Chapter 1

Handwriting Recognition Engine

1.0.1 Curve Handling

1.0.1.1 Detecting Curves in Strokes

Different curve matching techniques are described in section **??**. The technique used here refers to is a vector-based technique, that aims at the extraction of features to describe the curved stroke. In this prototype the direction feature plays a crucial role. It is generally follwing several approaches in on-line handwriting recognition literature. The direction feature as it is used in other applications has been discussed in section **??**). The approach for curve recognition used here defines direction vectors.

The general concept 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 for straight strokes is used. If the slope of the curve changes, i.e. the direction of the vectors changes, a curve has been detected. There is a threshold T_{min} for the minimal size of a *recognisable angle* between two vectors. Any angle smaller than that will be ignored completely. A second threshold T_{max} describes the minimal size for an angle γ that is taken into account. Any angle β_i between T_{min} and T_{max} will be stored. When the sum of the β_i exceeds T_{max} the position will be treated as a curve.

$$\alpha < T_{min} \Rightarrow \alpha \text{ will be ignored}$$

$$T_{min} < \beta_i < T_{max} \Rightarrow \beta_i \text{ will be stored}$$

$$T_{max} < \gamma \Rightarrow \gamma \text{ will be interpreted as an angle in the stroke}$$

$$T_{max} < \sum_{i=k}^{l} \beta_i \Rightarrow \sum_{i=k}^{l} \beta_i \text{ will be interpreted as an angle in the stroke}$$

A successful curve detection extracts the key feature of the curve description. The remaining features can be extracted or calculated, by using the detected curves.

1.0.1.2 Feature Extraction for Curved Strokes

The feature vector for curved strokes contains the following elements:

$$F := \begin{pmatrix} \text{Length} & & l \\ \text{Initial point} & & p_I \\ \text{Endpoint} & & p_E \\ \text{Velocity} & & v \\ \text{Corner angles} & \gamma_i & i := 1, \ldots, n \\ \text{Corner points} & c_i & i := 1, \ldots, n \\ \text{Direction sequence} & \vec{d_j} & j := 1, \ldots, m \end{pmatrix}$$

The partial feature vector l, p_I, p_E, v is identical to the partial feature vector of the straight strokes with the same elements and needs no further description (see section **??**). The corner angles have been calculated. In the case of a summation of angles the point with the greatest angle will be determined as the corner point. In the case of angles that have been observed directly without summation, the corner point is the point where the angular vectors meet. Therefore, the number of corner points m and corner angles is identical. The feature *Corner points* is defined as the sum of euclidian distances between the corner points from the stored datastructure with the candidate data structure.

$$c_p := \sum_{i=1}^{n} \sqrt{(x_{i,db} - x_{i,exp})^2 + (y_{i,db} - y_{i,exp})^2}$$

The corner angles feature is defined as the sum of the squares of the differences between the angles at the corner points.

$$c_a := \sum_{i=1}^{n} (\beta_{i,db} - \beta_{i,exp})^2$$

For both c_a and c_p the smaller the value, the closer the more similar the strokes. In fact, if c_a or c_p are equal to 0, the point analised point sequence was identical with the stored point sequence. The *direction sequence* feature takes into account all the direction changes, as opposed to the corner angles that use only the most significant ones. The direction sequence feature uses the angle of each line segment regarding the X-axis of the coordinate system. The angles are then matched against the time they were produced in.

1.0.2 Dynamic Time Warping

The technique of *dynamic time warping* (DTW) has been used for stroke matching by several studies and systems (see sections **??** and **??**). It is an elastic matching technique that seeks an optimal path through a matrix of point distances. Details of the algorithm have been described in literature.

1.0.2.1 Standard DTW

Following **?** (**?**) and **?** (**?**), the DTW technique is implemented for two pen trajectories, namely the trajectory stored in the database $P_{db} = (p_{1,db} \ldots, p_{n,db})$ and the input trajectory generated by the user's drawing $P_{exp} = (q_{1,exp}, \ldots, q_{m,exp})$. Any point pair $p_{i,db}$ and $p_{j,exp}$ matches if one of two constraints is met:

1. **Boundary condition**: The points $p_{i,db}$ and $p_{j,exp}$ match, if one of the following boundary conditions is fulfilled:
   - $p_{i,db}$ equals the initial point $p_{I,db}$ and $p_{j,exp}$ equals the initial point $p_{I,exp}$ of their respective point sequences P_{db} or P_{exp}
   - $p_{i,db}$ equals the endpoint $p_{E,db}$ and $p_{j,exp}$ equals the endpoint $p_{E,exp}$ of their respective point sequences P_{db} or P_{exp}
2. **Continuity condition**: The points $p_{i,db}$ and $p_{j,exp}$ match, if the follwoing equation is fulfilled

   $$\frac{M}{N} i - cM \leq j \leq \frac{M}{N} i + cM$$

   c is a constant between 0 and 1, indicating the strictness of the condition.

1.0.2.2 3-Dimensional DTW

In addition to the standard DTW algorithm, a second DTW classifier performs a 3-dimensional distance calculation. That is, a point sequence $P = (p_1 \ldots, p_n)$ is considered 3-dimensional in the way that it contains the relative time (in ms) from the beginning of the stroke as a third dimension. The point distance calculations are euclidian, ignoring the fact that the third dimension is a different type of coordinate.

$$d_i := \sqrt{(x_{i,db} - x_{i,exp})^2 + (y_{i,db} - y_{i,exp})^2 + (z_{i,db} - z_{i,exp})^2}$$

$< x_i, y_i >$ are the regular 2-dimensional point cooredinates, while the z_i are the relative times from the beginning of the stroke. The actual DTW algorithm works exactly the same, the difference lies in the distance calculation. This version of the algorithm has been conducted as an experiment. The direct comparision with the regular DTW is reported in section **??**.

1.0.3 Stroke Matching Summary

Four different classifiers have been implemented. The straight stroke matching via regression lines, the curve matching classifier with direction sequences and two variants of the dynamic time warping algorithm, a regular DTW algorithm and the three-dimensional variant DTW. The stroke matching occurs in NUMBER parts. Firstly, the straight stroke recogniser attempts to establish if a stroke is straight or not. Then the straight stroke matcher analyses if the input stroke and the database stroke are equal. Secondly, if the stroke is regarded as being curved, the curve detector performs a search for the significant points and angles in both sequences. After that, the curved stroke matcher compares the features of the curved strokes. If both recognisers fail to recognise the expected stroke, dynamic time warping is used as a fallback option.

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?