Chapter 1

Handwriting Recognition Engine

1.0.1 Recognition of Straight Strokes

1.0.1.1 Identification of a Straight Stroke

Since strokes here are essentially point sequences and not mathematical objects like functions, it is necessary to determine what shape a stroke has. In order to identify a straight stroke, a linear regression line is calculated from the point sequence with the Gaussinan method of least squares. This calculation is done the standard method described in literature. If the sum of the squares then is below a certain threshhold that has been determined empirically, the system assumes that the point sequence generally follows a straight line. If the line appears to be straight, straight stroke matching will be performed. If not the line is matched with curve analysis methods. (see section 1.0.2).

1.0.1.2 Straight Stroke Matching

For the matching of a straight stroke there are are a number of features that could be considered for a feature vector.

- Length
- Direction
- Initial point
- Endpoint
- Total number of measured coordinate points
- Velocity with which the stroke was drawn

In this prototype the direction feature plays a crucial role. Generally follwing several approaches in on-line handwriting recognition literature (see section refsec:strokedirection), 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, above another dynamic threshold, considering the total stroke length, the stroke is considered straight.

In order to math two straight strokes with each other, the vicinity of their start point and their end point is considered. Furthermore, the total length, the number of points and the velocity are considered. A feature vector F for the description of a stroke then consists of the following values:

$$F := \begin{pmatrix} \text{Direction vector} & \vec{d} \\ \text{Total number points} & n_p \\ \text{Length} & l \\ \text{Initial point} & p_I \\ \text{Endpoint} & p_E \\ \text{Velocity} & v \end{pmatrix}$$

Those values are used for matching with other straight strokes by comparison of their values. In the case of the points, the euclidian vicinity between the original point and the candidate point is used as a comparison value. The velocity is calculated from the time difference δ_T between sampling time of the initial point t_{p_I} and the endpoint t_{p_E}:

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

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

In the case of the direction vector $\vec{d}$ the deviation of direction is expressed as the angle α between the original and the candidate vector.

$\alpha := \vec{d_{original}} \angle \vec{d_{candidate}}$

1.0.2 Curve Handling

In section see **??**

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

1.0.3 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?