COSC134S11 Project Final Report: Geometric Shape Recognition

Ping Lin

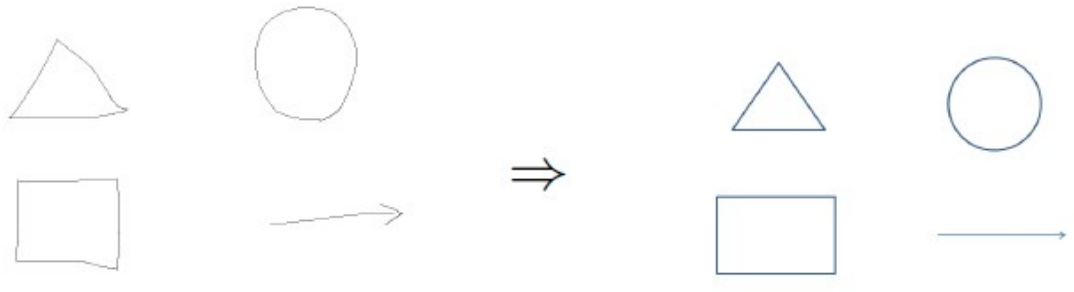

Figure 1: Schematic view of the problem.

The Problem

The goal for this project is to recognize simple 2D geometric shapes one sketches on a pen-based device like Tablet PC. One important consideration of the system is the ability to generalize to larger/different symbolic sets. The idea is to confine the domain of recognition to be the simple geometric shapes, like circle, rectangle, triangle, arrow, etc. so that the complexity of the problem is not overwhelming with the time constraints, but at the same time, try hard to make the system general-purpose augmentable. Schematically, the problem can be summarized in Figure 1.

Background

Free-hand sketch recognition is a problem that has been studied for a long time. But because of the diversity of the possible target objects to be recognized, there has not been any method that is "the" method to use.

This shape recognition problem is closely related to the familiar OCR problem. But there are differences between them. One such difference is that, in OCR, we rely on character's orientation to distinguish some symbols, say "6" and "9". In shape recognition, however, we really would like to have rotational invariant property. First, it is natural and often needed to draw the same shape in different orientations, say, an arrow from left to right and another from up to down. Second, the orientation of shapes tend to have a continuum in contrast to OCR, where the margin for heading difference of two characters is very large. In shape recognition, the shape could be tilted to any slope; there is no clear cut. So, it is best to identify the shapes different only in orientation.

Because anything relying on the temporal information will put constraints on the order strokes are written, which is not so desirable for geometric shape recognition. Also for simplicity of feature generation, in this project, the input is merely a 2D image.

Also, for generalizability, the proposed method is chosen to be in the category of statistical methods.

Methods

Moments method is popular in statistical methods of classification. Among them, Zernike moments, when taking its magnitudes, are naturally rotational invariant. So, Zernike moments, or more precisely, magnitudes of Zernike moments up to certain order of the input 2D binary image are served as the feature vector in this project. [1]

Zernike polynomials are a set of complex, orthogonal polynomials defined over the interior of the unit disk. And they are complete on the interior of the unit disk, so they form an orthogonal basis. Zernike moments are then obtained as projections of the input image on these Zernike polynomials.

Precisely, Zernike polynomials V_{nm} are computed as follows:

$$V_{nm} = V_{nm}(\rho, \theta) = R_{nm}(\rho) e^{jm\theta} \tag{1}$$

$$R_{nm} = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!} \rho^{n-2s} , \tag{2}$$

where n is a non-negative integer, m is an integer such that $n - |m|$ is even and $|m| \leq n$.

Then, Zernike moment A_{nm} of order n with repetition m is:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V_{nm}(x,y), \quad x^2 + y^2 \leq 1 . \tag{3}$$

The component of the feature vector is thus $|A_{nm}|$, where the largest order n needs to be specified to make our feature vector finite dimensional.

Notice that $A_{n,-m} = A_{nm}^*$, so $|A_{nm}| = |A_{n,-m}|$. We thus only need to consider $|A_{nm}|$ with $m \geq 0$. This together with the constraint: m is an integer such that $n - |m|$ is even and $|m| \leq n$, will establish the number of moments at a given order of Zernike moments, which is summarized in Table 1. This will give us the dimension of our feature vector at each order we specify.

Zernike moments are not, however, possess all invariances needed in practice. In particular, Zernike moments are not scale and translation invariant. We need to rely on some preprocessing to achieve these two invariances, which are indispensable in geometric shape recognition.

Order	Moments	No. of Moments
0	A_{00}	1
1	A_{11}	1
2	A_{20}, A_{22}	2
3	A_{31}, A_{33}	2
4	A_{40}, A_{42}, A_{44}	3
5	A_{51}, A_{53}, A_{55}	3
6	$A_{60}, A_{62}, A_{64}, A_{66}$	4
7	$A_{71}, A_{73}, A_{75}, A_{77}$	4
8	$A_{80}, A_{82}, A_{84}, A_{86}, A_{88}$	5
9	$A_{91}, A_{93}, A_{95}, A_{97}, A_{99}$	5
10	$A_{10,0}, A_{10,2}, A_{10,4}, A_{10,6}, A_{10,8}, A_{10,10}$	6

Table 1: Number of Zernike moments from order 0 to order 10

Preprocessing

Besides, translation and scaling normalization. Another thing that needs to be done in preprocessing is to make the input strokes have evenly distributed data points [2]. This is done by first using a data reduction filter to reduce data points that are overly close to each other and then applying a stroke interpolation procedure to add points properly so that no two adjacent points are too far away. The more evenly distributed data points on the image tends to give more consistent Zernike moments,which is crucial to this project. Obviously, the above processes need threshold values of the proper adjacency distances. To make these thresholds not depending the original scaling of the image, a prefix step is added to first map all x,y coordinates of the plot to the unit interval.

Then, data points of the image are translated so that their centroid is at the origin of the image (the center of the square matrix representing the image).

After this, the usual way is to resample the image so as to make its total pixel count equal to some fixed value; thus the scaling invariant is obtained [3]. But this process introduces sampling errors, which are annoying in the final moments generation. Instead, here we adopt a modified way [4] of treating the scaling invariance, that is, first go ahead and compute the Zernike moments and then normalize the moments directly by dividing them by the total number of pixel counts (in the binary case). From the experiments, this modification achieves scaling invariance very nicely.

To summarize, the procedure from raw plot data to the final feature vector is schemed in Figure 2.

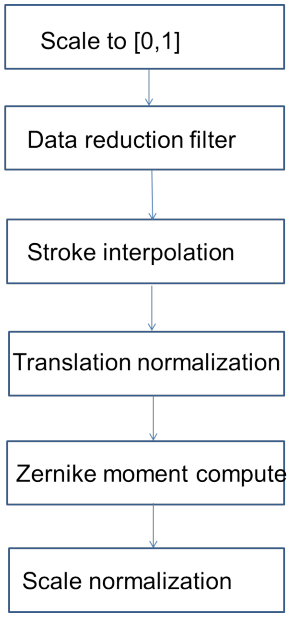

Figure 2: Schematic procedure of the method.

Results and Discussion

The key of success in the shape recognition turns out to be the preprocessing step. This makes sense in that the preprocessing (including the moment generation and normalization) produces the sole identifier of the input image, namely, the feature vector. Once the feature vector is generated with the desired invariances properties and so on, the classification stage is just cruising.

So the results start from the preprocessing. Figure 3 shows the intermediate results after different stages in preprocessing.

The input image could range in anywhere (in my case, the input area is normalized to $[0, 1]$, but the input image could only occupy a small area in the $[0, 1] \times [0, 1]$ box). The upper left plot is the one after mapping the data range to full scale of $[0, 1]$, without changing the number of data points. So, it has 149 points as that in the original plot.

Then, the upper right shows the plot after applying the data reduction filter. Points are too close to each other are filtered out and the total points is now reduced to 93. This also will make moments generation a little faster.

Then, seemingly counter intuitively, an interpolation procedure is employed to add points back. The idea is that some segment may be sketched too fast so the adjacent point distance is too large and interpolation is applied to add points in between these points so the final plot has no two adjacent points are more than some threshold distance away. The lower left plot shows the output of the interpolation and we see the total points increased a little to 101, but the points are much more evenly distributed. This, verified by experiments, will give more consistent Zernike moments.

The lower right plot shows the image after we translate the points so that the centroid is moved to the center of the image (the square). This is to take care of the translation invariance.

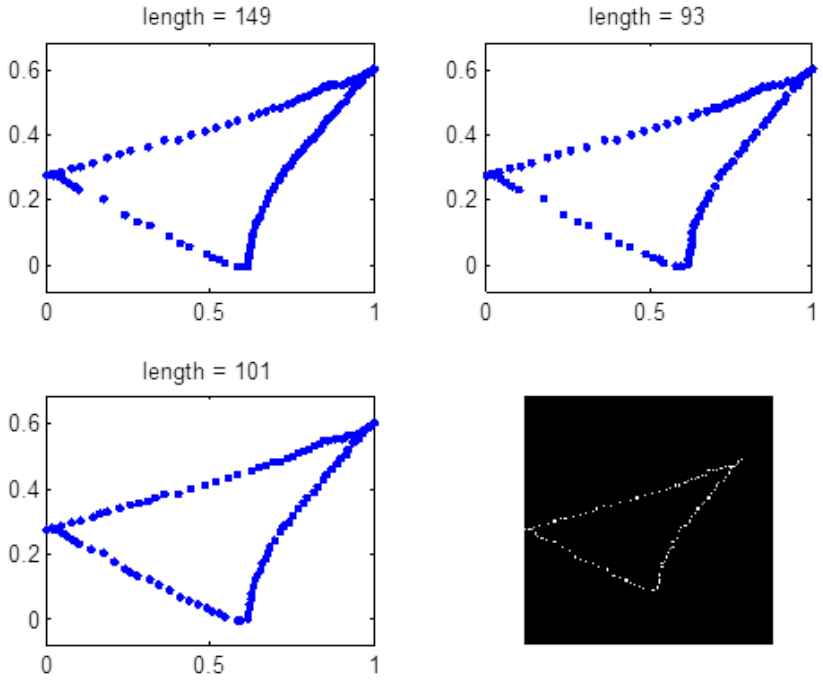

Figure 3: Preprocessing (upper left: map to [0,1], upper right: data reduction, lower left: interpolation, lower right: translation normalization).

Next, we show in Figure 4 that the final Zernike moments generated out of the above preprocessing are indeed invariant under rotation, scaling and translation. Rotation invariant is inherent in magnitude taking of Zernike moments. Translation is taken care of as mentioned above. The scaling invariance is achieved by the normalization on the computed Zernike moments.

In Figure 4, the left-hand side are two original plots input. They are quite different in scale, orientation and location. The right-hand side are the corresponding normalized Zernike moment magnitude up to order 10 (from Table 1, there are 36 moments). Notice that from the formulas and our normalization procedure, it can be shown that the first moment $|A_{00}|$ is the same for all images and A_{11} is always zero. So, in the final feature vector used for classification, the vector starts from the third moment (for instance, for order 10, there are going to be 34 features). Here, we see that the moments, especially the salient ones of the two are quite close to each other. So, in the classification, we are going to likely to group these two into the same class.

Finally, we are ready to do classification. With a training set containing 10 samples for each shape, which is drawn manually, the Zernike moments are computed up to order 10. The leave-one-out training results on different maximum Zernike moment orders are shown in Figure 5. Here, Minimum-mean-distance learning methods is compared with multi-class SVM by one-vs-one scheme. The MMD is actually weighted minimumu-mean-distance. For a test image, this method

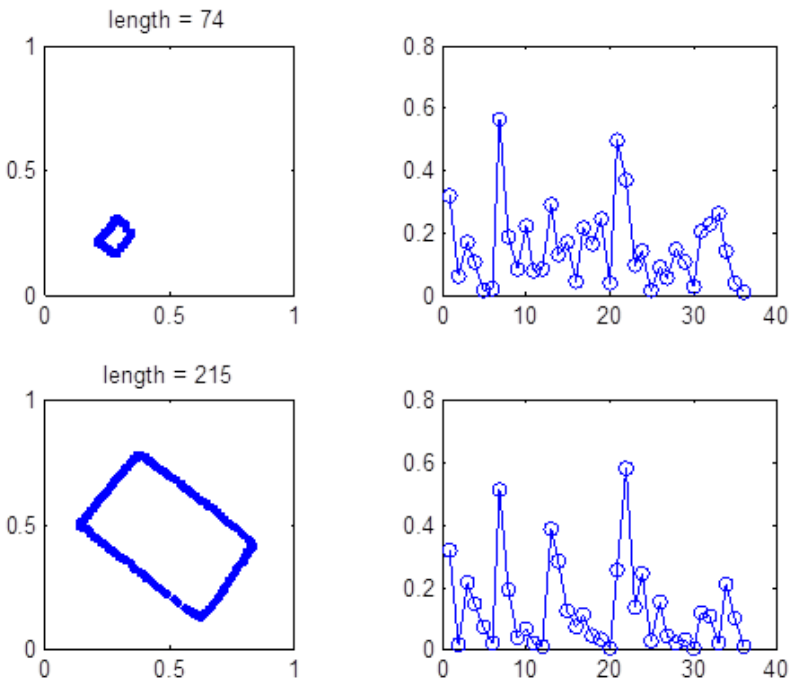

Figure 4: Rotation, scale and translation invariances.

measures a weighted Euclidean distance between the feature vector of the test image and the mean of the feature vectors of each classes, weighted feature-wise by the inverse of the variance in that feature.

Clearly, this shows that too low the order is not enough; also, it suggests, that, too high an order is also not necessary. From this figure, order 8 seems the optimal one that achieves the lowest error rate on both minimum-mean-distance and support vector machine classifiers.

Also from the above figure, the proposed method works pretty well on this small shape set. Especially, our careful preprocessing takes care of several invariances that Zernike moments do not have.

In terms of learning methods, SVM performances slightly better and should be the preferred method to use in practice. MMD has the advantage of being fast, so if real-time processing is at risk, MMD might be considered as an alternative.

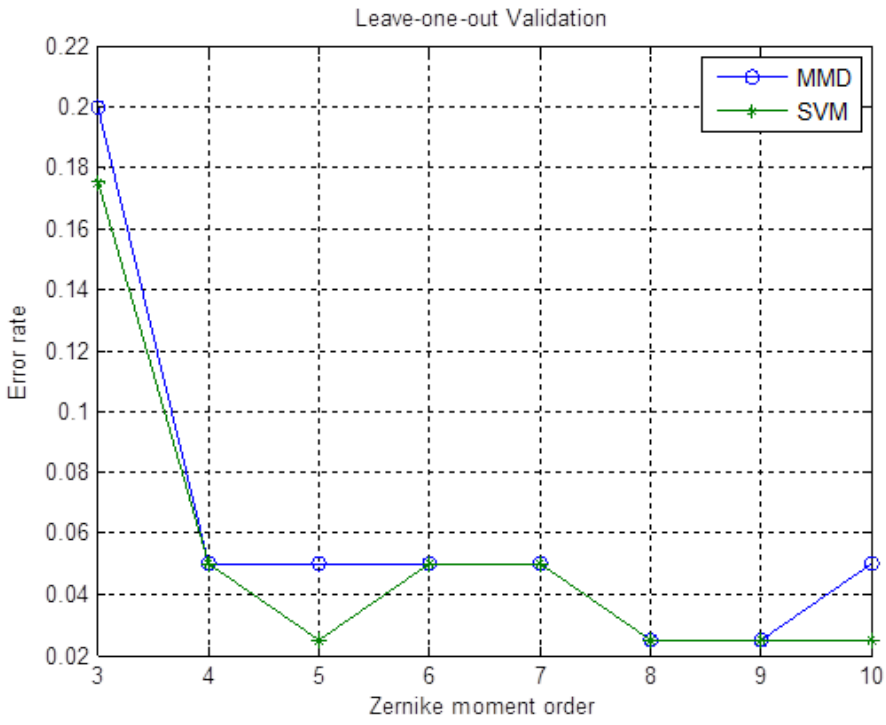

Figure 5: Training error vs Zernike moment order.

Future Work

The obvious first one is to expand the shape data set. This can be readily done given the nature of the method is statistical.

Another direction toward application is to add some functionality recognition such as delete, select and move, so that this system can be easily used to draw figures made of simple geometric shapes. This is also called gesture recognition in the literature.

Bibliography

[1] Heloise Hwawen Hse and A. Richard Newton. *Sketched symbol recognition using zernike moments.* In ICPR (1), pages 367-370, 2004. doi: 10.1109/ICPR.2004.1334128

[2] Hanaki, S., Temma, T. and Yoshida. *An on-line character recognition aimed at a substitution for a billing machine keyboard.* Pattern Recognition, 8. a63-71.

[3] Khotanzad, A, and Hong, Y.H. *Invariant image recognition by Zernike Moments.* IEEE Trans. on PAMI, 12(5). 489-497.

[4] Ye Bin and Peng Jia-Xiong. *Invariance analysis of improved Zernike moments.* J. Opt. A: Pure Appl. Opt. 4, 2002.