Chapter 1

Handwriting Recognition Engine

1.0.1 Normalisation

Many handwriting recognition systems perform some kind of normalisation. See section **??** for a review of common techniques. In this system, only size normalisation is performed. Other normalisation techniques reported in literature were often necessary, because the input devices returned unsufficient data. With modern input devices many of the normalisation techniques that were used to clean the data from different kinds of noise are not necessary.

In order not to lose any data, the complete list of points is maintained, there is no point reduction. Since the prototype is a learning application and not a pure handwriting recognition, the main issue seems the size of the character. The size normalisation is done in two steps. Each complex that is more complex than a point has a bounding box around that can be used for scaling.

1.0.1.1 Boxing

The term *bounding box* needs not much introduction. It is the smallest box surrounding a graphic object. In this prototype, bounding boxes are rectangles with the same height and width. Rectangles are defined as a 3-tuple of a handle point, a height and a width. $R :=<< x, y >, w, y >$ where the 2-tuple $< x, y >$ is a point in the coordinate system with the decimal values x and y as coordinates, h is the y-height of the rectangle and w is the x-width. In order to find an extended square bounding box of any sequence of points, a method finds the four outermost points C_i with the minimal and maximal values for x and y.

$C_{x_{min}} :=< x_{min}, y_1 >$
$C_{x_{max}} :=< x_{max}, y_2 >$
$C_{y_{min}} :=< x_1, y_{min} >$
$C_{y_{max}} :=< x_2, y_{max} >$

The intermediate handle point $H_{intermediate}$, the width w and height h of the minimal bounding box B' can be calculated as follows:

$H_{intermediate} :=< x_{min}, y_{min} >$
$w := x_{max} - x_{min}$
$h := y_{max} - y_{min}$
$s := \arg\max(w, h)$

Then, the handle point H can be calculated by:

$$H := \begin{cases} < x_{min} - \frac{s-w}{2}, y_{min} > & \text{if } s \text{ equals } h \\ < x_{min}, y_{min} - \frac{s-h}{2} > & \text{if } s \text{ equals } w \end{cases}$$

The minimal bounding box B' is a rectangle with the following values:

$B' :=< H_{intermediate}, w, h >.$

The extended, square bounding box B is a rectangle with the following values:

$B :=< H, s, s >$

The use of boxing is used multifunctional in the prototype. One of the uses is for distortion-free scaling (see section 1.0.1.2). Another use is for the comparison of positions of substructures of a character or Radical (see section 1.1).

1.0.1.2 Scaling

The bounding box of most Kanji is square, in cases where it is not, the space given to a smaller Kanji is still a square. In order to account for that, scaling is done with respect to the ideal of a square shape of a character. That concept is used for full characters, as well as Radicals and strokes. Before scaling the size of a stroke, a square bounding box is constructed. The handle point of the square bounding box is used as a vanishing point for vectorial projection. In a stroke data structure that consists of a point sequence $p_1, \ldots, p_n$ a number of vectors $\vec{v}_1, \ldots, \vec{v}_n$ is created. Each vector is defined by the handle point $H =< x_h, y_h >$ of a square bounding box and the p_i.

$$\vec{v}_k := p_k - H = \binom{x_{p_k} - x_h}{y_{p_k} - y_h}$$

In order to generate a scaled version of a sequence of points, all the $\vec{v}_k$ are multiplied with a scaling factor.

1.0.2 Recognition of Straight Strokes

1.0.2.1 Identification of a Straight Stroke

The notion of *stroke* essentially means *point sequence* in a technical sense. That is, strokes are not mathematical objects like functions. In order to perform stroke recognition, it is necessary to determine what shape a stroke has. For the identification of a straight stroke, a linear regression line is constructed from the point sequence with the Gaussian method of least squares. The construction is done with the standard method for calculating least squares, as described in literature. The method seeks for the minimal sum of squares of residuals. If the sum of residual squares is below a certain threshhold that has been determined empirically, the system assumes that the point sequence generally follows a straight line. In that case, straight stroke matching will be performed. If the line appears to be curved, it will be matched with curve analysis methods as described in section 1.0.3.

1.0.2.2 Straight Stroke Matching

For the matching of a straight stroke there are a number of features that could be considered for a feature vector. In order to describe a straight stroke and then match it with another one, a feature vector is constructed that has the following features:

- **Length**: The total lenght of the line. Sum of the distances of succeeding points.
- **Initial point**: The coordinates of the initial point
- **Endpoint**: The coordinates of the endpoint
- Total number of measured points: The number of sample points that were measured by the input device
- **Velocity**: The velocity with which the stroke was drawn
- **Gradient/Direction**: The slope of the linear regression line that can be constructed for the point sequence, or the general direction of the line, represented as a numerical vector.

However, some of the information seem redundant. In order to match two strokes by their feature vectors F_{DB} and F_{exp}, the following values are conidered:

$$F := \begin{pmatrix} \text{Length} & l \\ \text{Initial point} & p_I \\ \text{Endpoint} & p_E \\ \text{Velocity} & v \\ \text{Direction vector} & \vec{d} \\ \text{Sum of residual squares} & \sigma \end{pmatrix}$$

The features are used for matching with other straight strokes by comparison of their values. The length l is calculated as the sum of the euclidian distances d_i between the individual points p_i and p_{i-i} within a sequence of N points.

$$d_i := \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$
$$l := \sum_{i=1}^{N-1} d_i$$

In the case of the initial point and endpoint comparison of the two strokes, the euclidian distance between the original point $P_{I,DB}, P_{E,DB}$ and the candidate point $P_{I,exp}, P_{E,exp}$ is used as a comparison value. The velocity v is calculated from the time difference δ_T between sampling time of the initial point t_{p_I} and the endpoint t_{p_E}:

$$\delta_T := t_{p_E} - t_{p_I}$$

$$v := \frac{n_p}{\delta_T}$$

The velocities v_{DB} and v_{exp} are both real numbers and can be compared directly. The direction vectors of the regression line $\vec{d}_{DB}$ and $\vec{d}_{exp}$ are compared by the deviation of their directions. The deviation is expressed as the angle α between the original and the candidate vector.

$$\alpha := \vec{d}_{DB} \quad \angle \quad \vec{d}_{exp}$$

The sum of residual squares σ is calculated from the measured point sequence with respect to the regression line of the database point sequence. Given the regression line f_{DB}, the sum of the squares of the residuals for the input sequence is:

$$\sigma := \sum_{i=1}^{N} (f_{DB}(x_{i,exp}) - x_{i,exp})^2$$

The smaller σ the closer the measured points to the regression line from the database. This is only done for strokes that have been calculated to be straight, since any set of points can have a regression line, even curved ones. In the case of curved strokes, a linear regression line as it is used here does not seem to bear the potential helping identify a point seqence to be a certain shape.

1.0.3 Curve Handling

In this prototype the direction feature plays a crucial role. Generally follwing several approaches in on-line handwriting recognition literature (see section **??**), the approach used here defines direction vectors. The general approach is to define a direction vector between a point within the point sequence and a successor point. However, since the measured points are usually very close together, the direction vector is defined between a point and another point that is further away, defined by a dynamic threshold, considering distance, total number of points and the total length of the stroke.

If non of these vectors deviates from the general direction of the stroke, a dynamic threshold, considering the total stroke length, the stroke is considered straight. Nevertheless, for matching straight strokes, a feature vector is used.

Different curve matching techniques are described in section **??**. The technique used here refers to *curved strokes*, however, it is a vector-based technique, that aims at point reduction.

1.0.3.1 Identifying Curves in Strokes

休む

S 14, 16, 17 how is curver handling done? show requirements. what alternatives were there to consider? stroke matching with angles instead of point position. s. 24

1.0.4 Dynamic Time Warping

what's the similarity measure for points and strokes? show requirements. what alternatives were there to consider?

s. 51 how is dynamic time warping done here? pointer to papers or hwr - chapter, don't explain DTW here. show requirements why DTW? what alternatives were there to consider? none - it is the alternative. to all the other stuff I've been doing. however, what about 3D time warping?

1.1 Radical Recognition Process

1.2 Character Recognition Process

In order to scale to the normalised size, the length of an edge of the bounding box is compared to the length of the normalised character.

1.3 Error Handling

see section **??** in chapter **??** for possible sources of error

1.3.1 Error Recognition

why this section? to demonstrate own achievements of error recognition. the reader should know how it is done technically.

what goes into this section? the aspects of finding errors. finding errors is not a straightforward trivial task - whenever something does not match it is an error - doesn't work like that. instead, firstly, it needs to be made sure that it actually is an error. meaning - not a recognition error, but a user error. secondly, the type of error needs be identified. see section **??** (or handwritten page 58) for sources of error.

how will this section be written? technical - first describe how the error recognition integrates into the recognition process, then how errors are identified.

1.3.2 Error Processing

why this section? actually the 'handling' or 'processing' aspect could be described in the recognition section 1.3.1 as well. so this section is only for a better overview, for document structure, thematically they are the same section. thus they are put together under Error Handling 1.3.

what goes into this section?