$$D = \sqrt{\sum_{i=1}^{n_{ip}} [(x_{1,i} - x_{2,i})^2 + (y_{1,i} - y_{2,i})^2]}, \tag{1}$$

where n_{ip} is the total number of interpolation points in the gesture input vector and $x_{1,i}$ is the x-component of the ith interpolation point in the first of the two input vectors being compared. This is very similar to the *proportional shape matching* channel used in the SHARK^2 writing system [Kristensson and Zhai (2004)] and in the gesture clarity metric used by Smith et al. (2015).

Although this method can correctly identify when two gesture inputs match exactly, it could also return a large distance between two input vectors that are qualitatively very similar. For example, a user may start near the bottom of the key for the first letter of the word and end up shifting the entire gesture input pattern below the centers of the subsequent keys. This input pattern could pass over all of the correct keys but still result in a large Euclidean distance when compared to the perfect vector for the word.

The shortcomings of this approach made it clear that we were not utilizing all of the useful information contained in the input vectors. If a poor distance measure were to cause misidentifications that would not happen in practice then this could introduce significant biases during the optimization procedure, resulting in a keyboard that is not optimal for real use. Kristensson and Zhai accounted for this in the SHARK^2 writing system by incorporating language and location information in addition to proportional shape matching in their recognition algorithm. Similarly, In order to reduce the impact of these systematic effects, we needed to identify additional features that would improve our gesture input recognition.

3.1.2. Feature Set

Our first step was to uncouple the x and y coordinates and treat them individually. Given the anisotropic nature of most keyboards, the relative importance of biases between the two spatial dimensions is not clear a priori. Therefore, we decided that the first two features should be the Euclidean distance (Eq. 1) between two input vectors for the x and y components individually,

$$D_x = \sqrt{\sum_{i=1}^{n_{ip}} (x_{1,i} - x_{2,i})^2} \text{ and } D_y = \sqrt{\sum_{i=1}^{n_{ip}} (y_{1,i} - y_{2,i})^2}. \tag{2}$$

In order to address the issue with offset input vectors, translational symmetry needed to be taken into account. To do this we decided to look at the time derivatives of the input vectors with respect to their x and y coordinates. Since the input vectors are sets of sequential, discrete points we can easily estimate the derivative at each point. We can then construct a distance measure by taking the Euclidian distance between the time derivatives of two swipe patterns at each point. The equation for the derivative distance in the x-dimension is given by:

$$D_{\partial x} = \sqrt{\sum_{i=1}^{n_{ip}-1} \left[(x_{1,i+1} - x_{1,i}) - (x_{2,i+1} - x_{2,i})\right]^2}, \tag{3}$$

where x_1 and x_2 are the x-components of the two input vectors being compared. We assume a constant input velocity for the models considered here so we've implicitly rescaled the time coordinated such that $t_{i+1} - t_i = 1$ to simplify the equations.