2006

Raster to vector conversion: creating an unique handprint each time

William DiBacco

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

Raster to Vector Conversion:
Creating an Unique Handprint each time

Submitted by

William DiBacco

in partial fulfillment of the
Requirements of the degree
Masters in Computer Science

July 14, 2004

Committee:
Advisor: Dr. Peter Anderson
Reader: Dr. K. Bradley Paxton
Advisor: Dr. Vladimir Misic

School of Computer Science
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, New York, 14623
Curriculum Vitae

Bill has been with ADI, Ltd since 2001 and prior to that, its precursor RIT Research Corporation since 1998 and has worked on the Census project evaluating results of Lockheed-Martin system, creating test data for census centers, writing image viewing software, and creating and running internal OCR recognition software. He also investigated and made recommendations towards the task of converting experimental satellite image processing software to production use by the government (specifically, NIMA).

Prior to RIT Research Corporation, Bill worked for Xerox Corp. for 25 years in a number of positions as a business systems integrity and security auditor and business analyst designing and implementing computer information systems and managing a group of programmers. Bill worked with all aspects of Xerox businesses including engineering, manufacturing, procurement, accounting, and marketing.

Bill has a Bachelor of Science in Physics and Mathematics from SUNY Brockport and a Masters in Business Administration from Rochester Institute of Technology. Bill has also taught Computer Science at SUNY Brockport and at Rochester Institute of Technology.

Bill’s computer interests are in the areas of Computer Vision, Expert Systems, Genetic Algorithms, and Neural Networks. He has worked with many computer systems over the years, most recently including Visual Basic, C, Java, Lisp, MatLab, Visual Basic, Perl, and UNIX Scripting Tools.
Acknowledgements

I wish to thank Dr. Roger Gaborski who introduced me to digital image processing and provided me with some of the basic tools in the beginning.

I wish to thank K. Bradley Paxton for his support and suggestions and for permission to use the characters from the HCDC data base. He has provided several important ideas on how the system should work.

I wish to thank Vladimir Misic for keeping me organized and focused. His knowledge in both digital image processing and Matlab have been invaluable. He also had a number of very useful suggestions on organization and structure of this paper and how to efficiently produce it. Without his patience and help, this project would not have come this far. He provided many suggestions that assisted me in establishing direction and choosing courses of action that ultimately succeeded.

Most of all, I would like to thank Peter Anderson who always showed encouragement and flexibility in keeping me moving forward towards this goal. Peter has been a supporter from the beginning and has provided advice and council whenever it is needed.
Abstract

When a person composes a document by hand, there is random variability in what is produced. That is, every letter is different from all others. If the person produces seven a’s, none will be the same. This is not true when a computer prints something. When the computer produces seven a’s they are all exactly the same. However, even with the variability inherent in a person’s handwriting, when two people write something and they are compared side by side, they often appear as different as fonts from two computer families. In fact, if the two were intermixed to produce some text that has characters from each hand, it would not look right!

The goal of this application is to improve the ability to digitally create testing materials (i.e., data collection documents) that give the appearance of being filled out manually (that is, by a person). We developed a set of capabilities that allow us to generate digital test decks using a raster database of handprinted characters, organized into “hands” (a single person’s handprint). We wish to expand these capabilities using vector characters. The raster database has much utility to produce digital test deck materials. Vector characters, it is hoped, will allow greater control to morph the digital test data, within certain constraints. The long-term goal is to have a valid set of computer-generated “hands” that is virtually indistinguishable from characters created by a person.
## Contents

1 Introduction ............................................................................................................... 7  
   1.1 Motivation ........................................................................................................... 7  
   1.2 Development of Hand-printed Character Data Collection (HCDC) and character collection .......................................................... 8  
   1.3 Digital test decks are a useful testing tool ......................................................... 10  
   1.4 Organization ...................................................................................................... 11  

2 Useful definitions ..................................................................................................... 13  
   2.1 Digital character definition ............................................................................... 13  
   2.2 Font definition ................................................................................................... 13  
   2.3 Handprinted character definition ...................................................................... 14  
   2.4 Generating characters from a vector representation ........................................... 16  

3 Process/Algorithm ................................................................................................... 18  
   3.1 Process Summary .............................................................................................. 18  
   3.2 HCDC Data Collection & Preprocessing .......................................................... 19  
   3.3 Shrink the character to a skeleton. Grow a character perimeter to obtain an outline ......................................................................... 19  
   3.4 “Order” the points to establish the curves for the vector representation .......... 21  
   3.5 Determine Control Points via recognition of curvature and selection of key points which define the curves that make up the character ....................... 22  
   3.6 Develop methodology for introducing some variation in the characters .......... 24  
   3.7 Convert to (textual) parametric form (TrueType/Type 1 Font or LUT) .......... 26  

4 Experiment(s) .......................................................................................................... 27  
   4.1 Segmentation of characters ............................................................................. 28  
   4.2 Skeletonization of captured characters ......................................................... 28  
   4.3 Shaping the character ....................................................................................... 30  
   4.4 Generate the chain code .................................................................................. 31  
   4.5 Curvature and choosing control points: ......................................................... 33  
   4.6 Converting raster to vector: ............................................................................ 44  
   4.7 Reshaping the vector character ....................................................................... 45  

5 Appendices ............................................................................................................... 50  

6 Bibliography & Annotated references ..................................................................... 55
Figures:

Figure 1-1 Digitally Simulated Handprinted Text. Each letter was selected as required and pasted together into the desired textual string................................................................. 8

Figure 2-1 Sample of Font Families. Each E above represents a family of characters: a) Times New Roman, b) Old English, and c) Mare................................................................. 14

Figure 2-2 Examples of a Handprinted E. Each one is different from every other one. In addition, there can be lots of variability even though all the E’s were made by the same person........................................................................................................................................ 15

Figure 2-3 Handprinted Font Characters are all the same: top: one “hand”, bottom: different “hand” .............................................................................................................. 15

Figure 2-4 Handprinted manually collected Characters are all different, top: one “hand”, bottom: different “hand” ........................................................................................... 16

Figure 2-5 A set of scripted characters, created parametrically from curves ............. 17

Figure 3-1 Original raster characters and their skeletons. This is the first step towards development of ordering the pixels that comprise the character. ....................... 20

Figure 3-2 Create a character outline from the character's glyph/skeleton. Starting with the skeleton of the character, dilate and then subtract to obtain the outline................. 20

Figure 3-3 Chain Coding: (a) direction of connectivity, (b) 4-connectivity, (c) 8-connectivity. Generate the chain code by detecting the direction of the next-in-line pixel.......................................................................................................................... 21

Figure 3-4 Chain coding the boundry with 8-conn: 2,2,2, ... (start at p1, then in the clockwise direction) ............................................................................................................ 22

Figure 3-5 A sample of possible Control Points (CPs) for an machine print E ............ 23

Figure 3-6 Drawing the B from a set of curves. The uppercase B is composed of 3 curves ................................................................................................................................... 23

Figure 3-7 Creating a character from its glyph/skeleton. Starting with the skeleton, dilate to make it “fatter”. The new character will always approximate the shape of the original closely............................................................................................................. 24

Figure 3-8 Affine transform: the red 8 is the original, the blue 8 is created by a shearing in both the x and y directions.................................................................................. 25

Figure 4-1 Converting the raster HCDC characters to their skeleton representation .... 29

Figure 4-2 Spurs, due to the digital raster nature and shape of the object. These did not, in general create a problem for the conversion (raster to vector) ........................................ 29

Figure 4-3 The original 2 is in blue and the transformed 2 is in red.................................. 31

Figure 4-4 Endpoints (in blue) of the skeleton can be used to find places of high curvature on the character’s outline......................................................................................... 32

Figure 4-5 Junction Points (in orange) of the skeleton can be used to find places of high curvature on the character’s outline............................................................................. 33

Figure 4-6 High Curve points (in purple) are close to endpoint areas and junction areas. They are also in other areas. ................................................................. 33

Figure 4-7 Control Points for the E .................................................................................. 34

Figure 4-8 Secant line relationship to tangent line ........................................................ 35

Figure 4-9 Methods for determining curvature: (a) averaging from 1st point; (b) averaging from the middle point; (c) averaging over the span from the middle point .......................................................................................................................... 36
Figure 4-10 (a) Low and (b) High curvature ................................................................. 37
Figure 4-11 The 1 has two endpoints, at locations of very high curvature on the 1’s outline. The points on the outline are indicated by the light colored circles (orange above and green below). ................................................................. 38
Figure 4-12 The grown outline of the 2 ........................................................................ 38
Figure 4-13 The endpoints (blue) and junctions (purple) in the context of the grown character outline..................................................................................................................... 39
Figure 4-14 The light green arrows highlight the intersection/junction points exist........ 40
Figure 4-15 Spurs can indicate false junctions, When the 2 has a complete loop, there is no false junction..................................................................................................................... 40
Figure 4-16 Character Perimeters -inner and outer....................................................... 41
Figure 4-17 The 4, with two junctions, finds control points. (a) The 4, with its junctions circled in blue, (b) the analysis leading to finding outline points from the first junction, (c) the analysis leading to finding outline points from the second junction. All points are in positions of high curvature................................................................. 42
Figure 4-18 The 8, with a two junctions finds control points. (a) The 8, with its junctions circled in blue, (b) the 8, showing the curves which make up the glyph of the 8 surrounded by its outlines, (c) the analysis leading to finding outline points from the first junction, (d) the analysis leading to finding outline points from the second junction. All points are in locations of high curvature, however, the points spread on both the outer and inner outlines.................................................................................. 43
Figure 4-19 Control Points come from 3 processes: (1) the endpoint process, (2) the junction process, and (3) the curvature process.......................................................... 44
Figure 4-20 The control points are used to develop a parametric form of the character.. 45
Figure 4-21 Reshaping the character by changing just one parameter......................... 46
Figure 4-22 Taking a simple shape (the red 8), and generating many variants from it (in blue).......................................................................................................................... 47
Figure 4-23 The figure 8: the original vectorized 8 (comes from the HCDC data base); row two has shape modifications, each tilted up a greater amount as we progress from left to right; row three has shape modifications, each tilted down a greater amount as we progress from left to right; row three has shape modifications, each tilted down a greater amount as we progress from left to right. Summary and Future Research..................................................................................................................... 47
1 Introduction

When a person composes a document by hand, there is random variability in what is produced. That is, every letter is different from all others. If the person produces seven a’s, none will be the same. This is not true when a computer prints something. When the computer produces seven a’s they are all exactly the same. However, even with the variability inherent in a person’s handwriting, when two people write something and they are compared side by side, they often appear as different as fonts from two computer families. In fact, if the two were intermixed to produce some text that has characters from each hand, it would not look right!

What would be useful in order to produce realistic looking handprint from a computer is the ability, starting with a set of characters digitally collected from someone, to be able to change them in small ways so they do not look like the original set, but the message appears to be produced by the same person. In this project, we will convert digital characters from a raster format to vector and by shifting the points produce a set of characters that look different from one another.

1.1 Motivation

The goal of this application is to improve the ability to digitally create testing materials (i.e., data collection documents) that give the appearance of being filled out manually (that is, by a person). These materials are destined to assist in developing, scoring, and validating the output of document processing systems that specifically employ Intelligent Character Recognition/Optical Character Recognition/Optical Mark Recognition (ICR/OCR/OMR referred to as an OCR system) technology. The current technology is
employed extensively, but not yet to the point of complete confidence in the results. With these digitally created documents/forms, a systems baseline can be established, and improvement progress can be guided and monitored. Many of these systems, which last many years and are the main information gathering vehicle of many commercial concerns, could be improved to produce a higher quality output in less time, with less effort.

These testing materials will be referred to as digital test decks.

![Digital Test Data](image)

Figure 1-1 Digitally Simulated Handprinted Text. Each letter was selected as required and pasted together into the desired textual string.

1.2 Development of Hand-printed Character Data Collection (HCDC) and character collection

Initially inspired by a large collection of respondent-completed forms, we looked at the problem of generating test data. Test data consists of a completed form to submit to the document processing system. The form progresses through a set of preprocessing steps, which capture the data fields from the form, submission of these fields to the recognition engine, and subsequent post processing steps which verify and store the data. The data fields that are captured originate as the result of a respondent answering the questions in the space provided to increase the accuracy of the OCR system. While the respondent can answer the questions free-form in many instances, neatly handprinted answers yield the best results, and are typically requested of the respondent.
The lowest constituent part of the data field is an individual character. It is not until the respondent-marks on a completed data collection form are decomposed that they become useful in the generation of testing materials. While the data fields can be useful for the data collection system in which they originated, this data unit limits their usability for testing other data collection systems. As we began extracting the characters, we realized, in general, that there is only a subset of the universe of useful characters on a particular single form. But, when constructing a new test form from the collected set, if characters from multiple forms are used, it is noticeable and objectionable because each person has a characteristic style in writing (and printing). The form no longer appears as one created by a single person, but rather like the composite of several persons’ hands and not representative of actual documents that the system will be processing.

This led to development of the Handprinted Characters Data Collection (HCDC) instrument (see Appendix 1). It contained 5 pangrams (sentences which contain all 26 letters of the alphabet) designed to collect at minimum 5 of each character of the alphabet, both upper (A-Z) and lower (a-z) case and the numbers (0-9). We also collected some special characters (, ; ‘ _ -). Additional characters may be needed in the future. In addition to the special characters, other characters from such sources as the French, Spanish, and German alphabets may need to be added. We currently have a (raster) data base that has been divided into individual characters and organized by the “hand” that created it.
These “hands” are kept together for use in testing materials. As is generally accepted, every time a person writes is (slightly) different from every other time. We would like a representation scheme that allows us to use what we have in new and novel ways. Ideally, we want a methodology that permutes the shape of each occurrence of a character as it is produced for testing purposes. In this way, we can effect a simulation of a person’s handprint. However, if the change is too great, the characters do not look like they were created by a single person.

We next developed a set of capabilities that allow us to generate digital test decks using this raster database. This system takes as input a set of documents/form blanks designed to collect data for subsequent processing by an OCR system. Also as input is the test data that the OCR system should produce from the document processing efforts. The output of our system is a digital test deck and a TRUTH file. The digital test deck is a set of (paper) forms, created digitally which simulate real forms. The TRUTH file is a control check with which to score the OCR system.

1.3 Digital test decks are a useful testing tool

We wish to expand these capabilities using vector characters. The raster database has much utility to produce digital test deck materials. Vector characters, it is hoped, will allow greater control to morph the digital test data, within certain constraints, to test various OCR system features and provide useful feedback on system tuning parameters and system weaknesses. We can define the limits of the systems capabilities. The long-
term goal is to have a valid set of hands that is virtually indistinguishable from characters created by a person.

These marks/characters will provide digital test decks with great flexibility and validity. Test decks gathered by manual methods, that is filled out by (an army of) individuals – can take a long time, be filled with errors, are not easily reproducible, and do not initially contain a TRUTH file. Digital test decks come with all the power of a digital tool: excellent reproducibility on demand, completely known output, ability to interject characteristics as required, and the ability through regular improvement of the process to continually decrease the cost to produce a deck.

This project is expected to provide greater control and to extend the existing base of collected characters using digital character shape modification techniques. We also expect to develop additional methodologies that allow us to manage the characters we have collected.

1.4 Organization

Chapter 2 has a set of definitions that are useful in understanding this paper. Making the distinction between a handprinted font and the handprinted characters that are used within this project lays some groundwork useful in establishing its goals and the context of much of the work. Chapter 3 contains a description of the algorithms being used. There is relatively little work being pursued in this specific area of endeavor. Much of the work that has appeared in print, while using tried and true digital processing techniques, has
indicated that topics of research have not been studied in depth at this point. I believe there are opportunities to make useful contributions to this area of knowledge. I think that useful by-products will be commercially viable. Chapter 4 describes experiments to achieve working code and choose among several competing algorithms.

For completeness and consistency, I have introduced the concept of conversion from vector format to that of font technology. This is a future activity and is not within the scope of this project. Thus, I mention some goals and aspects of this task, but do not, during this project, develop code to implement the task.
2 Useful definitions

2.1 Digital character definition

Let us define characters. A character is a collection of curves

\[ L = \{c_1, c_2, \ldots, c_n\} \] \{2.1\}

These curves completely define the shape of the letter once they are grouped together appropriately, but do not have to be connected in general. Once the space within the connected curves is shaded/filled in the characters take shape – as, for example, the typical computer fonts with which we are so familiar. Generally, font technology takes advantage of this in ways that allow efficient representation and storage of documents, and printing technology extends this capability to the myriad ways of producing the printed page.

2.2 Font definition

Fonts have special characteristics/aspects in going from one font family to another (see Figure 2-1). The characters are constructed in very predictable ways. Some characters, such as the i and the j, have multiple parts. Fonts can be normal, bolded, italic, subscript and superscript, and many other features that distinguish them from one another, such as serif or sans serif, but all the characters of a font-group derive from the same “base shape and/or other distinguishing characteristic”. Fonts have evolved into a digital vector representation. This parametric representation gives them the versatility they have today.
Scalable or Vector Font

A font represented in an object-oriented graphics language such as PostScript or TrueType. Such fonts are called scalable because the representation of the font defines the shape of each character (the typeface) but not the size. Given a typeface definition, a scalable-font system can produce characters at any size (or scale).

Aside from offering innumerable sizes of each font, scalable fonts have an added advantage in that they make the most of an output device’s resolution. The more resolution a printer or monitor has, the better a scalable font will look.

Scalable fonts are often called outline fonts because the most common method of representing scalable fonts is to define the outline of each character. Scalable fonts are also called object-oriented fonts or vector fonts.


2.3 Handprinted character definition

Handprinted characters can easily be distinguished from fonts (see Figure 2-2) and in fact have different characteristics for each instance of a character.
The E above is composed of 3 distinct segments that the person who wrote this E used to comprise the letter. In the first instance, these are all part of one (touching) E. However, it is very possible for the person to use two or even three segments and all represent this person’s “hand”. Each of these handprint characters is of the person’s E, and each is different in some random (but not too random) way.

It is possible to create a handprint font. One constructs the alphabet and either using a freely downloadable program or paying someone a nominal amount can get a font of their “hand”. But, it is a font, one a, one b, and so forth to one z (see Figure 2-3). When words are written using the font, the characters (of a type size and other typographic characteristics) do not vary:

![Figure 2-3 Handprinted Font Characters are all the same: top: one “hand”, bottom: different “hand”](image)

This is not the same as the handprinted characters we have collected (see Figure 2-4). For instance, if we collected 20 E’s from a person’s hand, they are all different from one another:
An interesting approach to the organization and use of these hands is to convert them to vector format and use the power of the vector parameterization to manage small and random changes each time the character is used.

2.4 Generating characters from a vector representation

The approach within this project is to take those characters collected by the HCDC process and develop a glyph (i.e., skeleton) which will become a stem from which the family of character(s) can grow. An advantage of this approach is that multiple examples of the “hand” were collected. Thus, growing example characters from the same hand means applying the same growing methodology to multiple characters. This is expected to yield results that still appear to be in the same person’s writing.

The process will be to select a character from the HCDC data base, generate a glyph, grow a new character from this, then develop a parametric representation. Once the parametric curves are developed, they can be used to guide the creation of letters and data fields for the test deck.
Figure 2-5 A set of scripted characters, created parametrically from curves

With a vector representation (see Figure 2-5), such as typically used in computer fonts, characteristics are altered easily to the advantage of the printed document. Such characteristics as character spacing, weight, italics, serif, slants, size to name just a few are easily reviewed and adjusted to satisfy the needs of the author.
3 Process/Algorithm

3.1 Process Summary

The overall process used for this project is as follows:

1. Capture of handprinted characters from the HCDC process

2. Skeletonization of these (raster) character images and subsequent “growing” of the character outline.

   Subsequent Project: For this project, character dilation is chosen as proof of concept and additional methods are out-of-scope currently. This growing is an example of applying a “brush/pen” to the character skeleton. In this way, a methodology will be developed that allows one to modify the shape of the original character in prescribed ways, thereby creating new “hands” from the original.

3. Ordering the pixels, in a clockwise direction which lays the groundwork for generating the curves which comprise the character.

4. Choosing a set of control points to submit for curve parameterization

5. Generating the vector representation and generating the character from its parametric form
(6) shape morphing from the base shape

(7) choosing a storage format for sending this form to other applications

Subsequent Project: For this project, the internal format must be understood by MATLAB, but not by external applications. In a future project, the author expects to implement this storage format in a generalized form to allow usage by commercially available forms processing packages.

3.2 HCDC Data Collection & Preprocessing

The set of characters used in testing this system came from the HCDC process (see Appendix 1 for the HCDC form). These characters were rendered from a commercial process designed to support document scanning. Output of this process is very clean and well labeled and segmented characters. These characters are handprinted (mostly by college students). They are good examples (as opposed to sloppy) of the letters of the alphabet and numbers (see Appendix 2 for an example). A patent is pending and a further description of this process is not necessary to the project described within this paper.

3.3 Shrink the character to a skeleton. Grow a character perimeter to obtain an outline.

The first step in this process is for the extraction of a character glyph for subsequent processing (see Figure 3-1). This glyph is the character’s skeleton that is 1 pixel wide. The glyph conforms to the contour of the character faithfully, yet is a very compact representation. The glyph also will provide very useful information about the desired features from the original character. From this glyph, a character can be “grown”, the
boundary pixels can be ordered, and some to most of the control points can be determined.

Figure 3-1 Original raster characters and their skeletons. This is the first step towards development of ordering the pixels that comprise the character.

Initially we will “grow” a character from the glyph by simple dilation (see Figure 3-2). This may involve both the outer and inner contours. This will constitute the character outline. Useful output from this portion of the process is an outline of the characters that is 1 pixel wide.

Figure 3-2 Create a character outline from the character’s glyph/skeleton. Starting with the skeleton of the character, dilate and then subtract to obtain the outline.
3.4 “Order” the points to establish the curves for the vector representation

In this next step in the process, in order to parameterize the curves from which the character is composed, the data points that comprise the outline of the character must be ordered. Thus, if there are 100 points composing the outline, they must be ordered $P=\{p_1, p_2, p_3, \ldots, p_{100}\}$, where $p_n$ is adjacent to $p_{n+1}$ and next-in-line as we proceed in a clockwise direction. Once this is accomplished, we can fit a set of curves to the points.

To accomplish this, the object boundary/outline is represented using chain coding (using the Freeman chain code technique [Freeman,1961], see Figure 3-3 & example Figure 3-4). Chain coding provides the points in relative position to one another, independent of the coordinate system.

In this methodology of using a vector of connecting neighboring contour pixels, the points and the outline coding are captured. We choose to proceed in a clockwise direction.

Figure 3-3 Chain Coding: (a) direction of connectivity, (b) 4-connectivity, (c) 8-connectivity. Generate the chain code by detecting the direction of the next-in-line pixel.
3.5 Determine Control Points via recognition of curvature and selection of key points which define the curves that make up the character

The next step is to select a subset of the chain code points and use these as the control points for the parametric curves that define vector representation. While all points are useful in defining the shape of the character, not all are necessary and, in fact, some are redundant. Current font technology uses only a small set of *important* points to define font family shapes.

The concept of control points in font technology is simple (see Figure 3.5). These points control the parameters that describe the curves that comprise the characters. Choosing control points is very important to the shape of the character. More control points are needed to describe the curves at the positions of high curvature and fewer control points are needed for straight stretches of the character outline.
Generating the curves that represent the character is next. Font technology uses curves that are the result of quadratic and cubic equations. With the control points and the curve parameters a specific character can be generated.

Main Algorithm for finding the Control Points, for …say, a “B”:

1. Recognize the crosspoints (junctions):
2. Split the B into a set of distinct pieces/curves
3. Crawl along each curve and when the curvature changes “enough” make a control point
Once the control points are chosen, they can be used to (re)generate the character from a curve representation (see Figure 3-6).

3.6 Develop methodology for introducing some variation in the characters

The step which is expected to yield the most benefit to digital testing data is to then manipulate this parametric representation to change the shape of the character in new ways, yielding new yet realistic digital data.

One method chosen must simulate the writing instrument, such as a pen or pencil or “felt tip” pen. So, the skeleton representation is a good starting point. This is then modified with a “brush stroke/pen stroke” which simulates the handwriting after it has been introduced to the page (see Figure 3-7).

A second method is applying an affine transform (in this case, a shear operation in the x and y directions both at the same time) to the control points, shifting them in space and they generating the reshaped character’s vector representation (see Figure 3-8).
Figure 3-8 Affine transform: the red 8 is the original, the blue 8 is created by a shearing in both the x and y directions.

A third method is to use a projection transform from the existing plane onto a shifted plane (see Figure 3-9):

Figure 3-9 An 8. First the 8 is vectorized (i.e., establish the Control Points (CPs) and generate an 8 from parametric data). From the vector 8, using a simple transform of the vectorized shape, create a new 8.
Exploring this topic is worthy of another project and as such, only several methodologies were explored and utilized in this project. The ultimate goal of all this work is to implement this as part of the overall system. It is the author’s belief that, in the context of making a “new hand”, this is a complex area to investigate. It is a very worthwhile one also because in exploring the capabilities of the systems which are to be tested, the types of variability that can be implemented in a “hand” are to be generated in a highly controlled fashion.

3.7 Convert to (textual) parametric form (TrueType/Type 1 Font or LUT).

Finally, the information can be represented by either a Font (family) or a Look-Up-Table (LUT).

This task is out of scope for this project. It is mentioned for completeness sake as the ultimate goal of all this work is to implement this task as part of the overall system.
4 Experiment(s)

Segmentation:

Outlining:

Skeletonization of the handprinted characters data base (to produce character outline). Development of the perimeter of the character to obtain the outline.

Ordering the points:

Create a chain code to “order” the points on the character outline to establish the motion or flow in which the character was created

Finding the control points

Endpoints

Junction points

High curvature

Determine Control Points via recognition of curvature and selection of key points which define the curves that make up the character

Parametric representation

Generate Vector Format via the use of Bezier curves and verify the curves represent the characters

Develop methodology for introducing some variation in the characters

Conversion to a Font (out of scope)

Convert to (textual) parametric form (TrueType/Type 1 Font or LUT).
The need is for the extraction and processing of the contour of the character. This may involve both outer and inner contours. Some characters have only an outline, such as C and E, while others have outer outlines as well as inner outlines, such as A, B and D.

4.1 Segmentation of characters

During the data collection phase of this project, the HCDC form was used to gather samples of personal handprint. For the purposes of generating digital test decks, which are forms with the responses constructed and filled in from the hand print samples provided, the characters were labeled and separated from one another.

4.2 Skeletonization of captured characters

Our goal in this activity is to calculate the character representation that faithfully preserves the most features. Creating this glyph to represent the character is the first step in this process. This is because the exterior boundary of a character is not as interesting as its internal structure. Thus, we create a stick figure of the character.

Skeletonization takes the raster characters and “shrinks” them to 1-pixel wide while preserving the topology of the character (see Figure 4-1). Knowledge of the skeletonization algorithm and its structuring element provides knowledge of its inverse, so the original object can be reconstructed from its skeleton. From this representation, valuable information can be gathered to aid in generating the vector representation.
Figure 4-1 Converting the raster HCDC characters to their skeleton representation

As this applies to the characters within the HCDC data base, we shrink the characters to 1 pixel wide, while maintaining the general shape of the originally collected character. This is a simple single-step operation. It is not always possible with the skeleton operation to achieve a perfect implementation of the specification (see Figure 4-2), but Matlab appears to provide a very good one (note: the author did not delve into the specific Matlab algorithm). In explanation, Matlab has - to reduce all objects in an image to lines, without changing the essential structure of the image, use the bwmorph function. This process is known as skeletonization. Specifically, using the command

\[ \text{bwmorph(image, 'skel', N)} \]

Where \( N = \infty \), the algorithm removes pixels on the boundaries of objects without allowing objects to break apart.

![Skeletonization Diagram]

Figure 4-2 Spurs, due to the digital raster nature and shape of the object. These did not, in general create a problem for the conversion (raster to vector).
4.3 Shaping the character

This method of developing the outline of the character, that is, dilating the skeleton of the character to establish its shape demonstrates one of many methods of reshaping the character.

Another method that was pursued within the project was to apply various affine transforms to the skeleton of the object to achieve a changed shape (see Figure 4-3). An affine transform is a linear transform of the object, followed by a translation. Thus, its goal is to map straight lines to straight lines. One can apply transforms in series and achieve unique results and much additional study is required before the author masters this area. At this point, the focus has been on rotation and shearing, more so that translation or scaling. In a sample method displayed below to demonstrate the concept, a shear was applied to the 2 and, to the author, it appears to have achieved some of the characteristics of an italics character:
There are other transforms that could be applied as well as combinations of transforms that will change the shape in small (hopefully manageable) ways. One area of investigation is to change the shape of the skeleton. In this method, performing the raster to vector conversion as the first step would establish the CPs that could then be subsequently be “moved” to change the character’s shape. Initial, but very limited, reading by the author caused the focus on producing the outline image prior to the raster to vector conversion, and generating the vector of the character’s outline. This topic seems to have a lot of merit and so needs to be explored in more depth in the future.

4.4 Generate the chain code

Originally considered a compression method for object boundaries in a digital image, this method is applied here to help establish the control points for the handprint characters. It is also being calculated while “thinning” the boundary of some points. This method
eliminates the unneeded/redundant points from future calculations. These eliminated points are those that do not add either to the shape of the character, nor will they add to the determination of the control points for the purposes of the parametric representation.

If we then use the endpoints from the skeleton of the character (see Figure 4-4), combined with the outline created by thickening the skeleton to grow the character, we can determine control points by finding those points in close proximity to the endpoints. These points will be in areas associated with high curvature, thus determining in large part the overall shape of the character.

Figure 4-4 Endpoints (in blue) of the skeleton can be used to find places of high curvature on the character's outline

In addition, there will also be junctions where major strokes of the character skeleton come together (that is, cross as in 8 or X, but also in the H and the T where there are junctions unlike the X). These intersections can also be used to determine places of high curvature in the character outline (see Figure 4-5).
Figure 4-5 Junction Points (in orange) of the skeleton can be used to find places of high curvature on the character’s outline

Lastly, the outline itself can be used to establish points of high curvature (see Figure 4-6). These points will come in areas in close proximity to the endpoints. They will also come in areas in close proximity to the junction points. But, in addition, they will come in other areas.

Figure 4-6 High Curve points (in purple) are close to endpoint areas and junction areas. They are also in other areas.

4.5 Curvature and choosing control points:

We will look at four methods for picking the control points. Each indicates some curvature and thereby provide hints at which of the outline points make good control points. The control points control the parameters that describe the curves that comprise the characters. Choosing control points is very important to the shape of the character. More control points are needed to describe the curves at the positions of high curvature
and fewer control points are needed for straight stretches of the character outline (see Figure 4-7).

**Figure 4-7 Control Points for the E**

### 4.5.1 Curvature of the underlying outline

In analyzing curvature, it is recognized that the tangent line is the slope at a point. That is, the change in the vertical direction divided the change in the horizontal direction. For a straight line:

\[
y = mx + b \quad \{4.5.1\}
\]

\[
m = \text{slope} = \frac{\Delta y}{\Delta x} = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad \{4.5.2\}
\]
Figure 4-8 Secant line relationship to tangent line

The slope of the tangent at x is the limit of this expression as h tends to 0:

\[
\lim_{{h \to 0}} \frac{f(x+h) - f(x)}{h} \tag{4.5.3}
\]

Often, for numerical analysis, it is useful to use the secant line to approximate the slope.

Using this for the character (chain code), it is important to recognize that the actual slope is not critical, but rather curvature is high when the slope is changing rapidly and low when the slope is changing slowly.

Three methodologies were compared to determine curvature. All three were based upon averaging the slope lines over a number of points. In the first method (see Figure 4-8a), we took the first point in a five-point set, and calculated a running average of lengths.

\[
\| f(t+h) - f(t) \| \tag{4.5.4}
\]
In the second method (see Figure 4-8b), we took the middle point in a 5-point set, and calculated a running average of the tangent.

\[
\frac{(f(t+h)-f(t))}{h} \quad \{4.5.5\}
\]

In the third method (see Figure 4-8c), we took the middle point as the center, and tried a variation of the secant line method of determining the slope based upon the formula:

\[
\frac{f(t+h)-f(t-h)}{2h} \quad \{4.5.6\}
\]

where \( f(t) = <x(t), y(t)> \) \{4.5.7\}

Figure 4-9 Methods for determining curvature: (a) averaging from 1st point; (b) averaging from the middle point; (c) averaging over the span from the middle point

A fourth method was finally decided (see Figure 4-9). This method is based upon the fact that curvature will lessen the distance between points, we can use another formulation. In this method, we go out N points, and calculate the distance between point 1 and point N, we then calculate the distance between each of the successive points in between. The sum of these is the denominator.
Thus, we see in the above Figure 4-9, that the curvature $C_a = \frac{D_a}{\Sigma d_i}$ is less than that represented by $C_b = \frac{D_b}{\Sigma d_i}$.

### 4.5.2 Finding endpoints

Endpoints are important indicators of locations of high curvature. For instance, the character 1 has two endpoints and these indicate several points that bracket the areas of high curvature (see Figure 4-11).
Figure 4-11 the 1 has two endpoints, at locations of very high curvature on the 1’s outline. The points on the outline are indicated by the light colored circles (orange above and green below).

Another example (see Figure 4-12):

Figure 4-12 The grown outline of the 2

And the results of finding the points close to the endpoints is (see Figure 4-13):
One assumption is that the chain code for the outline will yield a smooth, unbroken curve as it grows a perimeter around the skeleton and completely encompasses the character. This is reasonable as the outline starts as a 1-pixel wide skeleton and moves outward as it grows. But when there is a loop in the character, such as that of a 2 or an 8 or an R, then there are multiple chain codes to represent the outline of the character.

4.5.3 Finding Junctions/Intersections:

An intersection, also referred to as a junction, is a location where the chain code goes in more than a single direction (see Figure 4-13). Generally, this is where multiple (shape-defining) curves meet. However, it is not generally clear from the analysis how many curves meet and which portions of the character boundaries belong to which curve. Also, there are instances when, due either to the rendering (i.e., scanning and interpretation software) or due to the motion of the person creating the character, when an intersection of curves is undetectable. This makes using junctions relatively unreliable in establishing the character’s signature or characteristic curves.
How does one use intersections to develop the “curves” for the character?

As you can see from the above figure (see Figure 4-14), the 0 and the 1 do not have any junctions and so, can be represented as a single curve.

The 2 has two junctions, with the lower junction clearly being an artifact of the lack of a loop in the original 2 (see also Figure 4-15):

Figure 4-15 Spurs can indicate false junctions, When the 2 has a complete loop, there is no false junction
The 3 has a single junction and is composed of two curves.

The E and H have 2 junctions each, but the E should have three junctions and be composed of 4 curves (that is, lines), while the H is composed of three curves.

The T has a single junction and is composed of two curves.

The 8 has a single junction and could be made from one or two curves.

All of the characters could be composed of more than the “ideal” number of curves.

However, it must be recognized that the characters are typically four or five pixels wide (in our examples), and so since the outlines of the characters are used in the creation of the chain code from which the control points are to be selected, as would be the case for fonts, the problem almost disappears (see Figure 4-16):

![Figure 4-16 Character Perimeters -inner and outer](image)

For this case, the problem of junctions is replaced by the problem of recognizing where the control points should be to capture the best-match curvature.

4.5.4 Using Junctions to find control points

Junctions are useful in finding some of the control points because they are located in close proximity to places of high curvature. The 4, for instance, has a junction where the lower horizontal line intersects the long vertical line. In our example, because of the shape of the horizontal portion, we have two junctions.
Figure 4-17 The 4, with two junctions, finds control points. (a) The 4, with its junctions circled in blue, (b) the analysis leading to finding outline points from the first junction, (c) the analysis leading to finding outline points from the second junction. All points are in positions of high curvature.

The 4 has a Y-style junction (see Figure 4-17). That is, in finding the outline points indicated by the junction point, we first find three points from the lines emanating from the junction, we then find the three midpoints of the lines connecting these three points,
from this, we project three lines which bisect these lines and when these intersect the outline, we find the points on the outline in areas of high curvature.

![Diagram of character outline and control points](image)

**Figure 4-18** The 8, with a two junctions finds control points. (a) The 8, with its junctions circled in blue, (b) the 8, showing the curves which make up the glyph of the 8 surrounded by its outlines, (c) the analysis leading to finding outline points from the first junction, (d) the analysis leading to finding outline points from the second junction. All points are in locations of high curvature, however, the points spread on both the outer and inner outlines.

The 8 has an X-style junction. This indicates four points on the character outline. However, the four points are not on the same outline, some are on the outer outline and some are on the inner outline.
4.6 Converting raster to vector:

Once the control points are determined, these can be used to generate the splines that define the character in vector format (the following 2 figures, Figure 4-19 and 4-20 graphically depict the identifying of the control points of interest and subsequent use of these control points to “create” a character outline):

Figure 4-19 Control Points come from 3 processes: (1) the endpoint process, (2) the junction process, and (3) the curvature process
Figure 4-20 The control points are used to develop a parametric form of the character

4.7 Reshaping the vector character

Now that we have the control points, we can apply character projection to reshape the character in new ways (4 examples of this reshaping follow, see Figure 4-21):
This method allows for parametric manipulation of the points. In this case, we shift the character up or down as we proceed from left to right across the character’s control points. The greater the number, the greater the shifting, while the leftmost column remains anchored to its original location. In this way, the character remains approximately the same size and maintains approximately the same characteristic micro-features. In this way, it is hoped we can simulate a left-hand or a right-hand modification to the characters, with the base “handedness” (i.e., the characters still look like they were written by the same person) remaining homogeneous.
Thus, we can get a very large number of shapes from a simple change in one or two parameters (see Figure 4-22). This can be done randomly to an entire hand at the time of data generation.

Thus, a “set” of 8-mods, by this method would look (see Figure 4-23):

Figure 4-22 Taking a simple shape (the red 8), and generating many variants from it (in blue)

Figure 4-23 The figure 8: the original vectorized 8 (comes from the HCDC data base); row two has shape modifications, each tilted up a greater amount as we progress from left to right; row three has shape modifications, each tilted down a greater amount as we progress from left to right; row three has shape modifications, each tilted down a greater amount as we progress from left to right.
5 Summary and Future Research

The method described in this paper appear to work for transforming the characters from raster to vector for the purposes desired. The unique or interesting parts are in using the HCDC data base as input and in implementing character morphing. More generally, having handprinted characters, collected in large numbers, provides useful information on how people’s handprint varies while remaining characteristic of a single “hand”. Choosing a particular methodology to morph the “hand” to another was inconclusive from the work in this paper.

The HCDC data base is a set of real handprinted characters and as such, the model for what we wish to achieve: a unique character each time it is used (or as is the hope, electronically generated). Developing code to codify the characters in new and unique ways was an interesting activity. Different aspects of the algorithms provide additional food for thought and areas for additional discovery, such as providing a vector representation of the character’s skeleton to submit to the modifying algorithms.

Much additional work is needed on reshaping the character, especially testing to determine if the all the characters from a single “hand” still appear to be from the same hand, albriet not matching the same person who contributed the hand originally.

This work will take two directions: (1) additional morphing of the skeleton of the character and (2) experimentation with the shifting of control points in the outline vector formulation.
Also, there is the work in converting the output into a font.
6 Appendices
Appendix 1 Handprinted Characters Data Collection Form (HCDC)

Handprint Character Data Collection Form

Instructions: (Upper case and numeral form)
This form is for the collection of a single hand PRINTED font.
Please use a black pen for the best results.
Print capitals in the space provided. Please keep all letters separate.
Do not scratch-out. Do not include commas or periods.
Follow the examples given at the top of the form.

ABCDEF
HIJKLMNOPQRSTUVWXYZ0123456789

THE QUICK BROWN FOX JUMPS OVER A LAZY DOG.
THE QUICK BROWN FOX JUMPS OVER A LAZY DOG

0123456789012345678901234567890123456789

THE FIVE BOXING WIZARDS JUMP QUICKLY.

PACK MY BOX WITH FIVE DOZEN LIQUOR JUGS.

PLAYING JAZZ VIBE CHORDS QUICKLY EXCITES MY WIFE.

SPHINX OF BLACK QUARTZ, JUDGE MY VOW.

BREEZILY JANGLING $3,416,857,209 WISE ADVERTISER

AMBLES TO THE BANK, HIS EXCHEQUER AMPLIFIED.
Appendix 2 A set of characters from a “hand”

Characters from the hand: BDibacco.

<table>
<thead>
<tr>
<th>AAAAAA</th>
<th>aaaaaa</th>
<th>000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBBB</td>
<td>bbbb</td>
<td>111111</td>
</tr>
<tr>
<td>CCC</td>
<td>cccc</td>
<td>222222</td>
</tr>
<tr>
<td>DDDD</td>
<td>dddd</td>
<td>333333</td>
</tr>
<tr>
<td>EE</td>
<td>ee</td>
<td>4444</td>
</tr>
<tr>
<td>FF</td>
<td>ff</td>
<td>555555</td>
</tr>
<tr>
<td>GG</td>
<td>gg</td>
<td>666666</td>
</tr>
<tr>
<td>HH</td>
<td>hh</td>
<td>777777</td>
</tr>
<tr>
<td>II</td>
<td>ii</td>
<td>888888</td>
</tr>
<tr>
<td>JJ</td>
<td>jj</td>
<td>999999</td>
</tr>
<tr>
<td>KK</td>
<td>kk</td>
<td>0000</td>
</tr>
<tr>
<td>LL</td>
<td>ll</td>
<td>1111</td>
</tr>
<tr>
<td>MM</td>
<td>mm</td>
<td>2222</td>
</tr>
<tr>
<td>NN</td>
<td>nn</td>
<td>3333</td>
</tr>
<tr>
<td>PP</td>
<td>pp</td>
<td>4444</td>
</tr>
<tr>
<td>QQ</td>
<td>qq</td>
<td>5555</td>
</tr>
<tr>
<td>RR</td>
<td>rr</td>
<td>6666</td>
</tr>
<tr>
<td>SS</td>
<td>ss</td>
<td>7777</td>
</tr>
<tr>
<td>TT</td>
<td>tt</td>
<td>8888</td>
</tr>
<tr>
<td>UU</td>
<td>uu</td>
<td>9999</td>
</tr>
<tr>
<td>VV</td>
<td>vv</td>
<td>0000</td>
</tr>
<tr>
<td>WW</td>
<td>ww</td>
<td>1111</td>
</tr>
<tr>
<td>XX</td>
<td>xx</td>
<td>2222</td>
</tr>
<tr>
<td>YY</td>
<td>yy</td>
<td>3333</td>
</tr>
<tr>
<td>ZZ</td>
<td>zz</td>
<td>4444</td>
</tr>
</tbody>
</table>
Appendix 3: Vector Characters and Shape-changed Vector Characters

3a Original HCDC Raster Characters

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
```

3b HCDC Raster to Vector Characters

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
```

3c Vector to Distorted Vector Characters

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
```
7 Bibliography & Annotated references

Letters & Numbers, ADI, LLC

As a part of the Digital Test Deck™ (DTD™) product strategy, many handprinted characters were collected using a methodology called Handprinted Characters Data Collection (HCDC™). The data base of collected characters was scanned and used as input data for subsequent operations/algorithms/testing for this thesis.
Previous masters projects or theses

Yampolskiy, R., Feature Extraction Methods for Character Recognition, Masters Thesis, Rochester Institute of Technology, May, 2004

A compendium of methods for analyzing and preparing characters for submission to OCR engines. The author has developed a taxonomy of methods.
Books

Pratt, William K., Digital Image Processing, John Wiley and Sons, 1978,

Considered a classic in the field of digital images. The book was written to serve as a text for computer science graduates or a reference for practicing engineers and scientists in digital image processing. This book covers in exhaustive detail, beginning with the mathematical representation of discrete images and continuing with all aspects of the application of various operators to improve image quality, analysis, and understanding.


Image processing can be considered as consisting of two components: (1) the application of a range of mathematical transformations to an image, to improve the quality of an image, such as rendering something that was not visible as visible; (2) extraction of features from the image for further analysis. This book combines the concepts of expert systems and image processing and presents them to final year and post graduate computer science students.


This textbook attempts to provide a “complete” coverage of the subject matter for students ranging from the undergraduate with very limited experience through the Masters student researching a specific topic.


This book is a tutorial in image processing, providing explanations of the basic concepts. In the process, the book provides the C algorithms and code to build an image processing software system which runs on a PC. The book is based upon a series of articles published in The C Users Journal from 1990 through 1993.


A thorough description of the MatlabTM programming language.


Page 57 of 62

A reference book on Latex processing
Papers


An image segmentation method based on directed image region partitioning is proposed


This paper presents an assessment of how use of color information can improve the segmentation results of images with shadows and surface curvature in the images which would negatively impact the results from standard gray-level analysis.


This paper presents parametric piecewise functions to represent curved shapes.


This paper presents a scheme which interpolates piecewise cubic polynomials based upon continuous first derivatives.


Describes methods for creating hand printed text using random variations of a font.


Review of the translation quality using a technique which provides an objective assessment by using statistical techniques.


The original article on chain coding techniques. This describes how to generate and interpret chain codes as object boundaries and use them in image understanding.
URLs

Course: Image Analysis and Understanding
Site: http://www.icaen.uiowa.edu/~dip/LECTURE/lecture.html
The University of Iowa, College of Engineering

A very comprehensive set of course notes, including exercises, beginning from basic image processing techniques and including digital motion picture processing techniques.

Course: Image Processing and Computer Vision
Site: http://iul.cs.byu.edu/morse/550-F95/node1.html
Brigham Young University

A comprehensive set of class lecture notes covering the fundamental techniques of image enhancement and image processing theory at both the mathematical and computer science application levels. These notes provide a lecture on color image processing techniques.

Article: A Tutorial on Freeman Chain Coding

An internet article/ad which describes the Basics of Chain Coding and Quantization Schemes.

Course: EEN 538 Digital Image Processing, Project 5, author: Hongsheng Zhang
Site: http://snapper.eng.miami.edu/~zhanghs/Old%20stuff/Project%205.htm
University of Miami, Coral Gables, Electrical and Computer Engineering Department, Underwater Vision and Imaging Laboratory

Contains information on Chain code extraction and character recognition.

OnLine Book: Chain codes
Site: http://www.icaen.uiowa.edu/~dip/LECTURE/Shape2.html
Chapter 6, Part II

Contains information on Chain codes, border representation, and curvature analysis.

Course: Connected component analysis, Exercise 5
Site: http://www.imm.dtu.dk/~jmc/04250/exercises/cca/cca.html
04250 Digital Image Analysis
Technical University of Denmark, Informatics and Mathematical Modelling

Shows how objects in binary images can be described, classified and represented.