected.

Increasing the size of *JPEG* images changes the quality of the color which affects the detection process and accordingly results in errors.

As mentioned in chapter 4 the color segmentation was not tested on black skins. The system makes mistakes when counting people in images that contain black persons except when the black person is lighter than usual or the lighting is such that it renders the face lighter than its original dark color.

It is very difficult to come with a color segmentation that works on all skin color ranges with different lighting and color systems. A great deal of research has been done in this area with no optimal solution for all cases. Hence, inevitably the images which are taken for a person who is swimming or a person that has a part of his face shaded would be miscounted.

Chapter 7

Conclusion and Future work

7.1. Conclusion:

We have presented an automatic face counter that gave reasonably good accuracy ranging from 70 % to 100% and a performance speed average 1.7. Face detection was done using color hue and saturation segmentation as the first cue then template (ellipse) matching is done. However, many aspects of the design are tuned for the constraints of the chosen thresholding that was done, which adversely affect the robustness. Our algorithm is sensitive to the color information in the image and will not work for a gray scale image.

7.2. Future work:

The topic of this research project paves the way for several possible and natural extensions that could be summarized as follows:

- Gender detection and counting of females vs. males was not one of the project objectives. The template matching techniques is insensitive to facial feature and hence to gender. However, one could possibly extend the research to count females vs. males in an image by applying some heuristics e.g. hair ... etc.

- Color segmentation using adaptive thresholding techniques that would increase both the robustness of larger variations of illumination and the portability between different camera systems.

- Detecting side views of the faces.

- Detect faces with suitable orientation but not vertical.

- Counting people with only a part of the face showing.

- Finding a way to better account for separation of overlapping faces, such as creating more accurate eye and nose kernel, can possible help with this.

- Resorting to different strategies in parallel, such as neural networks, may also increase the accuracy of the problem.

Appendix A

wxWindows

wxWindows is a set of libraries that allows C++ applications to compile and run on several different types of computer, with minimal source code changes. There is one library per supported GUI (such as Windows, GTK+, Motif, and Mac). As well as providing a common API (Application Programming Interface) for GUI functionality, it provides functionality for accessing some commonly-used operating system facilities, from copying and deleting files to socket and thread support. wxWindows is a 'framework' in the sense that it provides a lot of built-in functionality, which the application can use or replace as required, thus saving a great deal of coding effort. Basic data structures such as strings, arrays, linked lists and hash tables are also supported.

wxWindows is not a translator from one GUI to another; it cannot take a Motif application and generate a Windows application, for example. IT is a new API. However, the wxWindows API has been praised for its intuitiveness and simplicity, and can be far easier to learn and use than a native GUI API such as Motif or Windows. Porting from MFC is particularly easy due to its similarity.

Such a toolkit is not unique - there are others to choose from - but wxWindows is free, well-established, well-documented, and very broad in its coverage of GUI functionality. It has some extras that make it stand out from the crowd, such as the many convenience dialogs, built-in HTML display and printing, virtual filesystems, easy-to-use OLE automation controller class, Open GL support, and many other features that make it easier to write modern and user-friendly applications.

A.1 wxWindows 2 Features:

1.Basic GUI classes: wxWindows has a wide range of classes for creating standard windows and controls: for example, wxFrame, wxDialog, wxButton, wxListBox, wxGauge, wxComboBox, wxTextCtrl, wxTreeCtrl, and so on. Most of these are wrappers around the equivalent native widget; so a wxListBox uses the Motif listbox widget or Windows listbox control. Occasionally additional widget code has been added to the wxWindows library to support a control, such as the Motif combobox control or the Windows 3.1 gauge implementation. wxToolBar is supplied on each platform for providing good-looking and convenient toolbars. wxNotebook can be used for tabbed dialogs and other applications where tabs are required. wxSplitterWindow allows an application to have two windows with a moveable sash, which when moved completely to one edge triggers code to remove one of the windows, under application control. wxSashWindow provides an even more flexible way

to allow window resizing. wxScrolledWindow implements commonly-used scrolling behaviour, and the programmer can define different scrolling behaviour if required.

2.Event handling: In wxWindows, any class derived from wxEvtHandler may respond to events. Command events can be passed up from child to parent. The mapping between events and class member functions can be done either dynamically (using 'Connect') or statically (using event tables). This is similar to the MFC message map, but implemented in a more consistent manner. Event handler objects can be plugged into other objects, supporting a chain of objects each of which handles a different set of events.

3.Device contexts:wxWindows follows the Windows method of associating a *device context* with a window that is to be drawn into. The drawing code operates via the device context, and therefore by parameterising code by wxDC, you can obtain very portable drawing functions that can draw into a variety of device contexts, such as a wxMemoryDC (for drawing onto a bitmap) or wxPrinterDC (for drawing into a printer).
To draw into a device context, an application normally sets a wxPen (for specifying outline style and colour) and wxBrush (fill colour).

4.Printing: wxWindows offers a printing and print previewing framework, which make use of the wxPrinter, wxPrintout and wxPreview classes, internally using wxPostScriptDC or wxPrinterDC for printing.

5.Bitmaps: wxBitmap supports a variety of bitmap types, which differs according to platform. XPM is implemented on Unix and Windows; and PNG is implemented on both. wxIcon can load ICO or XPM images from file under Windows, and XBM or XPM files under Unix. The wxImage class can load JPEG, TIFF, PCX, GIF, PNM and BMP files and can also rotate and scale

images. wxImage is a platform-independent class with emphasis on loading multiple image formats, whereas wxBitmap wraps the appropriate platform-dependent bitmap.

6.Dialogs: There is a Dialog Editor for wxWindows to allow interactive positioning of controls. Positioning can also be done programmatically using absolute coordinates or the wxWindows constraint system. Convenience dialogs are supported, such as wxFileDialog, wxDirDialog, wxMessageBox, wxColourDialog, wxFontDialog, wxPrintDialog, wxPageSetupDialog.

7.Operating system functionality: A variety of OS functions are supported, such as wxExecute, wxRemoveFile, wxRenameFile, wxCopyFile, wxFileExists. Thread support is handled by a set of classes.

8.Data structures: wxWindows offers some commonly-used data structure classes, including wxList, wxStringList, wxString, wxHashTable.

9. Debugging facilities: wxWindows offers a set of logging functions that can write warnings and error messages to standard output, the Windows debug stream, or to a file. Memory operators are redefined to detect memory leaks, which are reported when the application closes (if compiled in debug mode).

10. Online help: wxWindows applications can support online help, with wxHelpInstance controlling WinHelp under Windows, and Netscape under Unix. The supplied utility Tex2RTF can be used to generate your own RTF, WinHelp RTF, and HTML files from Latex source. wxHTML incorporates a help viewer which can be used to implement on-line HTML help on non-Windows platforms.

11.Interprocess communication: Socket classes are provided on all platforms, plus classes for popular protocols such as HTTP and FTP. On Windows, DDE is encapsulated by a set of classes.

12.Document/view framework: These (optional) classes can help reduce the amount of tedious housekeeping work that an application has to do. An application derives from wxDocument and wxView and specifies the relationship between them. wxWindows implements default handlers for many behaviours such as file open, close, new, etc. Also included in this framework, but useable independently of it, is a collection of classes for implementing Undo/Redo.

13.Database:ODBC is supported on Unix and Windows via a comprehensive set of classes donated by Remstar. wxWindows users can also use the dBase clone Xbase.

14.Documentation: A comprehensive reference manual is supplied in HTML, PDF, and WinHelp formats.

15.Compiler support: Most popular compilers are supported, through makefiles and/or project files. Compatible compilers include VC++ (versions 1.5 up to 6.0), Borland C++, Watcom C++, CodeWarrior, Cygwin (formerly GnuWin32), Mingw32, gcc, egcs, Sun C++.

Appendix B

User Guide

B.1 System Requirements:

- A 1.0 GHz , with Pentium or Pentium Pro preferred.
- A minimum of 215 MB RAM.
- A 15", 8-bit color monitor supporting a minimum of 256 colors, and 640×480 resolution.
- Approximately 3-4MB of free hard disk space.
- A double-speed or faster CD-ROM drive to speed process.
- Windows XP /Windows 2000 or later.

B.2 user manual:

Lunch the system for a CD-Rom:

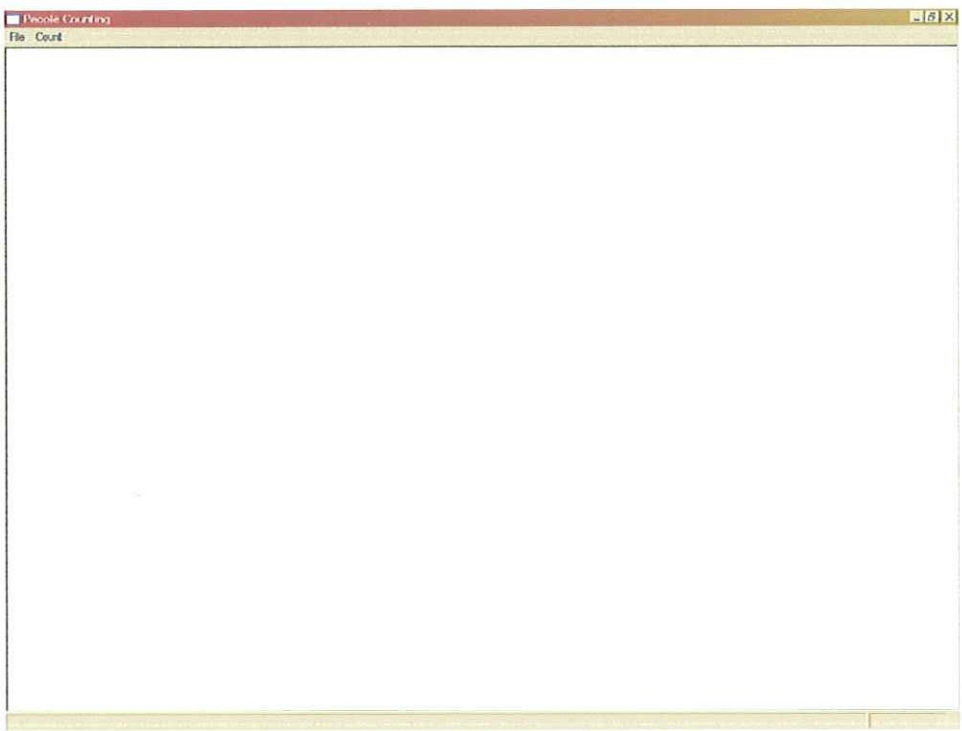
1 inserts the counting people CD-Rom in the CD-Rom drive.

2 double click the people counting program

Lunch the system for a Hard drive:

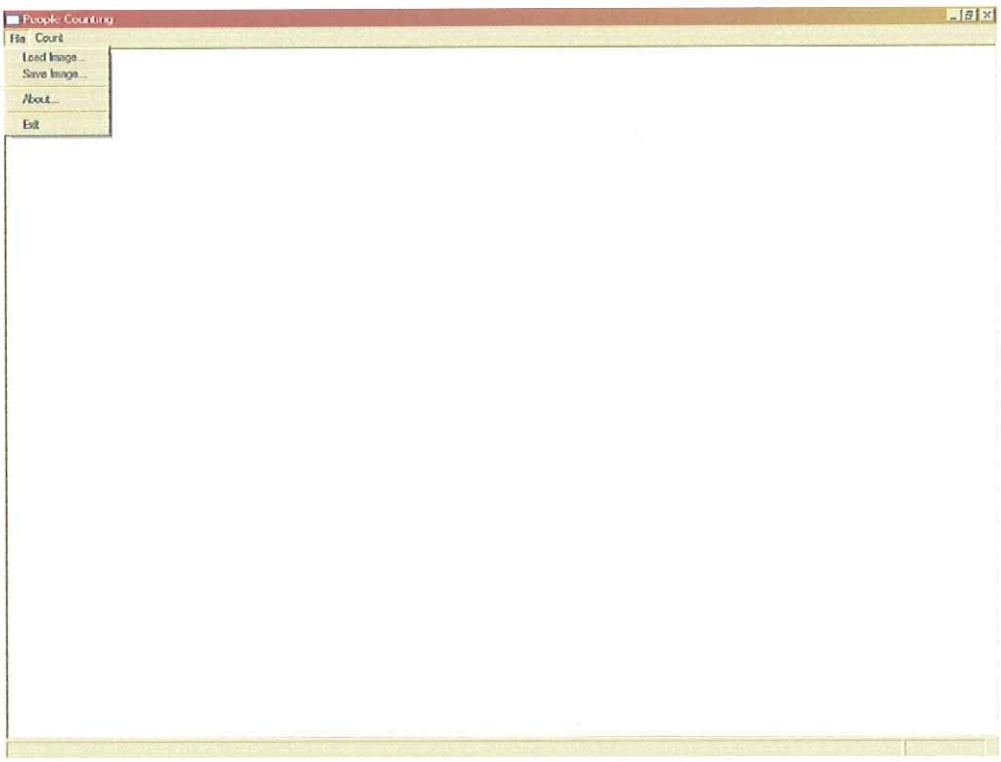
1 double click the people counting program icon in the desktop

The stat window will appear

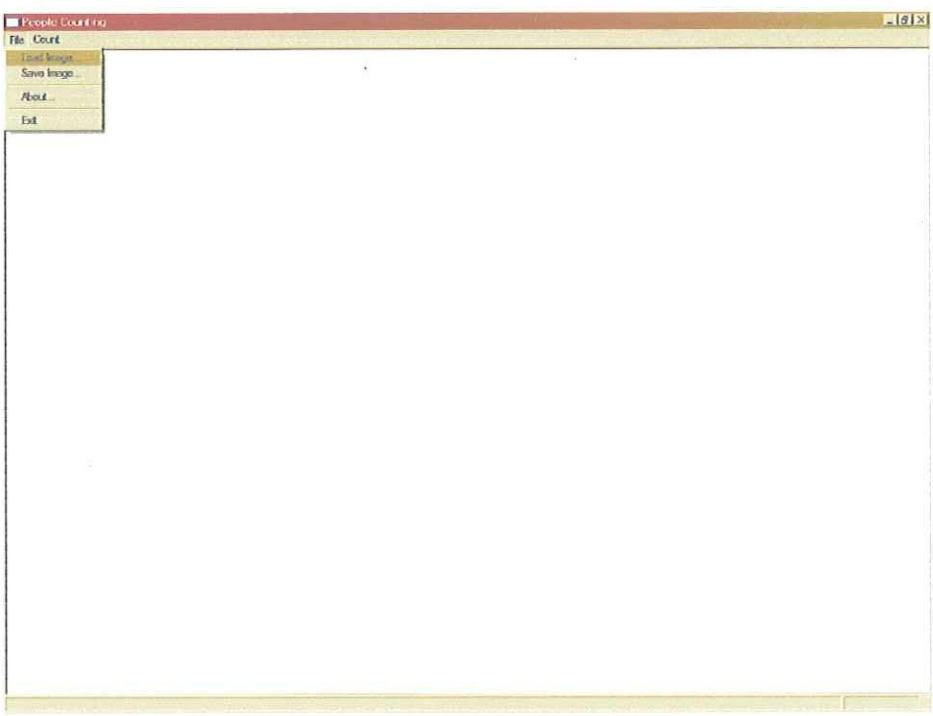


B.2.1 Load Image:

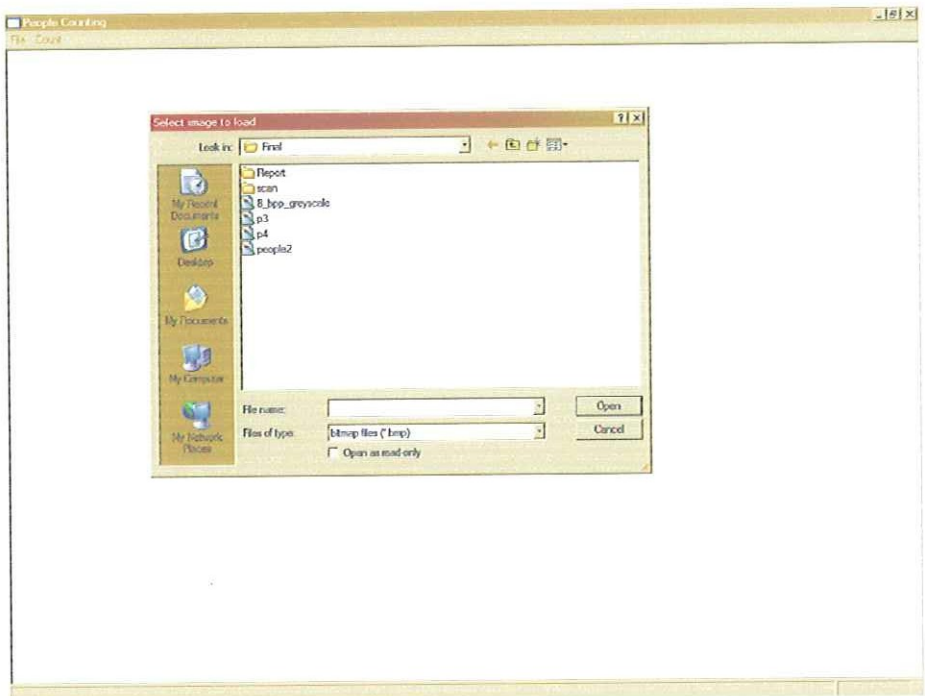
1 Select file menu



2 Select load image command



3 In the (load image) dialog box.



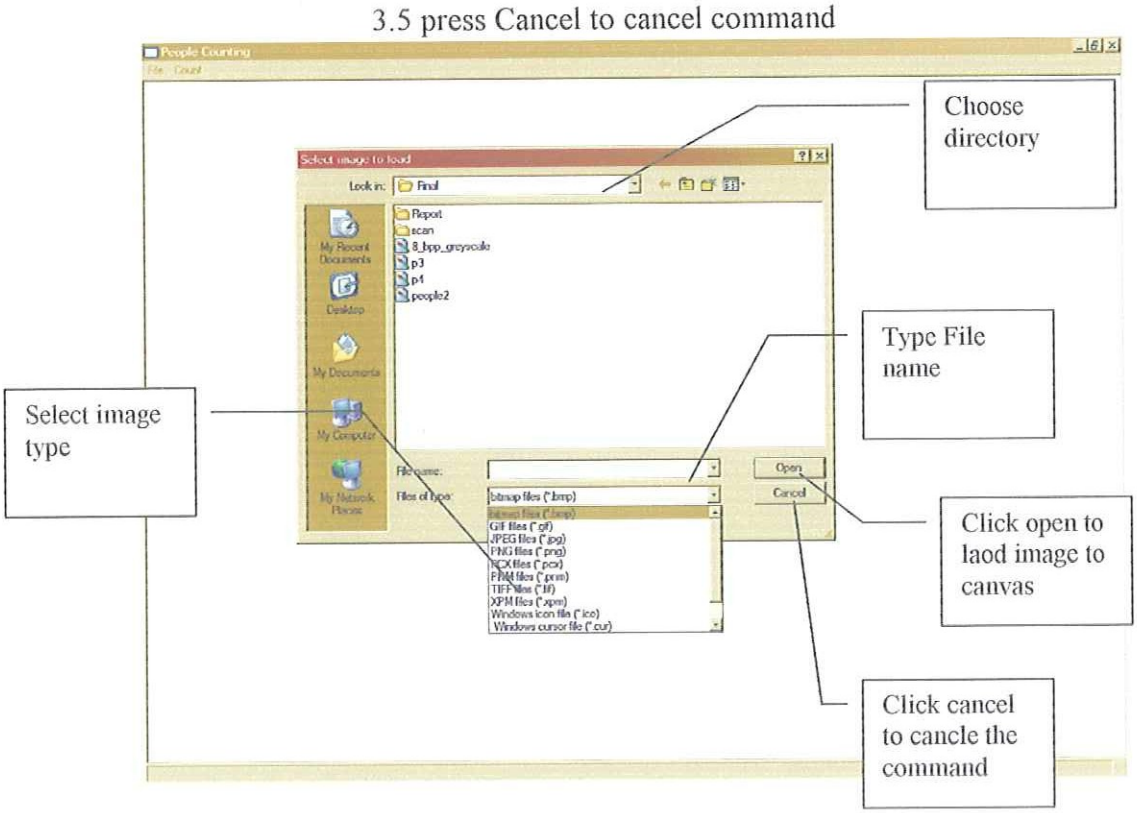
3.1 choose directory

3.2 type file name

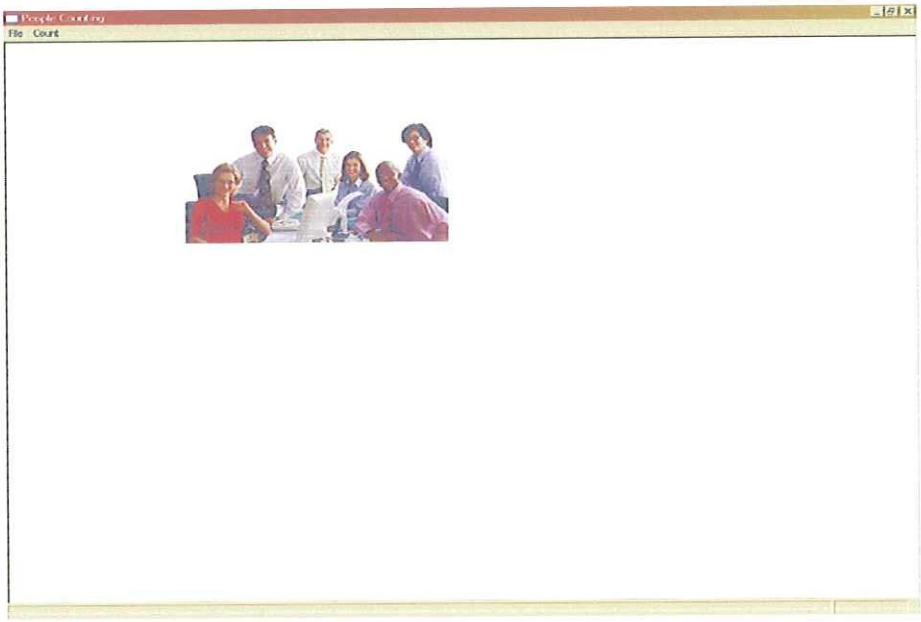
3.3 select image type

3.4 press open to load image to the canvas

3.5 press Cancel to cancel command

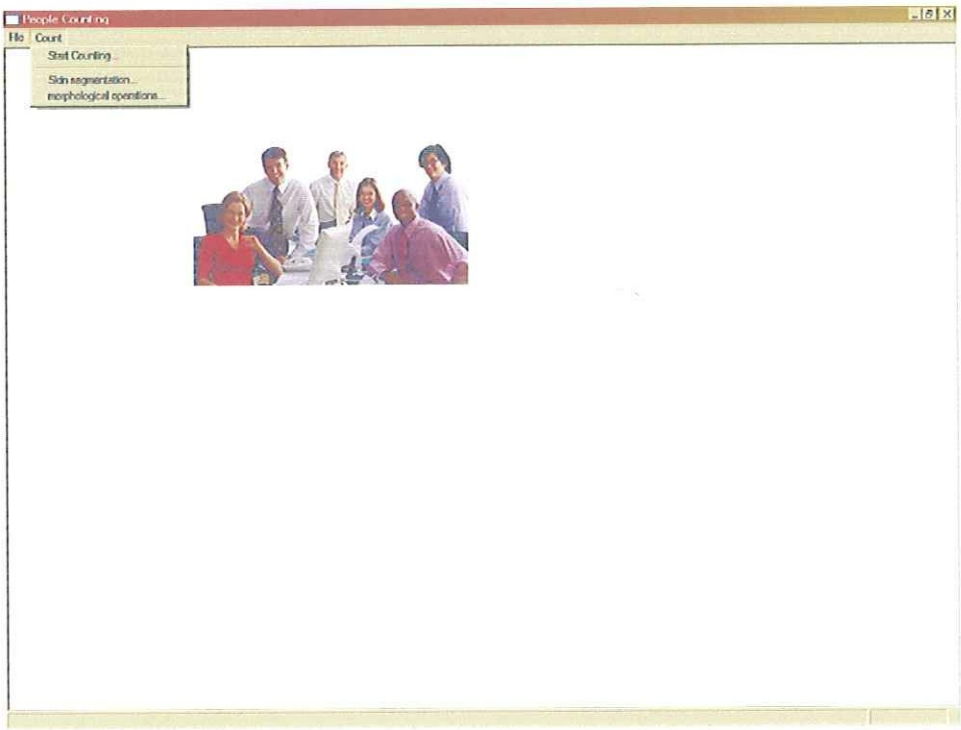


If you choose open the image will be loaded to the canvas

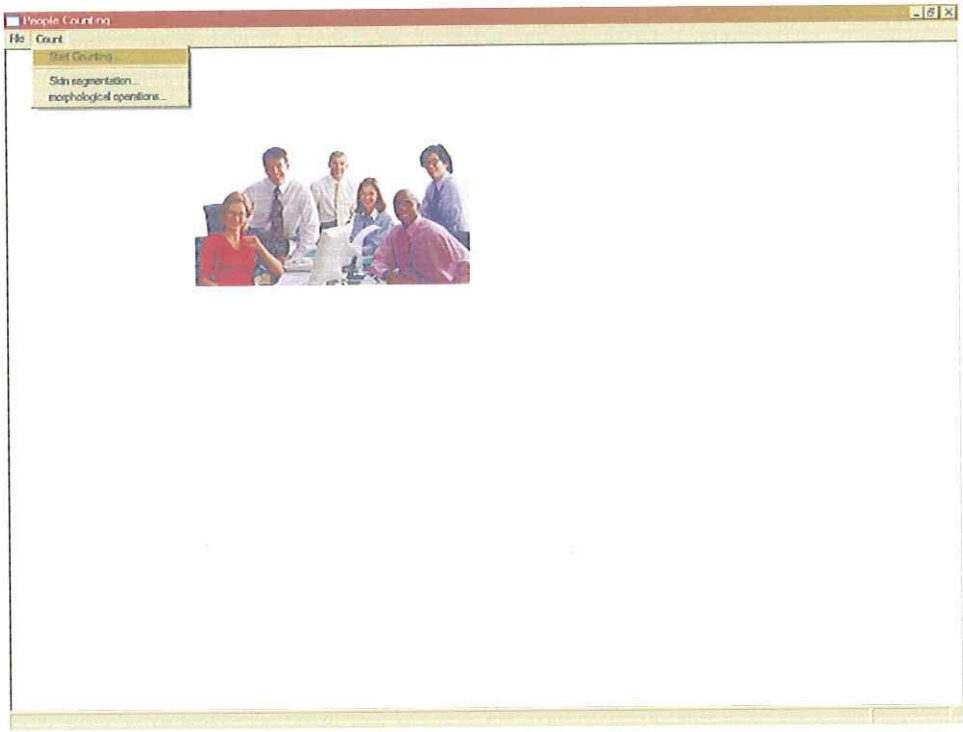


B.2.2 Display the number of people in the displayed image:

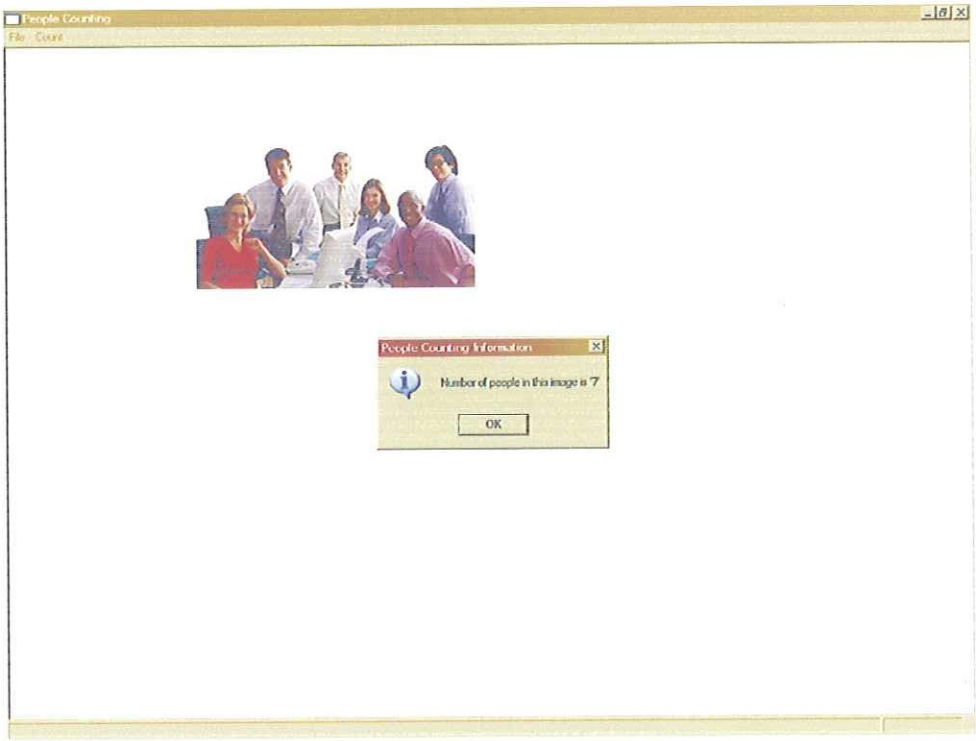
1 select count menu



2 Select start counting command



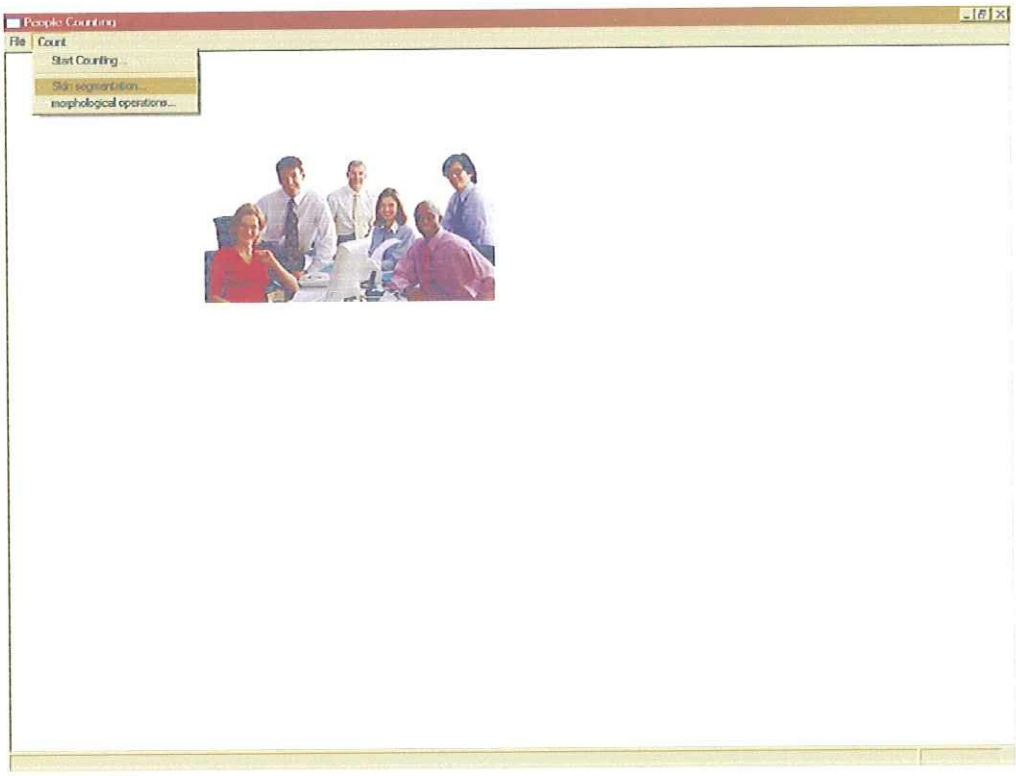
dialog box appear with the number of people in the image



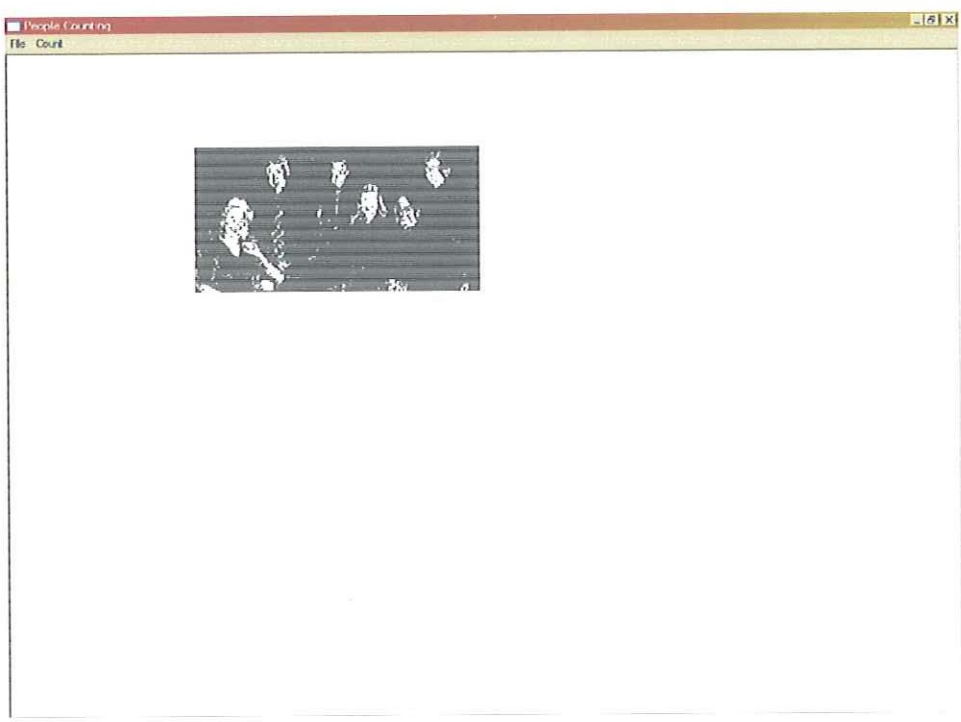
B.2.3 Display the skin color segmentation result of the image:

1 select count menu (as above)

2 Select skin segmentation command from the menu



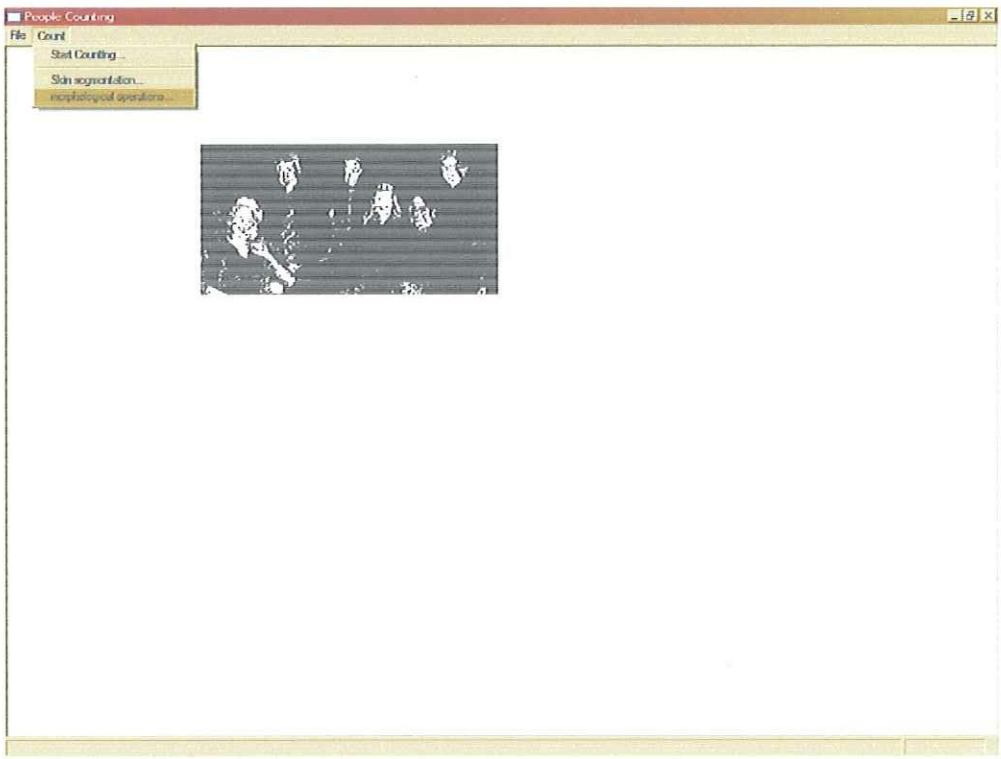
Skin color segmentation result will be displayed on the canvas



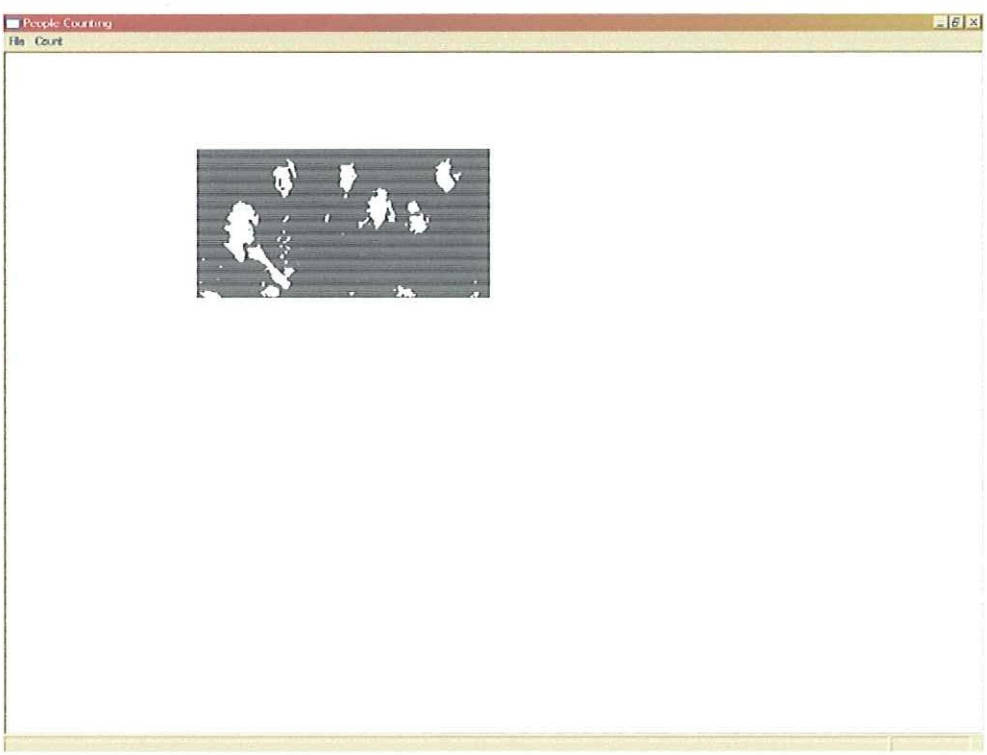
B.2.4 Display the morphological processing result of the image:

1 select count menu (as above)

2 Select morphological operations command from the menu.



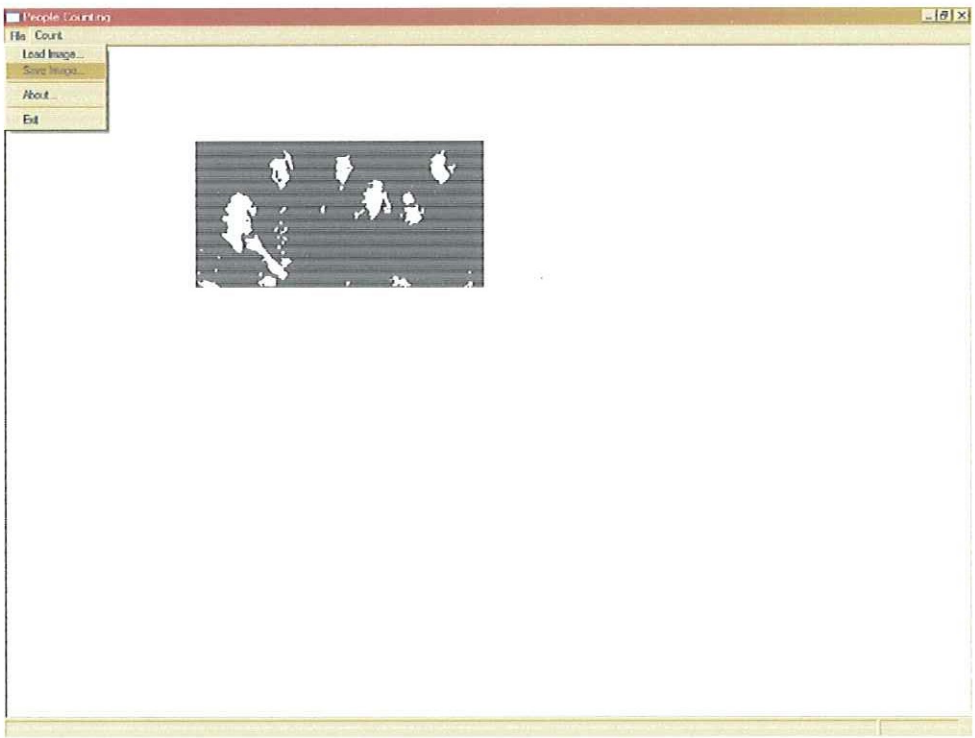
Morphological operations result will be displayed on the canvas



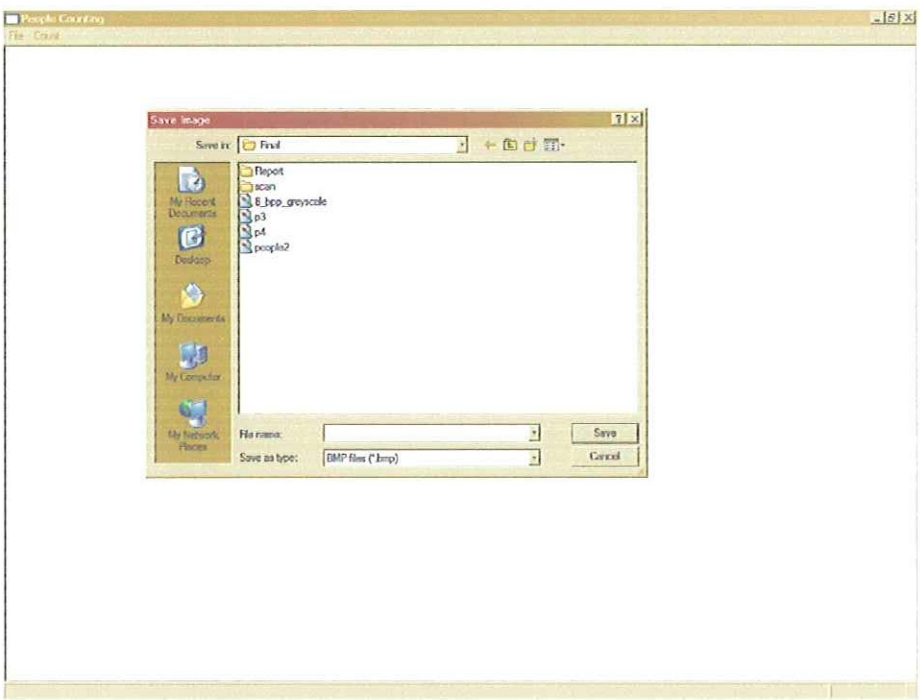
B.2.5 Save Image:

1 Select file menu (as above)

2 Select save image command



3 In the (save image) dialog box.



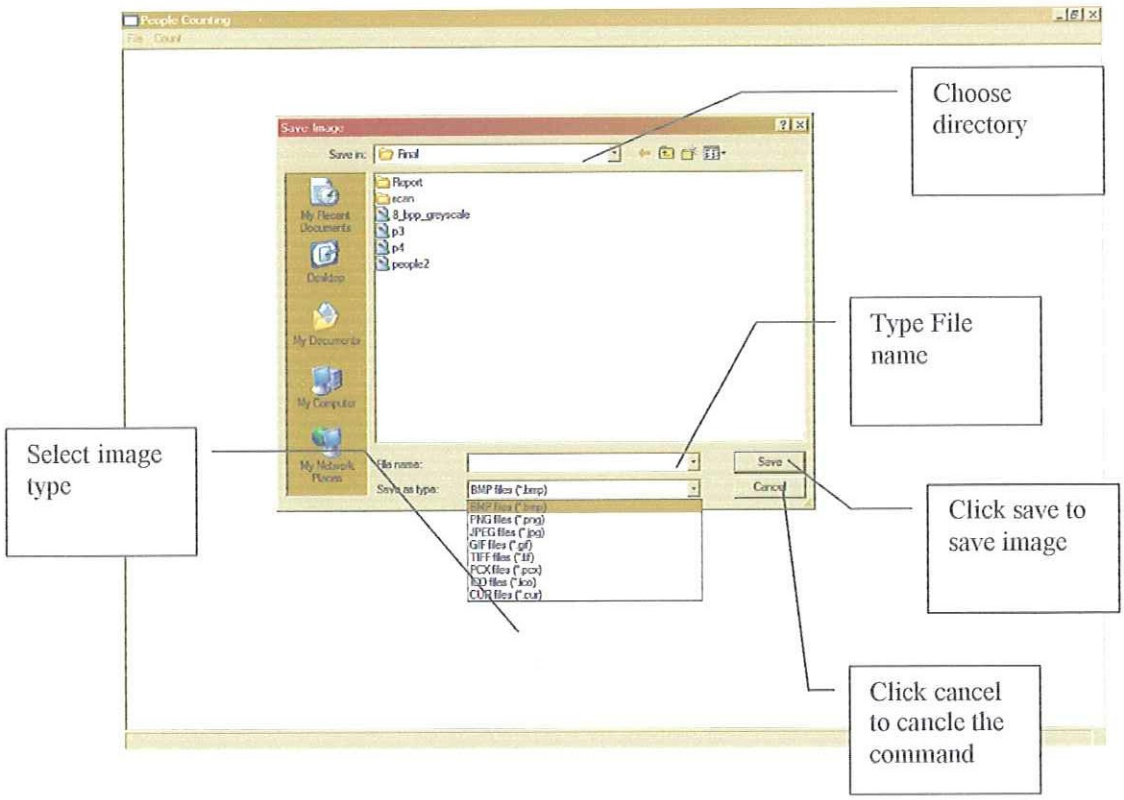
3.1 choose directory

3.2 type file name

3.3 select image type

3.4 press save to save image

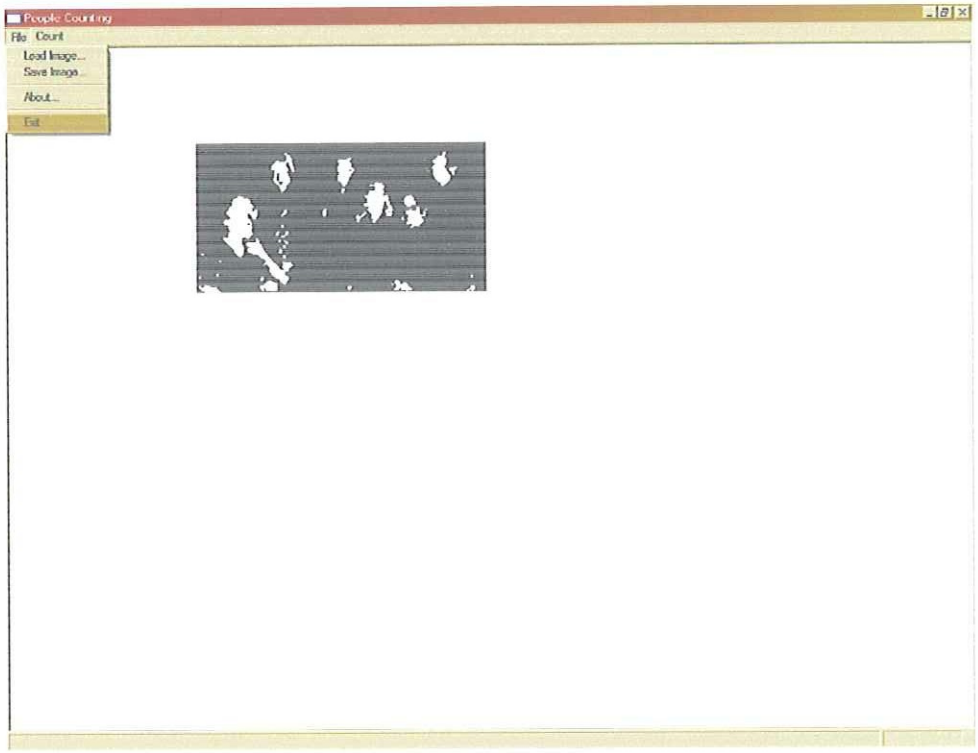
3.5 press Cancel to cancel command



B.2.6 Exit :

1 Select file menu (as above)

2 Select exit command


People Counting
File Count
Load Image...
Save Image...
About...
Exit

Bibliography

1. Rafeal C. Gonzalez Richard E. Woods, Digital Image Processing, *2nd Edition, Prentice Hall, New Jersey, USA, 2002*
2. F. S. Hill,JR. Computer Graphics using openGl, 2nd Edition, *Prentice Hall, New Jersey, USA, 2001*
3. Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, Human-Computer Interaction, 2nd Edition, *Prentice Hall, New Jersey, USA*
4. Bill McCarty, Visual C++6 core Language little Black Book, coriolis, USA
5. P. J. L. Van Beek, M. J. T. Reinders, B. Sankur, and J. C. A. Van Der Lubbe, Semantic segmentation of videophone image sequences, in *Proc. of SPIE Int. Conf. on Visual Communications and Image Processing, 1992*, pp. 1182–1193.
6. R. Brunelli and T. Poggio, Face recognition: Feature versus templates, *IEEE Trans. Pattern Anal. Mach. Intell.* **15**, 1993, 1042–1052.
7. G. Burel and D. Carel, Detection and localization of faces on digital images, *Pattern Recog. Lett.* **15**, 1994, 963–967.
8. M. C. Burl, T. K. Leung, and P. Perona, Face localization via shape statistics, in *Int. Workshop on Automatic Face and Gesture Recognition, Zurich, Switzerland, June 1995.*
9. 12. M. C. Burl and P. Perona, Recognition of planar object classes, in *IEEE Proc. of Int. Conf. on Computer Vision and Pattern Recognition, 6, 1996.*
10. R. Chellappa, C. L.Wilson, and S. Sirohey. Human and machine recognition of faces: A survey, *Proc. IEEE* **83**, 5, 1995.
11. Craw, H. Ellis, and J. R. Lishman, Automatic extraction of face-feature, *Pattern Recog. Lett.* Feb. 1987, 183–187.
12. Y. Dai and Y. Nakano, Face-texture model based on sgld and its application, *Pattern Recog.* **29**, 1996, 1007–1017.
13. L. C. De Silva, K. Aizawa, and M. Hatori, Detection and tracking of facial features by using a facial feature model and deformable circular template, *IEICE Trans. Inform. Systems* **E78–D(9)**, 1995, 1195–1207.
14. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
15. V. Govindaraju, Locating human faces in photographs, *Int. J. Comput. Vision* **19**, 1996.
16. H. P. Graf, E. Cosatto, D. Gibson, E. Petajan, and M. Kocheisen, Multi-modal system for locating heads and faces, in *IEEE Proc. of 2nd Int. Conf. on Automatic Face and Gesture Recognition, Vermont*, Oct. 1996, pp. 277–282.
17. G. Holst, Face detection by facets: Combined bottom-up and top-down search using compound templates, in *Proceedings of the 2000 International Conference on Image Processing, 2000*, p. TA07.08.
18. H. Hongo, M. Ohya, M. Yasumoto, Y. Niwa, and K. Yamamoto, Focus of attention for face and hand gesture recognition using multiple cameras, in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000.*
19. S. H. Jeng, H. Y. M. Liao, C. C. Han, M. Y. Chern, and Y. T. Liu, Facial feature detection using geometrical face model: An efficient approach, *Pattern Recog.* **31**, 1998.

20. V. Kumar and T. Poggio, Learning-based approach to real time tracking and analysis of faces, in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000.*
21. C. H. Lee, J. S. Kim, and K. H. Park, Automatic human face location in a complex background, *Pattern Recog.* **29**, 1996, 1877–1889.
22. D. Maio and D. Maltoni, Real-time face location on gray-scale static images, *Pattern Recog.* **33**, 2000, 1525–1539.
23. J. Ng and S. Gong, Performing multi-view face detection and pose estimation using a composite support vector machine across the view sphere, in *Proc. IEEE International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999.*
24. N. Oliver, A. Pentland, and F. B´erard, LAFTER: A real-time face and lips tracker with facial expression recognition, *Pattern Recog.* **33**, 2000, 1369–1382.
25. E. Osuna, R. Freund, and F. Girosi, Training support vector machines: An application to face detection, in *IEEE Proc. of Int. Conf. on Computer Vision and Pattern Recognition, 6, 1997.*
26. H. A. Rowley, S. Baluja, and T. Kanade, Neural network-based face detection, *IEEE Trans. Pattern Anal Mach. Intell.* **20**, January 1998, 23–38.
27. H. A. Rowley, S. Baluja, and T. Kanade, Rotation invariant neural network-based face detection, in *Proc IEEE Intl. Conf. on Computer Vision and Pattern Recognition, 1998*, pp. 38–44.
28. T. Sakai, M. Nagao, and T. Kanade, Computer analysis and classification of photographs of human faces, in *Proc. First USA—Japan Computer Conference, 1972*, p. 2.7.
29. H. Schneiderman and T. Kanade, Probabilistic modeling of local appearance and spatial relationships for object recognition, in *IEEE Conference on Computer Vision and Pattern Recognition, 6, 1998.*
30. H. Schneiderman and T. Kanade, A statistical model for 3D object detection applied to faces and cars, in *IEEE Conference on Computer Vision and Pattern Recognition, 2000.*
31. L. Sirovich and M. Kirby, Low-dimensional procedure for the characterization of human faces, *J. Opt. Soc. Amer.* **4**, 1987, 519–524.
32. Y. Sumi and Y. Ohta, Detection of face orientation and facial components using distributed appearance modelling, in *IEEE Proc. of Int. Workshop on Automatic Face- and Gesture-Recognition, Zurich, Switzerland, Jun. 1995*, pp. 254–255.
33. K.-K. Sung and T. Poggio, Example-based learning for view-based human face detection, *IEEE Trans Pattern Anal. Mach. Intelligence* **20**, 1998, 39–51.
34. J. Terrillon, M. Shirazi, M. Sadek, H. Fukamachi, and S. Akamatsu, Invariant face detection with support vector machines, in *Proceedings of the 15th International Conference on Pattern Recognition, 2000*, Vol. IV, p. 4B.
35. C. Wong, D. Kortenkamp, and M. Speich, A mobile robot that recognises people, in *IEEE Int. Conf. on Tools with Artificial Intelligence, 1995.*
36. G. Yang and T. S. Huang, Human face detection in a complex background, *Pattern Recog.* **27**, 1994, 53–63.
37. J. Yang and A. Waibel, A real-time face tracker, in *IEEE Proc. of the 3rd Workshop on Applications of Computer Vision, Florida, 1996.*
38. K. C. Yow and R. Cipolla, Feature-based human face detection, *Image Vision Comput.* **15**(9), 1997.
39. S. McKenna, S. Gong, and J. J. Collins, Face tracking and pose representation, in *British Machine Vision Conference, Edinburgh, Scotland, Sept. 1996.*
40. T. W. Yoo and I. S. Oh, A fast algorithm for tracking human faces based on chromatic histograms, *Pattern Recog. Lett.* **20**, 1999, 967–978.

41. J. Cai and A. Goshtasby , Detecting human faces in color images, *Image and Vision Computing, Volume 18, Issue 1, 8 December 1999, Pages 63-75*
42. Sobottka, K. and Pittas, I., A novel method for automatic face segmentation, facial feature extraction and tracking , *Signal Processing: Image Communication* 12, 1998, pp.263-281.
43. J.C. Terrillon, M. David, and S. Akamatsu. Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. *Proc. International Conf. on Automatic Face and Gesture Recognition*, pp. 112--117, 1998. 33