VISION BASED HANDWRITTEN CHARACTER RECOGNITION

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Özcan Öksüz

September, 2003

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Uğur Güdükbay(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Enis Çetin

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

VISION BASED HANDWRITTEN CHARACTER RECOGNITION

Özcan Öksüz
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. Uğur Güdükbay
September, 2003

Online automatic recognition of handwritten text has been an ongoing research problem for four decades. It is used in automated postal address and ZIP code and form reading, data acquisition in bank checks, processing of archived institutional records, automatic validation of passports, etc. It has been gaining more interest lately due to the increasing popularity of handheld computers, digital notebooks and advanced cellular phones. Traditionally, human-machine communication has been based on keyboard and pointing devices. Online handwriting recognition promises to provide a dynamic means of communication with computers through a pen like stylus, not just an ordinary keyboard. This seems to be a more natural way of entering data into computers.

In this thesis, we develop a character recognition system that combines the advantage of both on-line and off-line systems. Using an USB CCD Camera, positions of the pen-tip between frames are detected as they are written on a sheet of regular paper. Then, these positions are used for calculation of directional information. Finally, handwritten character is characterized by a sequence of writing directions between consecutive frames. The directional information of the pen movement points is used for character pre-classification and positional information is used for fine classification. After characters are recognized they are passed to LaTeX code generation subroutine. Supported LaTeX environments are array construction, citation, section, itemization, equation, verbatim and normal text environments. During experiments a recognition rate of 90% was achieved. The main recognition errors were due to the abnormal writing and ambiguity among similar shaped characters.

Keywords: pattern recognition, character Recognition, on-line recognition systems, LaTeX.

ÖZET

GÖRÜŞ TABANLI ELYAZISI HARF TANINMASI

Özcan Öksüz
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Asst. Prof. Dr. Uğur Güdükbay
Eylül, 2003

Etkileşimli otomatik el yazısı tanınması son 40 yıldır araştırma konusu olmuştur. Otomatik posta adresi ve ZIP kodu okunmasında, formlara girilen bilgilerin okunmasında, banka çeklerinden bilgi alınmasında, kurumsal arşivlerin işlenmesinde, otomatik olarak pasaportların denetlenmesinde vb kullanılmıştır. Avuçiçi bilgisayarların, sayısal taşınır bilgisayarların ve gelişmiş cep telefonlarının popüler olmasından dolayı son zamanlarda çok ilgi görmüştür. Geleneksel olarak insan ve bilgisayar iletişimi klavye ve fare ile saglanmaktadır. Kalem benzeri aletle etkileşimli elyazısı tanıma bilgisayarla, klavye dışında dinamik iletişim kurma yolları sunar. Bu bilgisayara veri girişinin daha doğal yolla yapılmasını sağlamaktadır.

Bu tezde, hem etkileşimli, hem de etkileşimsiz sistemlerin avantajını kullanan bir karakter tanıma sistemi sunulmuştur. Standart video kamera kullanılarak, karakterler kağıt üzerine yazılırken, kalemin uç noktası bulunup kaydedilmiştir. Sonra kaydedilen koordinatlar yönlerin hesaplanmasında kullanılmıştır. Sonunda, elyazısı karakter bir dizi yazı yönünün birbiri ardısıra görüntülerde degişmesiyle bulunmuştur. Kalem hareketinin yön bilgisi karakterin temel sınıflandırılmasında, pozisyonu ise karakterin bulunmasında kullanılmıştır. Karakterler bulunduktan sonra, LaTeX kodu hazırlama metoduna yollanmıştır. Desteklenen LaTeX çevrebirimleri, dizi hazırlanması, referans, bölüm, listeleme, formül, harfi harfine ve normal yazıdır. Deneylerde harflerin 90% oranında dogru tanıma yüzdesine erişilmiştir. Temel tanıma hataları düzensiz yazımdan ve benzer şekilli harflerin belirsizliğinden kaynaklanmaktadır.

Anahtar sözcükler: desen tanıma, karakter tanıma, etkileşimli tanıma sistemleri, LaTeX.

Acknowledgement

I would like to express my thanks and gratitude to Asst. Prof. Dr. Uğur Güdükbay and Prof. Dr. Enis Çetin for their help. Without their invaluable support this work could not have been possible.

I also would like to thank Prof. Dr. Özgür Ulusoy who was kind enough to read my thesis.

Finally I would like to thank to my wife and my family. Without their support and love, I could never taste any achievement in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online automatic recognition of handwritten text has been an ongoing research problem for four decades. It has been gaining more interest lately due to the increasing popularity of handheld computers, digital notebooks and advanced cellular phones. This seems to be a more natural way of entering data to the computers. Traditionally, manmachine communication has been based on keyboard and pointing devices. These methods can be very inconvenient when the machine is only slightly bigger or the same size as the human palm. An ordinary keyboard is very difficult to integrate in small devices and it usually determines the size of the whole apparatus. This is especially true when the number of the characters is very large as in Chinese and Japanese. A pointing device, for example a track ball or a pen, is insufficient or very slow when used alone in applications in which textual input is also desired. Online handwriting recognition promises to provide a dynamic means of communication with computers through a pen like stylus, not just a keyboard. This seems to be a more natural way of entering data into computers.

Because of these problems, new methods for input have been developed, for example systems that recognize speech and handwriting. Because both are very natural ways to communicate, people can easily learn to use them. Unfortunately, these recognition tasks are not that easy for computers whose artificial intelligence is different from that of humans. Neither one of the recognition problems have been completely solved yet. Very high accuracy is needed for such system before they can be commonly accepted. Handwriting recognition is the more attractive input method, especially in noisy environment and when privacy is needed.

Character and handwriting recognition has a great potential in data and word processing for instance, automated postal address and ZIP code reading, reading data entered in forms, data acquisition in bank checks, processing of archived institutional records, automatic validation of passports, etc.

The process of handwriting recognition involves extraction of some defined characteristics called features to classify an unknown character into one of the known classes. And at first sight, handwriting recognition does not appear to be a difficult problem. A recognition system should just choose the correct character, usually the one that most resembles the written one, from a limited set of characters.

In this thesis, we present an approach to on-line handwritten alphanumeric character recognition based on sequential features. In particular, positions of the pen-tip between frames are detected. Then these positions are used for calculation of directional information. Finally, handwritten character is characterized by a sequence of writing directions between consecutive frames. The directional information of the pen movement points is used for character preclassification and positional information is used for fine classification. After characters are found, corresponding LaTeX codes are generated.

1.1 Aims and overview of this thesis

The aims of this thesis are to discuss the essential background knowledge of the research field of handwriting recognition, make a literature survey of the most important and recent recognition methods, implement one of the stateofart methods in practice and produce resulting LaTeX codes.

The theoretical part of this work, presented in Chapter 2, mainly consists of the literature survey. The practical part of this thesis consists of creating a learning online recognition system by using vision based handwritten character recognition method is described in Chapter 3. Data structures, algorithms, user interface and LaTeX code generation are described in Chapter 4.

Some example of characters written and their LaTeX code describing the salient features of the character recognition system are given in Chapter 5. The major conclusion drawn from the results are collected in Chapter 6.

Chapter 2

Recognition of handwriting

Handwriting recognition systems can be subdivided into many different categories. Most important discriminating feature of the systems is the time when the text is recognized. Recognition can be done either offline or online. In addition, the different sequences of text which form the unit to be recognized as a single entity discriminate problem types: the text can be considered as a sequence of characters, words or whole sentences. The set of characters, writing style, and constraints on the writing also have a great influence on recognition methods. These distinctions are described in more detail in the following sections.

2.1 Off-line recognition

The handwriting recognition task heavily depends on the application. First applications were systems that recognized text that had originally been written on paper. In such systems, paper sheets are digitized into twodimensional images. Features for recognition are first enhanced and then extracted from the bitmap images by means of digital image processing. This type of recognition is called offline recognition as the text is not recognized at the same time as it is produced but after the writing task is completed. Offline handwriting recognition is a subset of optical character recognition (OCR) [17].

Offline methods are not suitable for manmachine communication because they do not facilitate realtime interactivity. They are suitable for automatic conversion of paper documents to electronic documents which then may be interpreted or postprocessed by computers. Typical applications for optical character recognition are systems which need to automatically handle a huge amount of information in paper form. Such recognition systems have been developed for numerous application areas but are mainly used for machineprinted text. Applications for handwritten text include, to name a few: automatic sorting of mail, handling of financial documents such as checks, writer recognition, signature verification, and shorthand transcription [5].

2.2 On-line recognition

In the case of online recognition, handwriting is collected and recognized in real time, i.e. at the same time it is produced. The writing medium is usually a tablet or a flat display which can capture information on the location and motion of a pen-like pointing device moving on the writing surface, are frequently sampled and sent to the recognition system. The use of a pressure sensitive switch on the pen tip indicates pen-up/pen-down status and disambiguates stroke segmentations. Location and possibly also pressure data are frequently sent to the computer which performs the recognition task. The applications of on-line handwriting recognition include various kinds of interactive user-interfaces in which a method for textual input is needed but a keyboard would not be a practical solution, for example as in the case hand-held computers.

The main applications for online character recognition allow only a limited interaction as they are typically just for filling out forms. In more advanced applications, a pointing pen and a handwriting recognition system can replace the mouse and keyboard by emulation. Such systems usually have a special tablet for writing and the recognition is carried out by the tablet's own microprocessor as the algorithms are very dependent on such things as the sampling rate and resolution. The recognition result is sent to the main application whose software needs only minor modifications if any [17].

The main advantage of on-line handwriting data over off-line is the dynamic information on writing process: the writing speed, acceleration, angle of the pen and its pressure against the writing surface are available for the system. Off-line data is just static images of handwriting. The image pixels do not contain any information on the writing direction or the writing order of the strokes. The values of pixels tell only if the pen point has ever visited the locations the pixels correspond to. It is not always clear which pixels belongs to which strokes, or even what is the number of strokes. The quality of off-line data depends heavily on how well the pen trace the image can be segmented from the background. In addition, too thick or smudged pen trace cannot capture small details of characters. In the case of off-line data, the pen trace is captured together with an image of the paper it has been written on. Thus, special image processing algorithms are needed for removing the background information and enhancing the actual handwriting information. The on-line data requires less memory resources than off-line data as only the coordinates of the sampled pen point positions and possibly some other features are stored. Another advantage of on-line recognition over off-line recognition is that there is close interaction between the user and the machine. The user can thus correct any recognition error immediately when it occurs.

2.3 On-line handwriting data acquisition and processing

The following sections will discuss the acquisition of on-line handwriting data and its different processing stages performed prior to the actual recognition stage.

2.3.1 Raw data

A typical format of on-line handwriting data is a sequence of coordinate points of the moving pen point. In addition to location, the pressure between the pen point and writing surface can be measured. Sometimes, not only the events when the pen is actually touching the writing surface are detected and recorded but handwriting data is collected also when the pen point is hovering just above the writing surface. Connected parts of the pen trace in which the pen point is touching the writing surface, or the pressure between them exceeds some threshold value, are called *strokes*.

The pen trace is usually sampled with a constant rate and thus data points are evenly distributed in time but not in space. When the speed of writing is slow, the sample points are located densely on the true pen trace, whereas quick writing produces more sparsely located points. Typically, writing speed slows down in sharp corners, in other points of extremal curvature, at the beginning and at the end of the stroke, but also by hesitation and pausing of the writer. Sampling rate and resolution should be so high that the sampled data points represent the true pen trace faithfully. If sampling rate is too low, odd corners will be introduced on the sampled pen trace and some of the real corners can be missed.

2.3.2 Preprocessing and normalization

The purpose of preprocessing and normalization methods is to simplify the recognition task by reducing the amount of information, eliminating imperfections, and removing uninformative variations in handwriting data.

On-line handwriting data collected with a high sampling rate contains redundant information as the pen trace could be represented well enough with fewer data points. Therefore, the data points are usually either *down-sampled* or *resampled*. Down-sampling means that only some of the original data points are kept while the rest are abandoned. In linear down-sampling, the data points to be kept are selected in a linear manner from the data point sequence. For example, only $\mathbf{0}$th, $\mathbf{n}$th, $\mathbf{2n}$th, ... points can be kept in a data point sequence. Nonlinear down-sampling methods select the data points in a nonlinear instead. For example, the criterion for keeping a data point can be a minimum distance along the pen trace to the previous preserved point. Resampling means that new data points are calculated on the basis of the original ones. For example, new equidistant data points can be obtained by interpolation from temporally equidistant original data points. equidistant data points can be obtained by interpolation from temporally equidistant original data points. Resampling and down-sampling may destroy or severely distort dynamic information in the original data points. Therefore, all dynamic features should be evaluated prior to such preprocessing.

Handwritten character samples must be normalized in respect to the variations in their size and location. Otherwise, they cannot be reasonably compared with each other and would not be compatible with the recognition system. Size normalization of isolated characters is typically just a linear scaling to a standard height preserving the aspect rations of the characters. Translation variations are normalized by moving the centers of the characters into the origin of the coordinate system. A character center can be defined to be the center of mass of the data points of the center point of the smallest box drawn around the character.

2.3.3 Features and representations

The recognition of handwritten patterns is based on the features extracted from the pre-processed and normalized raw data. The features are selected so that they represent the handwriting well and emphasize the inner-class differences and intra-class similarities.

Typically on-line handwriting data is represented by time series, i.e. sequences of feature vectors corresponding to the data points of the sampled pen trace. In the simplest case, the only features are the x- and y-coordinates of the sampled pen point positions.

On-line handwriting data patterns can also be represented by simple feature vectors, symbol strings, or graph structures. The components of feature vectors can either indicate the existence or absence of certain features or give measured values for them. These features can be as simple as the number of strokes and the rough orientations and the lengths of the strokes, or more complicated, such as the existence of loops, and different kinds of junctions of strokes.

2.4 Character sets

The character set which forms the text to be recognized affects crucially the recognition algorithms used. The recognition task is naturally easier when the number of characters is small and they are dissimilar. Recognition systems have been developed for numerous languages and character sets: Roman alphabets, Arabic numerals, Japanese, Chinese, Indian, Arabic, Korean, Cyrillic, Hebrew, Thai, Greek, Berber, and shorthand writing [5]. The most difficult characters are Chinese due to the enormous number of them over 50 000. Chinese characters can also be written in two different styles, block or cursive.

Other important characteristics of the character sets are the typical number of strokes in one character and the variety of the stroke forms. The Chinese characters have an average of 810 strokes in block style. It is considerably more than the average of about 2 strokes in capital Roman alphabets [17]. In Japanese hiragana, almost all strokes are curved whereas the strokes in the Chinese characters mainly consist of straight lines and only a few corners [20].

2.5 Writing style variations

Handwritten characters have enormous variety in shape compared to machine printed characters. Variations occur mostly between writers but also with one writer. Handwritten characters may be regarded as distorted versions of idealized character models and the distortion can be interpreted to be caused by several factors.

2.5.1 Variations of characters

A writing style is based on the alignment and variable forms of characters. Handwritten characters can vary in both their static and dynamic properties [17]. Static properties are the underlying, ideal model of the character and the geometrical properties such as relative position and size of the strokes, corners and retraces, ornamentals, aspect ratio, size, and slant. Dynamic properties are more involved with the generative aspects of the characters. Characters can look similar although their number of strokes, stroke order and direction may vary considerably. Because of these numerous variations, the number of styles to write a certain character can be considered as infinite.

2.5.2 Material factors

The writing instrument, surface, and form constitute the material factors of the writing style. The writing instrument has and effect on the writing style depending on its size and overall comfort. Writing surface's friction and position also affect the style. The form factors, such as the size of the blank writing area, the length of the writing line, or the size of the writing boxes for characters, can have a dramatic effect on the handwriting style.

2.5.3 Constraints on writing

As the variations are the key problem of automated handwriting recognition, writing style is usually more or less constrained in such systems. Usually, characters are written in boxes to ease their separation, or the system guiding lines to help the user write more consistently. These constraints are designed to improve the performance of the recognizer and they can be seen as mediate a *priori* desicion rules. If the constraints are too strict, they may cause the system to be rejected by the users. Constraints do affect the style and speed of writing and recognition results.

The user effort to learn printing with a constrained character set and will be minimal if the character set is designed to be as natural as possible. Once the user has learned the character set, the payback on input efficiency is very large. An example of a constrained writing style is *Graffiti characters* which are shown in Figure 2.1.

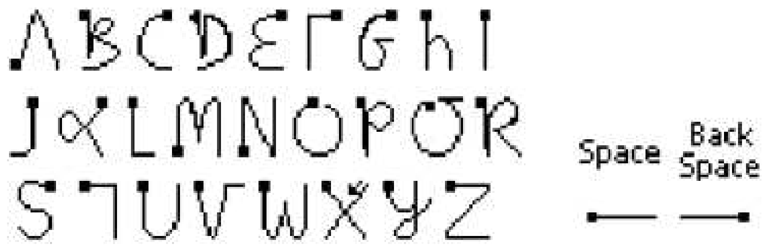


Figure 2.1: Graffiti characters

2.5.4 Recognition of Letters vs. Words

The recognition of cursive writing is almost impossible without the use of vectorized input, and it is easier to recognize whole words than single characters, because the difficulty in separating the characters is less. The recognition of words assumes the existence of a dictionary, because only words can be recognized that are known. Exactly this limits the usability of dictionaries: they are inevitable, but there will never be a single dictionary containing all the words that a person would want to write.

2.6 Recognition methods

The problem of online handwriting recognition can be defined in various ways. Variables in the problem definition include character set, writing style, and desired recognition rate. In general, each problem definition lends itself to different algorithmic approaches, which in turn make use of different features for classification.

Two essential components in a character recognition algorithm are the feature extractor and the classifier. Feature analysis determines the descriptors, or feature set, used to describe all characters. Given a character image, the feature extractor derives the features that the character possesses. The derived features are then used as input to the character classifier. The problems involved in this process are: 1) finding both descriptive and discriminating features, 2) selecting a way to compare them, and 3) creating the classification rules. Of course, these problems are highly dependent on each other.

Recognition methods can be roughly classified into four major groups: statistical, structural and syntactical, model matching, and neural network methods. Sometimes, different methods are combined, for example, simple methods are used for preclassification and final decision is made with more sophisticated methods.

In statistical methods, the recognition of a character is carried out by choosing the character set which is most probable or has the minimum measure of expected classification error or some other risk.

A simple statistical method for on-line character recognition is described by Odaka et al. (1982). Isolated characters are preclassified according to the number of strokes. Each stroke is then approximated with a small number, N, of points which divide the stroke into straight segments of equal length. The reference patterns consist of the mean vectors and covariance matrices of the strokes. Final classification is carried out by minimizing the distance between the feature vector of the unknown character and reference patterns.

Template matching, or model matching, is one of the most common classification methods. In template matching, individual image pixels are used as features. Classification is performed by comparing an input character image with a set of templates (or prototypes) from each character class. Each comparison results in a similarity measure between the input character and the template. One measure increases the amount of similarity when a pixel in the observed character is identical to the same pixel in the template image. If the pixels differ the measure of similarity may be decreased. After all templates have been compared with the observed character image, the character's identity is assigned as the identity of the most similar template.

Structural classification methods utilize structural features and decision rules to classify characters. Structural features may be defined in terms of character strokes, character holes, or other character attributes such as concavities. For instance, the letter P may be described as a vertical stroke with a hole attached on the upper right side. For a character image input, the structural features are extracted and a rule-based system is applied to classify the character. Structural methods are also trainable but construction of a good feature set and a good rule-base can be time-consuming.

Recognition systems based on neural networks (NN) use a similar basic idea as those based on model matching methods. Instead of explicitly storing models in a model base, the information on the models is stored implicitly in the weights of the neurons. The input of the network consists of the features extracted from the patterns to be recognized and the output can be the scoring values for each class or simply the index of the best-matching class. Several NN-based applications and methods for handwriting recognition are described in a journal issue edited by Guyon and Wang (1993). In most cases, NNs are used for optical character recognition and the input consists of preprocessed images or different features which are extracted from them. Due to the black-box nature of the some neural-network models, it is difficult to analyze their recognition process and increase the recognition performance by finding and eliminating the error sources. In addition, they need relatively large training databases.

Chapter 3

Vision Based Handwritten Character Recognition

In this thesis, we used a system that combines the advantages of both on-line and off-line approaches. Since writing on paper is the most natural way for handwriting, we allow users to write on any regular paper just like using the off-line system. However, instead of waiting for a whole page to be written and then scanned into a computer, we use an USB CCD Camera attached on the computer to capture a sequence of the character images while it is being written on the paper. Then using a set of video processing techniques, we extract character strokes from the video sequence. Unlike the on-line system writing board that has to be purchased separately, most computers nowadays have a small video camera attached for net meeting type of application, therefore no additional special equipment is required for our system.

The image sequences are captured using an USB CCD Camera. We obtain 6 frames per second with 320 x 240 pixels. The camera is positioned approximately 50 cm above the writing surface and for good illumination we placed two lamps to left and right side of the camera, Figure 3.1. The writing space visible in the video images covers an area of approximately 20 x 12 cm.

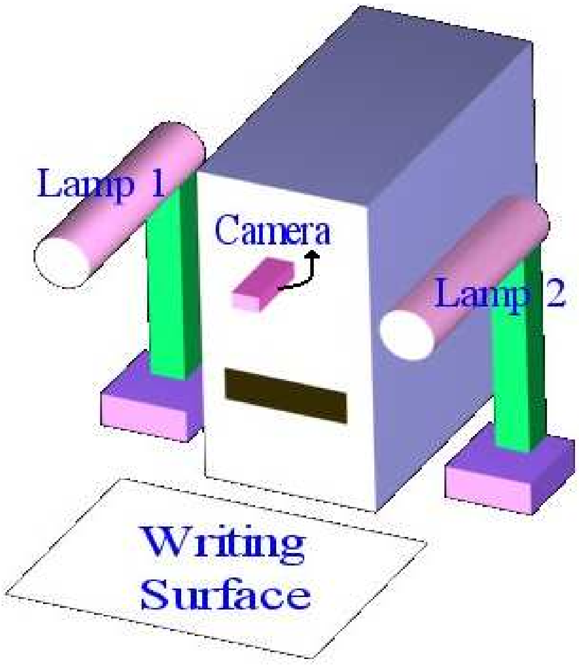


Figure 3.1: Camera and Lambs

3.1 Pen Tracking

In our handwriting character recognition system, we used an ordinary Board Marker having black color. In order to trace the pen movements easily, a blue band is attached near its tip Figure 3.2 and at each frame the position of this blue band is recorded.

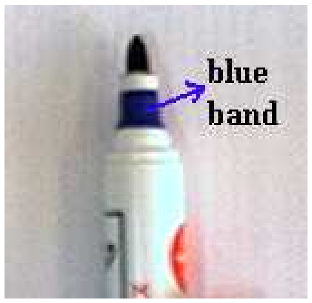


Figure 3.2: Boardmarker and blue band attached to it

3.2 Foreground Subtraction

For detection and removal of hand and pen from the video we used the method described in [4]. This method was a background subtraction algorithm used for detection of moving parts (car, human, bear etc.) and removal of background from image using a dynamically updating background model. As we are interested in what is written on paper, we used the method for detection of moving parts (hand and pen) and removal of them from the image. Following paragraphs describe the algorithm in detail.

The background model $B_n(x)$ is initialized by setting $B_0 = I_0$. After this, for each frame a binary motion mask image $M_n(x)$ is generated containing all moving pixels

$$M_n(x) = \begin{cases} 1, & |I_n(x) - I_{n-1}(x)| > T_{n-1} \ and \ |I_n(x) - I_{n-2}(x)| > T_{n-1} \\ 0, & otherwise \end{cases} \tag{3.1}$$

Where T is an appropriate threshold. After this, non moving pixels are updated using an IIR filter to reflect changes in scene (such as illumination)

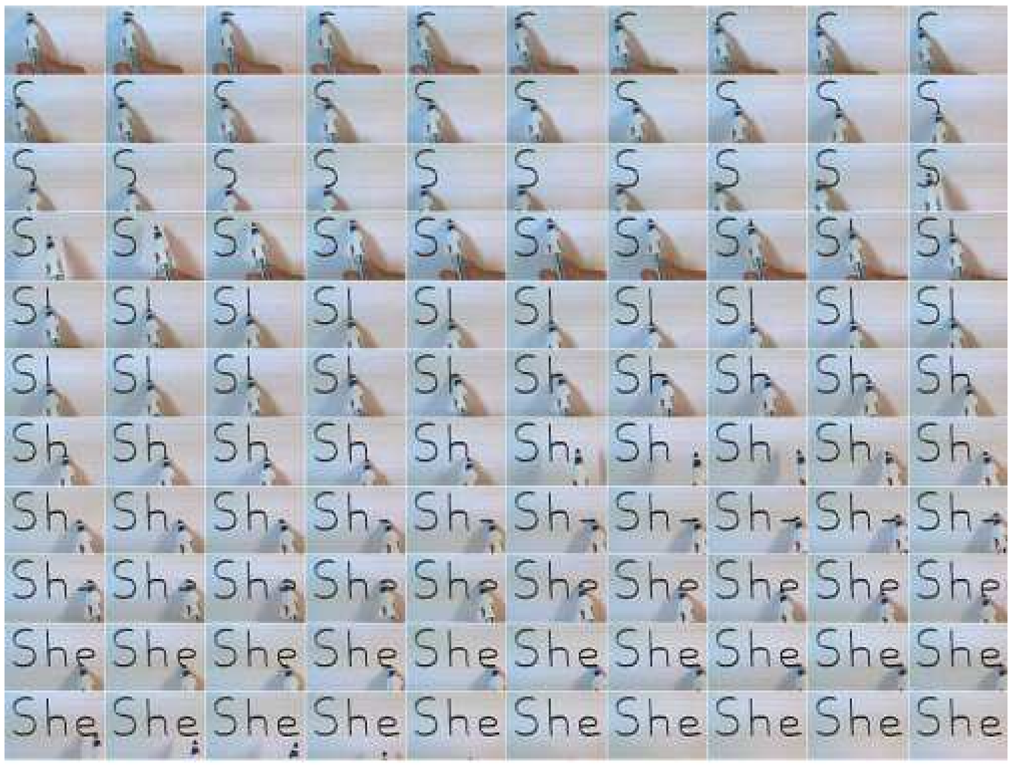
$$B_n(x) = \begin{cases} B_{n-1}(x), & M_n(x) = 1 \\ \alpha B_{n-1}(x) + (1\text{-}\alpha) I_n(x), & M_n(x) = 0 \end{cases} \tag{3.2}$$

where α is the filter's time constant parameter.

Then the threshold T is updated.

$$T_n(x) = \begin{cases} T_{n-1}(x), & M_n(x) = 1 \\ \alpha T_{n-1}(x) + 5 * (1\text{-}\alpha)|I_n(x) - B_n(x)|, & M_n(x) = 0 \end{cases} \tag{3.3}$$

where α is again the filter's time constant parameter. The dynamic background model contains the most recent background image information *for every pixel* - even the ones currently occluded. Consequently, removing moving parts from the video stream is easily achieved by simply replacing their pixels with the corresponding background pixels from B_n. Finally image is converted to black and white by using an appropriate threshold value. An example of the method used is shown in Figure 3.3.


(a)

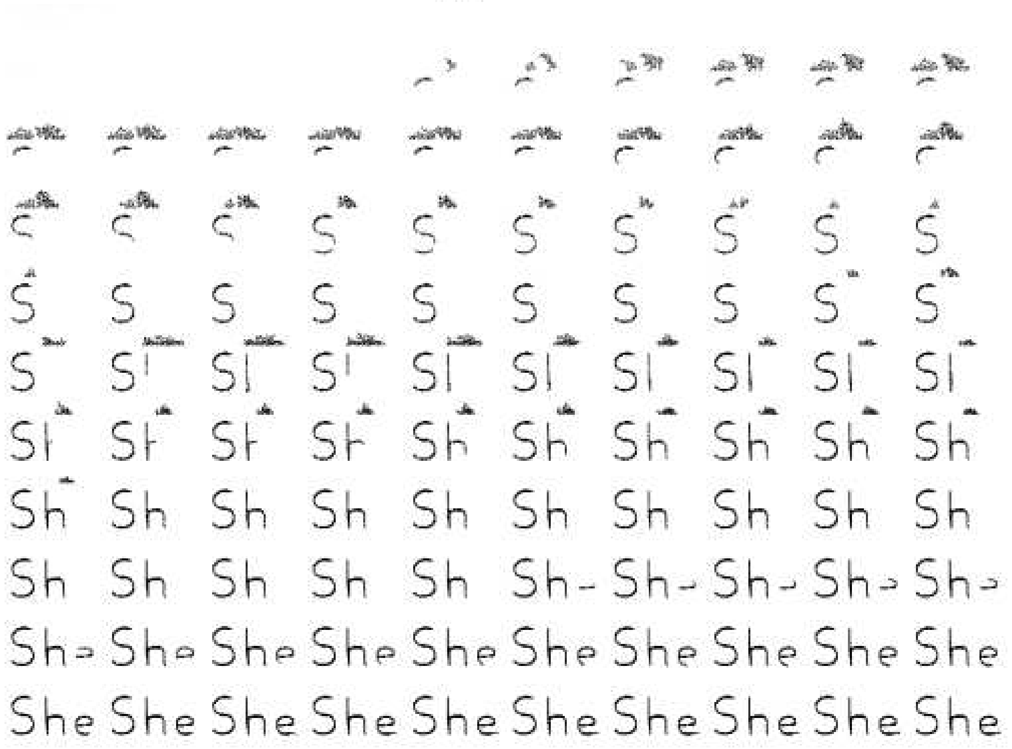

(b)

Figure 3.3: Foreground subtraction algorithm at work :
(a) sequence of images obtained while writing word **"She"** (b) corresponding foreground subtracted images

3.3 Reference Lines

In this recognition system we used a white paper which has 4 parallel dashed reference lines. And each reference line starts with a small red rectangle as shown in Figure 3.4. The aim of using red blocks is for detection of reference line start points in the video.

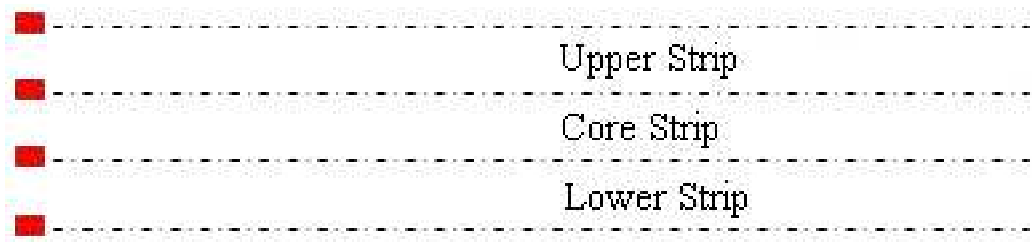


Figure 3.4: Reference lines on paper

Reference lines are used for two reasons

1. force the user write properly on paper so that character recognition rates are increased
2. preliminary classification of letters are performed according to position of letters and reference lines

3.4 Preliminary Classification

The aim of the preliminary classification is to reduce the number of possible candidates for an unknown character, to a subset of the total character set. For this purpose, the selected domain is categorized into five groups as in Figure 3.5. The classification is based on the relative heights of each character in the three-strip frame determined by the four reference lines.

Table 3.1 lists all the characters under consideration, classified into the above five preclassification groups.

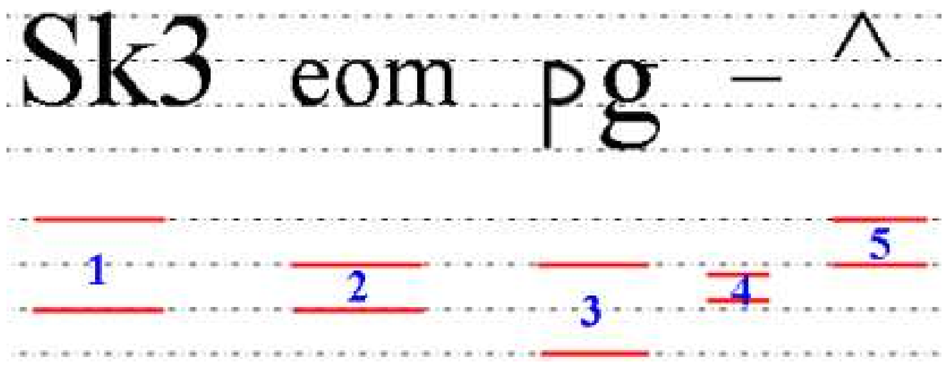


Figure 3.5: Five pre-classification groups

Group	Characters of the group
1	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, b, d, f, h, k, l, t, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (,), [,], {, }, $\surd$, $\sum$, $\prod$, $\int$, Δ, λ, Θ, γ, δ, β, Ω, Φ
2	a, c, e, i, m, n, o, r, s, u, v, w, x, z, *, +, /, <, >, =, '.', ',', \\, π, θ, α, ϵ, σ
3	g, j, p, q, y
4	-
5	ˆ

Table 3.1: Preliminary classification into 5 groups

The subsequent recognition stage concentrates only on the pre-classified group and treat the members of the group as possible recognition candidates. Characters belonging to other groups are assumed to be invalid matches to the unknown and are not considered for the recognition.

3.5 Data Processing

In the current context, a handwritten stroke refers to the locus of the pen from its pen-down to the next pen-up position. It can therefore be described as a sequence of consecutive points on the x-y plane: S = $p_1 p_2 \cdots p_L$ where p_1 and p_L are the pen-down and pen-up points, respectively. Based on the representation, a handwritten character can then be described as a sequence of strokes C =

$S_1 S_2 \cdots S_N$. [8].

After pen movement points are found they are processed for chain code extraction. A chain code is a sequence of numbers between 0 and 7 obtained from the quantized angle of the pen tip's point in an equally timed manner. Chain code values for angles are shown in Figure 3.6. Chain codes describe the charac-

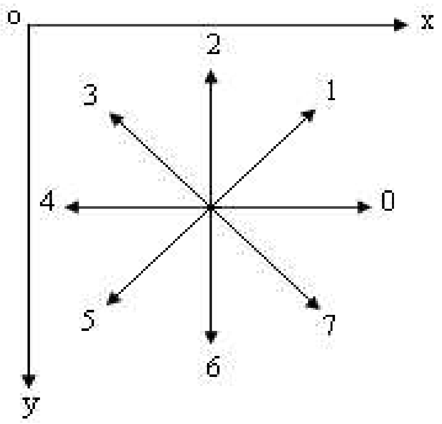


Figure 3.6: Chain Code values for the angles

ters drawn. Each character has a different chain code representation. Chain code values of some characters are shown in Figure 3.7.

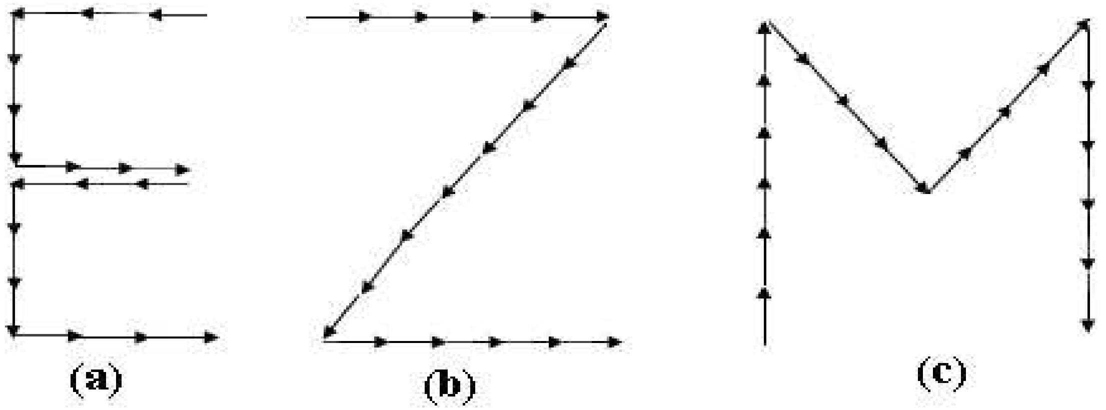


Figure 3.7: A simple chain coded representation of characters
(a) E=444666000444666000 (b) Z=00000555555500000
(c) M=2222227777111166666

3.6 Calculation of bounding boxes and sub-rectangles

Bounding box is the minimum rectangle that encloses the written character. After characters are written, by using the foreground subtracted image, bounding boxes of each character is calculated using a simple image processing algorithm. Bounding boxes are one of the key points in the character recognition algorithm. They are used to distinguish each character's coordinates on the paper. Then using these coordinates each character's movement points and chain codes are separated and feed to the Finite State Machine.

Pen movement points are not used just for chain code generation. Their positions in the bounding box are used in the classification period. First of all each calculated bounding box is divided into sub-rectangles. Bounding boxes belongs to the pre-classification group of 1 and 3 are divided into 9 different sub-rectangles. And bounding boxes belongs to the pre-classification group of 2 and 5 are divided into 5 sub-rectangles, Figure 3.8.

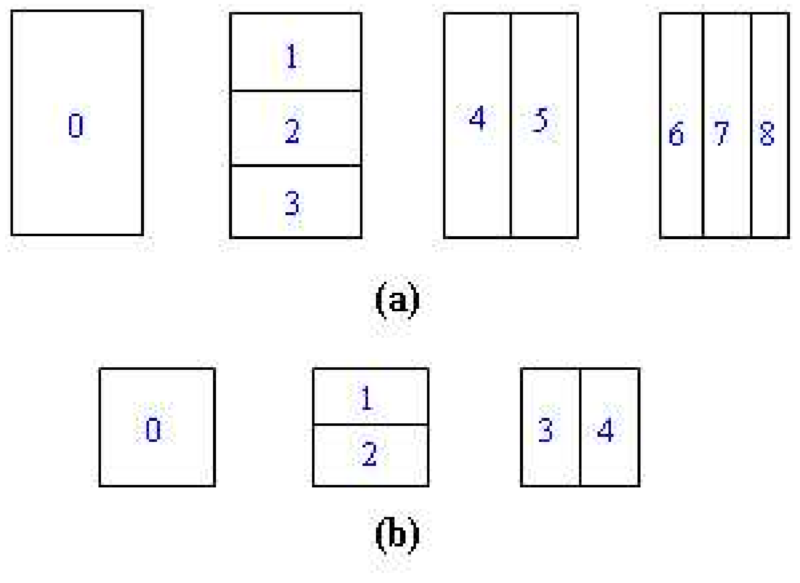


Figure 3.8: Bounding box sub-rectangles
(a) preclassification group 1 and 3 (b) preclassification group 2 and 5

Besides the chain codes, each character is described by their position in the bounding boxes. Using the position in the bounding boxes, recognition of characters are performed more easily.

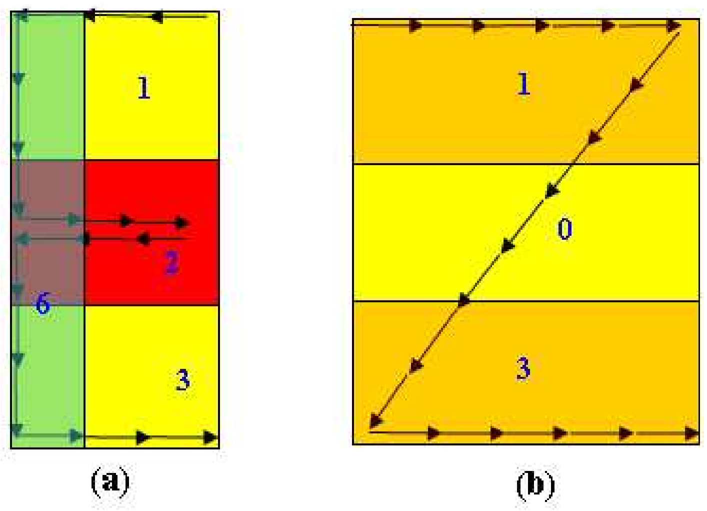


Figure 3.9: A simple region coded representation of characters
(a) E=111666222222666333 (b) Z=11111000000033333

3.7 Finite State Machines

The recognition system consists of finite state machines corresponding to individual characters. The FSMs generating the minimum error identifies the recognized character. The weighted sum of the error from the from a finite state machine determines the final error for a character in the recognition process. Finite state machines for some characters are shown in Figure 3.10.

During analysis using finite state machines, the system

1. applies the chain code and region code as input to each state machine,
2. determines state changes (additionally, the system increases an error counter by one if a change is not possible according to the current FSM),

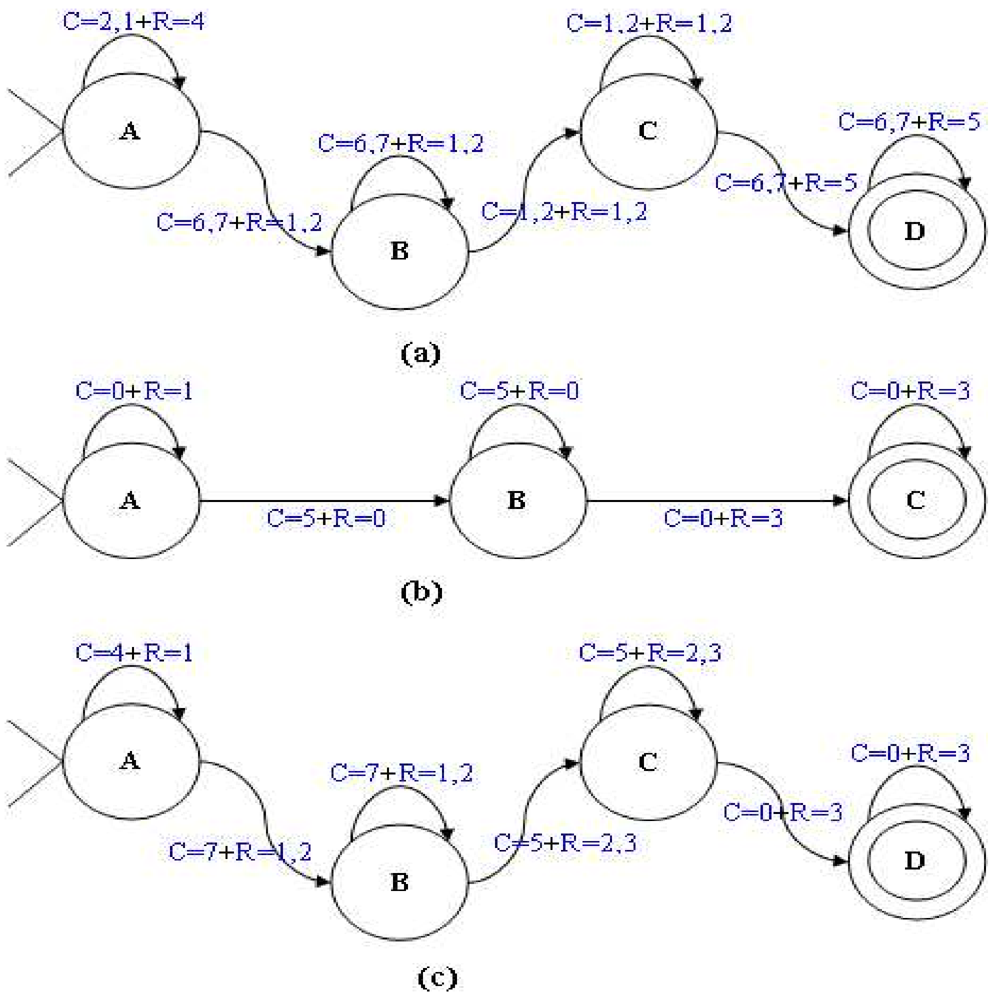


Figure 3.10: Finite State Machines of characters (a) M (b) Z (c) $\sum$ where C = Chain Code, R = Rectangle Index

3. eliminates the corresponding character, if a chain code does not terminate in the final state

4. adds up errors in each state to find the final error for each character.

When the 32222217771117666 chain code and the 44444441122115555 region code are applied as an input to M's machine, the first element, 3, generates an error and the error counter is set to 1. The second element of the chain 2 and region 4, are correct value at the FSM's starting state so the error counter remains at 1 after processing the input 2. The FSM remains in the first state with other 2s of chain and 4s of region code values. Also with the subsequent 1, because

1 and 2 are the inputs of the machine's first state for M. Input of chain code 7 and region code 1 makes the FSM go to the next state, and the subsequent three chain code 7s and region code 1,2 let the machine remain there. Whenever the chain code becomes 1 with region code 2, the FSM moves to the third slate. The machine stays in this state until the chain code is 7 and region code is 5, and this makes FSM go to the Final state. The rest of the input data, chain 6 and region 5, makes the machine stay in the final state, and when the input is finished, the FSM terminates. For this input sequence, 1 is the machine's error for character M. In practice, this sample chain and region code determines all other characters using FSMs. However, the other FSMs generate either greater or infinite error values. When this chain and region code is applied to the other characters they will never reach the final state and the error will he set to infinity.

Chapter 4

Implementation Details

4.1 The Microsoft Vision SDK

The Microsoft Vision SDK is a library for writing programs to perform image manipulation and analysis on computers running Microsoft Windows operating systems. The Microsoft Vision SDK was developed by the Vision Technology Research Group in Microsoft Research to support researchers and developers of advanced applications, including real-time image-processing applications. It is a low-level library, intended to provide a strong programming foundation for research and application development; it is not a high-level platform for end-users to experiment with imaging operations. The Microsoft Vision SDK includes classes and functions for working with images, but it does not include image-processing functions. The Microsoft Vision SDK is a C++ library of object definitions, related software, and documentation for use with Microsoft Visual C++.

The core of the VisSDK is the CVisImage class. Similar to the Windows' bitmap header, the CVisImage stores a variety of properties about an image and a pointer to the memory used to store the image data. The **Video For Windows (VFW)** digitizer in The Microsoft Vision SDK is used to capture live video. Acquired live video is stored in **CVisImage** Object.

4.2 Data Structures

In order to store the states of characters, we create a structure. At the outset, we create an array of pointers for holding each character's starting states. The following is the prototype for the structure to present a state of a character.

```
struct state
{
   int total_states;
   int states[3];
   int min_number_of_states;
   int total_rects;
   int rects[2];
   struct state* next_state;
};
```

The `total_states` is used to store the number of states in current character state. This can be a number between 1 and 3. `states[3]` holds the **Chain Code** values for the current state. Before passing to next state of the character `min_number_of_states` must be satisfied. This value is used for better recognition results. `total_rects` is used to store number of rectangles in which current state value will be searched. `rects[2]` holds the rectangle indexes. `next_state` is the pointer to the next character state.

When the program is run, file containing the states and rectangles is opened and for each character a linked list of `struct state` type is created. Then an array is created to store pointers of the linked list start points.

In character recognition algorithm positions of the pen is used. While characters are written position of the pen is found and it is stored in the following data structure.

```
struct point {
int x;
int y;
struct point *next;
};
```

The `x` field keeps the x coordinate, the `y` field keeps the y coordinate of the blue band sticked to the pen tip. `next` is the pointer to the next point.

4.3 Algorithms

In this thesis we mainly worked used three type of algorithms, foreground subtraction algorithm, bounding box calculation algorithm and character recognition algorithm. These algorithms are explained in detail in following sections.

4.3.1 Foreground Subtraction Algorithm

As we are interested in what is written on paper, for detection and removal of hand and pen we used the foreground subtraction algorithm. The pseudo-code of the algorithm is as follows.

```
- initialize background model Bn(x)
- initialize masking-threshold
for each frame
begin
  - generate a binary motion mask image Mn(x) which contains
    all moving pixels using threshold
  - update the non-moving pixels
  - update the masking-threshold
  - convert the background model to black and white
  - show the resulted image to user
end
```

4.3.2 Bounding Box Calculation Algorithm

Bounding boxes are one of the key points in the character recognition algorithm. According to their coordinates each character's movement points and chain codes are separated from the others. Input to this algorithm is foreground subtracted image, and output is a vector of bounding boxes. This algorithm processes all the pixels in the image row by row and uses a first-in, first-out queue. The pseudo-code of the algorithm is as follows.

```
for each pixel u in image
   - visited[u]  = not visited yet
for each pixel u in image
   if (pixel[u] is black) and (not visited yet)
   begin
     - push u to the Queue
     - visited[u] = visited and in the Queue
     - create and initialize a bounding box
     while Queue is not empty
     begin
       - pop p from Queue
       for each pixel v in the neighborhood of p
       begin
         if (pixel[v] is black) and (not visited yet)
         begin
            - push coordinate for pixel[v] to the Queue
            - visited[v] = visited and in the Queue
         end
       end
       - update the bounding box' properties using coordinate of pixel[p]
       - visited[p] = visited and out of queue
     end
     - store bounding box and try to find another one
   end
```

4.3.3 Character Recognition Algorithm

While characters are written at each frame position of the blue band sticked to the pen tip is found and stored in a linked list. Also at each frame, foreground subtraction algorithm is used to remove the pen and hand and the final image is stored. After writing is finished character recognition algorithm is called. The following is the pseudo-code of the character recognition algorithm.

```
- locate the reference lines
- pass Foreground subtracted image as input to the
  Bounding box calculation algorithm and calculate the bounding boxes
  of each character
for each bounding box
begin
   - find the pen trace points falling in the bounding box
   - construct chain codes using trace points
   - apply preliminary classification using position of bounding box
     and reference lines
   - calculate the coordinates of sub-rectangles
   for each character in the preliminary classification category
   begin
      while character states and pen trace points are not finished do
      begin
         if chain code is equal to current character state and
            current point lies in the current state rectangle
         begin
            pass to next state
         end
         else
            increase the recognition error and process to next point
      end
   end
   - post process the results for better results
   - find the best match having the least error with longest chain
end
```

4.4 User Interface

User interface of the program is created using MFC libraries of Microsoft Visual C++ 6.0. It consists of two separate windows, **Main Window** and **Recognized Text Window**. Their functionalities are described in following subsections.

4.4.1 Main Window

Main window is responsible for retrieving video image from camera, applying algorithms on grabbed image, and showing the resulted images to the user, Figure 4.1. When program is executed, 4 different images are shown (a) This part shows the frames grabbed from the camera. It is continuously updated. (b) This is the last frame taken while processing foreground subtraction algorithm. And pen trace points are shown on it with blue color. (c) Pen trace points falling within each bounding boxes. (d) Result of foreground subtraction algorithm and corresponding bounding boxes.

First thing to be noticed is that user is allowed to write a portion of the image. And this part is shown with a rectangle having sky blue color. Coordinates of this rectangle can be changed during the execution of the program.

4.4.2 Recognized Text Window

This dialog box is used for presenting the result to the user and providing some means of changing the text when required, Figure 4.2. It consists of 5 parts. (a) This editbox holds the LaTeX code created so far from the recognized letters. In Figure 4.2 lastly added word is bounded by blue rectangle. User can edit the content of this editbox when any recognition error occurred. (b) Red, green and

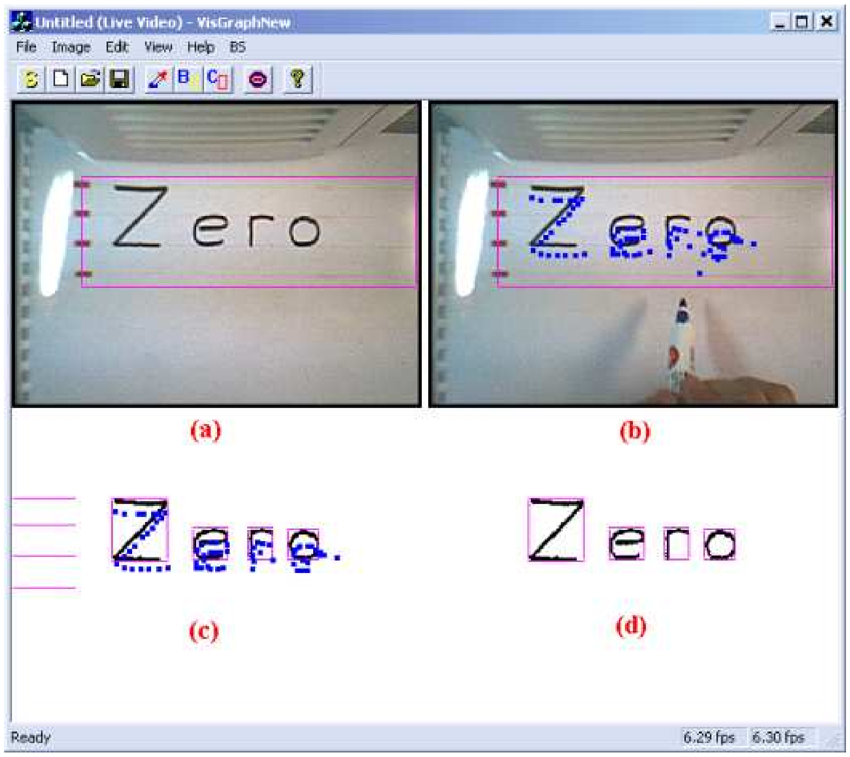
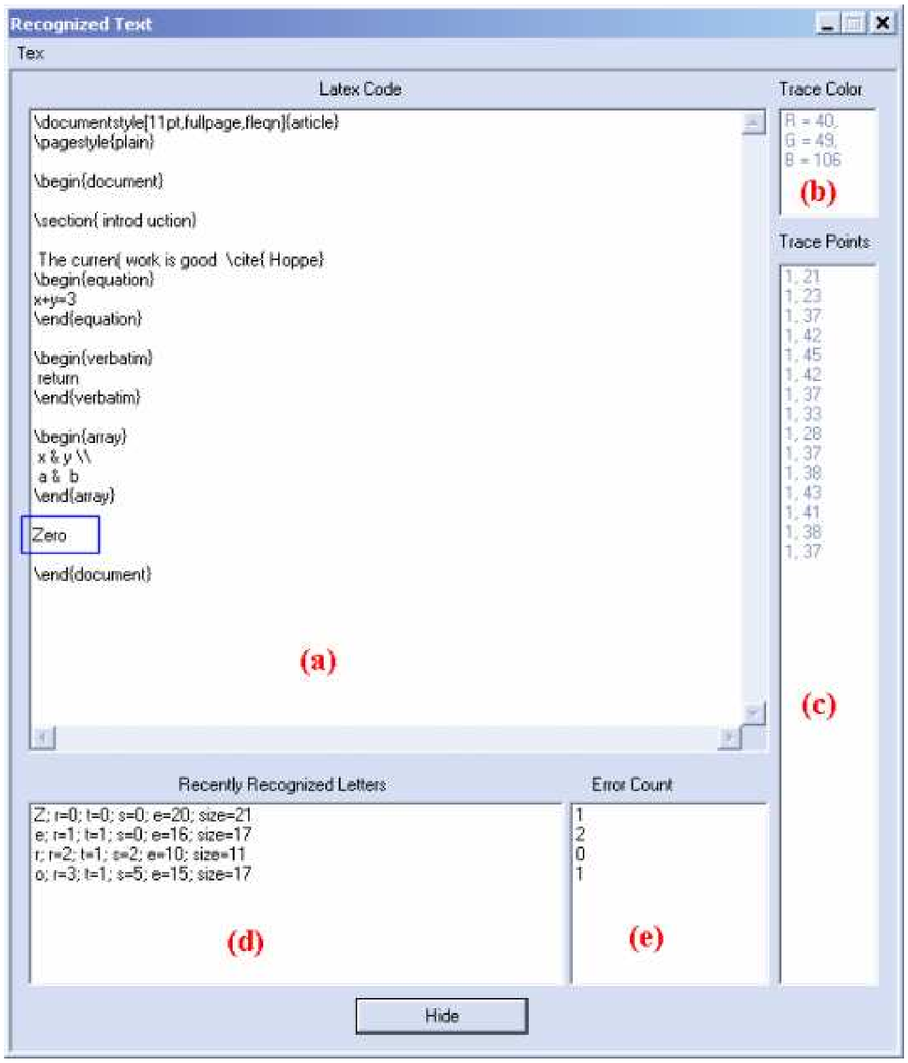

Untitled (Live Video) - VisGraphNew
File Image Edit View Help BS
(a)
(b)
(c)
(d)
Ready
6.29 fps 6.30 fps

Figure 4.1: Main window

blue components of the band sticked near to the pen-tip. (c) Number of pixels in which this trace color is detected with some threshold. (d) This listbox gives detailed information about the last recognition process. At each row following properties are written, recognized character, rectangle index from left, index of chain in which recognition of character started, index of chain in which recognition of character finished and total length of the chain. (e) Each row of this listbox holds the recognition errors associated with each character. In figure Z is recognized with error 1, e is recognized with error 2 . . .

4.5 LaTeX Code Generation

One of the aims of this thesis is construction of the **LaTeX** codes. After characters are recognized, they are passed to LaTeX code generation procedure. Some



```
\documentstyle[11pt,fullpage,fleqn]{article}
\pagestyle{plain}

\begin{document}

\section{ introd uction}

 The curren( work is good  \cite{ Hoppe}
\begin{equation}
x+y=3
\end{equation}

\begin{verbatim}
 return
\end{verbatim}

\begin{array}
 x & y \\
 a &  b
\end{array}

Zero

\end{document}
```

```
R = 40,
G = 49,
B = 106
```

```
1, 21
1, 23
1, 37
1, 42
1, 45
1, 42
1, 37
1, 33
1, 28
1, 37
1, 38
1, 43
1, 41
1, 38
1, 37
```

```
Z; r=0; t=0; s=0; e=20; size=21
e; r=1; t=1; s=0; e=16; size=17
r; r=2; t=1; s=2; e=10; size=11
o; r=3; t=1; s=5; e=15; size=17
```

```
1
2
0
1
```

Figure 4.2: Recognized Text Window

character combination are used for LaTeX keyword generation. Some of the characters have different effects in different environments. Supported LaTeX environments are array construction, citation, section, itemization, equation, verbatim and normal text environment. Character combination and corresponding LaTeX codes are shown in Table 4.1.

Characters	LaTeX code
<a	**begin{array}**
, in array mode	&
\ in array mode	\\
>a	**end{array}**
<c	**\cite{**
>c	}
<e	**begin{equation}**
>e	**end{equation}**
<i	**begin{itemize}** **\item**
\ in itemize mode	**\item**
>i	**end{itemize}**
<s	**\section{**
>s	}
<v	**begin{verbatim}**
>v	**end{verbatim}**

Table 4.1: Character groups and corresponding generated LaTeX code

Chapter 5

Results

In this chapter, some example of characters written and their LaTeX code describing the salient features of the character recognition system are given.

We used an ordinary board marker having black color and blue band attached near its tip, white background fabric, and an USB CCD Camera in our experiments. We used Amd 1400 processor with 256 Mbytes of memory for all processing.

There are three possible outcomes in the recognition: correct, incorrect, and rejected. Incorrect recognition means that algorithm returns the wrong character while rejected means that no answer is returned.

Following paragraph containing 470 characters is used as the test data.

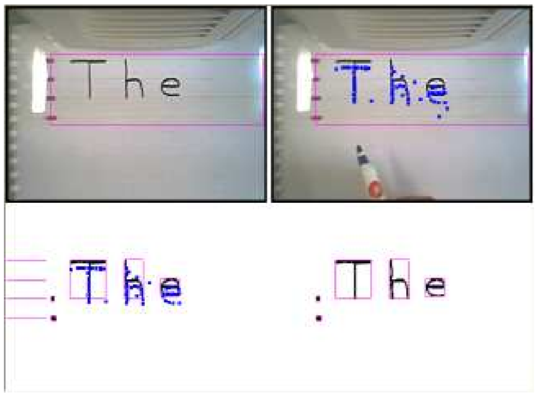
"By using an USB CCD Camera, positions of the pen tip between frames are detected as they are written on a regular paper. Then, these positions are used for calculation of directional information. The directional and positional information of the pen movement points are used for character recognition. After characters are recognized they are passed to LaTeX code generation subroutine. Supported LaTeX environments are array construction, citation, section, itemization, equation, verbatim and normal text environments. 0123456789 $\surd$ $\sum$ $\prod$ $\int$ Δ λ Θ γ δ β Ω α ϵ σ"

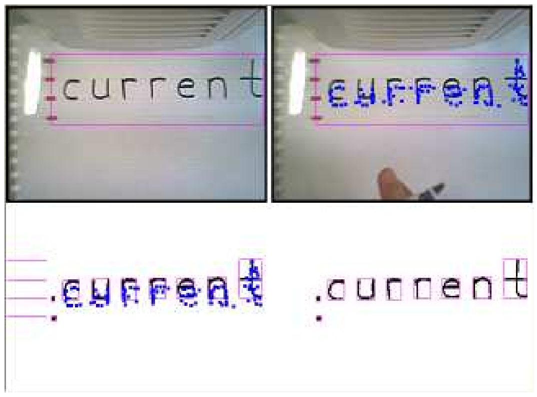
The recognition rate is 90.21%, incorrect rate is 7.02% and rejected rate is 3.77%, shown in Table 5.1. The main recognition errors were due to the abnormal writing and ambiguity among similar shaped characters. Most of the confusion was between character pairs such as "e" & "c", "5" & "S", "u" & "v". This could be avoided by using a word dictionary to look-up for possible character compositions. The presence of contextual knowledge will help to eliminate the ambiguity.

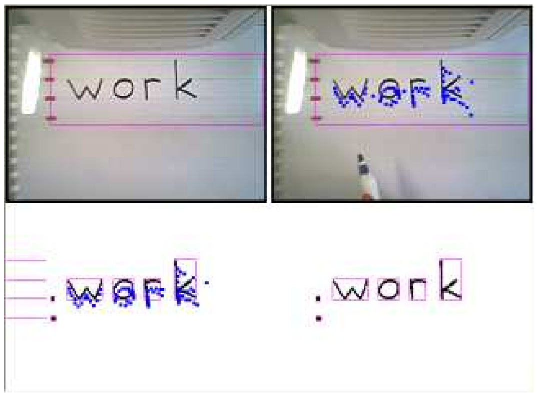
	Correct	Incorrect	Rejected
All	(424/470)*100 90.21%	(33/470)*100 7.02%	(13/470)*100 3.77%

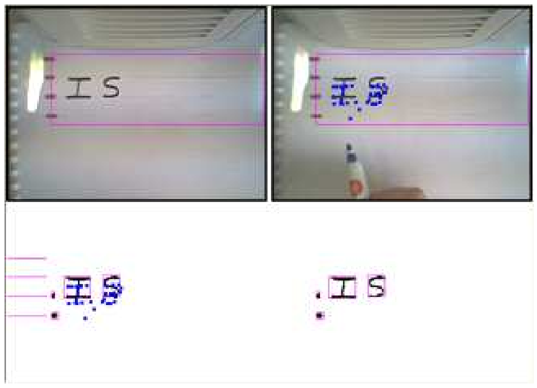
Table 5.1: Recognition Results

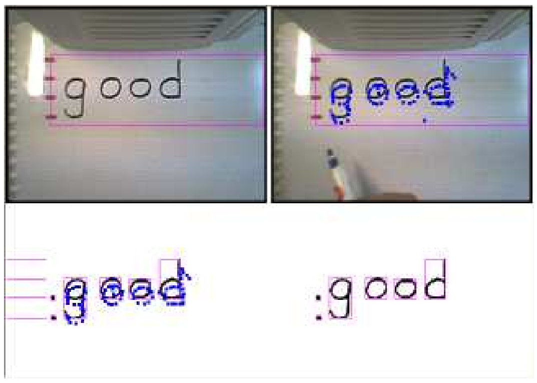
Figure 5.1 through Figure 5.8 presents LaTeX code creation of normal text, section, citation, array, item list, equation, verbatim text and mathematical expressions.


(a) The


(b) current

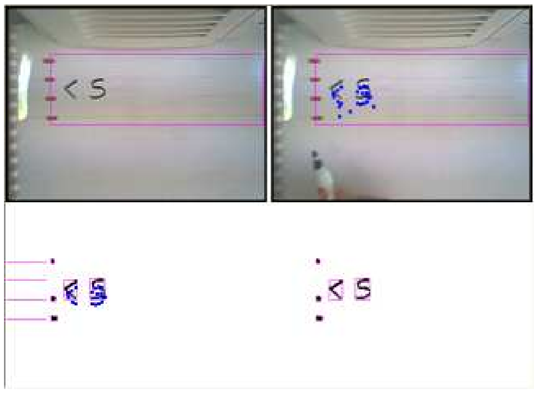

(c) work

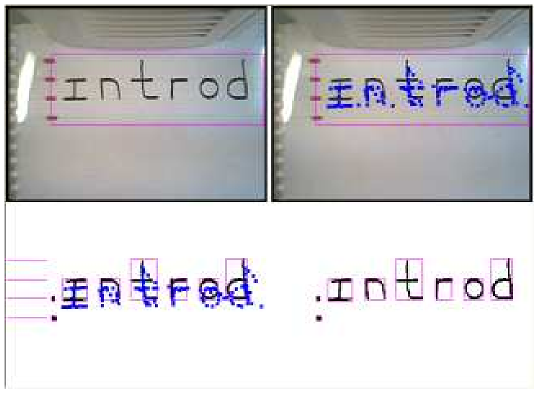

(d) is

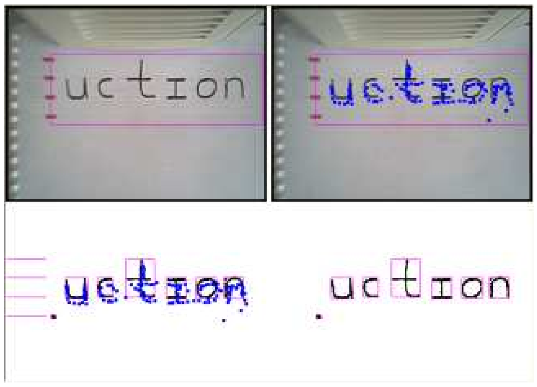

(e) good

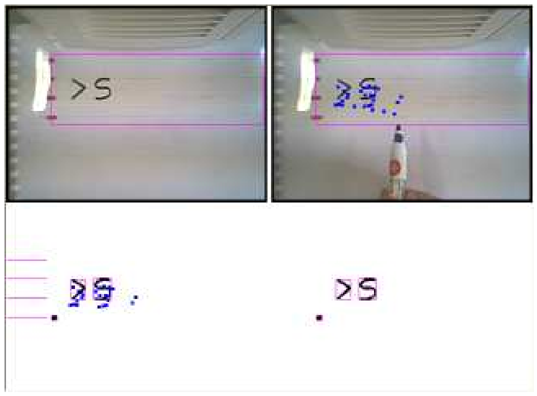
The current work is good
(f) LaTeX code

Figure 5.1: Normal text creation


(a) \section{

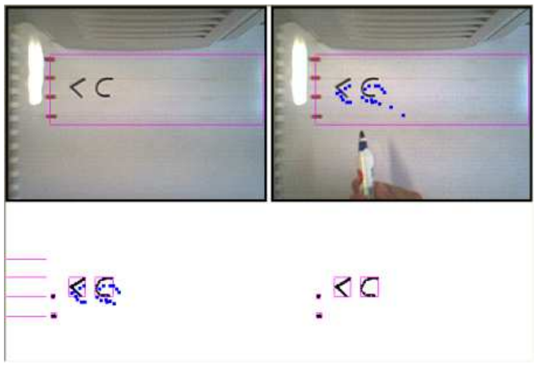

(b) Introd


(c) uction

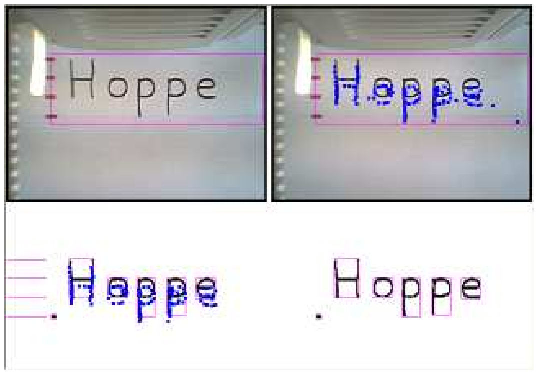

(d) }

\section{Introduction}
(e) LaTeX code

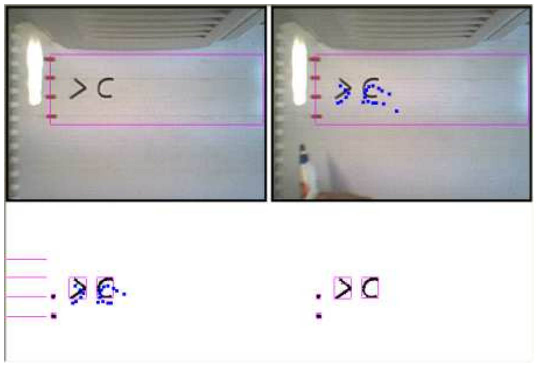
Figure 5.2: Section construction



(a) \cite{



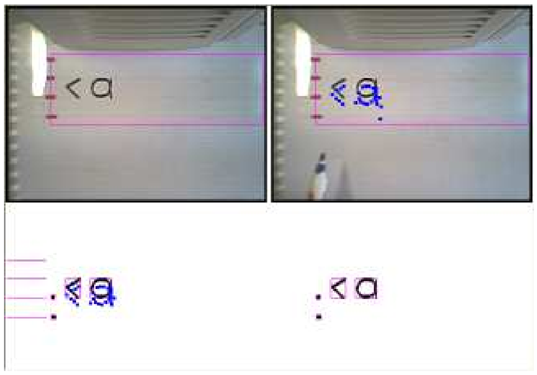
(b) Hoppe



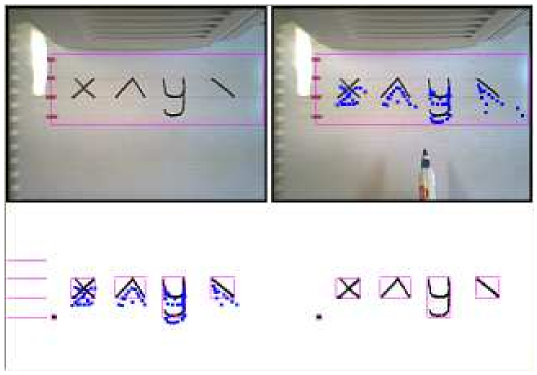
(c) }

\cite{Hoppe}
(e) LaTeX code

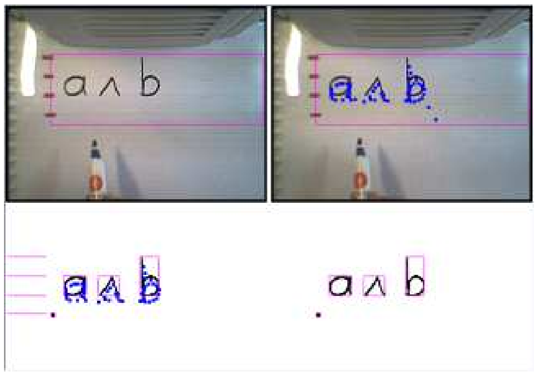
Figure 5.3: Cite construction



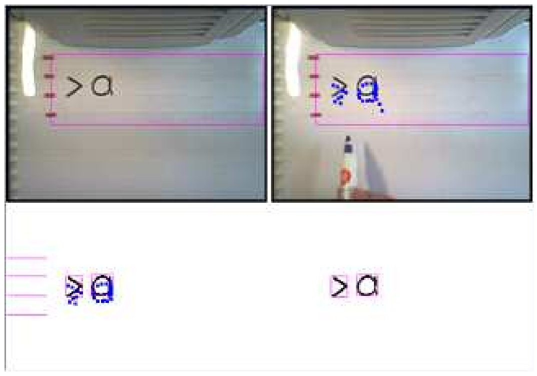
(a) \begin{array}{cccccc}



(b) x & y \\



(c) a & b



(d) \end{array}

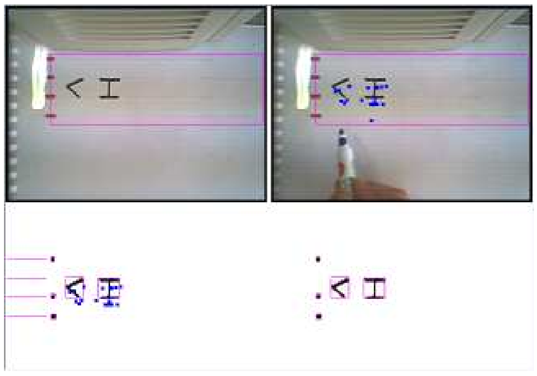
```
\begin{array}{cccccc}
   x & y \\
   a & b
\end{array}
```

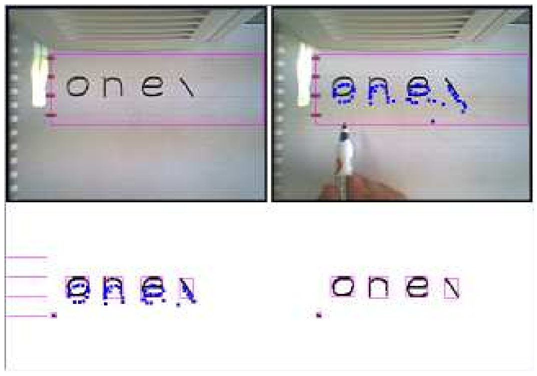
(e) LaTeX code

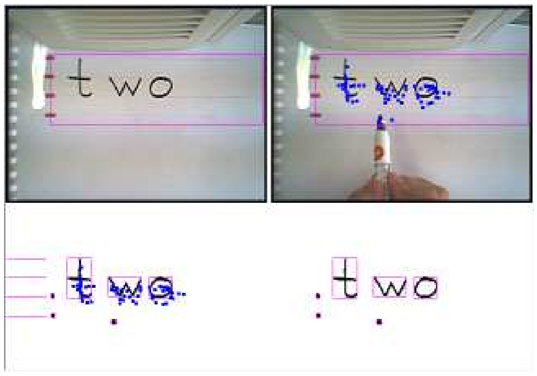
$$\begin{array}{cc} x & y \\ a & b \end{array}$$

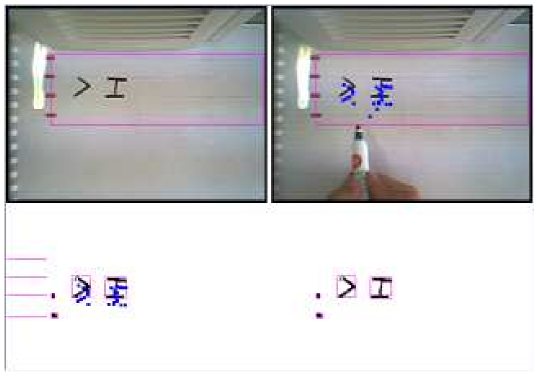
(f) Corresponding array created

Figure 5.4: Array construction


(a) \begin{itemize} \item


(b) one \item


(c) two


(d) \end{itemize}

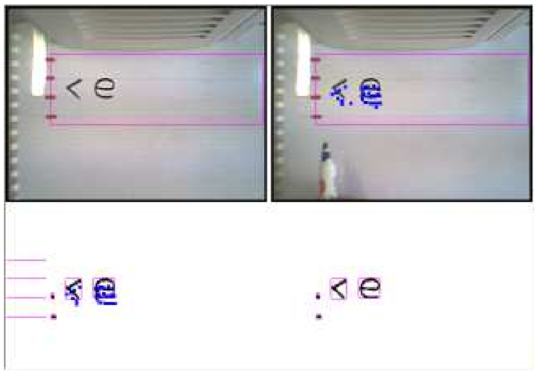
```
\begin{itemize}
   \item one
   \item two
\end{itemize}
```

(e) LaTeX code

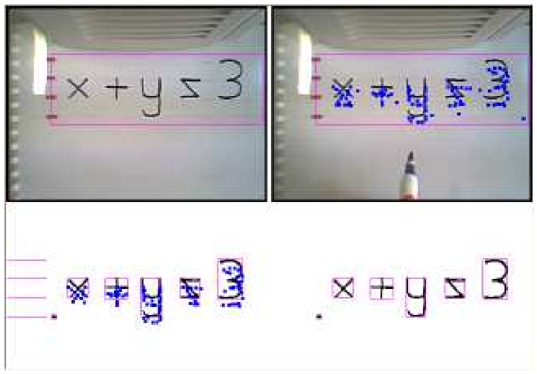
- one
- two

(f) Corresponding item list created

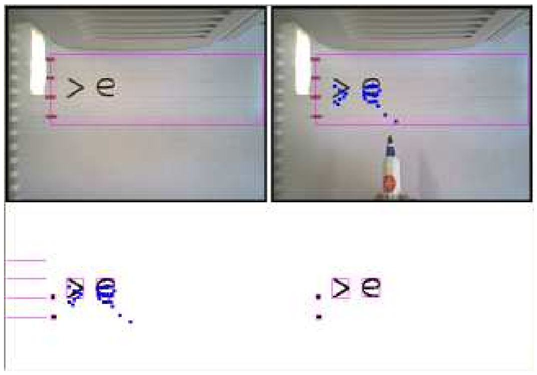
Figure 5.5: Item list construction



(a) \begin{equation}



(b) x + y = 3



(c) \end{equation}

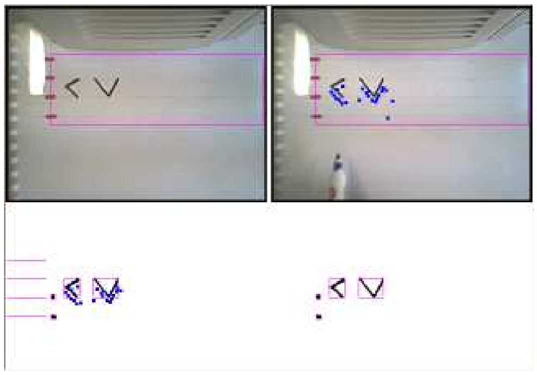
```
\begin{equation}
   x + y = 3
\end{equation}
```

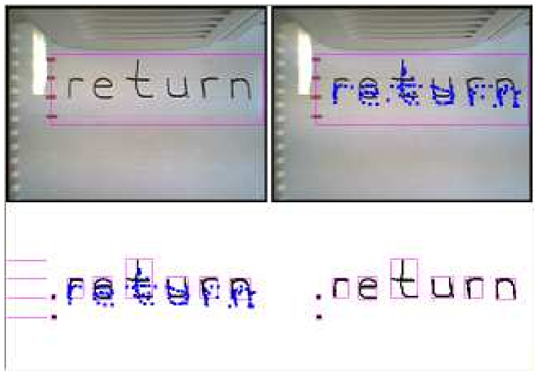
(d) LaTeX code

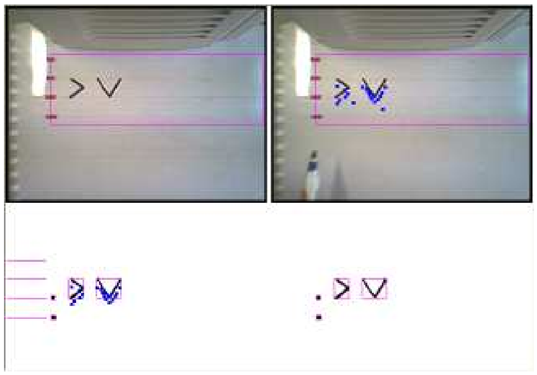
$$x + y = 3 \tag{5.1}$$

(e) Corresponding equation created

Figure 5.6: Equation construction


(a) \begin{verbatim}

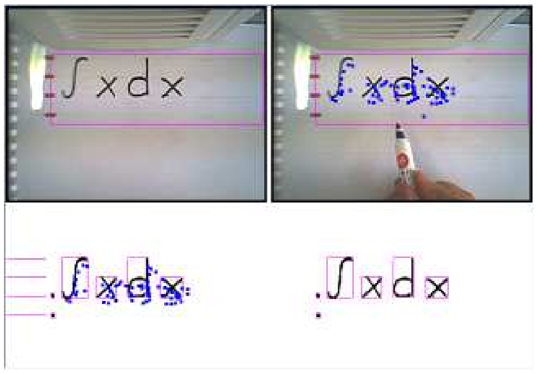

(b) return

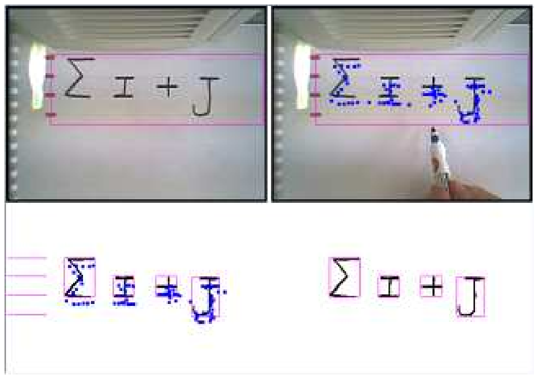

(c) \end{verbatim}

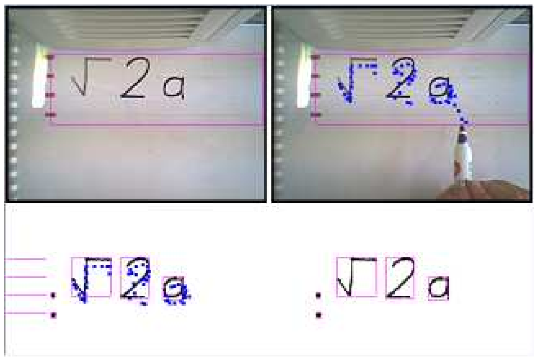
```
\begin{verbatim}
   return

 \end{verbatim}
```
(d) LaTeX code

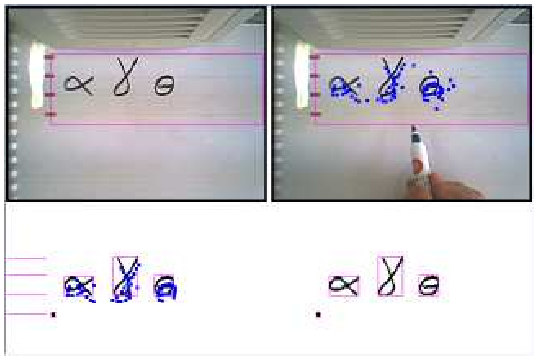
```
return
```
(e) Corresponding verbatim created

Figure 5.7: Verbatim construction


(a) \$\int{xdx}\$


(b) \$\sum{i + j}\$


(c) \$\sqrt{2a}\$


(d) \$\alpha\$ \$\gamma\$ \$\theta\$

$$\int xdx \sum i + j \sqrt{2a}\ \alpha\ \gamma\ \theta$$

(e) Corresponding output created

Figure 5.8: Mathematical expressions and some Greek letters

5.1 Limitations caused by the Equipment

The image sequences are captured using an USB CCD Camera and we obtained 6 frames per second with 320 x 240 pixels. As we obtained 6 frames per second, in order not to miss the details of the characters, the user needs to write the character slowly, 10 words per minute. The writing space visible in the video images covers an area of approximately 20 x 12 cm. Because of the limitations of the writing space, at most 4 uppercase or 6 lowercase characters can be written and then recognized at a time. Words containing more than 6 characters can be written in consecutive frames, see Figure 5.2.

Chapter 6

Conclusion

Traditionally, human machine communication has been based on keyboard and pointing devices. These methods can be very inconvenient when the machine is only slightly bigger or the same size as the human palm. A keyboard is very difficult to integrate in small devices and it usually determines the size of the whole apparatus. A pointing device, for example a track ball or a pen, is insufficient or very slow when used alone in applications in which textual input is also desired. Online handwriting recognition promises to provide a dynamic means of communication with computers through a pen like stylus, not just a keyboard. This seems to be a more natural way of entering data into computers.

In this thesis, we develop a character recognition system that combines the advantage of both on-line and off-line systems. Using an USB CCD Camera, positions of the pen-tip between frames are detected as they are written on a sheet of regular paper. Then these positions are used for calculation of directional information. Finally, handwritten character is characterized by a sequence of writing directions between consecutive frames. The directional information of the pen movement points is used for character preclassification and positional information is used for fine classification. After characters are recognized they are passed to LaTeX code generation subroutine. Supported LaTeX environments are array construction, citation, section, itemization, equation, verbatim and normal text environment.

The results show 90.21% correct recognition rate. The main recognition errors were due to the abnormal writing and ambiguity among similar shaped characters. Most of the confusion was between character pairs such as ”e” & ”c”, ”5” & ”S”, ”u” & ”v”. This could be avoided by using a word dictionary to look-up for possible character compositions. The presence of contextual knowledge will help to eliminate the ambiguity.

For the future work, there are still many issues need to be resolved before the system becomes of practical usage with high recognition rates. We need to carefully study many parameter settings in the system as well as continuous writing of long paragraphs on paper.

Bibliography

[1] F. Alimoğlu and E. Alpaydın. Combining multiple representations for pen-based handwritten digit recognition. In *4th International Conference Document Analysis and Recognition*, volume 1 and 2, pages 637–640, 1997.

[2] H. Bunke, T. von Siebenthal, T. Yamasaki, and M. Schenkel. Online handwriting data acquisition using a video camera. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 573–576, 1999.

[3] K. F. Chan and D. Y. Yeung. Elastic structural matching for on-line handwritten alphanumeric character recognition. In *Proceedings of the Fourteenth International Conference on Pattern Recognition*, pages 1508–1511, 1998.

[4] H. Fijiyohsi and A. Lipton. Real-time human motion analysis by image skeletonization. In *Proceedings of IEEE WACV98*, pages 15–21, 1998.

[5] V. K. Govindan. Character recognition - review. *Pattern Recognition*, 23(7):671–683, 1990.

[6] Graffiti. Graffiti, a handwriting recognition software by palm. http://www.palm.com/products/input/, 2002.

[7] K. Ishigaki, H. Tanaka, and N. Iwayama. Interactive character recognition technology for pen-based computers. *Fujitsu Scientific and Technical Journal*, 2:191–201, 1999.

[8] X. Li and D. Y. Yeung. On-line handwritten alphanumeric character recognition using feature sequences. In *Image Analysis Applications and Computer Graphics*, pages 197–204, 1995.

[9] X. Li and D. Y. Yeung. On-line handwritten alphanumeric character recognition using dominant points in strokes. *Pattern Recognition*, 30(1):31–44, January 1997.

[10] A. Lipton, H. Fijiyohsi, and R. S. Patil. Moving target detection and classification from real-time video. In *Proceedings of IEEE WACV98*, pages 8–14, 1998.

[11] I. Methasate and S. S. Tang. On-line thai handwriting character recognition using stroke segmentation with hmm. In *Applied Informatics International Symposium on Artificial Intelligence and Applications (AI2002)*, pages 59–62, 2002.

[12] W. Newman and P. Wellner. A desk supporting computer-based interaction with paper documents. In *Proceedings of CHI'92 Conference on Human Factors in Computing Systems*, pages 587–592, 1992.

[13] Ö. F. Özer, O. Özün, C. Ö. Tüzel, V. Atalay, and A. E. Çetin. Vision-based single-stroke character recognition for wearable computing. *IEEE Intelligent Systems*, May/June:33–37, 2001.

[14] S. Smithies, K. Novins, and J. Arvo. A handwriting-based equation editor. In *Proceedings of Graphic Interface (GI)*, 1999.

[15] X. Tang, C. Y. Chan, and J. Liu. Handwritten chinese character recognition through a video camera. In *Proceedings of the IEEE International Conference on Image Processing, 2000, Vancouver, Canada*, 2000.

[16] X. Tang and F. Lin. Video-based handwritten character recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002.

[17] C. Tappert, C. C. Suen, and T. Wakahara. The state of art in on-line handwriting recognition. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 12:787–808, 1990.

[18] C. C. Tappert. Adaptive on-line handwriting recognition. In *Proceedings of 7th International Conference on Pattern Recognition*, pages 1004–1007, 1984.

[19] V. Vuori and E. Oja. Analysis of different writing styles with selforganizing map. In *Proceeding of the 7th International Conference on Neural Information Processing*, volume 2, pages 1243–1247, 2000.

[20] T. Wakahara and K. Odaka. On-line cursive kanji character recognition using stroke-based affine transformation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19(12), pages 1381–1385, December 1997.

[21] M. Wienecke, G. A. Fink, and G. Sagerer. Video-based on-line handwriting recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 226–230, 2001.

[22] H. Yuen. A chain coding approach for real-time recognition of on-line handwriting characters. In *Proceeding of International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3426–3429, 1996.

Appendix A

Finite State Machines of characters

Following list shows the finite state machines of the characters. Each row begins with the character, the number after / shows the preclassification group (zero based) and following numbers seperated with spaces shows chain codes, within which rects, minimum number of repetitions.

```
A   /0 12 41 2 76 15 2 34 23 2 0 2 2
B   /0 6 6 3 2 6 2 01 1 1 76 12 1 54 12 1 076 23 1 54 3 1
C   /0 4 1 1 56 4 2 701 3 1
D   /0 6 6 3 2 6 2 01 1 1 76 15 2 54 3 1
E   /0 4 1 1 6 6 1 0 2 1 4 2 1 6 6 1 0 3 1
F   /0 4 1 1 6 6 3 2 6 1 0 2 1
G   /0 34 1 1 56 4 2 70 3 1 12 5 1 4 2 1
H   /0 6 6 3 2 6 1 0 2 1 2 5 1 6 5 2
I   /0 0 1 2 4 1 1 6 7 2 4 3 1 0 3 2
J   /0 0 1 2 4 1 1 6 5 2 54 3 1 32 3 1
K   /0 6 6 3 2 6 1 1 12 2 5 12 2 7 23 2
L   /0 6 6 3 0 3 2
M   /0 21 4 3 67 12 1 12 12 1 67 5 3
```

N	/0 21 4 3 7 0 3 21 5 3
O	/0 45 1 1 67 4 2 01 3 2 23 5 2 4 1 1
P	/0 6 6 3 2 6 3 01 1 1 76 5 1 54 21 2
Q	/0 67 4 2 01 3 2 23 5 2 456 1 2 7 0 3
R	/0 6 6 3 2 6 3 01 1 1 76 5 1 54 21 1 7 23 2
S	/0 4 1 1 5 12 1 6 12 1 70 0 2 65 3 1 4 3 1
T	/0 0 1 2 4 1 1 6 7 3
U	/0 6 4 2 70 3 1 1 3 1 2 5 2 6 8 3
V	/0 67 4 3 12 5 3
W	/0 67 4 3 12 23 1 67 23 1 12 5 3
X	/0 67 0 3 4 3 2 12 0 3
Y	/0 7 12 1 1 12 1 5 12 1 6 23 2
Z	/0 0 1 2 5 0 3 0 3 2
a	/1 23 14 1 45 13 1 67 3 1 01 2 1 2 4 1 6 4 2
b	/0 6 6 3 2 6 1 01 2 1 76 23 1 54 3 1
c	/1 34 1 1 56 3 1 70 2 1
d	/0 6 5 3 2 5 1 34 25 1 56 23 1 70 3 1
e	/1 0 0 2 12 4 1 34 1 1 56 3 1 70 2 1
f	/0 23 5 1 45 1 1 6 7 3 23 4 1 0 2 1
g	/2 23 15 1 45 1 1 67 12 1 01 2 1 2 5 1 6 5 2 54 3 1
h	/0 6 6 3 2 6 1 01 2 1 76 5 2
i	/1 0 1 1 45 1 1 6 0 2 45 2 1 0 2 2
j	/2 0 1 1 45 1 1 6 5 2 54 3 1
k	/0 6 4 3 2 4 1 1 23 1 5 23 1 7 3 2
l	/0 12 0 3 45 1 0 67 0 3
m	/1 6 3 2 2 3 2 10 1 1 76 0 1 2 0 1 10 1 1 76 4 2
n	/1 6 3 2 2 3 2 10 1 1 76 4 2
o	/1 45 1 1 67 3 1 01 2 1 23 4 1 4 1 1
p	/2 6 4 3 2 4 3 01 1 1 76 5 1 54 21 2
q	/2 23 15 1 45 1 1 67 12 1 01 2 1 2 5 1 6 5 3
r	/1 6 3 2 2 3 2 01 1 2
s	/1 4 1 1 56 13 1 70 0 2 65 2 1 4 2 1
t	/0 6 7 3 70 3 2 34 23 2 0 2 2

```
u   /1 6 3 2 70 2 1 12 24 2 67 4 2
v   /1 67 32 2 12 42 2
w   /1 67 3 3 12 3 1 67 4 1 12 4 3
x   /1 67 0 3 4 2 2 12 0 3
y   /2 6 12 2 70 2 1 12 28 1 6 5 2 54 3 1
z   /1 0 1 2 5 0 3 0 2 2
0   /0 23 1 1 45 1 1 67 4 2 01 3 2 23 5 2 5 0 3
1   /0 12 1 1 6 7 3 4 3 1 0 3 2
2   /0 1 12 1 0 1 1 76 12 2 5 0 3 0 3 2
3   /0 01 1 2 76 5 1 54 52 2 0 2 1 76 23 1 54 3 2
4   /0 56 0 3 0 3 2 2 8 1 6 8 2
5   /0 0 1 2 4 1 2 6 6 1 0 2 1 7 23 0 65 23 2 4 3 1
6   /0 56 0 3 01 3 1 23 23 1 45 23 1
7   /0 0 1 2 56 0 3 12 6 1 0 2 2
8   /0 4 1 1 56 12 1 70 0 2 65 23 1 4 3 1 32 3 1 10 23 2 23 12 1
9   /0 23 15 1 45 1 1 67 12 1 01 2 1 2 5 1 6 5 2 54 3 1
(   /0 5 0 1 6 0 2 7 0 1
)   /0 7 0 1 6 0 2 5 0 1
[   /0 4 1 2 6 4 3 0 3 2
]   /0 0 1 2 6 5 3 4 3 2
{   /0 56 0 2 4 2 1 0 2 1 76 0 2
}   /0 76 0 2 0 2 1 4 2 1 56 0 2
^   /4 1 0 2 67 0 2
*   /1 76 0 2 2 4 2 56 0 2
+   /1 0 0 2 34 0 2 6 0 2
-   /3 0 0 3
/   /1 5 0 3
<   /1 5 0 2 7 0 2
>   /1 7 0 2 5 0 2
=   /1 0 2 2 34 0 2 0 1 2
.   /1 1 0 2 67 0 2 4 0 2
,   /1 1 0 2 67 0 2
\\  /1 7 0 3
```

\sqrt	/0 67 23 1 2 67 2 0 1 2
\sum	/0 4 1 2 7 12 2 5 23 2 0 3 2
\prod	/0 0 1 2 4 1 1 6 4 2 0 3 1 2 5 2
\int	/0 45 1 1 6 0 3 54 3 1
Δ	/0 56 4 3 0 3 2 32 5 3
λ	/0 67 0 3 3 23 1 56 23 1
Π	/0 0 1 3 4 1 1 6 4 3 12 0 3 6 5 3
π	/1 0 1 2 4 1 1 6 3 2 12 0 2 6 4 2
σ	/1 45 1 1 67 3 1 01 2 1 23 0 1 4 1 1 0 1 2
Θ	/0 45 1 1 67 4 2 01 3 2 23 5 2 45 1 1 67 4 1 0 2 2
θ	/1 45 1 1 67 3 1 01 2 1 23 4 1 45 1 1 67 3 1 0 0 2
α	/1 5 0 2 43 2 1 2 3 1 01 1 1 7 0 2
γ	/0 67 0 3 54 3 2 6 3 0 12 0 3
δ	/0 34 1 1 56 12 1 70 0 2 65 3 1 4 3 1 32 3 1 0 2 1
β	/0 5 0 3 1 0 3 076 5 2 54 52 2 0 2 1 76 23 1 54 3 2
Ω	/0 0 3 1 32 4 2 10 1 2 765 5 3 0 3 1
ϵ	/1 34 1 1 56 3 1 70 2 1 43 0 1 0 0 2
Φ	/0 45 0 1 67 4 1 01 35 1 2 5 1 34 0 1 2 0 1 6 0 3