3. DATA COLLECTION AND ANALYSIS

The data collection procedure has an important impact on the handwriting recorded. We want to prevent unwanted influences that will cause "unnatural" inputs. The subjects must behave as they do in a real life situation, otherwise the samples taken will not reflect correctly the nature of handwriting and the success of the recognition algorithms will be poor.

3.1 Data Collection Procedure

We used a WACOM PL-100V pressure sensitive tablet with an integrated LCD display and a cordless stylus. The input and display areas are located in the same place. Attached to the serial port of an Intel 486 based PC, it allowed us to collect handwriting samples. This tablet sends x and y tablet coordinates and pressure level values of the pen at fixed time intervals (sampling rate) of 100 miliseconds.

Our data collection and displaying software was developed in MS-DOS environment using the Borland C++ 3.1 compiler. We ignored the pressure level values and we did not use them. The tablet resolution is 4157 by 3118 pixels. We mapped this to standard VGA resolution which is 640 by 480.

Subjects were volunteers from the Department of Computer Engineering of Boğaziçi University staff members and students. They were told that the writing would be recorded by the computer for "later usage."

We collected samples of ten handwritten digits from 44 writers. These writers were asked to write 250 digits in random order inside boxes of 500 by 500 tablet pixel resolution. Subject were monitored only during the first entry screens. Each screen

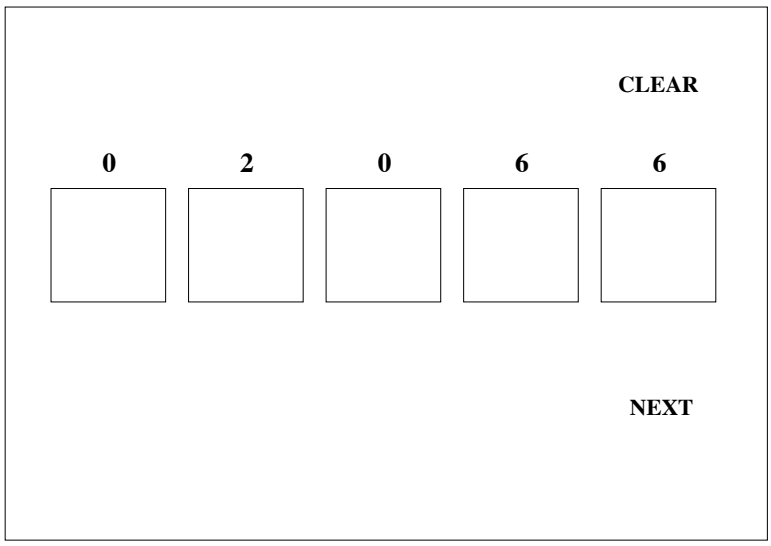

Figure 3.1: Tablet input screen.

contained five boxes with the digits to be written displayed above (Fig. 3.1). Subjects were told to write only inside these boxes. If they made a mistake or were unhappy with their writing, they were instructed to clear the content of a box by using an on-screen button. Progress was controlled using on-screen buttons. The first ten digits were ignored because most writers were not familiar with this type of input devices, but subjects were not aware of this.

3.2 Data Analysis

In our study the samples are digits of ten classes. But this does not imply that we have only ten shapes to distinguish. The digits were written by people having various styles of writing.

1. Every subject has one or more own prototype for each digit (Fig. 3.2).

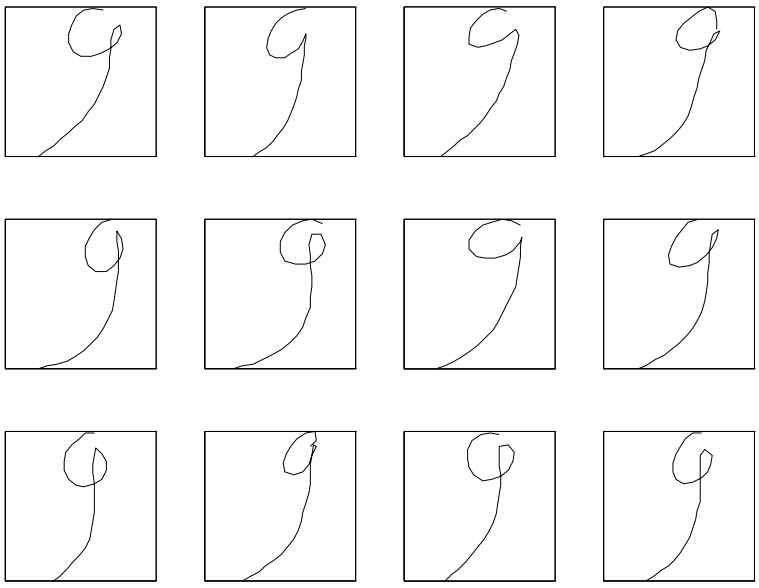

Figure 3.2: The digit "9" written by subject number 4.

2. Depending on various factors, the raw data is composed of vectors of different number of points.

3. Most of the subjects, but not all of them, wrote the digits about in the center of the boxes, however some of them wrote them bigger and some smaller.

These facts show us that we need preprocessing in order to convert data into a normalized form for improved recognition.

3.3 Data Inspection

The samples collected should be processed before using them for learning and recognition. We inspected our sample database using the Upview UNIPEN [12] viewer in SunOS 4.1.3_U1 environment. This inspection (authentication) allowed us to discard

samples which were written in wrong places.

4. PREPROCESSING

The raw data captured from the tablet contains enormous variations in writing speed. Different writers write with different speed characteristics. Training a recognizer with such data leads to poor performance. The purpose of the preprocessing is to create an intermediate representation, incorporating to the extent possible, the a priori knowledge of the designer. This includes two complementary processes [13]:

1. Reducing variability which does not help discrimination between classes, therefore building up some invariance.
2. Enhancing variability which does help discrimination between classes.

Various preprocessing methods for on-line handwriting were discussed by Tappert et al. [1], Guerfali and Plamondon [14]. In our problem, the steps of our preprocessing, namely normalization, resampling (for dynamic representation), converting to bitmap, blurring and downsampling (for static representation), greatly reduce the meaningless variability.

Since digits were collected in discrete input boxes, we do not correct rotations. Additionally, irregular samples are not considered, as explained in the previous chapter.

We used MATLAB for data visualization and prototyping our preprocessing routines. Main programs were written in C and and compiled using the GNU C compiler on a DTK SPARC Classic+ machine with SunOS 4.1.3_U1 operating system.

4.1 Dynamic Representation

In our study, we used only x and y coordinate information. The stylus pressure level values were ignored since they do not give additional information about the shape of a digit. They are more appropriate for authentication tasks such as signature verification and writer identification.

4.1.1 Normalization

First we applied normalization to make our representation invariant to translations and scale distortions. The raw data that we capture from the tablet consists of integer values between 0 and 500 (tablet input box resolution). In order to normalize data, first we calculate the median (x_m, y_m) of a digit:

$$\begin{aligned} x_m &= \frac{x_{max} + x_{min}}{2} \\ y_m &= \frac{y_{max} + y_{min}}{2} \end{aligned} \tag{4.1}$$

The spreads on the two dimensions are calculated:

$$\begin{aligned} \delta_x &= \frac{x_{max} - x_{min}}{2} \\ \delta_y &= \frac{y_{max} - y_{min}}{2} \end{aligned} \tag{4.2}$$

and the bigger one is chosen as the scaling factor:

$$\delta = max(\delta_x, \delta_y) \tag{4.3}$$

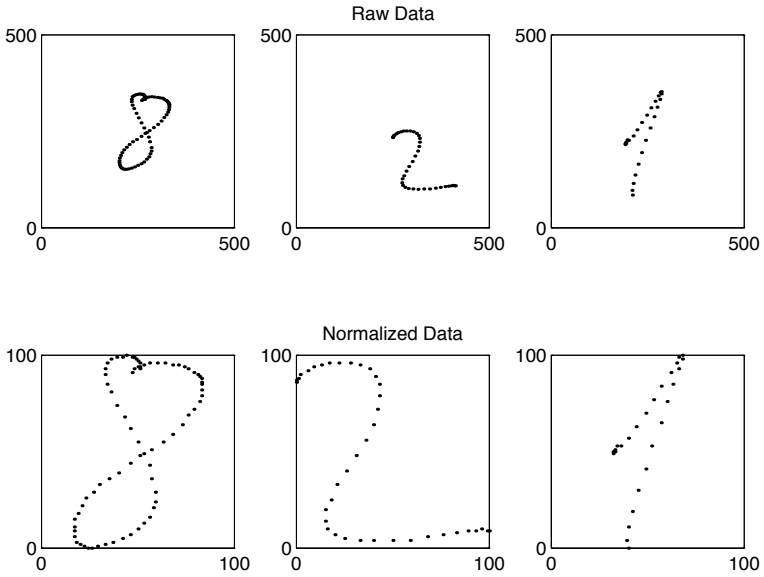

Figure 4.1: The normalization process.

The new coordinates are:

$$\begin{aligned} x_t &= 50 + 50\left(\frac{x_t - x_m}{\delta}\right) \\ y_t &= 50 + 50\left(\frac{y_t - y_m}{\delta}\right) \end{aligned} \tag{4.4}$$

such that the coordinate which has the maximum range varies between 0 and 100. Usually x stays in this range, since most characters are taller than they are wide. However, we observed the fact that the characters in which x has a greater range do exist and we decided to use the greatest δ value. (Fig. 4.1)

We could also have chosen using different scaling factors for the two axes:

$$\begin{aligned} x_t &= 50 + 50\left(\frac{x_t - x_m}{\delta_x}\right) \\ y_t &= 50 + 50\left(\frac{y_t - y_m}{\delta_y}\right) \end{aligned} \tag{4.5}$$

but this normalization would introduce distortions for very narrow digits such as "1" (Fig. 4.2).

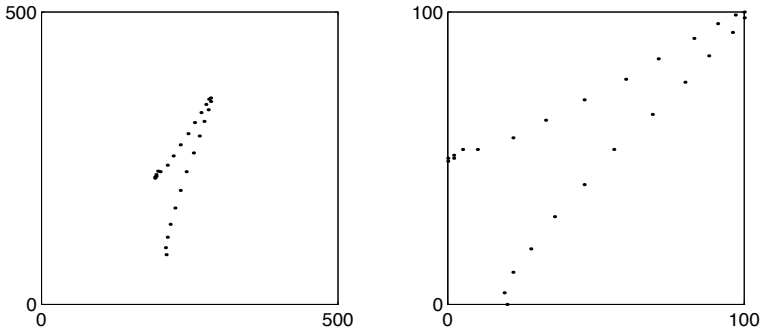

Figure 4.2: Normalization with different δ values causes distortion.

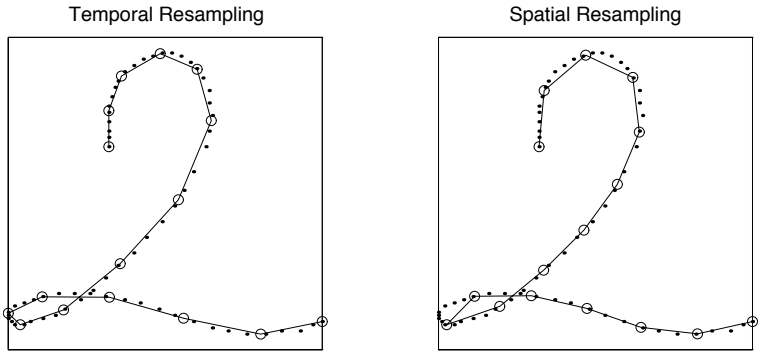

Figure 4.3: Temporal vs. spatial resampling to 16 points.

4.1.2 Resampling

In order to train and test our classifiers, we need to represent digits as constant length feature vectors. A commonly used technique leading to good results is resampling the (x_t, y_t) points. Temporal resampling (points regularly spaced in time) or spatial resampling (points regularly spaced in arc length) [2, 3, 5] can be used here (Fig. 4.3). Raw point data are already regularly spaced in time but the distance between them is variable. Previous research [15, 13] showed that spatial resampling to obtain a constant number of regularly spaced points on the trajectory yields much better performance, because it provides a better alignment between points.

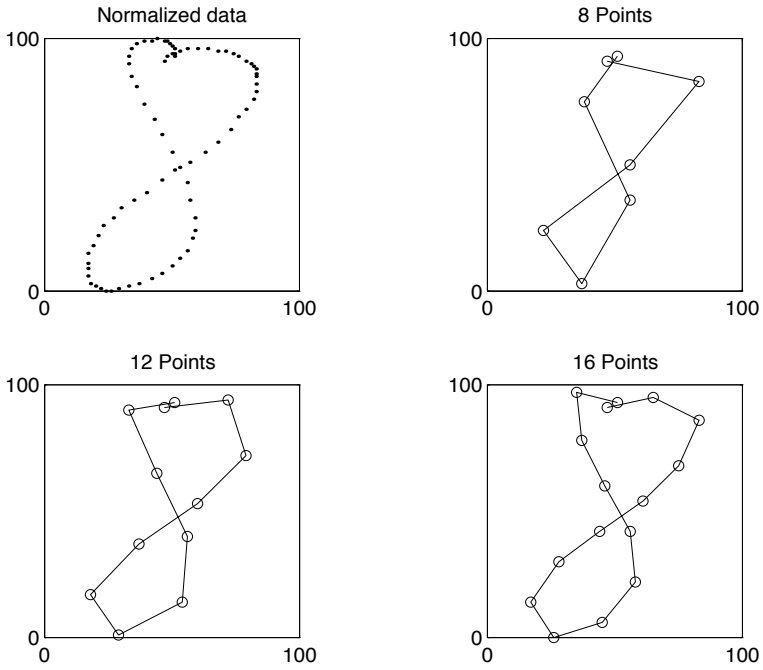

Figure 4.4: Spatial resampling to different number of points.

Our resampling algorithm uses simple linear interpolation between pairs of points. The resampled digits are represented as a sequence of N points (x_t, y_t), regularly spaced in arc length, as opposed to the input sequence, which is regularly spaced in time (Fig. 4.4).

For dynamic representation, the input vector is composed of ordered (x, y) pairs for each resampled point. So, the input vector size is $2N$, two times the number of points resampled. We considered spatial resampling to 8, 12 and 16 points in our experiments.

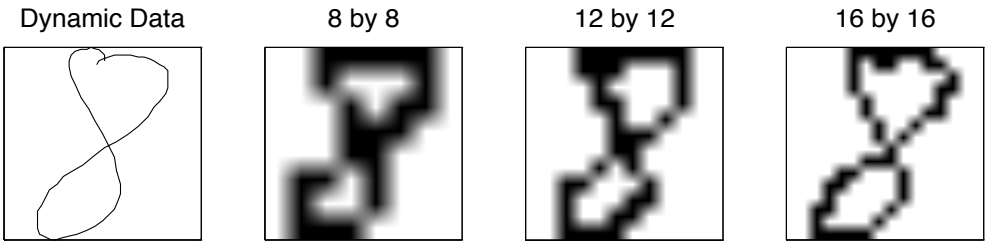

Figure 4.5: Converting to bitmaps of different resolutions.

4.2 Static Representation

The static representation differs from dynamic representation in the sense that it does not incorporate pen motion. Instead, it is based on solely the image of a character, as in off-line recognition tasks. First we convert dynamic data to bitmaps, then we blur it and perform downsampling on it.

4.2.1 Converting to Bitmaps

In order to obtain binary images of the characters, we used Bresenham's scan conversion algorithm [16] to create bitmaps from normalized on-line handwriting data. This algorithm is explained in Appendix A. We iterated through pairs of sequential point samples within each pen stroke.

Since every character in the database is normalized to a unit bounding box we use a square pixel array to store each image. The single parameter for this representation is the resolution (N by N) of the image. We considered three different image resolutions: 8 by 8, 12 by 12 and 16 by 16. Examples are shown in Fig. 4.5.

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

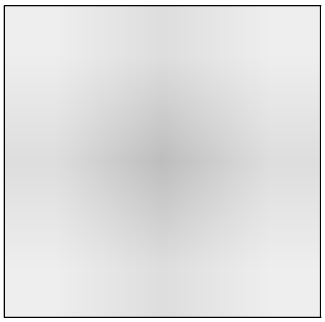

Figure 4.6: The 3 by 3 blurring kernel.

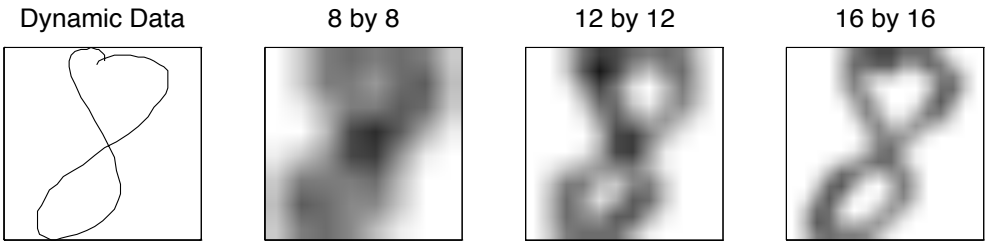

Figure 4.7: Blurred images.

4.2.2 Blurring

In preparing bitmaps for classification the spatial relationship was not preserved. Bitmaps which are usually similar can have very dissimilar feature vectors. Small variations in pen coordinates can produce different images because of quantization with hard thresholds. To solve this problem we smooth the bitmap so that a pixel's value is distributed across its neighbors. We accomplished this by convolving a unit volume blurring kernel (Fig. 4.6) with the image to produce a pixmap, in which pixels can take on arbitrary floating-point values. This kernel can be optimized for each bitmap resolution. A kernel of larger size may be appropriate for bitmaps of greater resolution. Examples are shown in Figure 4.7.

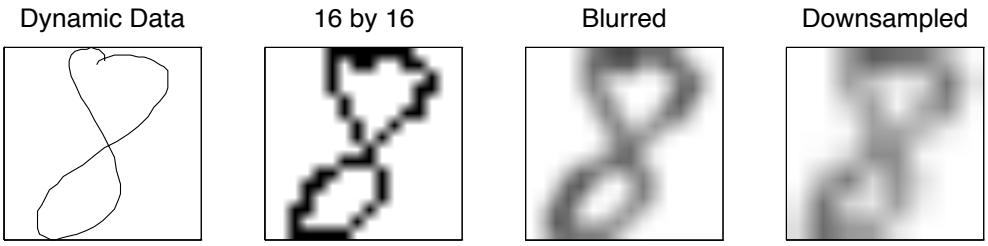

Figure 4.8: The process of obtaining static representation.

4.2.3 Downsampling

Downsampling reduces input dimensionality keeping extra information from a higher resolution blurred bitmap. We calculate the value of every new pixel by averaging the values of four correponding pixels of the larger image. So, the feature vector size is reduced by a factor of four. The output vector for static represenation is composed of N^2 input values for a N by N image (e.g. 8 by 8 bitmaps are obtained from 16 by 16 blurred bitmaps) (Fig. 4.8).

4.2.4 Summary

Figure 4.9 summarizes the final preprocessing method. For each sample, first we normalize the raw input. Then we apply spatial resampling to get the final dynamic representation. The feature vector size is two times the number of points resampled ($2N$).

The static representation is obtained by converting the normalized data to bitmaps and then applying blurring and downsampling. Note that the dynamic data are converted to a bitmap having a higher resolution than the final image $(2N)^2$. The

feature vector size is equal to the number of pixels in the final image (N^2).

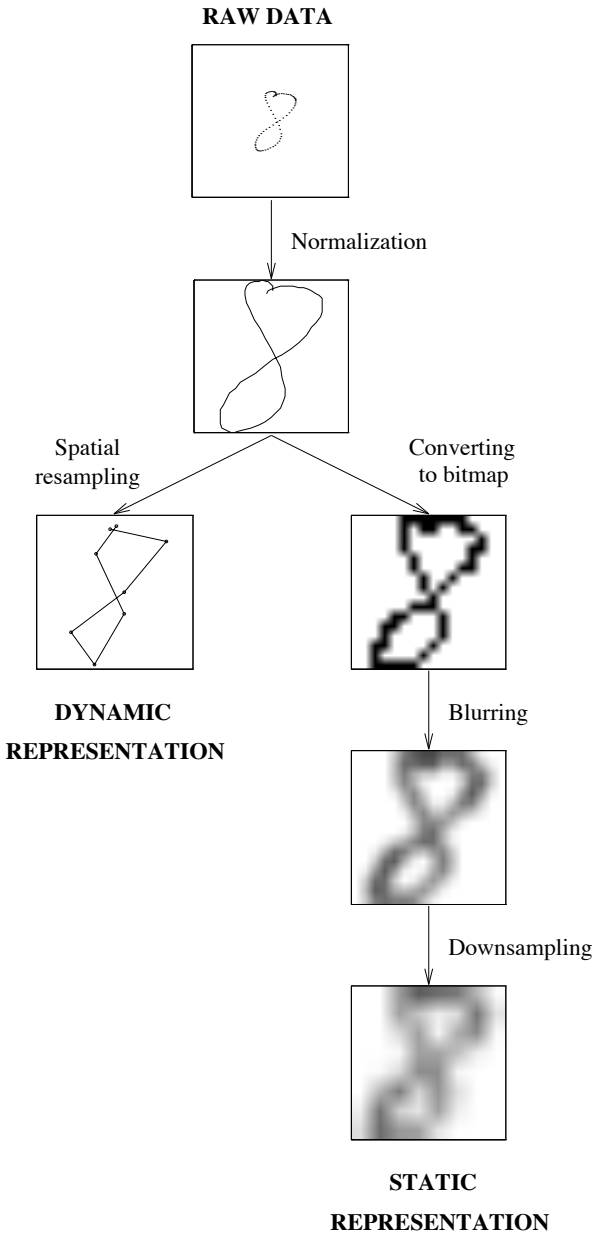

Figure 4.9: The stages of preprocessing.