


FEATURE BASED INDEXING OF HAND-WRITTEN TEXT IMAGES

NAGARAJAH KUNTHARSHAN

 This dissertation was submitted to the Department of Computer Science and Engineering of the University of Moratuwa in partial fulfillment of the requirement for the Degree of M Sc in Computer Science

University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2010

96425

Abstract

Identity management system which is maintaining a manual database system to store its data is facing a big challenge of searching a particular entry from the database if a query rose where the date of birth of the person is not known. Person Identification department, as an example, the existing database is in the form of Pre-designed hand written physical cards with the details. The cards are served separated by sex and ordered by date of birth.

In case of a person whose date of birth is not known, there is a challenge of going through the cards one by one to spot the exact record. So, person registration department decided to computerize the database and found out the following. Since the hand writing is very poor there is no chance of using character recognition software. The better option could be manually enter the data into a database, but quotations submitted for the tender called for this task was much higher than the potential level of the department.

Finally, feature extraction method is found out as an ideal solution for this task. In this approach all the cards are scanned and saved in the system using batch scanning. Each file is pre-processed and the number of characters in the name is saved to the database as index for the corresponding scanned image. The search operation on the database based on the number of characters of the name will list down the name of the possible card and the corresponding card also will be fetched from the saved location and previewed. Among the cards that are identified, user needs to find the cards manually.

This will narrow down the search. System will fail in counting the number of characters in the name if there is no space left between two characters or if one character is split "into two by mistake. To handle these challenges, a new intelligent algorithm should be generated with the ability to understand the order or the pattern of occurrences of the characters and make the decision based on them

Declaration

I, Nagarajah Kunatharshan hereby declare that the work included in this dissertation in part or whole has not been submitted for any other academic qualification at any institution.

N. Kunatharshan

Mr. N. Kunatharshan

Submitted for examination

UOM Verified Signature

DR. Chathura De Silva

(Supervisor)



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Acknowledgements

Having completed the MSc. Degree program at the Department of Computer Science and Engineering, University of Moratuwa, I would like to take this opportunity to thank those who helped me a lot in completing it at this level.

First of all, I would like to thank my supervisor, Dr. Chathura De Silva for the supervision and guidance despite his busy schedules. Also I would like to thank Ms. Visaka Nanayakkara, Dr. Gihan Dias and Dr. Sanath Jayasena for conducting the subject Research Seminar, which helped me a lot when browsing through the published research papers.

I should be grateful to all the members of the staff of the Department of Computer Science and Engineering for giving valuable feed back during the progress review sessions.

Finally, I would like to thank my family for supporting me in every aspect in completing my degree program and the research successfully.

Thank you all!



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Kunatharshan N

(06/8279)

Table of Contents

Declaration.....	2
Abstract.....	3
Acknowledgements.....	4
Table of Contents.....	5
List of Figures.....	7
1. Introduction.....	8
2. Literature Review	11
2.1 Edges Detection:	11
2.2 Lines Detection – Hough Transform:	12
2.3 Image Registration with Template matching	13
2.4 Cross-correlation.....	14
2.5 Open and Close.....	14
2.6 Erosion	15
2.7 Dilation	15
2.8 Opening.....	16
2.9 Closing	16
2.10 Color Scale Conversion – gray scale.....	17
3. Materials and Methods	18
3.1 Adding images to the system	19
3.1.1 Template image:.....	19
3.1.2 Printed Character Removal:	20
3.1.3 Removal of lines:.....	21
3.1.4 Closing:	23
3.1.5 Character counting:	24
3.2 Finding images.....	24
4. Observations	25

 University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

5. Analysis and Discussion of Results..... 26

 5.1 Binary image conversion: 26

 5.2 Closing: 27

 5.3 Line removal: 28

6. Conclusions and Recommendations for Future Research 29

7. References..... 30

8. Annexure..... 32



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

List of Figures

Figure 3.1: Flowchart of the model	18
Figure 3.2: Scanned image (Eyes and ID Numbers are masked in order to keep the confidentiality)	19
Figure 3.3: Template image	20
Figure 3.4: Cropped Image	21
Figure 3.5: Name surroundings after the removal of printed characters	21
Figure 3.6: Generated line image.....	22
Figure 3.7: Scanned image after the line removal	23
Figure 3.8: Name surroundings after the removal of lines	23
Figure 3.9: After the closing operation.....	23
Figure 3.10: After the noise removal.....	24
Figure 4.1: Processed segment of hand written characters.....	25
Figure 4.2: Processed segment of printed characters.....	25
Figure 5.1: Result with binary image	26
Figure 5.2: Result with gray image.....	27
Figure 5.3 close with 'disk' (radius 2) morphological structuring element	27
Figure 5.4: close with 'line' (length 10, angle 90 degrees) morphological structuring element	27
Figure 5.5: close with 'line' (length 3, angle 90 degrees) morphological structuring element	28

1. Introduction

Many Identity management systems of Sri Lanka need a system to filter or find the details of distributed identification documents. Currently their system is not automated and the searching is done manually. The details are maintained in papers which is a predesigned printed card filled in hand written Sinhala characters. Name, address, date of birth, place of birth, sex and the photographs are important contents of the cards. Even though the cards (template) are regularly printed if the hand writing is poor then there are possibilities where even human wouldn't be able to read and understand them properly. Now these filled cards are separated by sex and sorted in Date of Birth (DOB) order. When a person is reapplying for an Identity Card due to some reasons, if he/she knows the Date of Birth then it would be easy to find the corresponding card otherwise it will be a challenging task to locate the correct one. But not everyone remember their DOB correctly due to not having proper record in place or forgetting it as time goes.

Other than the difficulties in finding cards on request, another main thread for the manual database is getting damaged and losing information due to the expiration of the card material. When paper cards are lasting for more than eighty or ninety years it's getting damaged little by little and causing more problems in search process.

There were so many ideas suggested and tried to automate database in order to make the search process easy and keep the data safe. Optical Character Recognition (OCR) scan or using character recognition software cannot be used because of the poor quality and irregularity of the hand written characters. Exact solution for this scenario could be entering data manually to a new database. However, the cost factor is found as the limitation for this suggestion after the evaluation of the quotation submitted against the tender called by the Persons Registration Department for this data entry task.

Therefore it is finally concluded to go for a solution to narrow down the search by using possible features which could be extracted from the scanned text images (predesigned printed hand filled cards). Number of characters in the name was identified as a suitable feature to be extracted from the scanned card to narrow down the search. There were no related ideas or implementations for this particular scenario

were found during this study; however so many research works found related to feature extraction.

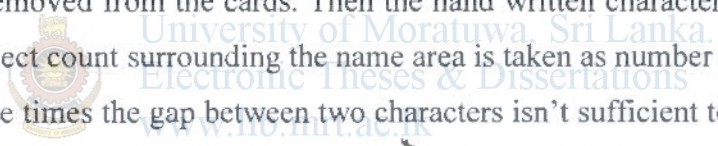
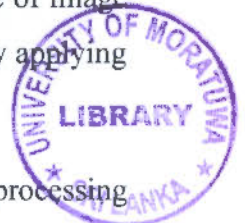
Hough transform is one of the well known techniques for any parametric shape detection. Image registration is the process of aligning two or more images of the same scene. Typically, one image, called the base image, is considered the reference to which the other images, called input images, are compared. The objective of image registration is to bring the input image into alignment with the base image by applying a spatial transformation to the input image.

Image registration is often used as a preliminary step in other image processing applications. But in this project the template (printed portion of the card) should be removed from the card in order to analyze the hand written part efficiently. Here, the image registration is implemented using cross correlation technique.

A template was created and certain parts (printed characters near to the name area) are selected and registered one after the other on the original cards. And the lines also detected and removed from the cards. Then the hand written characters are treated as objects and object count surrounding the name area is taken as number of characters in the name. Some times the gap between two characters isn't sufficient to separate them as two objects and some times two parts of one character is left more gaps to identify as two objects. 'Open' and 'Close' preprocessing techniques also used in order to minimize the error in object count.

The software has two major sections; one is addition of features of image to the database and the other is finding the card back from the database by using stored features. Above mentioned image processing techniques are used in the addition process. User will be allowed to add the cards to the database one by one or multiple cards at once.

System will automatically find the number of characters in the name and store in the database with the name of the image. Since the database contains very few attributes, the search will be very faster. When a person ask the system to search based on the given number of characters, system will perform the search within the database and list out the image's name, based on the image's name the correct card will be fetched from the local directory where all the cards are saved.



Since MATLAB has very useful image processing toolbox it is selected as the programming language. In addition to this it has the ability to connect the database and create the user interface. Microsoft Access is used as the backend database to store the extracted features.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2. Literature Review

Throughout the study, there were no works related to this problem domain found; however, the work related to the feature extraction is very wide and there are many researches and works which used the similar technologies in this work and proved their validity are found. Many well known techniques are implemented in many research works. Feature extraction techniques return information about the structure of an image. For example, this technology can be used to find edges, locations, and attributes of objects. Some of the image processing tasks are known as preprocessing tasks, which are usually performed prior to the main image processing actions in order to produce the good and correct results by enhancing the images by reducing the noises. Close, Open, Color scale conversion (RGB to Binary or Grayscale) is some of the well-known preprocessing methods.

Gray Image Reconstruction by Wabeed Abu Ulbeh, Akram Moustafa, and Ziad A. Alqadi paper proposed a methodology of eliminating the noise from the gray image and reconstructing the image in order to get the image without losing any piece of information from it. Morphological Operations such as Erosion, Dilation, Opening, and Closing which are explained below are suggested as the suitable methodologies to pre process the image in order to achieve the high quality result of this operation. [Erosion, dilation and related operators, by Mariusz Jankowski]

There are suggestions on the usefulness and the suitability of the morphological operations mentioned above and the suitable order of performing them. [[Separating Low Pass and High Pass Frequencies in the Image without Losing Information by Mohammed Abu Zalata and Ziad Alqadi] , [Morphological Image Processing by Ranga Rodrigo], [Application of Mathematical morphology operations for simplification and improvement of correlation of images in close range-photogrammetry by M.Kowalczyk, P.Koza, P.Kupidura, J.Marciniak]

2.1 Edges Detection:

The goal of edge detection is to mark the points in a digital image at which the luminous intensity changes sharply. Sharp changes in image properties usually reflect important events and changes in properties of the world. These include

- (i) Discontinuities in depth,

- (ii) Discontinuities in surface orientation
- (iii) Changes in material properties
- (iv) Variations in scene illumination



Edge detection is a research field within image processing and computer vision, in particular within the area of feature extraction.

Edge detection of an image reduces significantly the amount of data and filters out information that may be found as less relevant, while preserving the important structural properties of an image. There are many methods for edge detection, but most of them can be grouped into two categories. They are search-based and zero-crossing based. The search-based methods detect edges by looking for maxima and minima in the first derivative of the image, usually local directional maxima of the gradient magnitude. The zero-crossing based methods search for zero-crossings in the second derivative of the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression. Noises in the Scanned images are very significant (unwanted data), thus brightness changing very sharply. This technique is identified to play a big role in marking them in order to be removed during the image processing. [16]

Various detection technologies have been analyzed, discussed and compared with each other. The zero-crossing method has been found as the efficient one among all the traditional methods and also the accuracy of it proven. [Comparison for Edge Detection of Colony Images by Wang Luo]

2.2 Lines Detection – Hough Transform:

The Hough transform is a feature extraction technique used in digital image processing. The classical transform identifies lines in the image, but it has been extended to identifying positions of arbitrary shapes. The transform universally used today was invented by Richard Duda and Peter Hart in 1972, who called it a "generalized Hough transform" after the related 1962 patent of Paul Hough. The transform was popularized in the computer vision community by Dana H. Ballard through a 1981 journal article titled "Generalizing the Hough transform to detect arbitrary shapes". As the images contain many lines, in order to minimize the error in the result it is necessary to remove all the printed lines from the scanned images.

Lines Detection technique is found to be very useful in achieving this. Hough transform is used and proved to be a useful technique in the paper. Use of the Hough transformation to detect lines and curves in pictures by Richard O. Duda Peter E. Hart.

2.3 Image Registration with Template matching

Template matching is a technique in Digital image processing for finding small parts of an image which match a template image. It can be used in manufacturing as a part of quality control, a way to navigate a mobile robot, or as a way to detect edges in images.

There are different approaches to accomplishing template matching. Some are more preferment than others, and some find better matches.

The basic method is to perform template matching while looping through all the pixels in the search image and compare them to the pattern. While this method is simple to implement and understand, it is one of the slowest methods.

This method is normally implemented firstly by creating a sub image (the template), if the sub image is $w(x, y)$ where x and y represent the coordinates of each pixel. Then move the center of this sub image w over each (x, y) point in the candidate image, which is $o(x, y)$ and calculate the sum of products between the coefficients in o and the corresponding neighborhood pixels in the area spanned by the filter mask. This method is sometimes referred to as 'Linear Spatial Filtering'.

This type of spatial filtering is normally used only in dedicated hardware solutions because of the computational complexity of the operation, however this complexity can be lessen by filtering it in the frequency domain of the image, referred to as 'frequency domain filtering,' this is done through the use of the correlation theorem.

Another way to make the matching faster is to reduce the image into smaller images, and then search the smaller sub images for a match to the template, these smaller images are often referred to as an image pyramid (i.e. a 128 x 128px image can have a pyramid of smaller images which are 64x64, 32x32, 16x16 etc).

After finding matches in the smaller images, that information is used in the larger image as a center location. The larger image is then searched in a small window to

find the best location of the pattern. Other methods can handle problems such as translation, scale and image rotation.

Image registration with template matching technique has been used and the accuracy of them is proven to determine the appropriate shift for single – polarized SAR images.[PROCESSING OF POLARAMETRIC SAR IMAGES by Pamela A. Delaney] and Computational methods for nonlinear image registration by Ulrich Clarenz, Marc Droske, Stefan Henn, Martin Rumpf, and Kristian Witsch.

2.4 Cross-correlation

Cross-correlation is a very efficient tool to match images. It is quite robust to noise, and can be normalized to allow pattern matching independently of scale and offset in the images. It has serious drawbacks however, in the case images of faint sources.

One approach to identifying a pattern within an image uses cross correlation of the image with a suitable mask. Where the mask and the pattern being sought are similar the cross-correlation will be high. The mask is itself an image which needs to have the same functional appearance as the pattern to be found.

Issues may be, the process can be extremely time-consuming, and the 2D cross-correlation function needs to be computed for every point in the image. Calculation of the cross correlation function is itself a N^2 operation. Ideally the mask should be chosen as small as practicable. In many image identification processes the mask may need to be rotated and/or scaled at each position. Printed characters in the scanned image which are not the data to be processed should be removed in order to reduce the error; this technique is being used to remove them.

The correlation techniques have been used and the accuracy of them is proven to determine the appropriate shift for single – polarized SAR images.[PROCESSING OF POLARAMETRIC SAR IMAGES by Pamela A. Delaney]

2.5 Open and Close

Morphologic operators open and close are directly based on erosion and dilation. Operation open deletes “fuzzy” boundaries of an object, while close can be used to close small gaps within an object. Some of the hand written characters are split into

many and the parts are separated from each other and look like different characters. So this technique is being used to make them one and increase the accuracy. [1]

2.6 Erosion

The fundamental operation of mathematical morphology is erosion. All mathematical morphology depends on this notion. The erosion of an input image A by a structuring element B is defined as follows in equation (2.1):

$$A \ominus B = \{x : B + x \subseteq A\} \dots\dots\dots (2.1)$$

\subseteq - Subset or equivalent of

\ominus - Erosion

This means that in order to perform the erosion of A by B we translate B by x so that this lies inside A. The set of all points x satisfying this condition constitutes $A \ominus B$. The erosion of an image can also be found by intersecting all translates of the input image by the reflection of the structuring element:

$$A \ominus B = \bigcap \{A + b : b \in -B\} \dots\dots\dots (2.2)$$

\in - Element of

\bigcap - Intersection

\ominus - Erosion

[4, 13, 14, 15, 16]

2.7 Dilation

The dual operation to erosion is dilation. Dilation of an input image A by a structuring element B is defined as follows in equation (2.3):

$$A \oplus B = \bigcup \{B + a : a \in A\} \dots\dots\dots (2.3)$$

\in - Element of

\bigcup - Union

\oplus - Dilation

This means that in order to perform the dilation of A by B we first translate B by all points of A. The union of these translations constitutes $A \oplus B$. [4,13, 14, 15, 16]

2.8 Opening

A secondary operation of great importance in mathematical morphology is the opening operation. Opening of an input image A by a structuring element B is defined as follows in equation (2.4):

$$A \circ B = (A \ominus B) \oplus B \text{ ----- (2.4)}$$

- ⊖ - Erosion
- ⊕ - Dilation
- ∘ - Opening

An equivalent definition for opening is as in equation (2.5):

$$A \circ B = \bigcup \{B + x : B + x \subseteq A\} \text{ ----- (2.5)}$$

- ∘ - Opening

∪ - Union



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk



⊆ - Subset or equivalent of

This means that in order to open A by B we first translate B by x so that this lies inside A. The union of these translations constitutes, for instance, the opening of a triangle A by a disk B (the origin coincides with the centre of the disk) is the triangle A with rounded corners. In general, opening by a disk rounds or eliminates all peaks extending into the image background. [4, 14, 15, 16]

2.9 Closing

The other important secondary operation is closing. Closing of an input image A by a structuring element B is defined as follows in equation (2.6):

$$A \bullet B = (A \oplus B) \ominus B \text{ ----- (2.6)}$$

- - Closing
- ⊖ - Erosion
- ⊕ - Dilation

For instance, closing a triangle A by a disk B (the origin is on the centre of the disk) yields the same triangle A. In this case and we say that A is B-close. In general, closing by a disk rounds or eliminates all cavities extending into the image foreground. [4, 14, 15, 16]

2.10 Color Scale Conversion – gray scale

The grayscale processing method is also used in image processing to make the process easy and reduce the processing power. Some grayscale processing methods are based on the brightness graduation data of the image captured by the camera. The binary conversion method recognizes only white or black (1 or 0) data. The grayscale processing method divides the brightness graduation into 8 bits (256 levels), and obtains a differentiation result based on all the data. Therefore this method offers more shade resolution and therefore more accurate detection. [14]



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

3. Materials and Methods

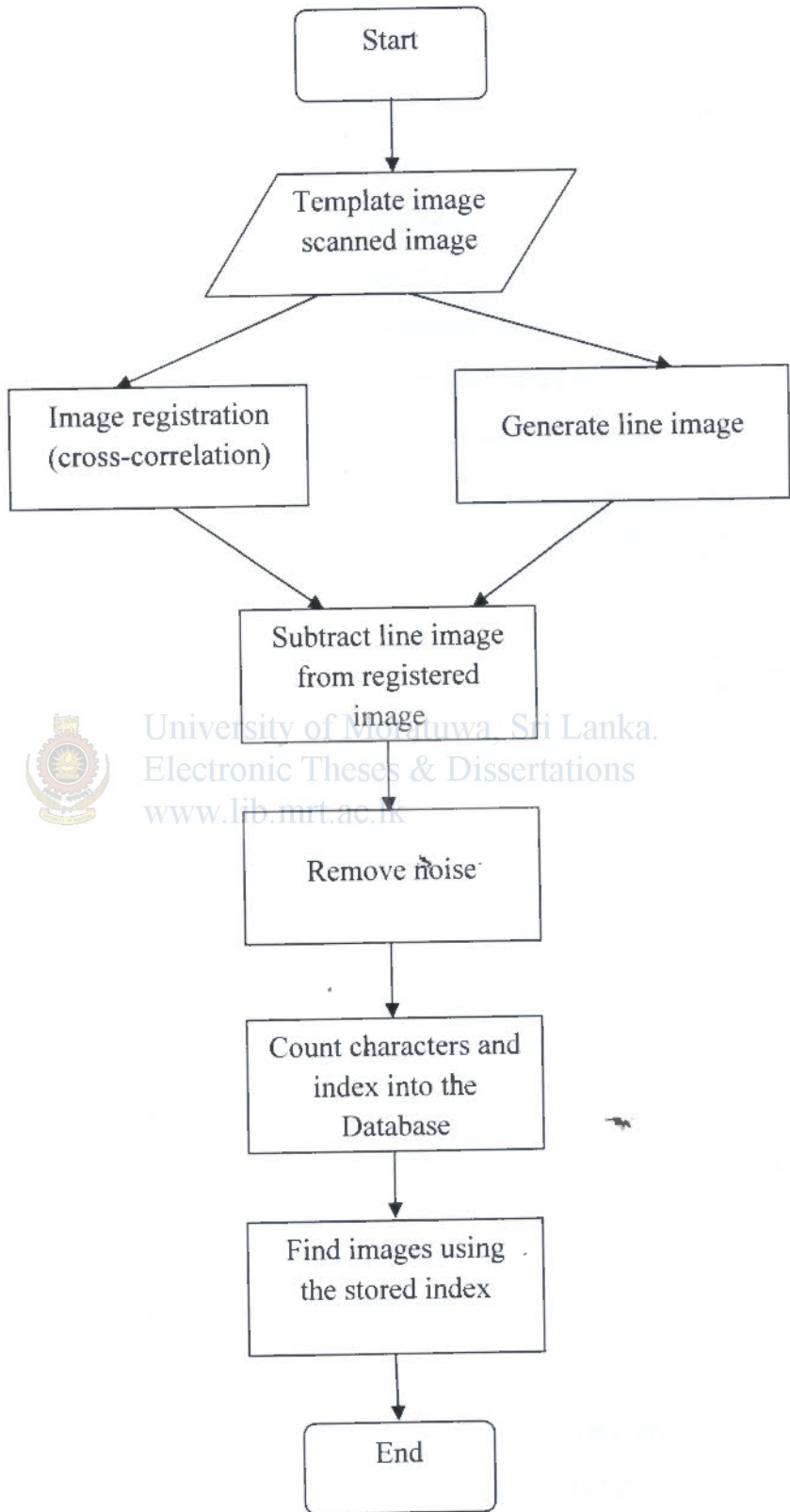


Figure 3.1: Flowchart of the model

Card samples are collected from Persons Registration Department for this project. There are many preprocessing involved due to the poor quality of the samples which don't meet the requirements of the image processing task. Physical cards which are, printed and hand writing filled are batch scanned and saved as digital images. Since many cards are lasting for long years they became blurred over the years. During the batch scanning there is a thread of images affected with geometric distortion due to the poor alignment of the card. The other problem may be the poor quality of hand writing. In some cases, there could be a situation where longer space left between two characters and in some cases the space between two characters may not be sufficient. There can be special cases where one character breaks into two.

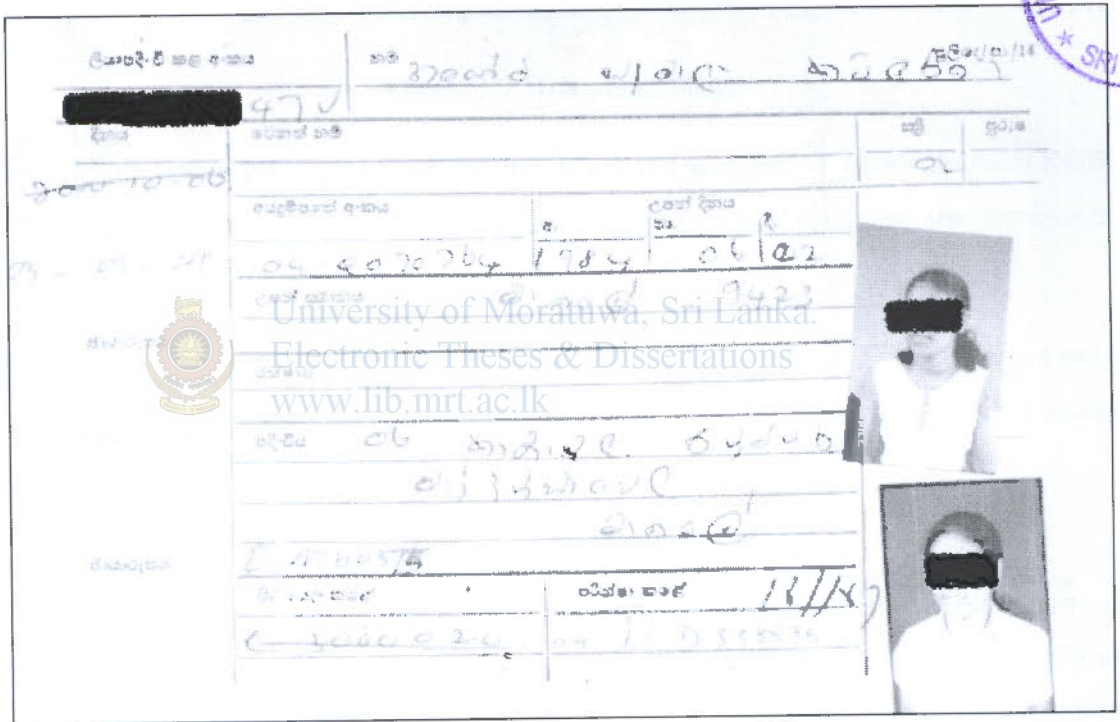


Figure 3.2: Scanned image (Eyes and ID Numbers are masked in order to keep the confidentiality)

3.1 Adding images to the system

3.1.1 Template image:

Template image contains only the printed characters and this image will be used to identify and remove the printed characters from the image samples.

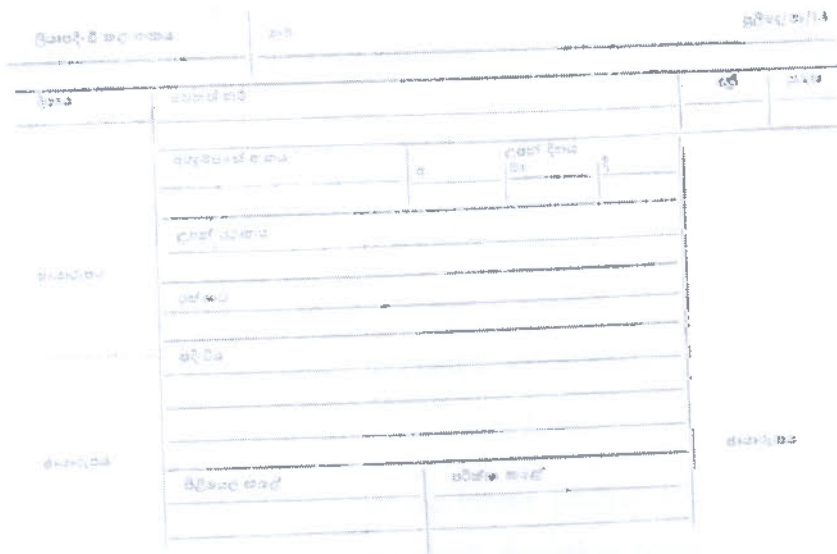


Figure 3.3: Template image

During the first preprocessing action, the image is converted to grayscale from RGB. This will eliminate the hue saturation information while retaining the luminance which will make processing easier and efficient.

In the first attempt RGB images are converted into Binary image. Since there are only two values in binary image, threshold application for noise removal wasn't a success as expected.

3.1.2 Printed Character Removal:

Match or Register entire template at once with the scanned image is not practical because scanned images are geometrically destroyed and intensity of the images also differs. That means the pixel pattern of the template is not exactly same as the scanned image. And the handwritten characters are treated as noise in this case, because template image contains only the printed characters and lines, and the other scanned images which need be registered with the template contains printed as well as the handwritten characters. So there are no possibilities of matching the whole image at once with the template. For the entire template matching, cross-correlation fails with unacceptable noise level. The total offset or translation between images depends on the location of the peak in the cross-correlation matrix, and on the size and position of the sub images.

Therefore to overcome these problems, the small portions of template image is cropped and registered with the original image using cross-correlation technique.

33

Figure 3.4: Cropped Image

The cropped image will be the template, and it must be smaller than the scanned image. The peak of the cross-correlation matrix occurs where the cropped image is best correlated.

This also gives unacceptable result for the binary images and better result for the grayscale images. Binary images show much deviation between template and the scanned image.

Image registration (cross-correlation) locates the matching area for the sub image (Cropped image) in the scanned images.

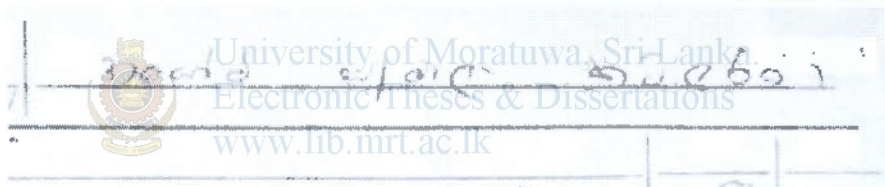


Figure 3.5: Name surroundings after the removal of printed characters

3.1.3 Removal of lines:

After removing the printed characters from scanned image the remains are lines and the hand written characters. Printed lines always introduce some noise while giving incorrect results in character count.

Removing Horizontal Lines:

Entire image is horizontally traversed starting from the left top corner and the each pixel examined for their intensity values. Pixels which have the intensity more than 220 are considered as part of the printed lines or the hand written characters. If these pixels go on with more than 15 pixels then this is identified as a horizontal line and a new same size image is created with this pixel values.

Removing Vertical Lines:

Same procedure is followed but this time traversal carried out vertically. Identified vertical lines also copied to the same new image which is created from horizontal lines.

Now the new image contains only the lines (vertical and horizontal). This new image subtracted from the original scanned image. Now the resultant image contains only the hand written characters.



Figure 3.6: Generated line image

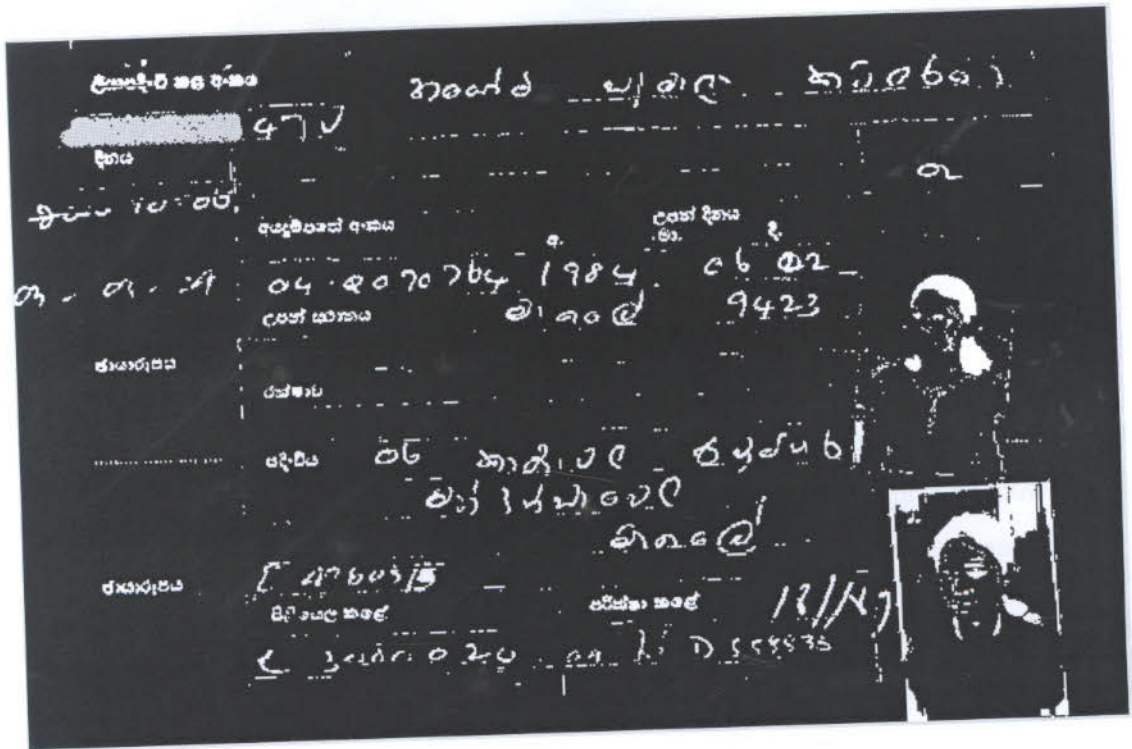
Problem1:

It was tried to remove the lines without creating the line image for subtraction. However, there were problems found such as the remaining sectored vertical lines after removal of horizontal lines, i.e., dash lines. Hence it was much difficult to locate the vertical line. Therefore it was decided to create a same sized line image and subtract from the original image.

Problem2:

It was also tried to use the Hough transform to locate the lines. Hough transform can locate the lines correctly and measure the angle of rotation also. After the rotation it is

difficult to do the subtraction and get the good result, due to the noise left in subtraction process.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk
Figure 3.7: Scanned image after the line removal



Figure 3.8: Name surroundings after the removal of lines

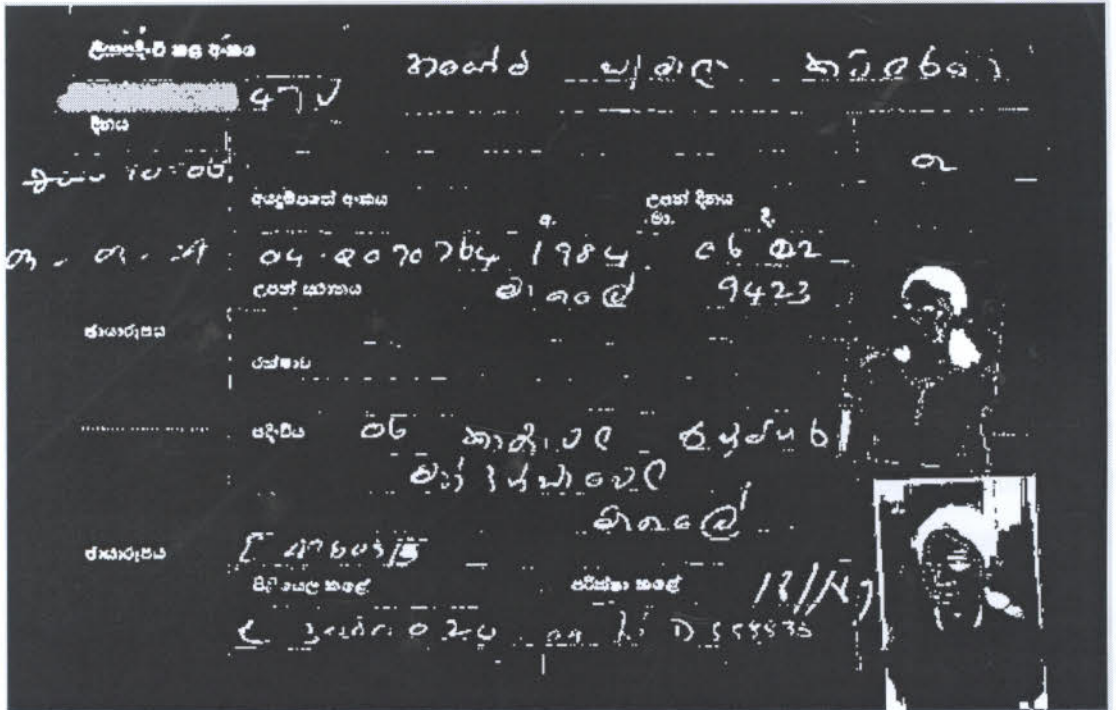
3.1.4 Closing:

The Closing operation is used to shape up the discontinuities in the characters in order to eliminate the possible errors. The figure 7 shows the image before closing operation while the figure 8 shows the same image after the closing is performed.



Figure 3.9: After the closing operation

difficult to do the subtraction and get the good result, due to the noise left in subtraction process.



University of Moratuwa, Sri Lanka.
 Figure 3.7: Scanned image after the line removal
www.lib.mrt.ac.lk

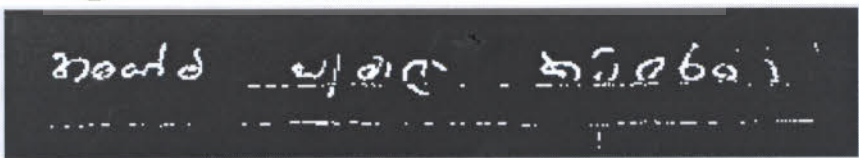


Figure 3.8: Name surroundings after the removal of lines

3.1.4 Closing:

The Closing operation is used to shape up the discontinuities in the characters in order to eliminate the possible errors. The figure 7 shows the image before closing operation while the figure 8 shows the same image after the closing is performed.

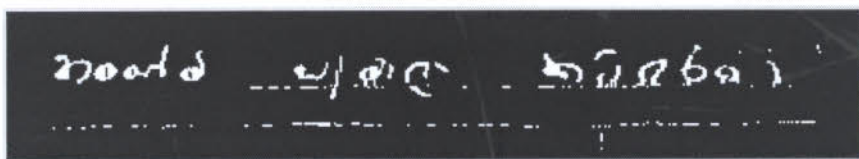


Figure 3.9: After the closing operation

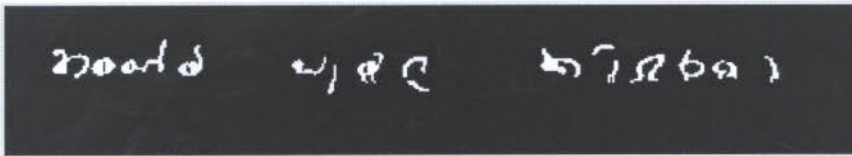


Figure 3.10: After the noise removal

3.1.5 Character counting:

The characters found in the black background in the above picture will be treated as objects and would be counted to get the number of characters in the name.

3.2 Finding images

Actually finding the exact card is impossible when the database is big, therefore the objective of this project is to narrow down the search. In this project number of characters in the name is chosen as input to find card or narrow down the search.

System will accept the number of characters in the name as the input and build a query to perform the search in the database. The finding will be the name of the scanned images which contains same number of characters as the given.

By using this name another search will be performed on the directory where the scanned images are stored. The result of the search will be displayed on the screen. If more than one card is found in the later search, then the user can traverse one by one pressing the next/previous buttons accordingly, and select the appropriate image among the images which also will be displayed accordingly during the traversal accordingly.

No delay/waiting is needed in search action. There are two reasons for this; one is very simple database and the data also was very small (images are not stored instead images' name and the number of characters are stored). Therefore system consumes only a small CPU time for the search. The other reason is no image processing task or any calculations happened in find mode.

4. Observations

It was observed that the implemented system correctly identifies the number of characters in the name with the 72% of accuracy. The failing scenario is due to the limitations in the handwritten names in the card such as the space between two characters, when the space between two characters is less than 2px size. Finding is working fine with zero errors.

We can conclude that the result of search functionality is 100% accurate provided that the addition of information to the database is accurate.



Figure 4.1: Processed segment of hand written characters



Figure 4.2: Processed segment of printed characters



5. Analysis and Discussion of Results

The result of the project is achieved with 80% accuracy of finding the number of characters of the name in the scanned card. The possibility of the error is due to the limitations in separating the hand written characters or merging the separated characters due to human error.

The processes which were applied to achieve the result and the limitations or the inaccuracy found during the process are discussed below.

There are two activities involving in this project. The first one is to insert the scanned cards to the system and the other one is to find the cards based on the requirement (e.g. number of characters in the name) using the stored index.

During the insertion activity, images are processed in many steps to find the number of characters in the name.

5.1 Binary image conversion:

Initially, the first option chosen was to convert the scanned cards into binary images. This option was tried out because image processing techniques are generally very efficient and powerful with the binary images. However, the outcome of that procedure was not acceptable. This is because when applying the threshold on the binary images many useful data also was lost with the noise while some of the noise was treated as useful data since the noise and useful data has the similar properties which make them very difficult to be distinguished.

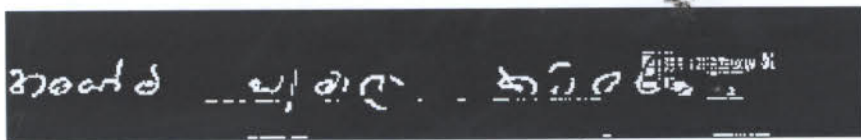


Figure 5.1: Result with binary image

As another option, the scanned images are converted to gray images and it was found the accuracy of the result was improved to an acceptable level. Unlike the direct conversion to binary images the gray scale images have many levels of values for the conversion before converting them into the binary images. Therefore the possibility of the noise removal was high.



Figure 5.2: Result with gray image

5.2 Closing:

Closing was tried with “disk” as morphological structuring element with radius 2. Since the space between the two characters are very small, this operation ended up in joining some adjoining characters in the character set as shown in the resulted image (Figure 12).



Figure 5.3 close with ‘disk’ (radius 2) morphological structuring element

Because of the limitations found with the morphological structuring element “disk”, the morphological structuring element “line” was tried with the length of 10 and angle of 90 degrees and it was also found that the resulting image wasn’t correct and mostly stacked up with the lines. That is because of the line length which is higher for the data we have.



Figure 5.4: close with ‘line’ (length 10, angle 90 degrees) morphological structuring element

With the trial and error attempts, it was found that length of 3 and angle of 90 degrees for morphological structuring element “line” was producing the result with expected level of accuracy of closing.



Figure 5.5: close with 'line' (length 3, angle 90 degrees) morphological structuring element

The limitation found in the observation is due to the space between two characters. This is due to the close operation performed in the image to increase the accuracy of the result which will shape up the discontinuities in the characters. It was also found that the possibilities of error were too high when the closing operation is not used and it was eliminated to the high extent when it is used.

5.3 Line removal:

Another important process involved is line removal; well known line detection operation Hough Transform was tried first. Even though the Hough transform detects the line, it treats the maximum possible length of the detected line as the real length of it. Therefore it gives problem when removing the line image from the original image i.e., the extended part of the detected line also will be removed therefore the useful information in the extended part of the line will be lost.

Therefore there is a new programming technique has originated to distinguish the lines in the image. This method is to traverse through the image horizontally as well as vertically to recognize the unbroken pixels of high intensity. The minimum value of intensity for this operation was identified as 220 through a numerous tries. When the smaller value of intensity is used, the noise added in the image is also detected as lines similarly when the higher value is used many lines were left undetected. The minimum unbroken length of pixels of high intensity was determined as 15 through number of trials. When smaller length is used, parts of certain hand written characters are detected as lines, likewise when higher value is used some lines are left undetected.

6. Conclusions and Recommendations for Future Research

The accuracy of this implemented system is found to be 72% based on the trials carried out with text images of hand written characters. Whereas it shows 100% accuracy for text images of printed characters.

When the test is being carried out with text images with printed characters, as there is no overlapping or splitting of characters in the image, the system identifies the number of characters accurately. However, due to the overlapping or splitting in the text images of hand written characters and the restrictions in handling such characters in the algorithm that is used to count the characters, the accuracy is limited.

The test has been carried out with a set of 100 text images of hand written characters and a number of characters of 72 of them were identified accurately. This test has been carried out repeatedly and same result was observed. As part of this trail a set of 25 text images of printed characters was used and number of characters of all of them was identified accurately.

In order to increase the accuracy, the algorithm which is used to identify the number of characters needs to be efficient enough to differentiate the characters from the adjacent characters, capable of recognizing the composite characters and splitting them and identifying the characters which are split and merge them.

This is where an investment of future research required in developing such an algorithm which has the above mentioned capabilities to recognize the hand written characters. Having an algorithm which is capable of identifying the characters 100% accurately might lead to further improvements to the result of this system as well.

7. References

- [1] Image Analysis and Mathematical Morphology: J. Serra
- [2] Erosion, dilation and related operators, by Mariusz Jankowski
- [3] Morphological Image Processing by Ranga Rodrigo
- [4] Application of Mathematical morphology operations for simplification and improvement of correlation of images in close range-photogrammetry by M. Kowalczyk, P. Koza, P. Kupidura, J. Marciniak
- [5] Morphological operations (Image processing Toolbox user's guide)
- [6] Document Text Extraction from Document Images Using Haar Discrete Wavelet Transform by S. Audithan and R. M. Chandrasekaran
- [7] Comparison for Edge Detection of Colony Images by Wang Luo
- [8] Computational methods for nonlinear image registration by Ulrich Clarenz, Marc Droske, Stefan Henn, Martin Rumpf, and Kristian Witsch
- [9] Gray Image Reconstruction by Waheeb Abu Ulbeh, Akram Moustafa, and Ziad A. Alqadi
- [10] Processing Of Polarametric SAR Images by Pamela A. Delaney
- [11] Mathematical Morphology and its Applications to Image Processing, by J. Serra and P. Soille
- [12] Image Corner Detection Using Hough Transform By Sung Kwan Kang, Young Chul Choung, and Jong An Park
- [13] A Fast Thresholded Linear Convolution Representation of Morphological Operations By Branislav KisaEanin and Dan Schonfeld
- [14] Use of the Hough transformation to detect lines and curves in pictures by Richard O. Duda Peter E. Hart
- [15] <http://www.mmorph.com/mmtutor1.0/html/mmtutor/mm030gray.html>

[16] Separating Low Pass and High Pass Frequencies in the Image Without Loosing Information by Mohammed Abu Zalata and Ziad Alqadi

[17] <http://www.mathworks.com/> - Image Processing Toolbox, Video and Image Processing Blockset, Database toolbox

[18] http://en.wikipedia.org/wiki/Main_Page - The free encyclopedia - Cross-correlation, Hough transform

[19] http://en.wikipedia.org/wiki/Morphological_image_processing Creating, compiling and linking MATLAB executables (MEX files) – By Peter Carbonetto

[20] Hough Transform for straight lines – By David Young



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

8. Annexure

personfinder.m

```
function varargout = personfinder(varargin)
% PERSONFINDER M-file for personfinder.fig
% DEMO, by itself, creates a new DEMO or raises the existing
% singleton*.
%
% H = DEMO returns the handle to a new DEMO or the handle to
% the existing singleton*.
%
% DEMO('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in PERSONFINDER.M with the given input
% arguments.
%
% PERSONFINDER('Property','Value',...) creates a new PERSONFINDER or raises
% the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before personfinder_OpeningFunction gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to personfinder_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help personfinder

% Last Modified by GUIDE v2.5 05-Jul-2007 12:28:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @personfinder_OpeningFcn, ...
    'gui_OutputFcn', @personfinder_OutputFcn, ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

end
% End initialization code - DO NOT EDIT

% --- Executes just before personfinder is made visible.
function personfinder_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to personfinder (see VARARGIN)

% Choose default command line output for personfinder
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes personfinder wait for user response (see UIRESUME)
% uiwait(handles.personfinder);

% --- Outputs from this function are returned to the command line.
function varargout = personfinder_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
im = imread('./disny.jpg');
image(im);
axis off

% -----
function File_Callback(hObject, eventdata, handles)

% -----
function Add_Callback(hObject, eventdata, handles)
add;

% -----
function Find_Callback(hObject, eventdata, handles)
delete(handles.txtName)
delete(handles.txtPerFinder)
delete(handles.txtDpt)
set(handles.txtNoChar, 'Visible', 'on');
set(handles.edtNoChar, 'Visible', 'on');
set(handles.pbFind, 'Visible', 'on');
set(handles.popFileName, 'Visible', 'on');
set(handles.pbNext, 'Visible', 'on');
set(handles.pbPrev, 'Visible', 'on');
set(handles.Find, 'Enable', 'off');
% -----

```




```

function Exit_Callback(hObject, eventdata, handles)
selection = questdlg(['Close ' get(handles.personfinder,'Name') '?'],...
    ['Close ' get(handles.personfinder,'Name') '...'],...
    'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.personfinder)

% -----
function Help_Callback(hObject, eventdata, handles)
% hObject   handle to Help (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

%--- Executes during object creation, after setting all properties.
function edtNoChar_CreateFcn(hObject, eventdata, handles)
% hObject   handle to edtNoChar (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edtNoChar_Callback(hObject, eventdata, handles)

% --- Executes on button press in pbFind.
function pbFind_Callback(hObject, eventdata, handles)
%x = str2num(get(handles.edtNoChar,'String'));
global file_count
file_count = 1;
set(handles.popFileName,'Value',file_count);
set(handles.pbNext,'Enable','on');
set(handles.pbPrev,'Enable','on');
x = get(handles.edtNoChar,'String');
conn = database('PersonDetails', '', '');
stm = strcat('select FileName from Details where NoOfChar = ',x);
curs = exec(conn, stm);
curs = fetch(curs);
global file_list;
file_list = curs.Data;
set(handles.popFileName,'String',file_list);
if (strcmp(file_list , 'No Data') ~= 1)

```

```

file_name = strcat('./Images/',file_list(file_count));
file_name = char(file_name);
image(imread(file_name));
axis off
else
im = imread('./disny.jpg');
image(im);
axis off
end

```

```

% --- Executes during object creation, after setting all properties.
function popFileName_CreateFcn(hObject, eventdata, handles)
% hObject handle to popFileName (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.

```

```

if ispc
set(hObject,'BackgroundColor','white');
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function popFileName_Callback(hObject, eventdata, handles)
global file_list;
file_name = strcat('./Images/',file_list(get(handles.popFileName,'Value')));
file_name = char(file_name);
image(imread(file_name));
axis off

```

```

% --- Executes on button press in pbNext.
function pbNext_Callback(hObject, eventdata, handles)
global file_count
len = size(get(handles.popFileName,'String'));
if file_count < len(1)
file_count = file_count + 1;
set(handles.popFileName,'Value',file_count);
set(handles.pbPrev,'Enable','on');
global file_list;
file_name = strcat('./Images/',file_list(file_count));
file_name = char(file_name);
image(imread(file_name));
axis off
else
set(handles.pbNext,'Enable','off');
end

```

```

% --- Executes on button press in pbPrev.
function pbPrev_Callback(hObject, eventdata, handles)

```



```

global file_count
if file_count > 1
    file_count = file_count - 1;
    set(handles.popFileName,'Value',file_count);
    set(handles.pbNext,'Enable','on');
    global file_list;
    file_name = strcat('./Images/',file_list(file_count));
    file_name = char(file_name);
    image(imread(file_name));
    axis off
else
    set(handles.pbPrev,'Enable','off');
end

```

autoposition.m

```

function [im1, im2, varargout] = autoposition(im1, im2)

```

```

[y x z] = size(im2);
[y1 x1 z1] = size(im1);

```

```

c = normxcorr2(im2,im1);

```

```

% offset found by correlation
[max_c, imax] = max(abs(c(:)));
[ypeak, xpeak] = ind2sub(size(c),imax(1));
corr_offset = [(xpeak-size(im2,2)) (ypeak-size(im2,1))];

```

```

offset = corr_offset;
xoffset = offset(1);
yoffset = offset(2);

```

```

% Images begin at (1,1), so add one to the offset values for coordinates
xbegin = xoffset + 1;
xend = xoffset + size(im2,2);
ybegin = yoffset + 1;
yend = yoffset + size(im2,1);

```

```

im1 = im1(ybegin:yend,xbegin:xend,:);

```

```

if nargout > 2
    varargout{1} = offset;
end

```

find_no_of_char.m

```

function[n] = find_no_of_char(img_bw)
roi = [226,14,442,77];

```

```

f2=imcrop(img_bw,roi);
se1 = strel('line',3,90);
f2 = imclose(f2,se1);
[labeled,numObj] = bwlabeln(f2,8);
obj_data = regionprops(labeled,'basic');
all_obj = [obj_data.Area];
small_obj = length(find(all_obj < 20));
n = numObj - small_obj;

```

insert.m

```

function [] = insert(no_of_char,file_name)
conn = database('PersonDetails', '', '');
exdata = {no_of_char,file_name};
colnames = {'NoOfChar', 'FileName'};
insert(conn, 'Details', colnames, exdata);

```

remove_lines_gray.m

```

function [im_temp] = remove_lines_gray(im)

```

```

size_im = size(im);
im_temp = ones(size_im);
for i = 1 : size_im(2)
    b_count = 0;
    for j = 1 : size_im(1)
        if(im(j,i)< 220)
            b_count = b_count + 1;
        else
            if(b_count > 15)
                for k = j-b_count : j-1
                    im_temp(k,i) = im(k,i);
                end
            end
            b_count = 0;
        end
    end
end
end

```

```

for i = 1 : size_im(1)
    b_count = 0;
    for j = 1 : size_im(2)
        if(im(i,j)< 220)
            b_count = b_count + 1;
        else
            if(b_count > 15)

```



```

        for k = j-b_count : j-1
            im_temp(i,k) = im(i,k);
        end
    end
    b_count = 0;
end
end
end
end

```

remove_printed_characters.m

```
function [image] = remove_printed_characters(image,im_temp,roi)
```

```
im_temp = imcrop(im_temp,roi);
```

```
[im3 im4 offset] = autoposition(image, im_temp);
```

```
im5 = imabsdiff(im4,im3);
```

```
xoffset = offset(1);
```

```
yoffset = offset(2);
```

```
% Images begin at (1,1), so add one to the offset values for coordinates
```

```
xbegin = xoffset + 1;
```

```
xend = xoffset + size(im4,2);
```

```
ybegin = yoffset + 1;
```

```
yend = yoffset + size(im4,1);
```

```
for i = 1:size(im4,1)
```

```
    for j = 1:size(im4,2)
```

```
        if (im5(i,j) < 65)
```

```
            im5(i,j) = 250;
```

```
        end
```

```
    end
```

```
end
```

```
image(ybegin:yend,xbegin:xend,:) = im5;
```

add.m

```
function[] = add()
```

```
[fileName,pathName,FilterIndex] = uigetfile('* .jpg','Select file','MultiSelect', 'on');
```

```
if FilterIndex == 1
```

```
    fileNameChar = char(fileName);
```

```
    load tempelate_image_data.mat;
```

```
    im_temp = rgb2gray(im_temp);
```

```
    if class(fileName) == 'char'
```



```

file_size = 1;
else
file_size = length(fileName);
end
hbar = waitbar(0,'Addition in progress – Please wait...');
for l = 1 : file_size
img = imread([pathname fileNameChar(l,:)]);
img = rgb2gray(img);

roi = [577,29,67,16];
img = remove_printed_characters(img,im_temp,roi);

roi = [232,30,25,11];
img = remove_printed_characters(img,im_temp,roi);

roi = [556,74,21,15];
img = remove_printed_characters(img,im_temp,roi);

roi = [609,74,28,15];
img = remove_printed_characters(img,im_temp,roi);

im_gray = uint8(remove_lines_gray(img));
im = ~imabsdiff(im2bw(img,.85),im2bw(im_gray,.5));
n = find_no_of_char(im);
insert(n,fileNameChar(l,:));
waitbar(1/file_size);
end
close(hbar);
end

```

