Recognizing Simple Geometric Figures Using Fuzzy Expert System

Zhang Linghao
13307130225@fudan.edu.cn

1 Introduction

This is the report for the project of ***Introductin to Artificial Intelligence*** *course (2015 Fall) at Fudan University.*

Our task is to build a rule-based expert system to recognize simple geometric figures, such as circles, triangle, rectangles, etc. In this report, we describe the design and implementation of a fuzzy expert system for solving the task, evaluate its performance, and finally discuss its limitations.

For the convenience of narration, this report will use *we* as the first person pronoun and present tense with all verbs.

2 Image Generation

2.1 Overview

Training images are vital to our task. Computer generated images could be too standard so as to have very little noise, making our model vulnerable to possible noisy test images; hand-drawn sketches are too noisy, on the other hand, raising challenges for image feature extraction and complicating our rules. Both senarios could diminish the generalization ability of our model. Therefore we manage to generate three different kinds of images.

First in the system prototyping stage, we use a painting tool to generate several images of circles, ellipses, triangles, rectangles and squares as the **development set**. Note that these images are not generated in vector format on purpose, introducing small noises on a pixel level. We hope that by analyzing these quasi-standard images, we can derive the basic framework of our system.

Then in the tuning stage of the project, we use image processing techniques to distort and rotate the standard figures, obtaining a great amount of images and split them into **training set and testing set**. Then we use the training set to improve the system.

Finally in the testing stage, we use the testing set obtained in the previous stage. We also

ask different people to **draw sketches with a digitized tablet**. We decide not to use mouse because that would produce overly noisy images.

2.2 Details

2.2.1 Computer-generated Images

Images in training set and testing set are generated as follows:

First we produce a random image of the target figure. Then we distort the contour of the figure by applying a Sine function on each row and column of the image, with a given distortion factor to control the distortion effect. At last we rotate the image by a random angle.

Finall we obtain 200 images for each figure, and we split them into training set and testing set with a $9:1$ ratio. The images look like this (Figure 1).

2.2.2 Hand-drawn Sketches

The sketches drawn by one (under development) difference person(s) look like this (Figure 2).

3 Feature Extraction

Feature extraction is mainly done with OpenCV.

3.1 Contours

Contours are probably the most starightforward feature that one can be think of. With the help of OpenCV, once we've found the contour of the target figure, we can calcuate the following:

- Perimeter: apply cv2.arcLength on the contour gives us an approximation of the perimeter.
- Area: apply cv2.contourArea on the contour gives us an approximation of the area.

3.2 SIFT

SIFT allows us to easily find keypoints(corners) in the image. At first glance it seems to be a great features, since it can distinguish circle/ellipse from triangles and rectangles/squares. However, the extraction of this feature it's highly unstable. A little noise in the image could increase the number of keypoints by, like, 20. Although it's possible to set some thresholds to tune out the undesired keypoints, still we couldn't put too much confidence on this feature.

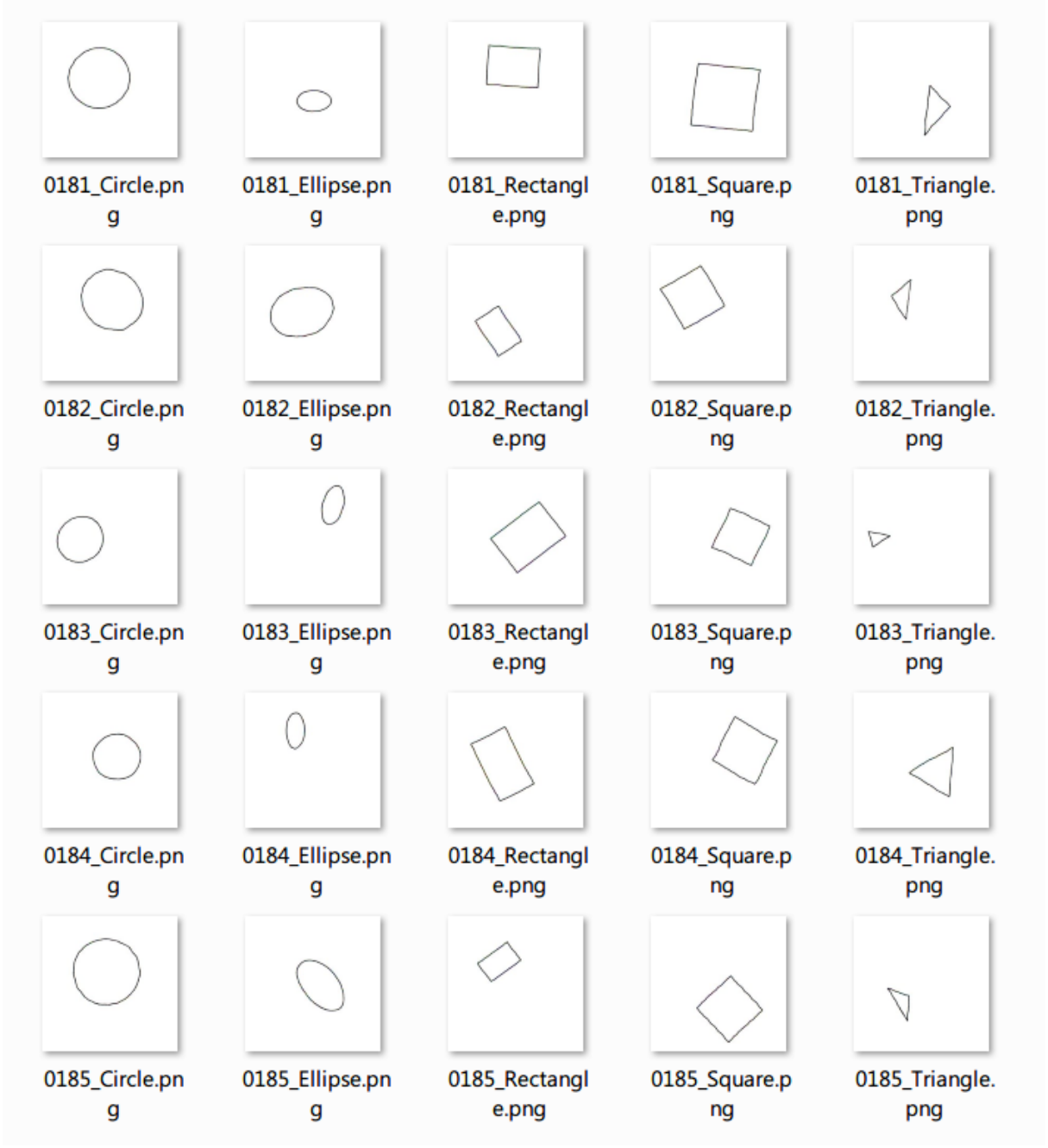


Figure 1: Computer-generated Images

3.3 Feature Expansion

In computer vision, when it comes to geometric figures, there are several frequently used features that we can easily obtain using the existing features. Some of them turn out to be useful for our task:

- $Thinness = \frac{P^2}{A}$: Thinness measures the degree to which the figure is 'economical' in using its perimeter to enclose an area.

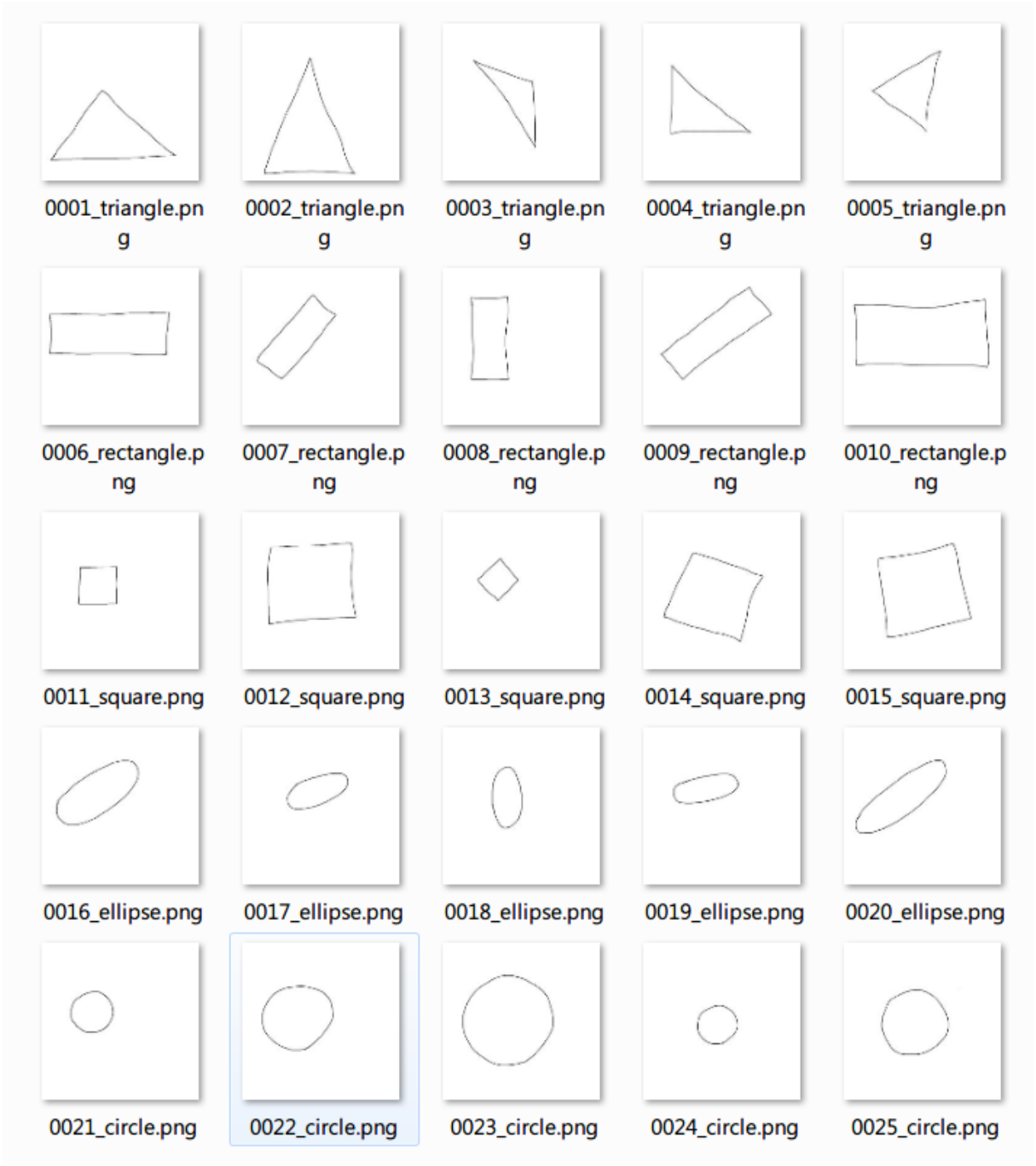


Figure 2: Hand-drawn Sketches

- $Extent = \frac{A}{A_{BR}}$: Extent measures the how much space the figure takes up in its bounding rectangle (the rectangle of the smallest area that bounds the target figure).

Technically, we can generate more features based on current ones, as long as they are useful for in our task. Currently features are selected based on geometric knowledge (maybe "common sense" is a better word) and observation of the training data. It is entirely possible to apply some machine learning techniques to autonomously generate and select new features. We will

discuss this in Section 7.

4 Rule Derivation

4.1 Manual Rule Derivation

After observing a couple of patterns from data in the development set and reasoning about their validity, we come up with several heuristics.

First we notice that *Extent* is a very good feature to start with: triangles have a *Extent* ratio close to 0.5, rectangles and squares have a *Extent* ratio close to 1, while circles and ellipses have a *Extent* ratio of around $0.75 \sim 0.85$.

Then we use *Thinness* ratio to distinguish circles and ellipses. Among common 2D figures, circle has the smallest thinness ($= 4\pi$). Similarily, when rectangles have *Thinness* ratio close to 16, they become squares.

Finally, we obtain 5 simple rules:

- *IF Thinness IS LIKE Circle AND Extent IS LIKE ellipse THEN Shape IS Circle*
- *IF Thinness IS NOT LIKE Circle AND Extent IS LIKE ellipse THEN Shape IS Ellipse*
- *IF Extent IS LIKE Triangle THEN Shape IS Triangle*
- *IF Thinness IS LIKE Square AND Extent IS LIKE rectangle THEN Shape IS Square*
- *IF Thinness IS NOT LIKE Square AND Extent IS LIKE rectangle THEN Shape IS Rectangle*

4.2 Autonomous Rule Derivation

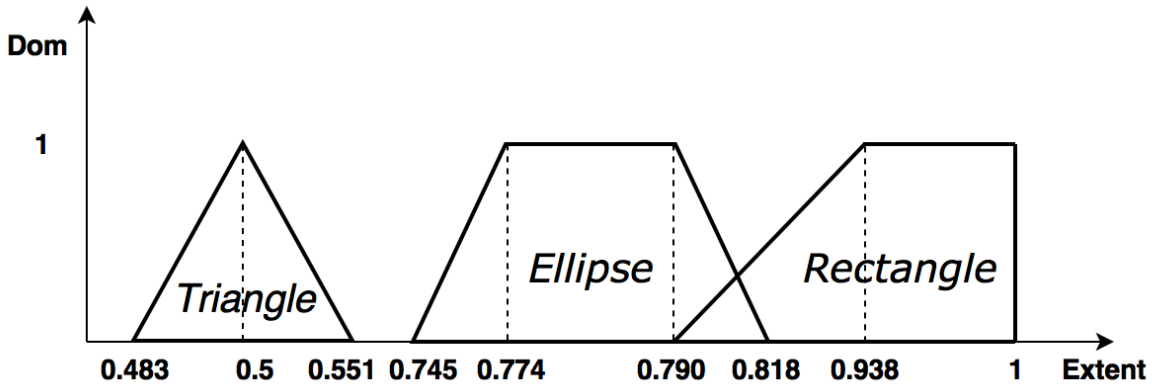
We will discuss the possibility of autonomous rule derivation in Section 7.

5 Fuzzy Inference

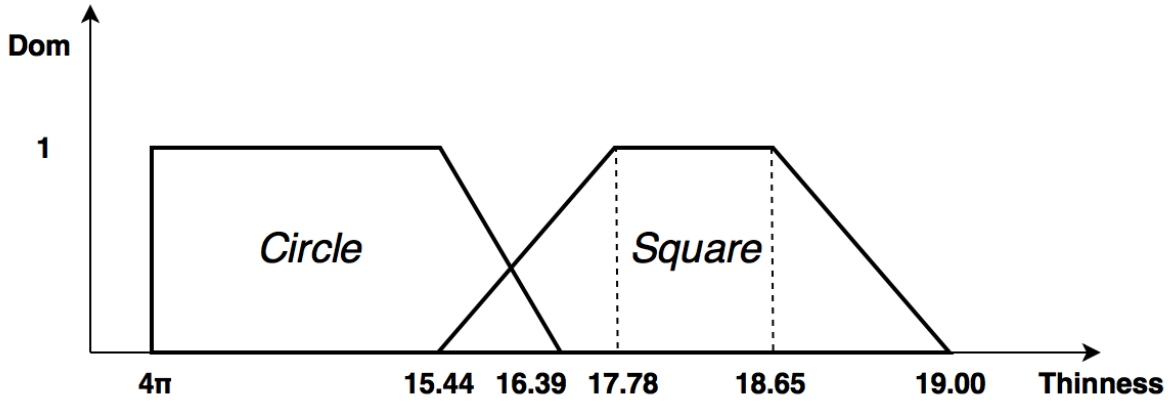
5.1 Determining Fuzzy sets

As mentioned in the previous section, the noises in the image inevitably introduces uncertainties to our system. Therefore the feature values referenced in our rules have to be represented in fuzzy sets.

To determine the **best fuzzy sets capturing the appropriate degree of uncertainties**, we use the statistics from the training data to find the optimal threshold.



(a) Fuzzy Set of Extent



(b) Fuzzy Set of Thinness

For *Thinness*, we use 3 fuzzy sets for ellipse-like, rectangle-like and triangle-like *Extent* ratio. For *Extent*, we use 2 fuzzy sets for circle-like and square-like *Thinness* ratio. The shapes and boundries of these fuzzy sets are determined based on the **distribution of feature values in the training data and their 25-th, 75-th and extreme percentiles**. The final fuzzy sets are illustrated in Figure 3 and Figure 4 (Figures not drawn to scale).

5.2 Implementing Inference Engine

Now that the fuzzy sets are determined, we have to implement the inference engine. However, the specific nature of our task requires us to make some adaptations to the the usual fuzzy inference engine.

Firstly, unlike the example from the textbook, the set of possible outputs in our task is **not** a set of linguistic values which can be represented by a series of contiguous intervals. We wouldn't want to compute a COG-like output and decide which category it falls in, because you can't assign shapes with a reasonable order.

Secondly, rules derived in the previous section are all of the format:

IF X IS LIKE A AND Y IS NOT LIKE B THEN Shape IS C

Note that the consequent of each rule is an assertion about the shape of the input figure and that **each rule corresponds to the recognition process for one specific shape**. In fact a fuzzy inference system of this kind is essentially a flattened decision tree, with each rule acting as a filter to calculate the possibility of one specific shape. The only difference is that in our expert system rules are separated from inference, making it easily maintainable.

Thus, in order to gain the advantages brought by fuzzy inference without over-complicating our task, we choose to implement **a simplified Sugeno-style inference engine**. In the final defuzzification stage, we don't compute a weighted average of the rule outputs. Instead we examine all the possible values for the target variable *Shape* and choose the one with the highest possibility as the output.

5.3 Choosing Operators

Under development.

6 Evaluation

The evaluation consists of two parts: one with computer-generated test set, one with hand-drawn sketches.

6.1 Computer-generated Test Set

6.1.1 Experimental Result

The confusion matrix for the evaluation on computer-generated test set is in Table 1. The test set contains 100 figures with 20 figures from each category.

Label \ Recognized	Circle	Ellipse	Triangle	Square	Rectangle
Circle	1.00	0.00	0.00	0.00	0.00
Ellipse	0.10	0.90	0.00	0.00	0.00
Triangle	0.00	0.00	1.00	0.00	0.00
Square	0.00	0.00	0.00	0.75	0.25
Rectangle	0.00	0.00	0.00	0.30	0.70

Table 1: Confusion Matrix on Testing Data

6.1.2 Error Analysis

We see that the main cause of error is that squares and rectangles are sometimes mixed up. This makes sense because some of rectangle images in our data do look quite square. To deal

with this type of error, we may consider using new features. Under development.

6.2 Hand-drawn Sketches

Under development.

7 Discussion

This section discusses the limitations (mainly about the expandability of the system) of our project and how it can be improved.

7.1 Limited Set of Shape Categories

Currently we can only recognize between 5 figures. We plan to incorporate more figures into the system, as well as try autonomous feature generation and rule derivation on the new data.

Under development.

7.2 About Image Processing

Currently, for the feature extraction to work smoothly, we make several assumptions. Mainly the contour lines in the image cannot be to thin, otherwise OpenCV would have problem finding the desired contour. While it is possible to apply some image transformations before feature extraction, it still remains a quite complicated (and out of the scope of this project) task to handle all kinds of visually valid images. Therefore we will only use thick-lined images in our project.

7.3 How to Derive New Rules

Since rules and inference are separated in our system, there's no doubt that if we obtain some new rules we could just add them to the system. However, the main challenge here is how to derive these rules. Currently, we don't have a fully automatic method to do this. Rules must be derived with human intervention. However we could make this process a lot easier by using a decision tree model to help identify features that have discriminative power.

Under development.

7.4 How to Generate New Features

Another problem is how to generate new features. Currently we use a subset of commonly used features. Of course we could use the whole set and select the useful ones, but that leaves us with two problems: 1. How to integrate these newly found features into our system automaticlly?

2. Can we geneate features outside that set? I'm afraid I haven't got an autonomous solution that meets satisfaction.

With the second problem, I believe we could utilized genetic algorithms. A new feature could be represented by the product of some old features raised to a certain power. So we can encode the exponets of each base feature into a chronosome. And the fitness could be computed by testing how much information gain (as in the decision tree) could be achieve by using the feature encoded in the chronosome.

Under development.

8 References

1. Joaquim A Jorge and Manuel J Fonseca. A simple approach to recognise geometric shapes interactively. In *Graphics Recognition Recent Advances*, pages 266–274. Springer, 2000

2. Manuel J Fonseca, Joaquim Jorge, et al. Using fuzzy logic to recognize geometric shapes interactively. In *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, volume 1, pages 291–296. IEEE, 2000