

Evaluation of OCR Algorithms for Images with Different Spatial Resolutions and Noises

by

Qing Chen

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of

Master of Applied Science

Ottawa-Carleton Institute for
Electrical Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

To my parents

Abstract

Various shaped-based image invariants are popular algorithms used in optical character recognitions (OCR), 3D object recognitions, and pattern recognitions. The shape-based image invariants can be divided into two different categories: boundary-based image invariants such as Fourier descriptors and chain code; region-based image invariants including various moment-based invariants such as Hu's seven moment invariants and Zernike moments.

This thesis introduced and evaluated different shape-based image invariants from the perspective of their invariance property to image transformations including scaling, translation, rotation and different image spatial resolutions. The influence caused by salt and pepper noises of different intensities is also analyzed. The image reconstruction ability of Zernike moments is implemented and discussed as well. An OCR engine is implemented with MATLAB scripts to perform the image feature extraction and image recognition. The OCR engine is also used in the overall performance evaluation of Fourier descriptors and Hu's seven moment invariants.

Acknowledgement

First of all, my sincere gratitude goes to my academic supervisor Dr. Emil M. Petriu, who helped and guided me to finish this work. His conversation and encouragements will be always remembered.

Special thanks goes to Rami Abielmona, the fellow Ph.D student of the SMR Lab, who helped me on computers, Internet connections and software.

I would like to give my appreciation to my parents, my wife, my brother and sister. Their encouragements and support accompanied me through the two years' study.

Acronyms

2D — 2 Dimensional

3D — 3 Dimensional

AGV — Automated Guided Vehicle

DPI — Dots Per Inch

FFT — Fast Fourier Transform

MCR — Magnetic Character Recognition

OCR — Optical Character Recognition

PDA — Personal Digital Assistant

PRBA — Pseudo Random Binary Array

Table of Contents

Abstract	III
Acknowledgement	IV
Acronyms	V
Table of Contents	VI
List of Figures	X
List of Tables.....	XV
Chapter 1 Introduction.....	1
1.1 OCR in 3D Model-Based Object Recognition.....	2
1.2 Motivation.....	5
1.3 Research Scope	6
1.4 Contributions.....	8
1.5 Thesis Contents	9
Chapter 2 Background Review	10
2.1 Optical Character Recognition.....	10
2.1.1 Different Families of Character Recognition	11
2.1.2 OCR's History	13
2.2 Image Features	15
2.3 The OCR Engine.....	17

2.4 Affine Transformation.....	19
2.5 Image Resolution.....	21
Chapter 3 Image Invariants	23
3.1 Introduction.....	23
3.2 Boundary-Based Invariants.....	25
3.2.1 Chain Code.....	25
3.2.2 Fourier Descriptors.....	27
3.3 Region-Based Invariants.....	31
3.3.1 Regular Moments and Central Moments.....	32
3.3.2 Hu's Seven Moment Invariants.....	34
3.3.3 Zernike Moments.....	35
3.3.4 Pseudo-Zernike Moments.....	37
3.3.5 Complex Moments.....	38
Chapter 4 Experiment Implementations.....	40
4.1 Introduction.....	40
4.2 Image Samples.....	42
4.3 Implementation Descriptions.....	46
4.3.1 Fourier Descriptors.....	46
4.3.2 Hu's Seven Moment Invariants.....	47
4.3.3 Zernike Moments.....	48

4.4 The OCR Engine	51
Chapter 5 Experiment Results	56
5.1 Introduction	56
5.2 Results of Fourier Descriptors	57
5.2.1 Value Distribution	57
5.2.2 Invariance to Different Spatial Resolutions	59
5.2.3 Sensitivity to Salt and Pepper Noises	62
5.2.4 Overall Performance Evaluation	63
5.3 Results of Hu's Seven Moment Invariants.....	64
5.3.1 Value Distribution	64
5.3.2 Invariance to Different Spatial Resolutions	66
5.3.3 Invariance to Salt and Pepper Noises	71
5.3.4 Overall Performance Evaluation	75
5.4 Results of Zernike Moments	77
5.4.1 Reconstruction Property Evaluation.....	77
Chapter 6 Conclusions	80
6.1 Conclusions	80
6.2 Future Work	82
Appendix MATLAB Scripts.....	83
A.1 ocr	83

A.2 getfd	90
A.3 trace_boundary	91
A.4 resample.....	92
A.5 moment.....	94
A.6 centroid.....	94
A.7 central_moment	95
A.8 normalized_moment.....	95
A.9 hu	95
A.10 bi.....	96
A.11 z_zmoment	97
A.12 z_invariant	99
A.13 z_zmrecon.....	100
References	102

List of Figures

Figure 1.1: The OCR engine in the 3D model-based object recognition system.....	2
Figure 2.1: The basic processes of an OCR system.....	10
Figure 2.2: The different families of character recognition.....	11
Figure 2.3: The structure of Tausheck’s “Reading Machine”.....	13
Figure 2.4: Examples of affine transformations.....	20
Figure 2.5: A 512×512 image sub-sampled down to 16×16.....	21
Figure 2.6: Results of stretching the sub-samples to the same size of the original 512×512 image.....	21
Figure 3.1: Chain code numbering schemes: (a) Direction for 4-connectivity chain code. (b) Direction for 8-connectivity chain code.....	25
Figure 3.2: An example of the 8-connectivity chain code.....	26
Figure 3.3: A re-sampled boundary and its representation as a complex sequence.....	27
Figure 3.4: Examples of reconstruction from Fourier descriptors. P is the number of Fourier coefficients used in the reconstruction process.....	28
Figure 4.1: The approximation process of images with multiple contours.....	42

Figure 4.2: A set of image samples from spatial resolution 512×512 to 16×16.....	43
Figure 4.3: Images corrupted by salt and pepper noise and Gaussian noise.....	44
Figure 4.4: Images corrupted by salt and pepper noise with different densities	44
Figure 4.5: Block diagram of computing Fourier descriptors.....	46
Figure 4.6: Block diagram of computing Hu’s seven moment invariants.....	47
Figure 4.7: Block diagram of computing Zernike moments.....	48
Figure 4.8: Comparison between the original image and the translation-normalized image.....	49
Figure 4.9: Comparison between the original image and the scale-normalized image.....	49
Figure 4.10: The computation process of the unit disk mapping for Zernike moments....	50
Figure 4.11: The screenshot of the OCR engine.....	51
Figure 4.12: The working process of the OCR engine.....	52
Figure 4.13: Classification result of image “A” of 128×128 rotated by 45 degree (counterclockwise).....	53
Figure 4.14: Classification result of image “C” of 128×128 rotated by 45 degree (clockwise).....	54

Figure 4.15: Classification result of image “F” of 128×128 rotated by 90 degree (counterclockwise).....	54
Figure 4.16: Classification result of translated image “L” of 190×190 rotated by 90 degree (counterclockwise).....	55
Figure 4.17: Classification result of translated image “S” of 190×190 rotated by 45 degree (clockwise).....	55
Figure 5.1: The Fourier descriptors’ distribution of the image “A” with spatial resolution of 512×512.....	57
Figure 5.2: The Fourier descriptors’ distribution of the image “Z” with spatial resolution of 512×512.....	57
Figure 5.3: The distribution of the first 10 Fourier descriptors of the 26 capital English letters’ images with spatial resolution of 512×512.....	59
Figure 5.4: The first 10 Fourier descriptors of the image “A” with spatial resolutions from 512×512 to 32×32.....	60
Figure 5.5: The first 10 Fourier descriptors of the image “Z” with spatial resolutions from 512×512 to 32×32.....	61

Figure 5.6: The filtering result of a 3×3 median filter on a salt and pepper affected image	62
Figure 5.7: The distribution of Hu’s 7 moment invariants of the 26 English capital letters’ images with spatial resolution of 512×512.....	65
Figure 5.8: The distribution of the 26 capital English letters with the first three moment invariants as the coordinates	66
Figure 5.9: The Hu’s moment invariants of the image “C” with spatial resolutions from 512×512 to 16×16.....	68
Figure 5.10: The Hu’s moment invariants of the image “T” with spatial resolutions from 512×512 to 16×16.....	69
Figure 5.11: The Hu’s moment invariants of the image “V” with spatial resolutions from 512×512 to 16×16.....	70
Figure 5.12: The Hu’s moment invariants of the image “C” affected by salt and pepper noises with intensities from 0.01 to 0.07.....	72
Figure 5.13: The Hu’s moment invariants of the image “T” affected by salt and pepper noises with intensities from 0.01 to 0.07.....	73
Figure 5.14: The Hu’s moment invariants of the image “V” affected by salt and pepper noises with intensities from 0.01 to 0.07.....	74

Figure 5.15: The distribution of the Zernike moments' magnitudes up to order
30 of image "B" with resolution of 64×64.....77

Figure 5.16: Reconstruction results with Zernike moments of
order 5, 10, 20, 30, 40.....78

List of Tables

Table 3.1: Basic properties of Fourier descriptors under some transformations.....	29
Table 5.1: The first 10 Fourier descriptors of the 26 capital English letters’ images with spatial resolution of 512×512.....	58
Table 5.2: The first 10 Fourier descriptors of the image “A” with spatial resolutions from 512×512 to 32×32.....	60
Table 5.3: The first 10 Fourier descriptors of the image “Z” with spatial resolutions from 512×512 to 32×32.....	61
Table 5.4: The recognition result using Fourier descriptors for the images of 26 capital English letters with resolutions from 512×512 to 32×32.....	63
Table 5.5: The Hu’s 7 moment invariants of the 26 capital English letters’ image samples with spatial resolution of 512×512.....	64
Table 5.6: The Hu’s 7 moment invariants of the image “C” with spatial resolutions from 512×512 to 16×16.....	68
Table 5.7: The Hu’s 7 moment invariants of the image “T” with spatial resolutions from 512×512 to 16×16.....	69
Table 5.8: The Hu’s 7 moment invariants of the image “V” with	

spatial resolutions from 512×512 to 16×16.....	70
Table 5.9: The Hu's 7 moment invariants of the image "C" affected by salt and pepper noises with intensities from 0.01 to 0.07.....	72
Table 5.10: The Hu's 7 moment invariants of the image "T" affected by salt and pepper noises with intensities from 0.01 to 0.07.....	73
Table 5.11: The Hu's 7 moment invariants of the image "V" affected by salt and pepper noises with intensities from 0.01 to 0.07.....	74
Table 5.12: The recognition result using Hu's moment invariants for the images of 26 capital English letters with resolutions from 512×512 to 32×32.....	75
Table 5.13: The recognition result using Hu's moment invariants for the images of 26 capital English letters corrupted by salt and pepper noises with intensities from 0.01 to 0.04.....	76

Chapter 1

Introduction

Optical character recognition (OCR) is a popular research topic in pattern recognition [18]. OCR is aimed to enable computers to recognize optical characters without human intervention. The applications of OCR have been found in automated guided vehicles (AGV), object recognitions, digital libraries, invoice and receipt processing, and recently on personal digital assistants (PDAs) [23]. OCR includes essential problems of pattern recognition, which are common to other related topics such as 3D object recognition and image retrieval systems.

Various OCR algorithms have been proposed to achieve better recognition results. These algorithms include template matching, image signatures, geometric features, and shape-based image invariants [33]. Among all of the OCR algorithms, the shape-based image invariants are of particular interests as they have the invariance property, which can improve the recognition performance even the images have undergone a certain class of image transformations such as translation, scaling and rotation. There are two types of shape-based image invariants: boundary-based and region-based [39]. The boundary-based image invariants focus on the properties contained in the image's contour while the region-based image invariants take the whole image area as the research object.

1.1 OCR in 3D Model-Based Object Recognition

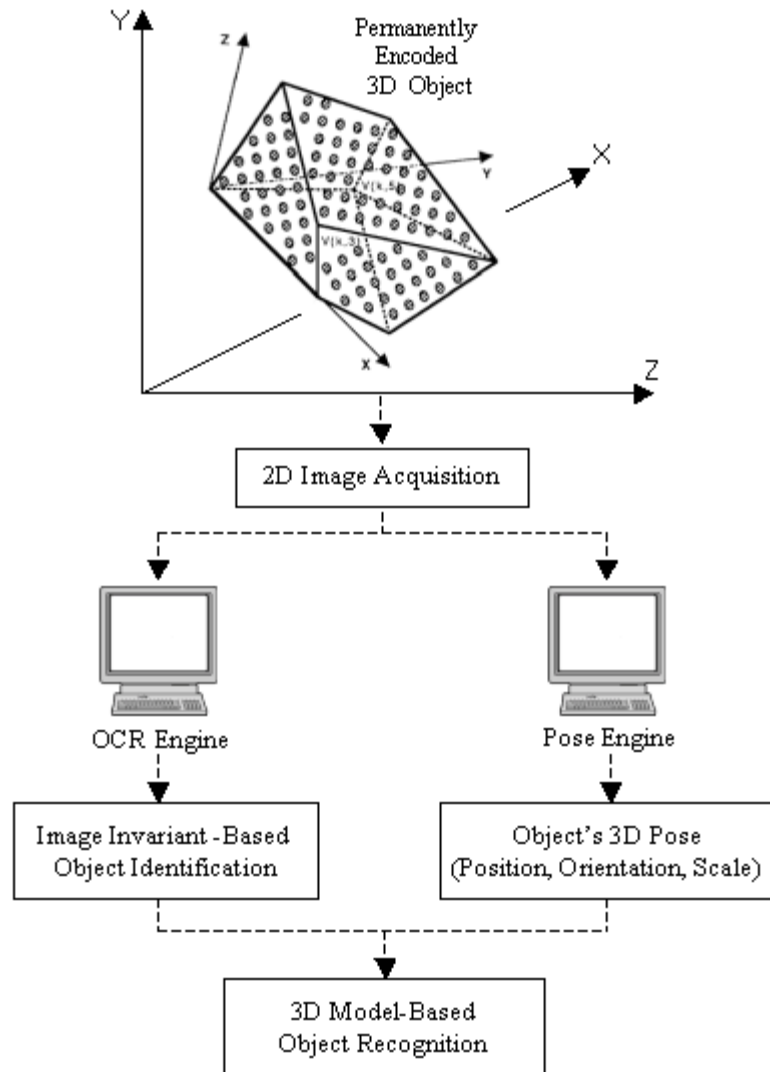


Figure 1.1: The OCR engine in the 3D model-based object recognition system.

OCR can play a very important role in 3D Model-based object recognition systems as Figure 1.1 shows. Two basic operations are included in 3D Model-based object recognition: identification and location [34]. Identification determines the identity of the

imaged objects by the comparison of the image data with a database of models. Location determines the location of the 3D imaged object in the space.

A practical and rapid method for visual recognition of 3D objects is the use of surface encoding with Pseudo Random Binary Array (PRBA) and feature matching with the model database [36]. After the object is visually recognized, a pose engine provided with tactile sensors can find its positional parameters.

The PRBA code is Braille-like symbols embossed on the object surface. The shape of the symbols is specially designed for easy visual and tactile recognition. For an efficient pattern recognition, the particular shapes of the binary symbols were selected in such a way to meet the following conditions [36]:

1. there is enough information at the symbol level to provide an immediate indication of the grid orientation.
2. the symbol recognition procedure is invariant to position and orientation changes.
3. the symbols have a certain peculiarity so that other objects in the scene will not be mistaken for encoding symbols.

One key problem of symbols recognition is that the appearance of the symbols depends on imaging conditions like viewpoint and orientations. If we can find image features that do not change with imaging conditions, the problem would be solved.

The image invariants can be described as functions of geometric configurations which do not change under a certain class of image transformations. The OCR engine based on

image invariants supports direct, feature-based model indexing, and therefore well-suited to identify the specific subsets of the PRBA codes embossed on the object's surfaces. However, one basic limitation of image invariants is that they are only invariant to a certain class of image transformations. Defining useful invariants for all image transformations is not easy at all. In this thesis, we consider only scaling, translation and rotation/orientation of the geometric image transformations.

After the encoded symbols are recognized, it is possible to immediately recover the identity, position and orientation of the imaged object by consulting a database which stores the PRBA mapping relationship [36].

1.2 Motivation

Faced with all of the OCR algorithms, one question to be answered is: which algorithm is the best choice for a given application? This question led the thesis to characterize the available OCR algorithms so that the most promising methods can be sorted out for different applications. An experimental approach is developed to compare and evaluate the performance of different shape-based image invariants.

Meanwhile, as the region-based image invariants take the whole image area into account, the total number of pixels contained in the image is a decisive factor for the algorithms' computation cost. An image with 1024×768 pixels will certainly take much longer time to compute than a 64×32 image. The investigation of the spatial resolution threshold for the region-based image invariants is also a major motivation for this thesis.

The OCR algorithms may perform well for the ideal clean images. However, in the practical image acquisition systems and conditions, noise is a common phenomenon in the acquisition process. The observed image represents only a degraded version of the original scene. Recognition of objects and patterns that are corrupted by various noises has been a goal of recent research [28]. Evaluating the performance of the shape-based image invariants under noise-affected images is also of the thesis's interest.

1.3 Research Scope

The research in OCR usually includes two stages, “low-level” and “high-level”. Low-level involves extracting features from images such as the boundary of a character or regions with the same texture. The task of “high-level” is then to recognize these objects with the extracted features.

This thesis is concerned with both “low-level” and “high-level” stages, in particular with finding properties of an image which are invariant to image transformations including scale, translation and rotation. The idea of this arises from a project of the SMRLAB at University of Ottawa. The name of this project is “Visual Position Recovery for an Automated Guided Vehicle (AGV)” [15]. In this project, an AGV can recover its absolute position anywhere by a pseudo-random binary sequence encoded with two graphical symbols “H” and “E”. The perspective of the camera installed on the AGV to the visually encoded guide-path is fixed at a certain angle. With the position change of the AGV, the scale, position and orientation of the graphical symbols may be changed. Finding mathematical representation functions of the image that are invariant to above transformations would thus provide the AGV with more stable position recovery ability.

A full performance evaluation of each OCR algorithm in terms of classification accuracy, invariance property against different resolutions and noise as well as the computation efficiency is not possible to be included in this thesis. Instead, we have to make a representative selection to illustrate the major principles.

In this thesis, we contained our research on gray level image samples of 26 capital English letters. For boundary-based image invariants, we mainly focused on the chain code and Fourier descriptors. For region-based image invariants, we implemented central moments, Hu's seven moment invariants and Zernike moments. For noise-corrupted images, we mainly investigated the impacts caused by salt and pepper noises with different intensities.

1.4 Contributions

The main contributions of this thesis can be summarized as follows:

1. evaluated the invariance property against image transformations including scaling, translation, rotation and different image spatial resolutions for Fourier descriptors and Hu's seven moment invariants. We implemented the algorithms of Fourier descriptors, the central moments and Hu's seven moment invariants.
2. found the resolution threshold for Fourier descriptors and Hu's seven moment invariants. The resolution threshold is 64×64 for Fourier descriptors, and 128×128 for Hu's seven moment invariants.
3. investigated the sensitivity to salt and pepper noises for Fourier descriptors and Hu's seven moment invariants. The image features are collected and plotted from image samples with different noise intensities.
4. implemented the algorithm of Zernike moments and investigated their reconstruction ability. The thesis proposed a proper feature vector size for Zernike moments which can meet the recognition requirements based on the analysis.
5. developed a generic OCR engine which can be used for different purposes. This OCR engine includes two basic functions: feature extraction and image recognition. It integrated Fourier descriptors and Hu's seven moment invariants together. The users can use this OCR engine to evaluate and compare their overall performances on gray level "JPEG" images and "TIFF" images.

1.5 Thesis Contents

Chapter 1 introduces the motivation, research scope, and contributions of the thesis.

Chapter 2 is the general background review. It gives a general introduction to OCR, its classifications and history. Then it introduces the concept of image features and the OCR engine. Other related concepts including image resolutions, affine transformations are also introduced in this chapter.

Chapter 3 focuses on the image invariants from the theoretical point of view: their definitions, the mathematical proofs for their invariance properties to scaling, translation, and rotation, their advantages and disadvantages.

Chapter 4 describes the experiment implementation process of the image invariants. The image library used in the experiment is also introduced.

Chapter 5 presents and analyzes the experiment results.

Chapter 6 gives the conclusion of the thesis.

Chapter 2

Background Review

2.1 Optical Character Recognition

Optical character recognition (OCR) is an important research area in pattern recognition. The objective of an OCR system is to recognize alphabetic letters, numbers, or other characters, which are in the form of digital images, without any human intervention [25]. This is accomplished by searching a match between the features extracted from the given character's image and the library of image models. Ideally, we would like the features to be distinct for different character images so that the computer can extract the correct model from the library without any confusion. At the same time, we also want the features to be robust enough so that they will not be affected by viewing transformations, noises, resolution variations and other factors. Figure 2.1 illustrates the basic processes of an OCR system.

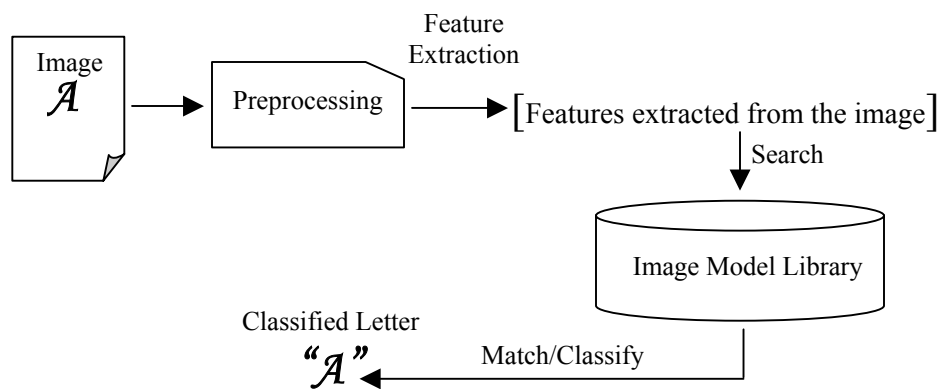


Figure 2.1: The basic processes of an OCR system.

2.1.1 Different Families of Character Recognition

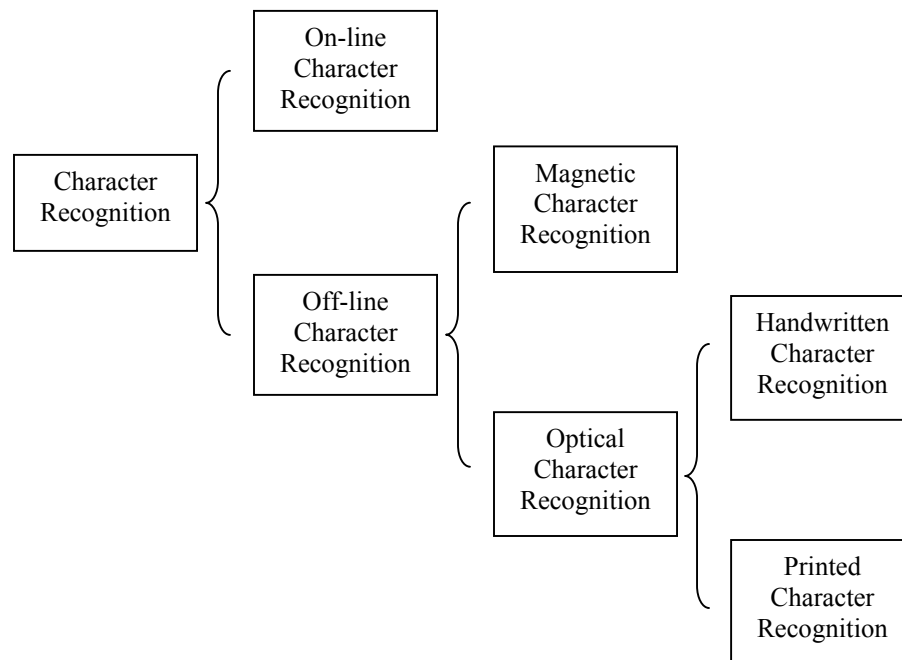


Figure 2.2: The different families of character recognition.

Figure 2.2 shows the different families of character recognition. Two different families are included in the general term of character recognition [13]:

- On-line character recognition
- Off-line character recognition

On-line character recognition deals with a data stream which comes from a transducer while the user is writing. The typical hardware to collect data is a digitizing tablet which is electromagnetic or pressure sensitive. When the user writes on the tablet, the successive movements of the pen are transformed to a series of electronic signal which is memorized and analyzed by the computer [29].

Off-line character recognition is performed after the writing is finished. The major difference between on-line and off-line character recognition is that on-line character recognition has time-sequence contextual information but off-line data does not. This difference generates a significant divergence in processing architectures and methods. The off-line character recognition can be further grouped into [41]:

- Magnetic character recognition (MCR)
- Optical character recognition (OCR)

In MCR, the characters are printed with magnetic ink. The reading device can recognize the characters according to the unique magnetic field of each character. MCR is mostly used in banks for check authentication..

OCR deals with the recognition of characters acquired by optical means, typically a scanner or a camera. The characters are in the form of pixelized images, and can be either printed or handwritten, of any size, shape, or orientation [21]. The OCR can be subdivided into handwritten character recognition and printed character recognition [23]. Handwritten character recognition is more difficult to implement than printed character recognition due to the diversified human handwriting styles and customs. In printed character recognition, the images to be processed are in the forms of standard fonts like *Times New Roman*, *Arial*, *Courier*, etc.

2.1.2 OCR's History

As early as 1929, Tausheck obtained a patent named “Reading Machine” in Germany [30]. This patent reflects the basic concept of today’s OCR. The principle of Tausheck’s patent is template matching which is still used in some applications even today. Figure 2.3 illustrates the detailed structure of this patent.

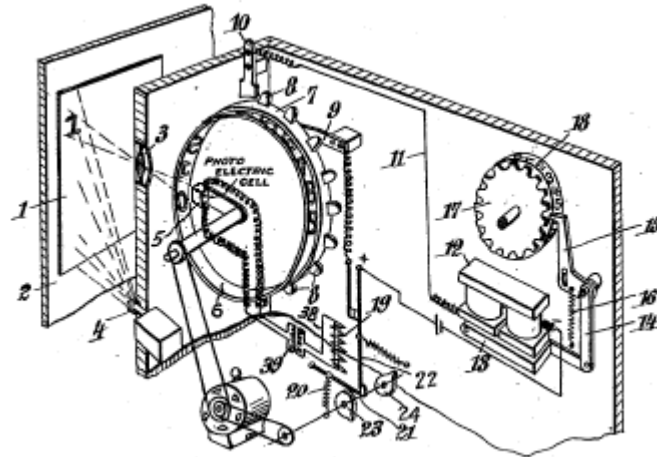


Figure 2.3: The structure of Tausheck’s “Reading Machine” (from [30]).

In this machine, a sheet of paper 1 printed with numbers (a binary image) is placed on the board 2 in such a manner that the number to be read is disposed opposite the lens 3. The number is illuminated by a strong source of light from 4. On the other side of the lens 3, a photo-electric cell 5 is arranged at a suitable distance which only reacts to the minimum illumination. A comparison wheel 6 provided with stencil-recesses corresponding to the numbers from 0 to 9 rotates in front of cell 5. During the rotation of wheel 6, rays of light will periodically pass to the cell 5 by the ways of the successive stencil-recesses of numbers. If a sheet of blank paper is placed opposite lens 3, the cell 5

is exposed to full light in the numbers shape. When a number, for instance the number “1” which is shown in the figure, does not completely fill the stencil-recess, the light will still pass to the cell 5. However, a minimum illumination will take place as soon as the stencil-recess corresponding to the number to be read is disposed exactly in front of the cell 5. Under this situation, the cell 5 will send out a signal through the relay circuit, drive the mechanical section, and rotate the manifest figure wheel 18 into proper position to display the current number.

The template matching method utilized by Tausheck’s reading machine is a fairly standard image processing technique. Although simple, template matching has some obvious limitations. One template is only capable of recognizing characters of the same size and position. On the other hand, template matching is also very vulnerable to noises and small variations that occur among characters from the same class. Meanwhile, in image processing, since the template matching needs to compare every input character to all of the templates from pixel to pixel, this method demands a lot of computation power and time.

2.2 Image Features

To overcome the vulnerabilities of template matching and reduce the computation cost, image features are employed to meet the requirements. Image features are unique characteristics that can represent a specific image. Image features are meaningful, detectable parts of the image. Meaningful means the features are associated to interesting elements via the image formation process. Two types of characteristics are usually referred by image features [34]:

1. a global property of an image, for instance the average gray level of all pixels included in a gray level image.
2. a part of the image with special properties, for example the boundary length of an image.

Sometimes the image features are not directly or obviously associated to any part of the image, but still reflect particular image properties, like the image moments. Detectable means the extraction algorithm corresponding to the feature must exist, otherwise the feature is useless. Different features are associated with different extraction algorithms that output collections of the feature descriptors. Good image features should satisfy the following conditions [26]:

1. robust to transformations – the image features should be as invariant as possible to image transformations including translation, rotation, and scaling, etc.

2. robust to noise – the image features should be robust to noises and various degraded situations.
3. feature extraction efficiency – image features can be computed efficiently.
4. feature matching efficiency – the matching algorithms should only require a reasonable computational cost.

The selection of image features and corresponding extraction methods is probably the most important step in achieving high performance for an OCR system. At the same time, the image feature and the extraction method also decide the nature and the output of the image-preprocessing step. Some image features and the extraction algorithms work on color images, while others work on gray level or binary images. Moreover, the format of the extracted features must match the requirements of the classifier [8]. Some features like the graph descriptions and grammar-based descriptions are well suited for structural and syntactic classifiers. Numerical features are ideal for statistical classifiers. Discrete features are ideal for decision trees.

2.3 The OCR Engine

An OCR engine is a system which can load an image, preprocesses the image, extracts proper image features, computes the “distances” between the extracted image features and the known feature vectors stored in the image model library, and recognizes the image according to the degree of similarity between the loaded image and the image models.

The preprocessing stage aims to make the image be suitable for different feature extraction algorithms. Some feature extraction algorithms only deal with the contours of the image while some algorithms calculate every pixel of the image. On the other hand, the initial image may be noise affected, or blurred by other reasons. The preprocessing stage which includes thresholding, binarizing, filtering, edge detection, gap filling, segmentation and so on can make the initial image more suitable for later computation.

The classification step, which corresponds to the matching stage of object recognition systems, assigns each loaded character image to one of the possible image models. The decision is made on the basis of the similarity measure. Given a shape representation scheme, the degree of similarity of any two shapes can be calculated by using a similarity measuring metric. Metric is commonly defined as follow:

A metric on a set A is a function d on $A \times A \rightarrow R$, where R is the set of real numbers, which satisfies the following conditions for all $(x, y) \in A \times A$ and $z \in A$ [20]:

(a) $d(x, y) \geq 0$

$$(b) \ d(x, y) = 0, \text{ if } x = y$$

$$(c) \ d(x, y) = d(y, x)$$

$$(d) \ d(x, y) \leq d(x, z) + d(z, y)$$

Euclidean distance is one of the most common metric distance functions and is defined as [9]:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2-1)$$

In the above definition, it is assumed that the image feature values are numerical.

2.4 Affine Transformation

Images are the projection of 3D objects onto 2D planar surfaces. The spatial status of a solid 3D object is described by six parameters, three for translations and three for rotations. Image points are affected by these six parameters, and different views of coplanar object points are related by planar projective transformations. When the distance of the coplanar object points from the camera (depth) does not vary much compared with their average depth, the planar projective transformation can be approximated by the affine transformation [25].

The affine transformation is an important class of linear 2D geometric transformation that maps variables (*e.g.* pixel intensity values located at position (x,y) in an input image) into new variables (*e.g.* (x',y') in an output image) by applying a linear combination of translation, rotation, scaling and/or shearing operations. The affine transformation preserves straight lines and parallelism. In other words, if you take a number of points that fall on a straight line from the input image and apply an affine transformation, the output image will still keep the points on a straight line. The parallel lines in the input image will still be parallel in the output image. A general affine transformation can be expressed as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2-2)$$

Since the affine transformation is linear, image invariant features are much easier to find than the case of projective transformations. Examples of affine transformations are shown in Figure 2.4. Note that straight lines are still straight. Parallel lines remain parallel under affine transformations.

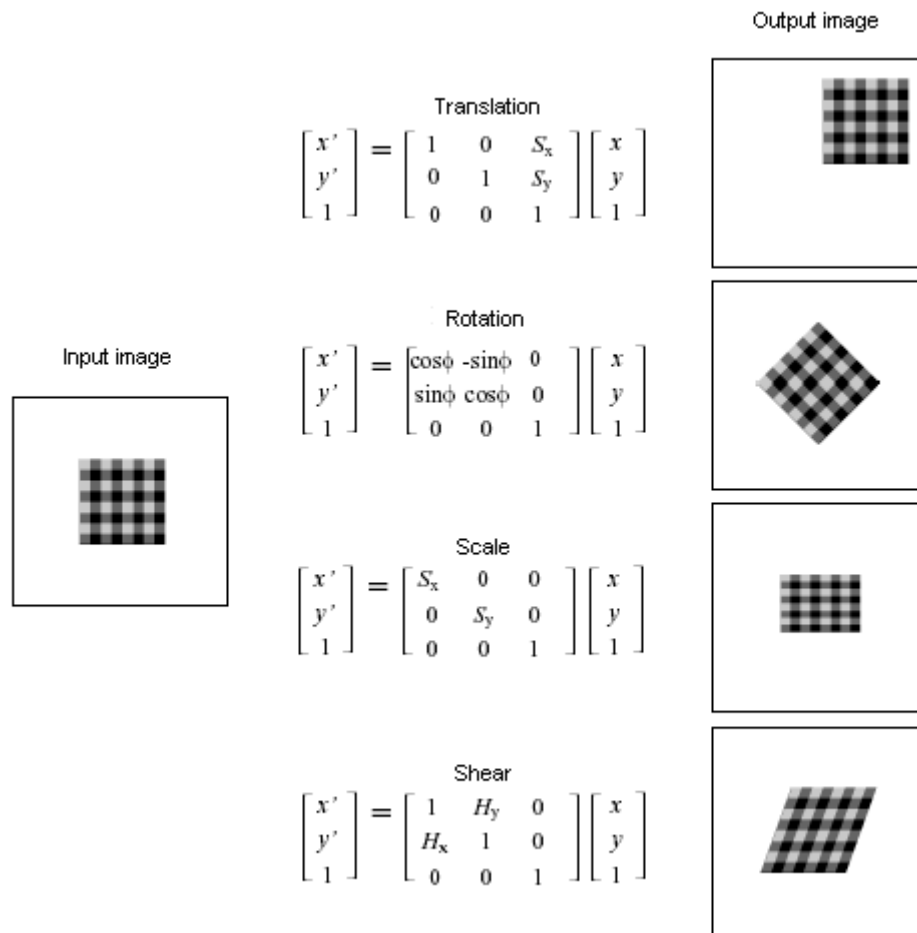


Figure 2.4: Examples of affine transformations.

2.5 Image Resolution

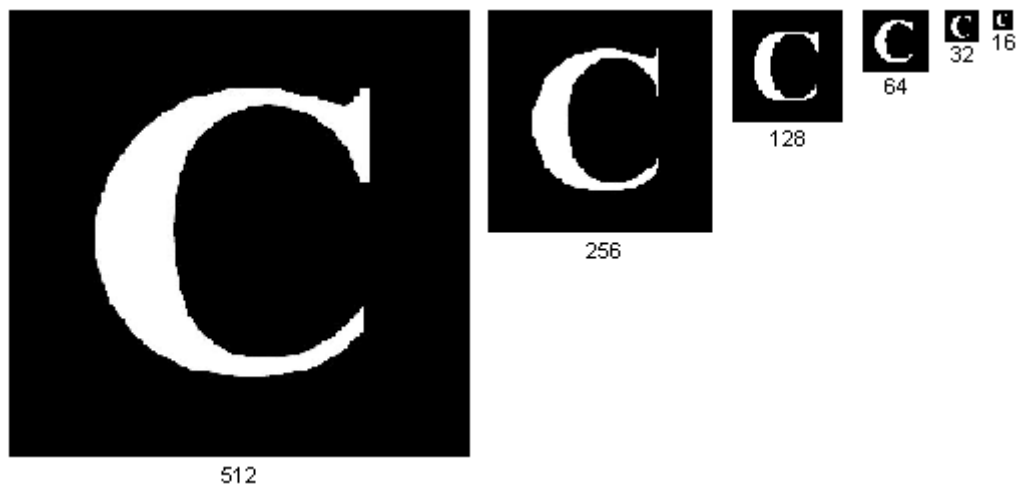


Figure 2.5: A 512×512 image sub-sampled down to 16×16.

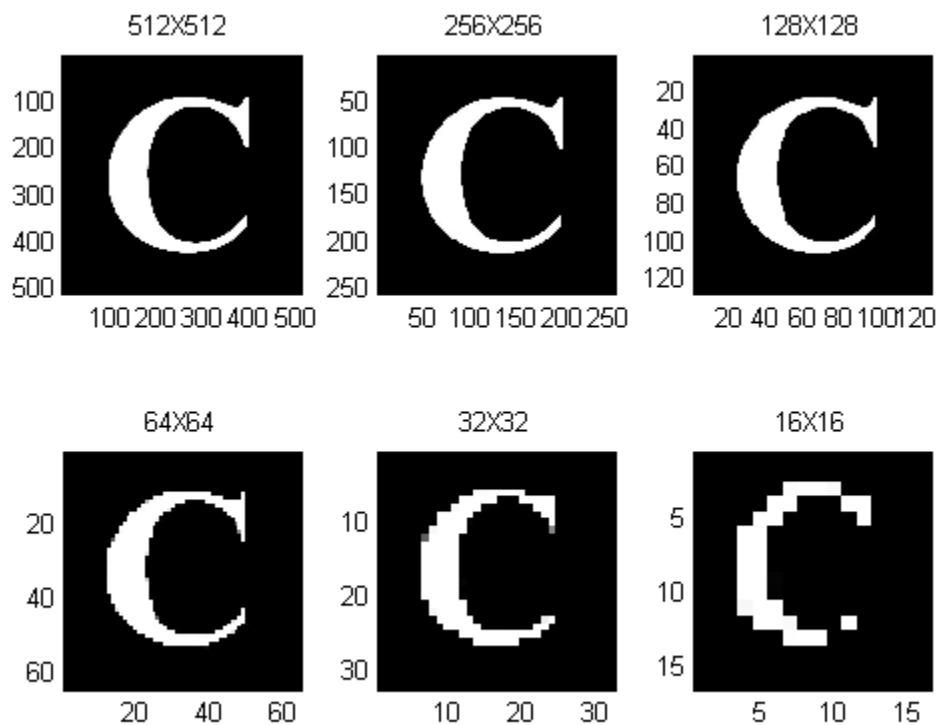


Figure 2.6: Results of stretching the sub-samples to the same size of the original 512×512 image.

Digital images are made up of small squares called pixels. Image quality is based on image resolutions [10]. There are two types of image resolutions: spatial and output.

Spatial resolution is defined in terms of width and height. An image of 640 pixels wide and 480 pixels high has a spatial resolution of 640×480 and contains 307,200 pixels. The higher the spatial resolution, the more pixels are available to create the image. This makes more detail possible and creates a sharper image. If we sub-sample this image by deleting every other row and column from the original 512×512 image, we can get an image of 256×256 . The 128×128 , 64×64 , 32×32 images were generated by the same way. Figure 2.5 shows the changes of the image dimension, but it make it difficult to see the effects resulting from the reduction of samples. If we stretch all of the images to the same size as the original 512×512 image, we can see the result of jagged edges and chessboard pattern when the spatial resolution goes down to 16×16 as Figure 2.6 shows.

Output resolution is defined in terms of the number of dots/pixels per inch, or dpi. The idea is to match the resolution of an image to the resolution of the output device, usually a computer screen or printed document. The higher the dpi of the image (up to the limit of the output device), the better the quality of the printed image. Typical ink-jet and laser printers can have output resolutions of 1200 dpi. Typical monitor resolution is 96 dpi.

From now on, the word “resolution” used in this thesis will only mean the spatial resolution of the image.

Chapter 3

Image Invariants

3.1 Introduction

An important problem in OCR is the automatic recognition of a character in a scene regardless of its position, size, orientation, etc. In order to recognize different variations of the same character, image features which are invariant to certain transformations need to be used. Image invariants are features which have approximately the same values for samples of the same image which are, for instance, translated, scaled, rotated, skewed, blurred, or noise affected [25], [33].

Shape-based image invariants are the most commonly used image features. Image recognition based on these invariants includes three major issues: shape representation, shape similarity measure and shape indexing [40]. Among these issues, shape representation is the most important issue. Various shape representation methods and shape descriptors exist in literatures. These methods can be classified into two categories: boundary-based invariants and region-based invariants.

In boundary-based invariants, only the contour information of the shape is explored. The common boundary-based invariants include chain code and Fourier descriptors. In region-based invariants, all the pixels within a shape are taken into account to obtain the mathematical representation. The most popular region-based methods include various

moment-based invariants such as Hu's seven moment invariants, Zernike moments, complex moments, etc.

The fundamental element of all these schemes is definition of a set of mathematical functions for image representation and data reduction. Usually additional transformations are required to achieve the desired invariant properties for the image features [16].

This chapter will focus on the theoretical derivation of these shape-based image invariants along with the evaluation of their advantages and disadvantages.

3.2 Boundary-Based Invariants

Boundary is one of the most straightforward and important image features as human beings tend to perceive scenes as being composed of different individual objects, which can be best identified by their boundaries. Meanwhile, as far as the implementation is concerned, boundary is also very simple to calculate.

3.2.1 Chain Code

Chain code was introduced by Freeman [7] as a means to represent lines or boundaries of shapes by a connected sequence of straight line segments of specified length and direction. A chain code has two components: the coordinates of the starting point and a chain of codes representing the relative position of the starting pixel and its followers. The chain code is generated by using the changing direction of the connected pixels contained in a boundary. The representation is based on 4- connectivity or 8- connectivity of the segments. The direction of each line segment can be coded by using a numbering scheme like the ones shown in Figure 3.1. Figure 3.2 shows an example of 8- connectivity chain code.

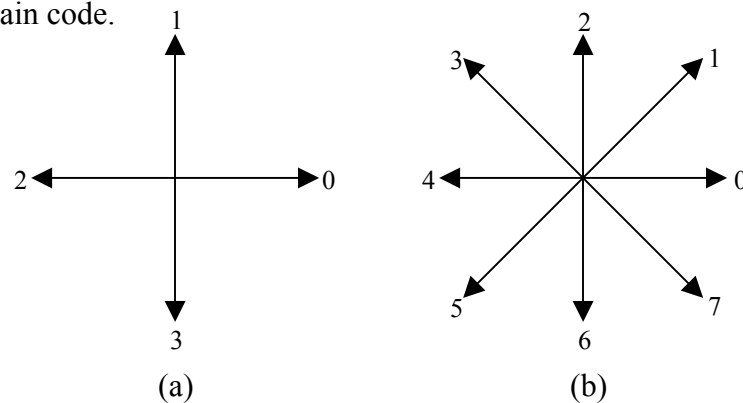
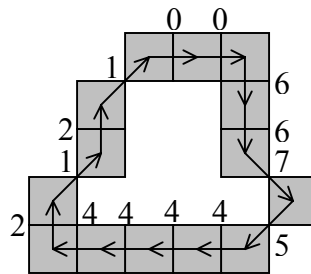


Figure 3.1: Chain code numbering schemes: (a) Direction for 4-connectivity chain code. (b) Direction for 8-connectivity chain code.

For a closed boundary, its chain code obviously depends on the starting point of the boundary. To make it invariant to the starting point, the chain code can be normalized according to the following method [10]: A chain code can be treated as a circular sequence of direction numbers. Therefore, the starting point can be redefined so that the resulting sequence of numbers forms a minimum integer (refer to the example shown by Figure 3.2).



Chain Code:	21210066754444
Normalized Chain Code:	00667544442121
First Difference:	70601670006717

Figure 3.2: An example of the 8-connectivity chain code.

The chain code can also be normalized for rotation by using the *first difference* instead of the code itself [10]. The *first difference* is generated by counting the number of direction changes (counterclockwise) of two adjacent pixels. The first code can be computed by counting the direction changes between the end point and the beginning point of the chain (refer to Figure 3.2). The *first difference* will keep the chain code invariant to rotation only if the boundary itself is invariant to rotation change.

3.2.2 Fourier Descriptors

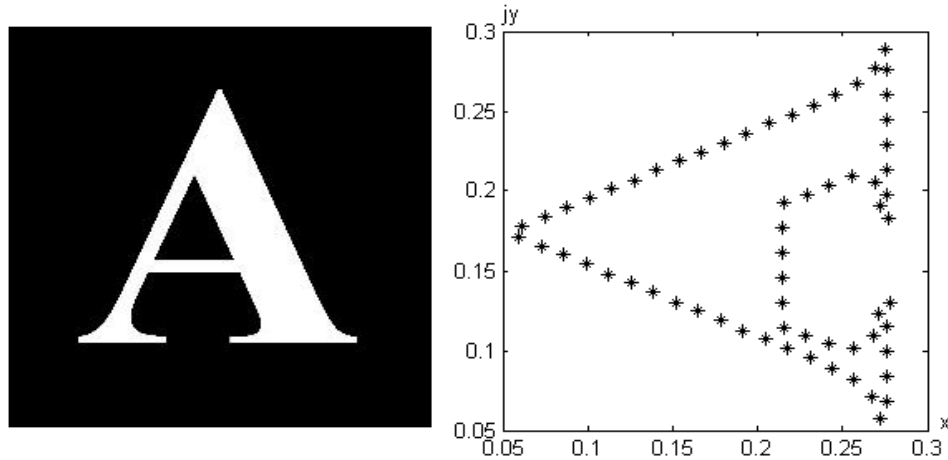


Figure 3.3: A re-sampled boundary and its representation as a complex sequence.

Fourier descriptors were introduced by Zahn and Roskies [37], Persoon and Fu [22], Wallace [35], Arbeter *et al* [3] to describe the shape of a closed, planar figure. Given a close shape in a 2D Cartesian plane, the boundary s can be traced and re-sampled according to, say, the counterclockwise direction, to obtain an uniformly distributed K points as shown in Figure 3.3. Each point's coordinates can be expressed as (x_0, y_0) , (x_1, y_1) , $(x_2, y_2), \dots, (x_{K-1}, x_{K-1})$. These coordinates can be expressed in the form $x(k) = x_k$ and $y(k) = y_k$. Under this condition, the boundary can be expressed as a sequence of complex numbers [10]:

$$s(k) = x(k) + j y(k), \quad k = 0, 1, 2, \dots, K-1 \quad (3-1)$$

That means the x -axis is treated as the real axis and the y -axis as the imaginary axis of a sequence of complex numbers. The coefficients of Discrete Fourier Transform (DFT) of this complex sequence $z(u)$ is:

$$z(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K}, \quad u = 0, 1, 2, \dots, K-1 \quad (3-2)$$

The complex coefficients $z(u)$ are called the Fourier descriptors of the boundary.

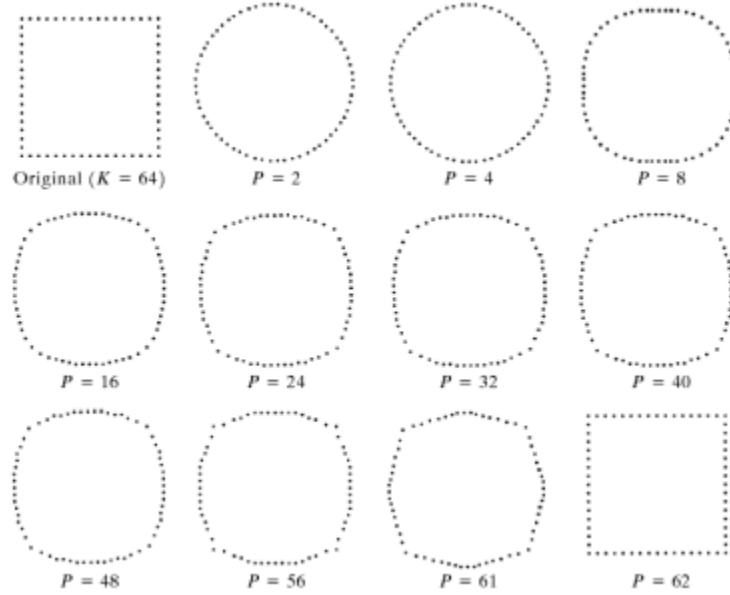


Figure 3.4: Examples of reconstruction from Fourier descriptors. P is the number of Fourier coefficients used in the reconstruction process (adapted from [10]).

According to the property of the Fourier transform, low-frequency components determine global shape, and high-frequency components determine the fine details [10]. We can reconstruct the boundary using the first P coefficients with the inverse Fourier transform:

$$s'(k) = \frac{1}{K} \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/K}, \quad k = 0, 1, 2, \dots, K-1 \quad (3-3)$$

Although only P coefficients are used to obtain the boundary $s(k)$, k still ranges from 0 to $K-1$. In other words, the same number of points is in the approximated boundary, but

not as many terms are used in the reconstruction of each point. Figure 3.4 shows the example of the reconstruction results of a square boundary with different number of coefficients from 2 to 62. From this example we see that when higher-order coefficients used, the reconstructed contour is more like the original shape.

The Fourier descriptors have the advantage of reducing a 2-D problem to a 1-D problem. The significant Fourier descriptors can be used as the basis for classifying different boundary shapes.

As stated before, we want the image invariants to be as insensitive as possible to scale changes, translation and rotation. Fourier Descriptors are not directly insensitive to these image transformations, but the changes in these parameters can be related to simple operations on the Fourier descriptors as follows:

Transformation	Boundary	Fourier Descriptor
Identity	$s(k)$	$z(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$z_r(u) = z(u)e^{j\theta}$
Translation	$s_t(k) = s(k)\Delta_{xy}$	$z_t(u) = z(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$z_s(u) = \alpha z(u)$
Starting point	$s_p(k) = s(k-k_0)$	$z_p(u) = z(u)e^{-j2\pi u k_0/K}$

Table 3.1: Basic properties of Fourier descriptors under some transformations (from [10]).

The symbol $\Delta_{xy} = \Delta x + j\Delta y$. $\delta(u)$ is an impulse function which only has non-zero values at the origin and zero values everywhere else.

From the table, we can see that the magnitudes of $z(u)$ are invariant to image rotations because:

$$|z_r(u)| = |z(u)e^{j\theta}| = |z(u)| \cdot |e^{j\theta}| = |z(u)| \cdot 1 = |z(u)|$$

The image translation consists of adding a consistent displacement to all sample coordinates of the boundary. The translation has no effect on the descriptors except for $u=0$ because of the impulse function $\delta(u)$. The first component of the Fourier descriptors depends only on the position of the shape, it is not useful in describing the shape and can be discarded.

The invariance to scale changes can be achieved by forming the ratio of two coefficients so that we can get rid of the parameter α .

The magnitudes of $z(u)$ are invariant to the change of starting point because:

$$|z_p(u)| = |z(u)e^{-j2\pi k_0 u}| = |z(u)| \cdot |e^{-j2\pi k_0 u}| = |z(u)| \cdot 1 = |z(u)|$$

By summarizing above analysis, we can use:

$$c(u-2) = \frac{|z(u)|}{|z(1)|}, \quad u = 2, 3, \dots, K-1 \quad (3-4)$$

to keep the Fourier descriptors invariant to rotation, translation and scaling at the same time. In the OCR engine developed by ourselves, we adopted the first 10 elements of the $c(u-2)$ series to compose the feature vectors for the image models.

3.3 Region-Based Invariants

Boundary-based invariants such as chain code and Fourier descriptors explore only the contour information; they cannot capture the interior content of the shape. On the other hand, these methods cannot deal with disjoint shapes where single closed boundary may not be available; therefore, they have limited applications. For region-based invariants, all of the pixels of the image are taken into account to represent the shape. Because region-based invariants combine information of an entire image region rather than exploiting information just along the boundary pixels, they can capture more information from the image. The region-based invariants can also be employed to describe disjoint shapes.

Moment-based invariants are the most common region-based image invariants which have been used as pattern features in many applications [2], [5], [12], [24]. Hu first introduced a set of moment-based invariants using nonlinear combinations of regular moments in 1961 [12]. Hu's invariants have the desirable properties of being invariant under image translation, scaling, and rotation. However, it is found that to compute the higher order of Hu's moment invariants is quite complex, and to reconstruct the image from Hu's invariants is also very difficult. To resolve these problems, Teague suggested the concept of orthogonal moments to recover the image from moment invariants based on the theory of orthogonal polynomials [31]. Teague introduced Zernike moments that allow independent moment invariants to be constructed easily to an arbitrarily high order. Pseudo-Zernike moments are another form of orthogonal moments proposed by Teh and Chin [32] which are less sensitive to image noise than the conventional Zernike moments.

Complex moments introduced by Abu-Mostafa and Psaltis [2] are more simple and straightforward in deriving moment invariants.

Moment-based invariants explore information across an entire image rather than providing information just at single boundary points, they can capture some of the global properties missing from the pure boundary-based representations like the overall image orientation. In this section, we will introduce regular moments, Hu's seven moment invariants, Zernike moments, Pseudo-Zernike moments and complex moments.

3.3.1 Regular Moments and Central Moments

Regular moments (also be referred to as geometric moments) are defined as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy, \quad p, q = 0, 1, 2, \dots \quad (3-5)$$

where m_{pq} is the $(p + q)$ th order moment of the continuous image function $f(x, y)$.

The central moments of $f(x, y)$ are defined as:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy, \quad p, q = 0, 1, 2, \dots \quad (3-6)$$

where $\bar{x} = m_{10} / m_{00}$ and $\bar{y} = m_{01} / m_{00}$, which are the centroid of the image.

For digital images the integrals are replaced by summations and m_{pq} becomes:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y), \quad p, q = 0, 1, 2, \dots \quad (3-7)$$

Then the central moments are changed to:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y), \quad p, q = 0, 1, 2, \dots \quad (3-8)$$

The central moments are computed using the centroid of the image, which is equivalent to the regular moments of an image whose center has been shifted to coincide with its centroid; therefore the central moments are invariant to image translations.

In affine transformations, scale changes are caused by:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

To obtain scale invariance, we let $f'(x', y')$ represent the image $f(x, y)$ after scaling the image by $S_x = S_y = \alpha$, so $f'(x', y') = f(\alpha x, \alpha y) = f(x, y)$, and $x' = \alpha x$, $y' = \alpha y$, then we have:

$$\begin{aligned} m'_{pq} &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x'^p y'^q f'(x', y') dx' dy' \\ &= \alpha^{p+q+2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \\ &= \alpha^{p+q+2} m_{pq} \end{aligned} \quad (3-9)$$

Similarly,

$$\mu'_{pq} = \alpha^{p+q+2} \mu_{pq}, \quad \mu'_{00} = \alpha^2 \mu_{00}$$

We can define normalized central moments as:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \quad \gamma = (p+q+2)/2, \quad p+q = 2, 3, \dots \quad (3-10)$$

η_{pq} is invariant to changes of scale because:

$$\eta'_{pq} = \frac{\mu'_{pq}}{\mu'^{\gamma}_{00}} = \frac{\alpha^{p+q+2} \mu_{pq}}{\alpha^{2\gamma} \mu_{00}^{\gamma}} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} = \eta_{pq}$$

In summarize, the normalized central moments are invariant to image translation and scaling.

3.3.2 Hu's Seven Moment Invariants

Based on normalized central moments, Hu introduced seven nonlinear functions which are translation, scale, and rotation invariant. The seven moment invariants are defined as [12]:

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Hu's seven moment invariants are invariant to image transformations including scaling, translation and rotation. However, this set of moment invariants is not invariant to contrast changes.

Hu's seven moment invariants have been widely used in pattern recognition, and their performance has been evaluated under various deformation situation including blurring [28], spatial degradations [6], random noise [11], [19], skew and perspective transformations [27].

As Hu's seven moment invariants take every image pixel into account, the computation cost will be much higher than boundary-based invariants. As stated before, image's spatial resolution decides the total amount of pixels, and to reduce the computation cost, the evaluation of the performance of Hu's moment invariants under different image spatial resolutions will be a major topic in the thesis.

3.3.3 Zernike Moments

Zernike introduced a set of complex polynomials $\{V_{nm}(x, y)\}$ which form a complete orthogonal set over the unit disk of $x^2 + y^2 \leq 1$ in polar coordinates [38]. The form of the polynomials is:

$$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho)e^{im\theta} \quad (3-11)$$

where n is positive integer or zero; m is integers subject to constraints $n - |m|$ is even, and $|m| \leq n$; ρ is the length of the vector from the origin to the pixel (x, y) ; θ is the angle

between the vector ρ and x axis in counterclockwise direction; $R_{nm}(\rho)$ is Radial polynomial defined as:

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s} \quad (3-12)$$

The Zernike moment of order n with repetition m for function $f(x, y)$ is defined as:

$$A_{nm} = \frac{n+1}{\pi} \iint_{x^2+y^2 \leq 1} f(x, y) V_{nm}^*(x, y) dx dy \quad (3-13)$$

where $V_{nm}^*(x, y) = V_{n,-m}(x, y)$.

To compute the Zernike moment of a digital image, we just need to change the integrals with summations:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x, y) V_{nm}^*(x, y), \quad x^2+y^2 \leq 1 \quad (3-14)$$

The defined features of Zernike moments themselves are only invariant to rotation. To achieve scale and translation invariance, the image needs to be normalized first by using the regular Zernike moments.

The translation invariance is achieved by translating the original image $f(x, y)$ to $f(x + \bar{x}, y + \bar{y})$, where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$. In other words, the original image's center is moved to the centroid before the Zernike moment's calculation.

Scale invariance is achieved by enlarging or reducing each shape so that the image's 0th regular moment m'_{00} equals to a predetermined value β . For a binary image, m_{00} equals to the total number of shape pixels in the image. According to equation (3-9), for a scaled image $f(\alpha x, \alpha y)$, its regular moments $m'_{pq} = \alpha^{p+q+2} m_{pq}$, m_{pq} is the regular moments of $f(x, y)$. Since the objective is to make $m'_{00} = \beta$, we can let $\alpha = \sqrt{\beta / m_{00}}$. By substituting $\alpha = \sqrt{\beta / m_{00}}$ into m'_{00} , we can obtain $m'_{00} = \alpha^2 m_{00} = \beta$.

Suppose we know all Zernike moments A_{nm} of $f(x, y)$ up to order N . Due to the orthogonal property of Zernike moments, we can reconstruct the image based on this set of Zernike moments by:

$$f'(x, y) = \sum_{n=0}^N \sum_m A_{nm} V_{nm}(x, y) \quad (3-15)$$

3.3.4 Pseudo-Zernike Moments

Zernike moments, which are invariant to the rotations, are polynomials in x and y . A related orthogonal set of polynomials in x, y and ρ derived by Bhatia and Wolf [4] has the properties similar to Zernike moments. This set of polynomials which are named Pseudo-Zernike moments by Teh and Chin [32] differs from that of Zernike in that the real-valued radial polynomials are defined as:

$$R'_{nm}(\rho) = \sum_{s=0}^{n-|m|} (-1)^s \frac{(2n+1-s)!}{s!(n-|m|-s)!(n+|m|+1-s)!} \rho^{n-s} \quad (3-16)$$

The Pseudo-Zernike moments are then defined by:

$$A'_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x, y) R'_{nm}(x, y) e^{-im\theta} \quad (3-17)$$

where n is positive integer or zero; m is an integers subject to constraints $|m| \leq n$; ρ is the length of the vector from the origin to the pixel (x, y) ; θ is the angle between the vector ρ and x axis in counterclockwise direction.

Note that the condition of Zernike moments “ $n-|m|$ is even” has been removed. Therefore, for a given order n , the set of Pseudo-Zernike moments contains approximately twice as many as the polynomials of conventional Zernike moments. Due to this, the reconstructed image with Psudo-Zernike moments will catch more details than the conventional Zernike moments of the same order. Teh and Chin also proved that Pseudo-Zernike moments are less sensitive to image noise than are the conventional Zernike moments [32].

3.3.5 Complex Moments

The complex moments of order $(p+q)$ are defined as [1]:

$$C_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x+iy)^p (x-iy)^q f(x, y) dx dy, \quad p, q = 0, 1, 2, \dots, \infty \quad (3-18)$$

If we map the complex moments to polar coordinates with $x+iy = re^{i\theta}$, (3-18)

becomes:

$$\begin{aligned}
C_{pq} &= \int_0^\infty \int_0^{2\pi} r^{p+q+1} e^{i(p-q)\theta} f(r, \theta) dr d\theta \\
&= \int_0^\infty r^{p+q+1} \left\{ \int_0^{2\pi} f(r, \theta) e^{ik\theta} d\theta \right\} dr, \quad k = p - q \\
&= \int_0^\infty r^{p+q+1} h_k(r) dr
\end{aligned} \tag{3-19}$$

where

$$h_k(r) = \int_0^{2\pi} f(r, \theta) e^{ik\theta} d\theta, \quad k \text{ integer} \tag{3-20}$$

$h_k(r)$ is the circular Fourier transform of the image $f(r, \theta)$ represented in polar coordinates.

From (3-19) we see that $|c_{pq}|$ is invariant to rotations of $f(r, \theta)$. Another property is that the complex moment of order $(p+q)$ is a linear combination with complex coefficients of the regular moments m_{pq} satisfying $r + s = p + q$:

$$C_{pq} = \sum_{r=0}^p \sum_{s=0}^q \binom{p}{r} \binom{q}{s} i^{p+q-(r+s)} \cdot (-1)^{q-s} m_{r+s, p+q-(r+s)} \tag{3-21}$$

Reiss [25] also proved that Hu's moment invariants can be easily derived from complex moments.

Chapter 4

Experiment Implementations

4.1 Introduction

Image invariants based on boundaries and regions, which have been described in the previous chapter, show the invariance property under different affine transformations including translation, rotation and scale changes. To make meaningful performance evaluation for the different image invariants under different transformations and compare their advantages and disadvantages, we implemented and tested several image invariants on a common set of image samples. The implementation process of the experiments will be presented and discussed in this chapter.

Because the computation costs of most region-based image invariants are mainly depend on the total number of pixels contained in the image, we will put more emphasis on the research of the their sensitivity to different image spatial resolutions in the experiments.

It has been proved that the image invariants possess various invariance properties under noise-free condition. However, in the presence of noise, the computed image invariants are expected not to be strictly invariant. It is also of interest to study how robust the invariants are against the influence of the noise. In this chapter, we will

investigate the performance of Fourier descriptors and Hu's seven moment invariants under the influence of salt and pepper noises of different densities.

To investigate the overall performance of Fourier descriptors and Hu's seven moment invariants, an OCR engine is implemented with MATLAB scripts. With this OCR engine, users can collect image features based on Fourier descriptors and Hu's moment invariants, and save them into the model library in the format of a MATLAB data file, which can be loaded later for image recognition.

4.2 Image Samples

The image samples we used in the experiment are gray level images of 26 capital English letters from “A” to “Z”, which only have a single contour and thus meet the requirement of Fourier descriptors. For the image with multiple contours like the small English letter “i”, we need to obtain its single boundary first to calculate its Fourier descriptors. One method to get its single boundary is to dilate the image so that we can connect the two separated parts together. By this way, we can get its approximated single contour. Figure 4.1 illustrated the dilation and approximation process.

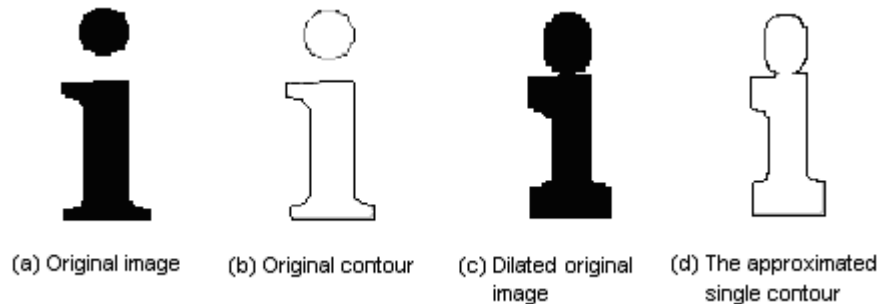


Figure 4.1: The approximation process of images with multiple contours.

As more emphasis is put on the research of the image invariants’ sensitivity to different spatial resolutions and noises, we try to simplify the image preprocessing complexity. Usually the preprocessing step includes thresholding, segmentation, and boundary approximation.

The spatial resolutions of the image samples used in the experiment are at 16×16 , 32×32 , 64×64 , 128×128 , 256×256 and 512×512 . Figure 4.2 shows a set of image samples of the capital letter “F”.

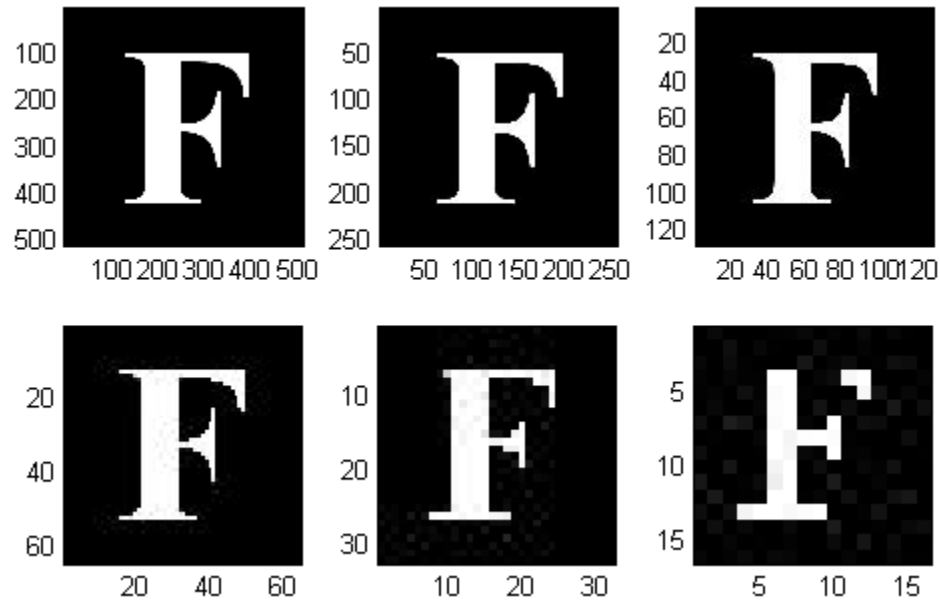


Figure 4.2: A set of image samples from spatial resolution 512×512 to 16×16 .

Noise is the errors occurred during the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. Digital images are prone to be corrupted by a variety of types of noises. Common types of noises include salt and pepper noise and Gaussian noise. Salt and pepper noise contains randomly distributed occurrences of both black and white impulses. Gaussian noise contains intensity variations that are drawn from a Gaussian distribution and is a very good model for many kinds of sensor noises caused by camera electronics errors [14] (see examples in Figure 4.3).

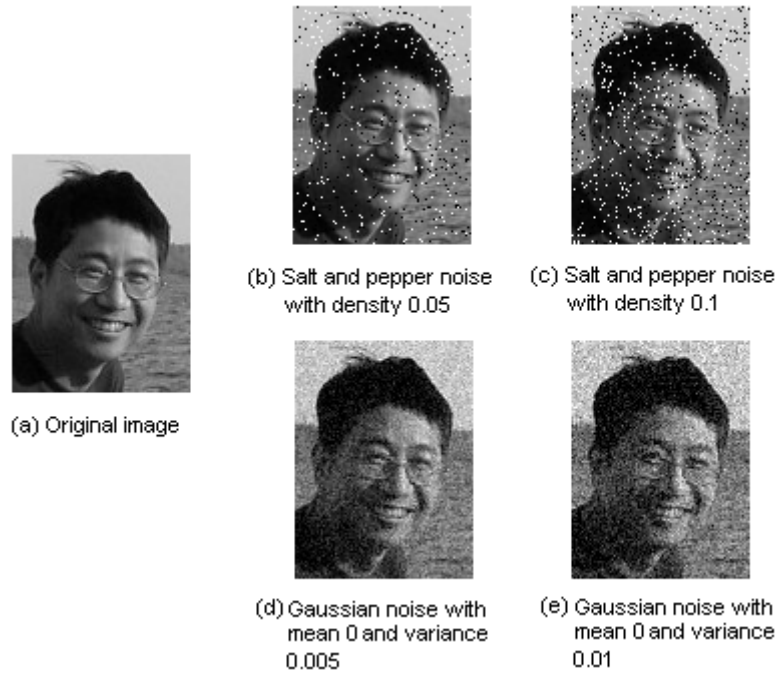


Figure 4.3: Images corrupted by salt and pepper noise and Gaussian noise.

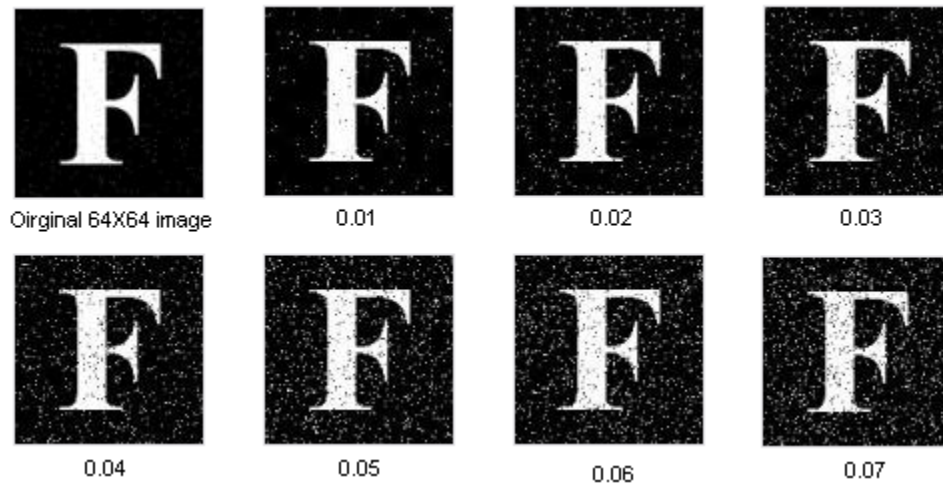


Figure 4.4: Images corrupted by salt and pepper noise with different densities.

Because all of the images are first converted to binary images in our experiment, we will only use the image samples corrupted by salt and pepper noises of different densities

from 0.01 to 0.07. Figure 4.4 gives a set of salt and pepper noise corrupted image samples used in the experiment.

All together, the image samples used in the experiment includes 156 clean gray level images with spatial resolutions range from 16×16 to 512×512 and 1092 gray level images corrupted by salt and pepper noise with intensity from 0.01 to 0.07 of the same spatial resolution range.

4.3 Implementation Descriptions

For boundary-based image invariants, we implemented the algorithms of Fourier descriptors. For region-based image invariants, we implemented regular moments, normalized moments, Hu's seven image invariants and Zernike moments. The implementation process of these algorithms will be described in detail in this section.

4.3.1 Fourier Descriptors

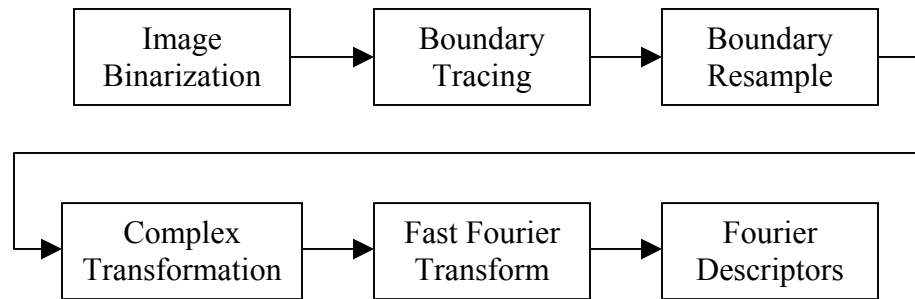


Figure 4.5: Block diagram of computing Fourier descriptors.

The block diagram of computing Fourier descriptors is shown in Figure 4.5. At the beginning of the implementation, all of the gray level images are first converted to binary images by thresholding according to the image's histogram. For the binarized image, the pixel values are either "1" for white or "0" for black.

Boundary tracing returns the X and Y coordinates for all pixels along the boundary of the capital English letter. The tracing process starts from the most left-up pixel of the image and traces down in counterclockwise direction. All of the coordinates are saved into a $2 \times n$ matrix. n is the total number of the boundary pixels. The first row of the matrix stores the X coordinates and the second row stores the Y coordinates.

To calculate Fourier transforms of the complex series composed by the X and Y coordinates of the boundary pixels, it is necessary to uniformly resample the image contour pixels at a number of power of 2 [24] [35]. The boundary resample step enumerates all of the boundary pixels, and re-samples them according to the number defined by the users. In our experiment, the number is set to 64, and the re-sampled pixels' coordinates are saved into a 2×64 matrix.

The rest steps convert the coordinates of the pixels to a series of complex numbers according to the format of $contour = x*j + y$, and calculate the Fourier descriptors of the series according to the method described in Section 3.2.2.

4.3.2 Hu's Seven Moment Invariants

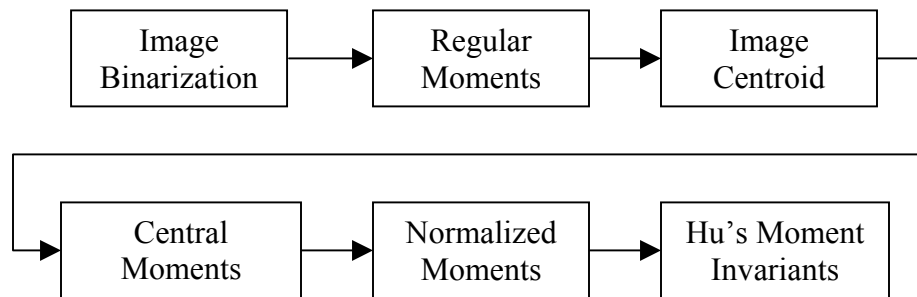


Figure 4.6: Block diagram of computing Hu's seven moment invariants.

Figure 4.6 shows the computing process of Hu's seven moment invariants. As same as Fourier descriptors, the image is first converted to binary format. The function to compute the regular moments is in the format: $[m] = moment(fig, p, q)$. fig is the input binary image, and p, q are predefined moment's order. With these parameters available, we do the summation according to the regular moments' definition

$m_{pq} = \sum_x \sum_y x^p y^q f(x, y)$ by going through all of the pixels in the image. For a binary

image, m_{00} is the total number of white pixels of the image.

The centroid of the binary image is computed according to $\bar{x} = m_{10} / m_{00}$ and $\bar{y} = m_{01} / m_{00}$. Based on the centroid of the image, similar to the regular moments, the central moments function is in the format: $[\mu] = \text{central_moment}(fig, p, q)$. This is computed according to the definition: $\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$.

Once all of above results are available, the Hu's seven moment invariants can be calculated according to the equations given by Section 3.3.2.

4.3.3 Zernike Moments

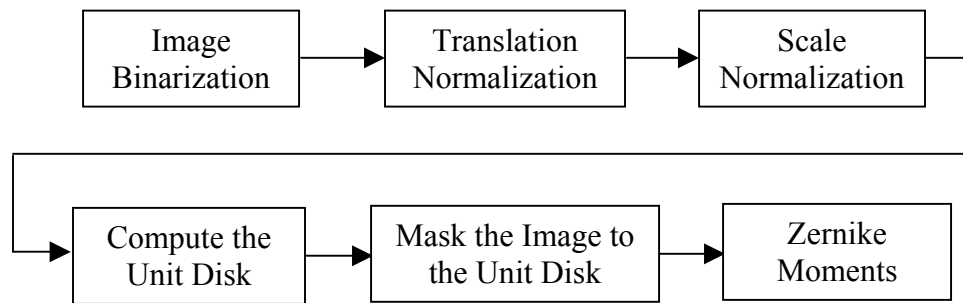


Figure 4.7: Block diagram of computing Zernike moments.

Figure 4.7 is the block diagram of Zernike moment's computation. Compared with Hu's seven moment invariants, the computation of Zernike moments is more complicated. The major reason for this is the image normalization process. In Hu's moment invariants, the whole concept is based on the central moments which have

integrated the translation and scale normalization in the definitions. The Zernike moments, however, are only invariant to image rotation for themselves. To achieve translation and scale invariance, extra normalization processes are required.

The translation normalization is achieved by moving the image center to the image centroid. Figure 4.8 compared the translation-normalized image with the original image.

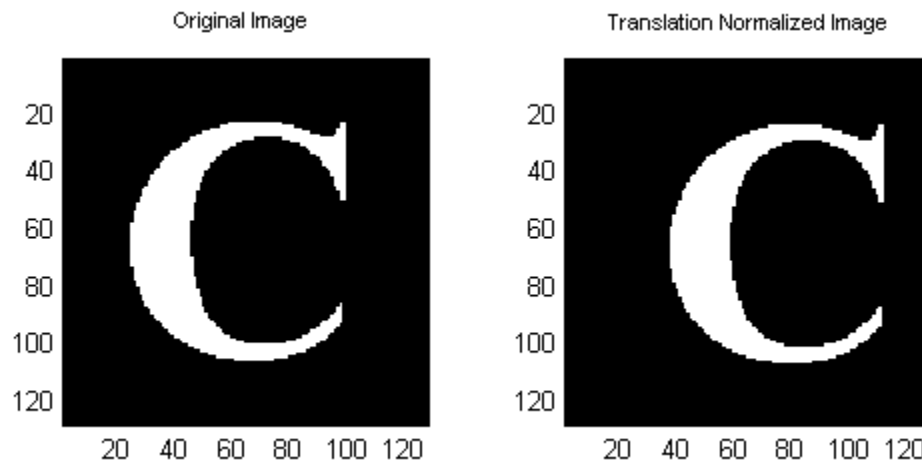


Figure 4.8: Comparison between the original image and the translation-normalized image.

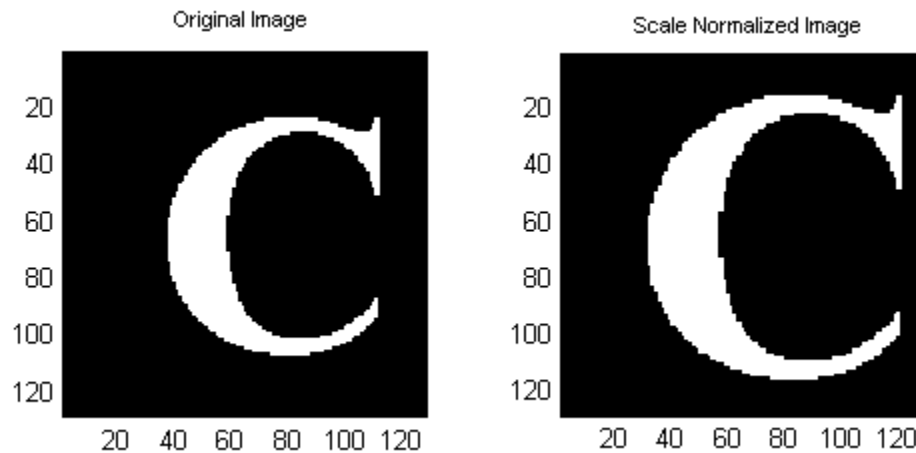


Figure 4.9: Comparison between the original image and the scale-normalized image.

The scale normalization is achieved by set the image's 0th order regular moment m_{00} to a predetermined value. Figure 4.9 compared the original image and the scale-normalized image. Because m_{00} is the total number of white pixels for binary image, we use interpolation to set m_{00} to the predetermined value.

Different from the regular moments which employ the summation within a square range of pixels, Zernike polynomials take the unit disk $x^2+y^2 \leq 1$ as their computation domain. To compute the Zernike moments of a digital image, the range of the image should be mapped to the unit circle first with its origin at the image's center. The pixels falling outside the unit circle are discarded in the computation process. In our implementation of Zernike moments, we use binary images with spatial resolution of 64×64 . All of these binary images are normalized into a unit circle with fixed radius of 32 pixels. Figure 4.10 shows the process.

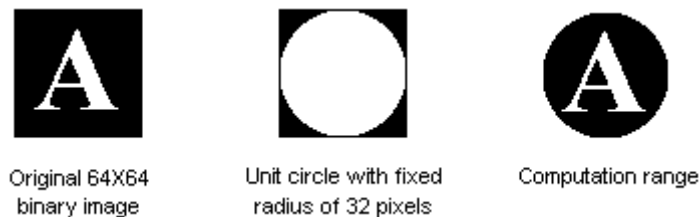


Figure 4.10: The computation process of the unit disk mapping for Zernike moments.

Zernike moments themselves are not invariant to translation and scale change. The images need to be preprocessed first to achieve the property of translation and scaling invariance. The advantage of Zernike moments lays in the image reconstruction priority. For this reason, we focus our research on their reconstruction ability rather than their sensitivity property to different spatial resolutions.

4.4 The OCR Engine

To test the overall performance of Fourier descriptors and Hu's seven moment invariants for the image samples of different transformations, spatial resolutions as well as and noises influences, we implemented an OCR engine with MATLAB scripts. Figure 4.11 is the screenshot of the OCR engine.

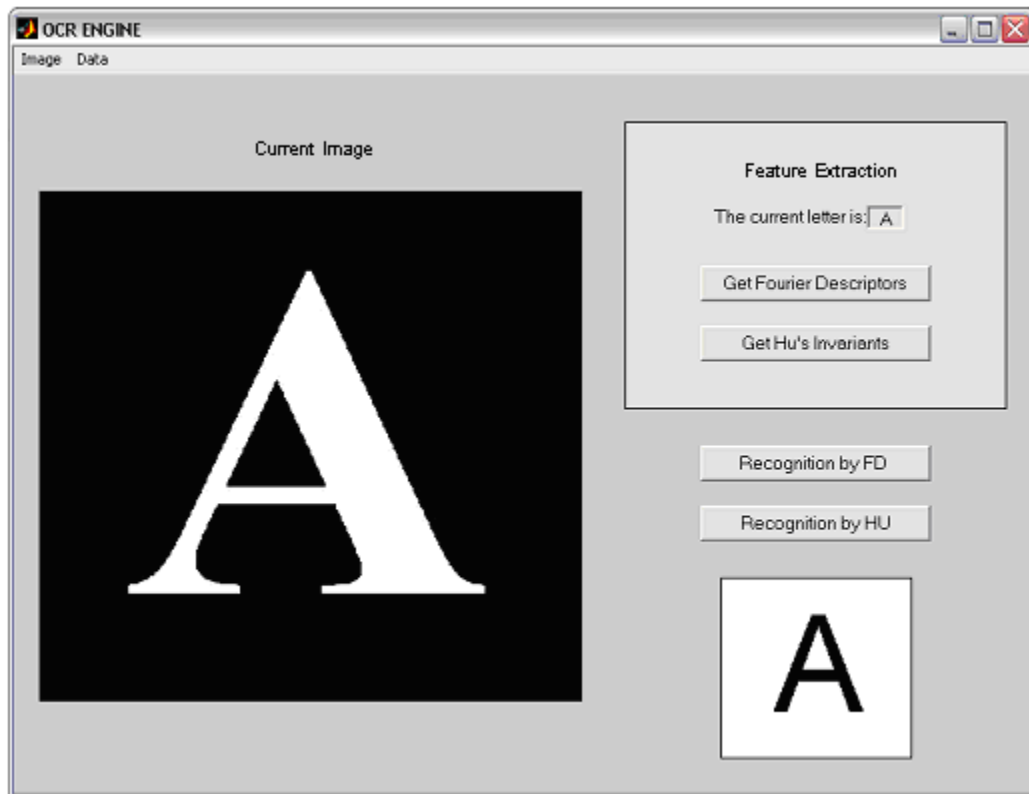


Figure 4.11: The screenshot of the OCR engine.

This OCR engine includes two basic functions: feature extraction and image recognition. Figure 4.12 illustrates the working process of the OCR engine.

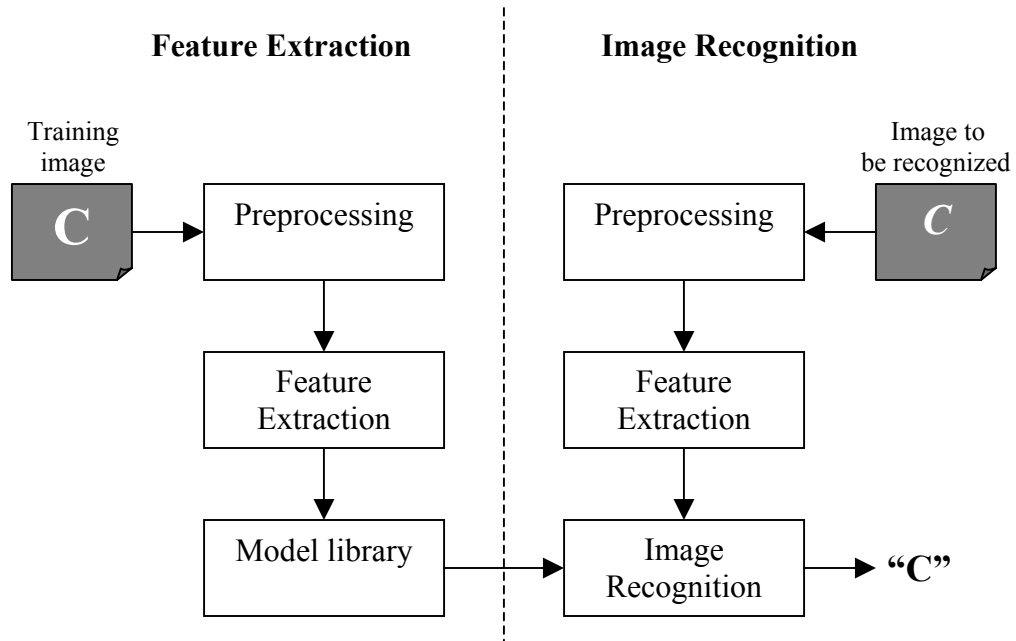


Figure 4.12: The working process of the OCR engine.

Images are loaded into the OCR engine by clicking the “Image” menu and selecting the “Open” item. Both “TIFF” and “JPEG” image formats are supported by this OCR engine.

The feature extraction is basically a training process. After a training image is loaded, the user needs to input the capital English letter corresponding to the image. By clicking the “Get Fourier Descriptors” or “Get Hu’s Invariants” buttons, the Fourier descriptors or Hu’s seven moment invariants of this image can be computed and collected correspondingly. The computed results will be saved to a model library as a MATLAB “.mat” file by clicking the “Data” menu and selecting the “Save” item. The data is in the format of feature vectors which can be indexed for each trained image.

At the image recognition step, the user loads the model library first by clicking the “Data” menu and load the model library first. Then the user can open an image and implement the recognition. By clicking the recognition button, the OCR engine can extract the features of the current image and compute the Euclidean distance by indexing the feature vectors stored in the model library to make the classification.. The OCR engine will show the classification result according to the minimum Euclidean distance.

Figure 4.13 to Figure 4.17 are some screenshots of the OCR recognition results for some translated and rotated images.

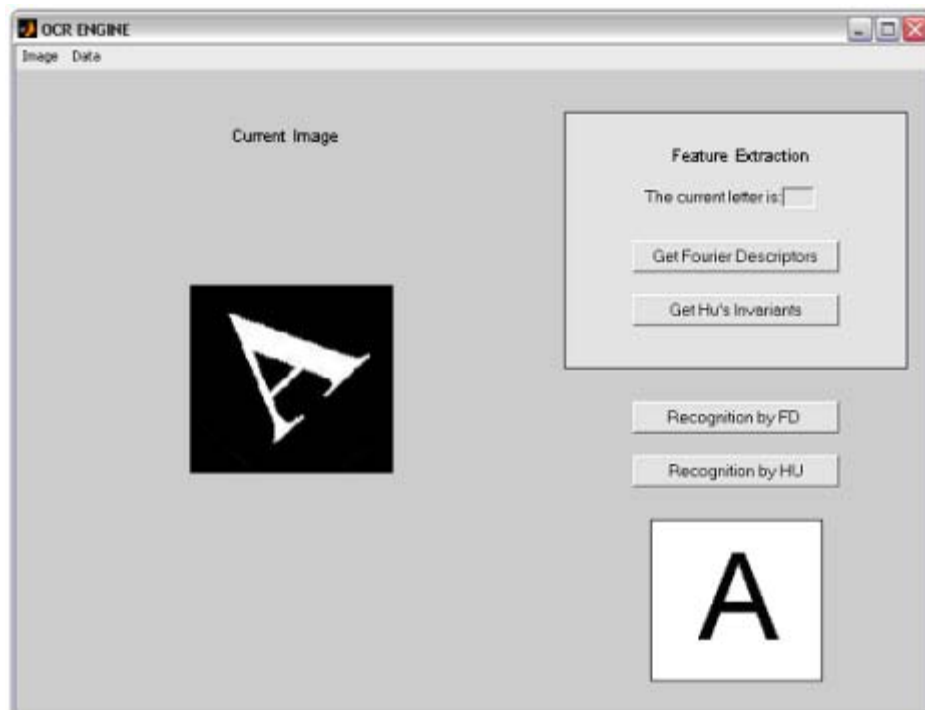


Figure 4.13: Classification result of image “A” of 128×128 rotated by 45 degree (counterclockwise).

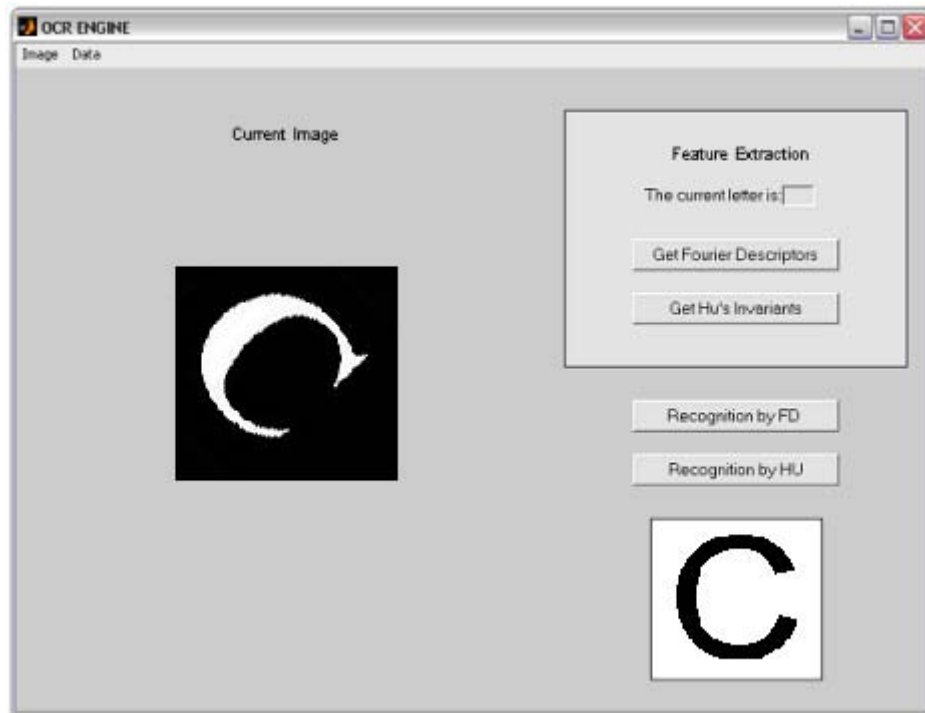


Figure 4.14: Classification result of image “C” of 128×128 rotated by 45 degree (clockwise).

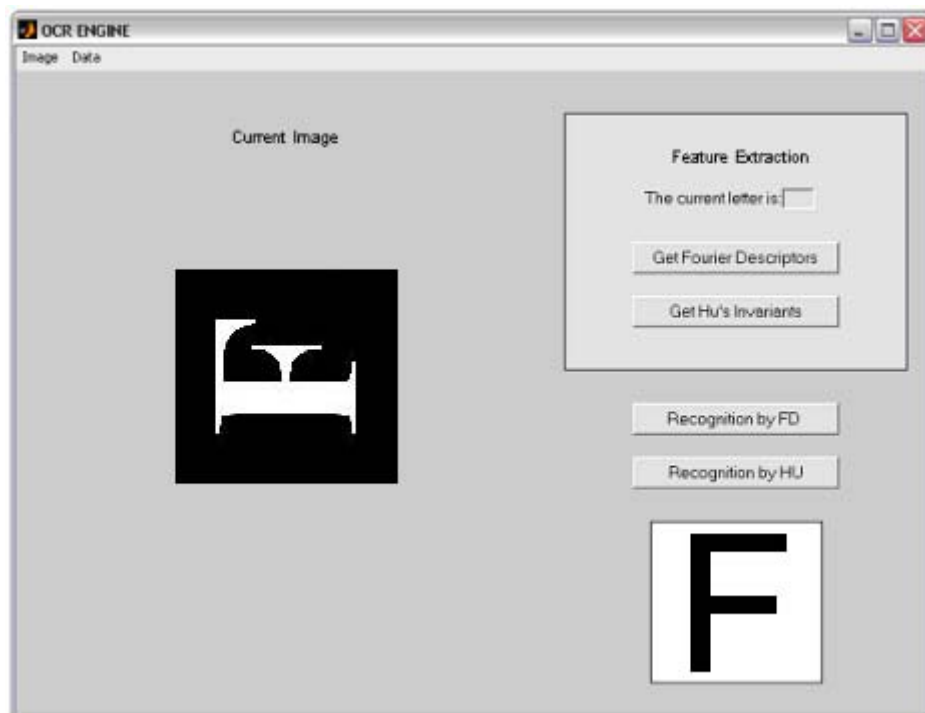


Figure 4.15: Classification result of image “F” of 128×128 rotated by 90 degree (counterclockwise).

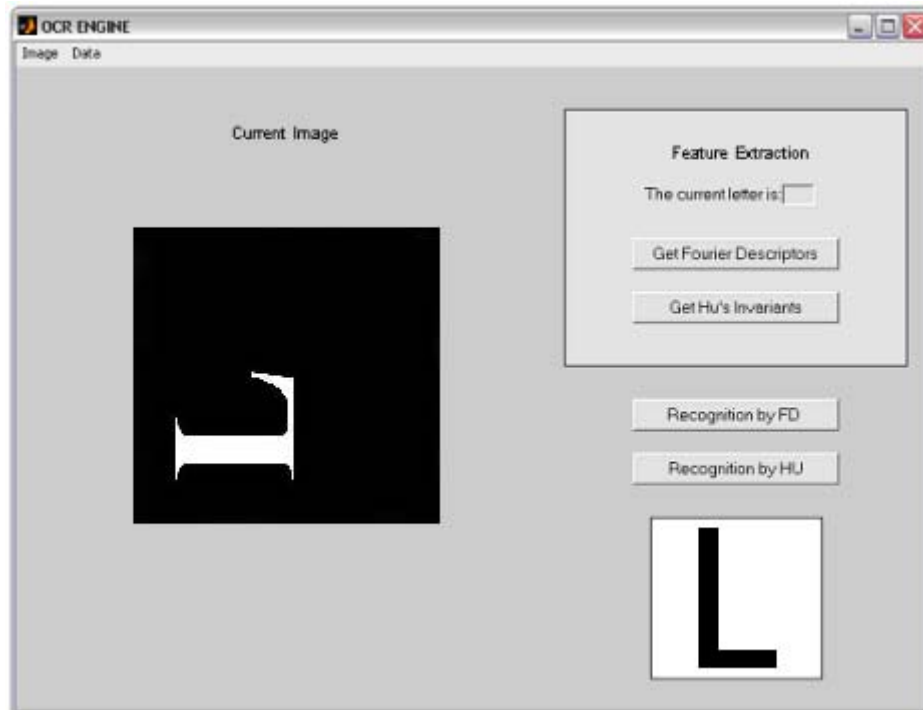


Figure 4.16: Classification result of translated image “L” of 190×190 rotated by 90 degree (counterclockwise).

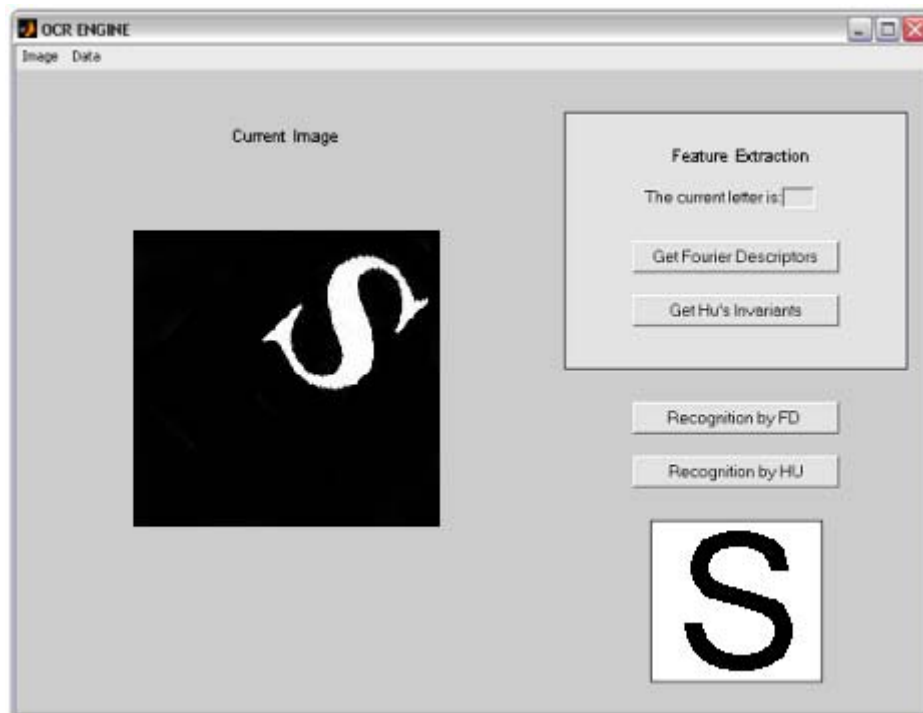


Figure 4.17: Classification result of translated image “S” of 190×190 rotated by 45 degree (clockwise).

Chapter 5

Experiment Results

5.1 Introduction

The experiment results are mainly based on two programs: the feature extraction and image recognition. The results of feature extraction include the calculation of Fourier descriptors and Hu's seven moment invariants of the loaded images of different spatial resolutions as well as noise-affected images. For image recognition, the recognition rate is achieved by using the OCR engine to recognize the 26 capital English letters' images of different resolutions as well as noise-affected images with Fourier descriptors and Hu's seven moment invariants.

For Zernike moments, the experiment focused on the research of their image reconstruction ability. Several reconstructed image examples based on Zernike moments of different orders are given and analyzed.

5.2 Results of Fourier Descriptors

5.2.1 Value Distribution

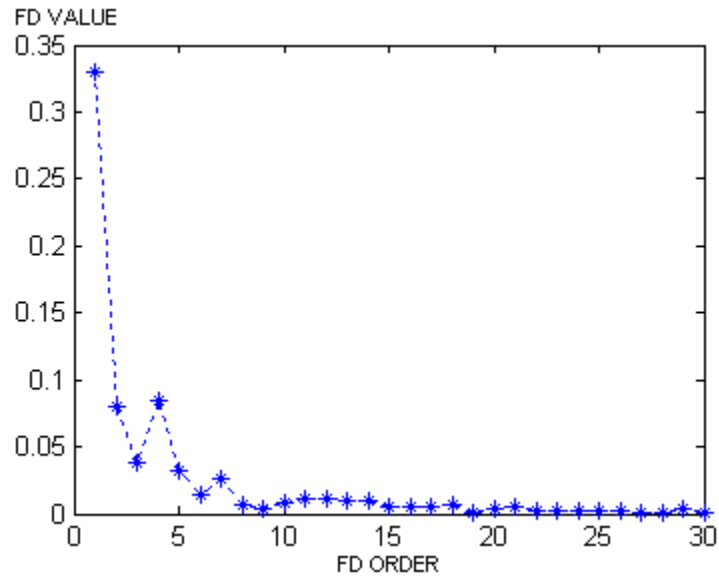


Figure 5.1: The Fourier descriptors' distribution of the image "A" with spatial resolution of 512×512 .

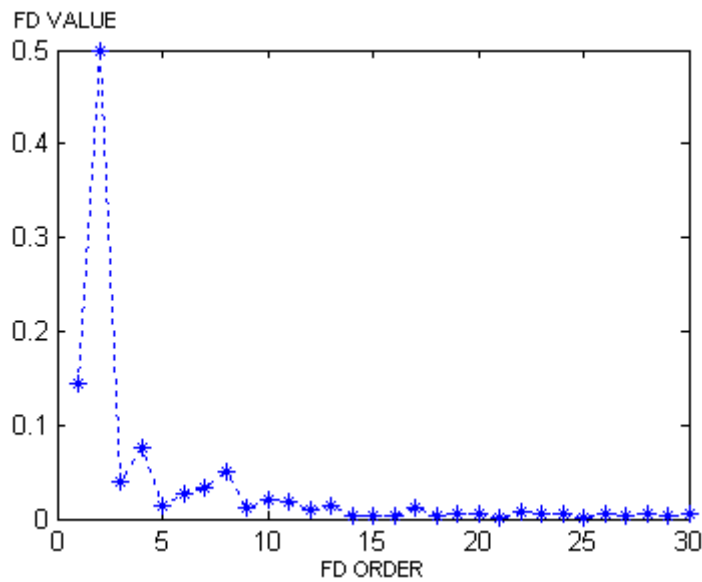


Figure 5.2: The Fourier descriptors' distribution of the image "Z" with spatial resolution of 512×512 .

Figure 5.1 and Figure 5.2 plot the overall distributions of the Fourier descriptors for randomly selected images “A” and “Z” with spatial resolution of 512×512. From the two examples, we see that the most significant values of the series are distributed at the first 10 Fourier descriptors along the curve. The values of the rest Fourier descriptors are very close to zero.

Table 5.1 lists the first 10 Fourier descriptors for the 26 capital English letters’ images with spatial resolution of 512×512.

FD	1	2	3	4	5	6	7	8	9	10
A	0.3296	0.0803	0.0379	0.0839	0.0319	0.0149	0.0260	0.0067	0.0043	0.0080
B	0.0381	0.1239	0.0602	0.0468	0.0384	0.0082	0.0080	0.0108	0.0089	0.0078
C	0.6497	0.1362	0.1197	0.0137	0.0471	0.0234	0.0255	0.0224	0.0112	0.0170
D	0.1020	0.0169	0.0098	0.0032	0.0165	0.0079	0.0031	0.0094	0.0085	0.0001
E	0.5777	0.3097	0.1774	0.0810	0.0324	0.1030	0.0741	0.0093	0.0084	0.0362
F	0.4699	0.1580	0.1354	0.1018	0.0898	0.0438	0.0201	0.0340	0.0208	0.0205
G	0.9483	0.4274	0.1452	0.0498	0.0794	0.0115	0.0275	0.0332	0.0105	0.0221
H	0.0192	0.4181	0.0027	0.0609	0.0048	0.0750	0.0024	0.0270	0.0036	0.0183
I	0.0001	0.0940	0.0001	0.0454	0.0001	0.0046	0.0001	0.0236	0.0001	0.0191
J	0.3489	0.1539	0.1342	0.0401	0.0372	0.0303	0.0187	0.0117	0.0202	0.0061
K	0.0560	0.2400	0.1750	0.1492	0.0183	0.0470	0.0183	0.0327	0.0172	0.0162
L	0.3329	0.1594	0.0482	0.0468	0.0049	0.0431	0.0114	0.0064	0.0115	0.0111
M	0.3858	0.0648	0.5478	0.0517	0.1044	0.0737	0.0323	0.0084	0.0299	0.0026
N	0.1663	0.5425	0.0906	0.0969	0.0298	0.0263	0.0329	0.0302	0.0325	0.0346
O	0.0007	0.0029	0.0016	0.0030	0.0002	0.0012	0.0002	0.0013	0.0004	0.0002
P	0.1395	0.1485	0.0535	0.0109	0.0256	0.0097	0.0123	0.0019	0.0175	0.0076
Q	0.1947	0.0635	0.0570	0.0384	0.0194	0.0038	0.0015	0.0012	0.0047	0.0078
R	0.3578	0.0855	0.1604	0.0487	0.0398	0.0178	0.0205	0.0129	0.0118	0.0150
S	0.0797	0.6049	0.0304	0.0646	0.0138	0.0533	0.0069	0.0389	0.0037	0.0248
T	0.0333	0.2505	0.0696	0.0086	0.0817	0.0251	0.0232	0.0059	0.0047	0.0118
U	1.2393	0.1046	0.0331	0.0859	0.0621	0.0222	0.0467	0.0281	0.0127	0.0249
V	0.7366	0.1176	0.0147	0.0622	0.0817	0.0483	0.0198	0.0328	0.0114	0.0074
W	0.2756	0.0864	0.4233	0.0493	0.0370	0.0663	0.0473	0.0055	0.0141	0.0340
X	0.0102	0.1423	0.0165	0.1933	0.0221	0.0397	0.0060	0.0264	0.0072	0.0151
Y	0.3244	0.1468	0.1303	0.0604	0.0041	0.0096	0.0421	0.0246	0.0143	0.0098
Z	0.1440	0.4992	0.0395	0.0757	0.0144	0.0277	0.0335	0.0508	0.0117	0.0196

Table 5.1: The first 10 Fourier descriptors of the 26 capital English letters’ images with spatial resolution of 512×512.

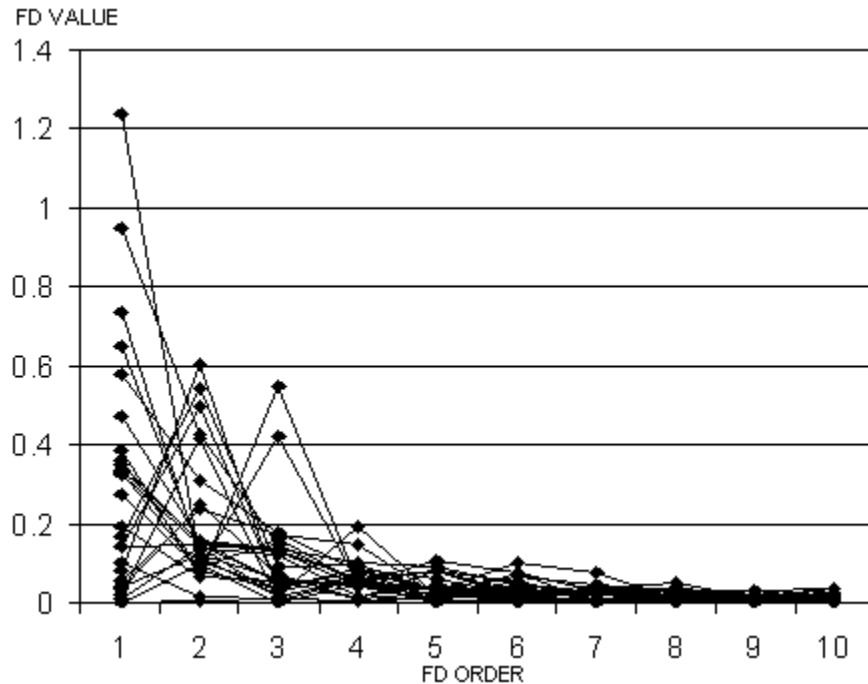


Figure 5.3: The distribution of the first 10 Fourier descriptors of the 26 capital English letters' images with spatial resolution of 512×512 .

Figure 5.3 plots the overall distribution of the first 10 Fourier descriptors for the 26 capital English letters' images with spatial resolution of 512×512 .

When implementing the image recognition, based on the distribution of the most significant values of the Fourier descriptors, we only collect the first 10 elements of the Fourier series to compose the feature vectors. In the study of Fourier descriptors' invariance property to different spatial resolutions, we also focused on the most significant Fourier descriptors which also only include the first 10 elements of the series.

5.2.2 Invariance to Different Spatial Resolutions

To study Fourier descriptors' invariance property to different spatial resolutions, we collected the Fourier descriptors of all image samples with spatial resolutions from

512×512 to 32×32. In this section, we only provide two examples to show the Fourier descriptors' value change versus the different spatial resolutions.

A	1	2	3	4	5	6	7	8	9	10
512×512	0.3296	0.0803	0.0379	0.0839	0.0319	0.0149	0.0260	0.0067	0.0043	0.0080
256×256	0.3371	0.0797	0.0347	0.0854	0.0311	0.0142	0.0254	0.0067	0.0044	0.0080
128×128	0.3521	0.0773	0.0293	0.0889	0.0293	0.0108	0.0282	0.0084	0.0057	0.0059
64×64	0.3417	0.0860	0.0324	0.0893	0.0361	0.0193	0.0307	0.0125	0.0120	0.0028
32×32	0.5781	0.1667	0.0345	0.1149	0.0200	0.0363	0.0453	0.0358	0.0086	0.0177
MEAN	0.3877	0.0980	0.0338	0.0925	0.0297	0.0191	0.0311	0.0140	0.0070	0.0085
STD DEV	0.0955	0.0345	0.0028	0.0114	0.0054	0.0090	0.0073	0.0111	0.0029	0.0050

Table 5.2: The first 10 Fourier descriptors of the image “A” with spatial resolutions from 512×512 to 32×32.

Table 5.2 gives the first 10 Fourier descriptors as well as the mean value and standard deviation of the image “A” with spatial resolutions from 512×512 to 32×32. Figure 5.4 plots the distributions. From the figure, we see the resolution of 64×64 is apparently a threshold for Fourier descriptors. The values keep flat from resolution 512×512 to 64×64, then apparent changes occurred when the resolution drops to 32×32.

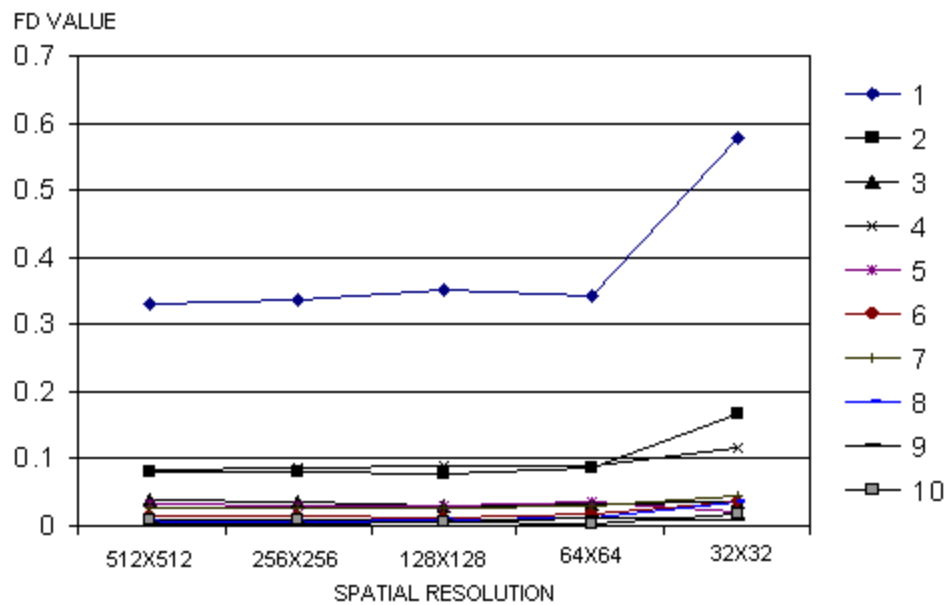


Figure 5.4: The first 10 Fourier descriptors of the image “A” with spatial resolutions from 512×512 to 32×32.

Table 5.3 gives the values of the Fourier descriptors for another example of the image “Z”. Figure 5.5 shows a similar result except the fluctuation after the resolution 64×64 is not so strong compared with last example.

Z	1	2	3	4	5	6	7	8	9	10
512×512	0.1440	0.4992	0.0395	0.0757	0.0144	0.0277	0.0335	0.0508	0.0117	0.0196
256×256	0.1400	0.5012	0.0397	0.0737	0.0143	0.0276	0.0327	0.0503	0.0125	0.0180
128×128	0.1367	0.5023	0.0363	0.0723	0.0117	0.0260	0.0318	0.0518	0.0097	0.0188
64×64	0.1234	0.5062	0.0380	0.0767	0.0124	0.0290	0.0221	0.0572	0.0150	0.0202
32×32	0.1362	0.5020	0.0422	0.0843	0.0008	0.0712	0.0385	0.0434	0.0242	0.0070
MEAN	0.1361	0.5022	0.0392	0.0765	0.0107	0.0363	0.0317	0.0507	0.0146	0.0167
STD DEV	0.0069	0.0023	0.0020	0.0042	0.0051	0.0175	0.0053	0.0044	0.0051	0.0049

Table 5.3: The first 10 Fourier descriptors of the image “Z” with spatial resolutions from 512×512 to 32×32.

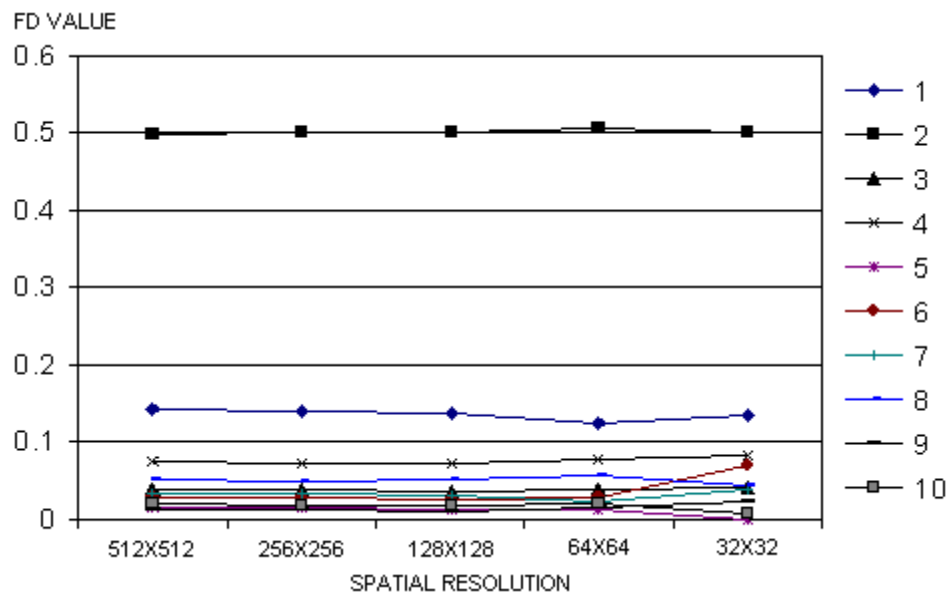


Figure 5.5: The first 10 Fourier descriptors of the image “Z” with spatial resolutions from 512×512 to 32×32.

5.2.3 Sensitivity to Salt and Pepper Noises

Fourier descriptors only capture the information of the image's boundary by sampling the pixels along the boundary. A small change in the image's boundary may lead to a very different result. Because of this, the Fourier descriptors are very sensitive to salt and pepper noise.

To overcome the sensitivity to salt and pepper noise, the noise-affected images need to be filtered before the computation Fourier descriptors. An effective way to eliminate salt and pepper noise is the median filter [14]. The median filter sets each output pixel's value according to values of its neighborhood pixels. The value of the output pixel is determined by the *median* of the neighborhood pixels, rather than the mean. The median filter is much less sensitive to extreme values such as the impulses of the salt and pepper noise. Median filter is therefore better able to remove these impulses without reducing the sharpness of the image. Figure 5.6 shows the effectiveness of a 3×3 median filter.



Figure 5.6: The filtering result of a 3×3 median filter on a salt and pepper affected image.

5.2.4 Overall Performance Evaluation

To evaluate the overall performance of the Fourier descriptors, we implemented the image recognition using our OCR engine developed with MATLAB. The evaluation process includes two steps. First, we collected image features in the format of Fourier descriptors with the 26 capital English letters' images with resolution of 256×256 , and saved it to a model library which is in the MATLAB data file format. This model library is loaded first before we open an image and implement the recognition.

Table 5.4 lists the overall performance result. We see that for images with resolutions from 512×512 to 64×64 , the recognition rate keeps at 100%. It drops to 53.8% at resolution of 32×32 , which verifies the spatial resolution invariance analysis in last section.

SPATIAL RESOLUTION	IMAGES MISSED	RECOGNITION RATE
512×512	0/26	100%
256×256	0/26	100%
128×128	0/26	100%
64×64	0/26	100%
32×32	12/26	53.8%

Table 5.4: The recognition result using Fourier descriptors for the images of 26 capital English letters with resolutions from 512×512 to 32×32 .

5.3 Results of Hu's Seven Moment Invariants

5.3.1 Value Distribution

Table 5.5 gives the Hu's seven moment invariants for the 26 capital English letters' images with spatial resolution of 512×512. Figure 5.7 plots the overall distribution of the values.

HU	1	2	3	4	5	6	7
A	0.3669	0.0076	0.0347	0.0018	0.0000	0.0001	0.0000
B	0.2838	0.0036	0.0006	0.0000	0.0000	0.0000	0.0000
C	0.5330	0.0173	0.0239	0.0280	-0.0004	-0.0025	0.0006
D	0.3330	0.0001	0.0033	0.0000	0.0000	0.0000	0.0000
E	0.3874	0.0217	0.0006	0.0022	0.0000	-0.0003	0.0000
F	0.3698	0.0374	0.0106	0.0008	0.0000	-0.0001	0.0000
G	0.4868	0.0031	0.0039	0.0052	0.0000	-0.0001	0.0000
H	0.3354	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
I	0.3933	0.1153	0.0000	0.0000	0.0000	0.0000	0.0000
J	0.3925	0.0785	0.0095	0.0012	0.0000	-0.0002	0.0000
K	0.3335	0.0049	0.0014	0.0014	0.0000	-0.0001	0.0000
L	0.4687	0.0671	0.0444	0.0065	0.0000	-0.0012	-0.0001
M	0.3791	0.0100	0.0009	0.0003	0.0000	0.0000	0.0000
N	0.4008	0.0074	0.0010	0.0009	0.0000	-0.0001	0.0000
O	0.4076	0.0061	0.0000	0.0000	0.0000	0.0000	0.0000
P	0.3263	0.0216	0.0140	0.0008	0.0000	0.0001	0.0000
Q	0.4075	0.0024	0.0057	0.0000	0.0000	0.0000	0.0000
R	0.2996	0.0024	0.0005	0.0000	0.0000	0.0000	0.0000
S	0.3765	0.0170	0.0008	0.0001	0.0000	0.0000	0.0000
T	0.4261	0.0532	0.0355	0.0002	0.0000	0.0000	0.0000
U	0.4771	0.0100	0.0063	0.0083	0.0001	-0.0008	0.0000
V	0.4156	0.0124	0.0463	0.0043	0.0001	-0.0003	0.0000
W	0.3544	0.0087	0.0057	0.0005	0.0000	0.0000	0.0000
X	0.3973	0.0253	0.0004	0.0000	0.0000	0.0000	0.0000
Y	0.4097	0.0332	0.0359	0.0009	0.0000	-0.0001	0.0000
Z	0.4305	0.0277	0.0041	0.0002	0.0000	0.0000	0.0000

Table 5.5: The Hu's 7 moment invariants of the 26 capital English letters' image samples with spatial resolution of 512×512.

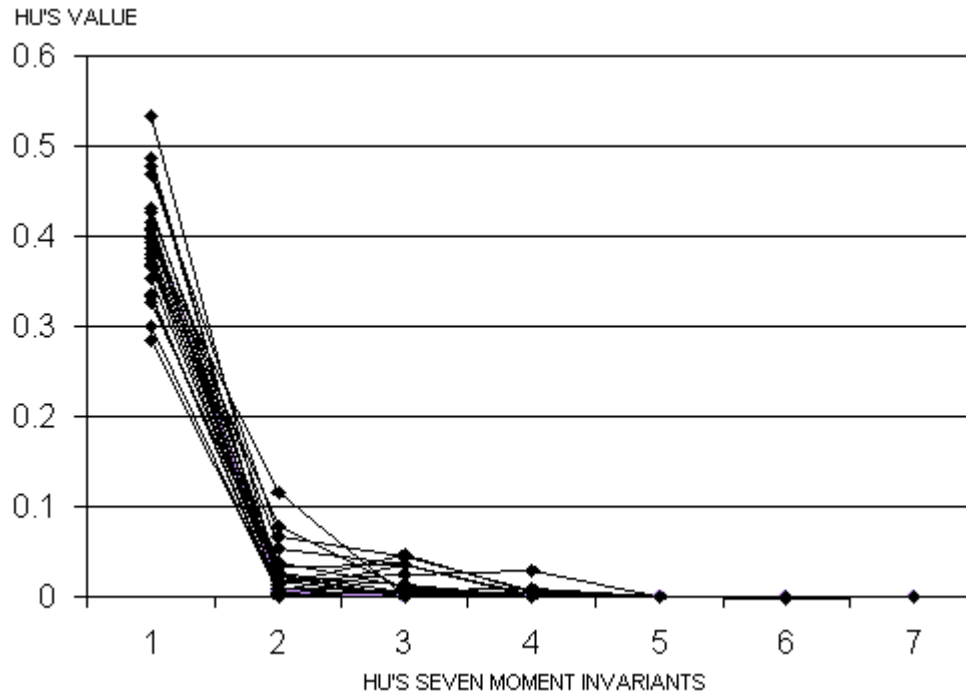


Figure 5.7: The distribution of Hu's 7 moment invariants of the 26 English capital letters' images with spatial resolution of 512×512 .

From Table 5.5 and Figure 5.7 we see the values of the computed moment invariants are usually small. We also find the most significant values concentrate at the first three moment invariants, and the values from fourth to seventh moment invariants are close or equal to zero.

As the first three moment invariants are the most significant elements, we plotted the distribution of the 26 capital English letters images with the first three moment invariants as the coordinates. Figure 5.8 shows the result. We see from the figure that the 26 letters set apart very well without apparent position overlapping. Based on this point, we can conclude that Hu's seven moment invariants can meet the recognition requirements.

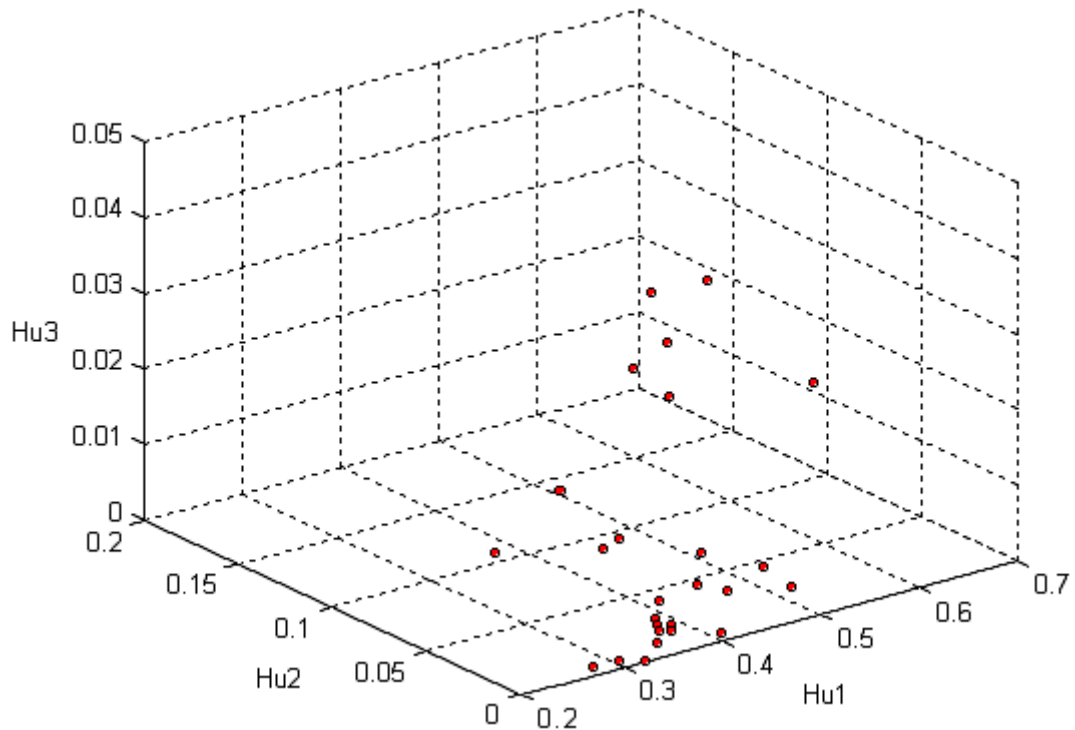


Figure 5.8: The distribution of the 26 capital English letters with the first three moment invariants as the coordinates.

5.3.2 Invariance to Different Spatial Resolutions

To study Hu's moment invariants' invariance property to different spatial resolutions, we calculate the seven Hu's moment invariants of 26 capital English letters' images with spatial resolutions from 512×512 to 32×32 .

In this section, we provided the results of randomly selected image samples of capital letters "C", "T", and "V". Table 5.6, Table 5.7 and Table 5.8 tabulate the results. Figure 5.9, Figure 5.10 and Figure 5.11 plot the corresponding diagrams according to the values listed in the tables.

From these figures, we see that the resolution of 128×128 is the threshold for Hu's seven moment invariants. From the resolution of 512×512 to 128×128 , the values are kept in a comparatively stable situation and they crossed up at the resolution of 64×64 , and we can predict the recognition rate using Hu's seven moment invariants will drop when the image's resolution reaches to 64×64 .

C	1	2	3	4	5	6	7
512×512	0.5330	0.0173	0.0239	0.0280	-0.0004	-0.0025	0.0006
256×256	0.5319	0.0171	0.0234	0.0281	-0.0004	-0.0026	0.0006
128×128	0.5324	0.0176	0.0242	0.0281	-0.0005	-0.0027	0.0006
64×64	0.5358	0.0166	0.0223	0.0294	-0.0006	-0.0030	0.0005
32×32	0.5373	0.0236	0.0274	0.0254	-0.0005	-0.0032	0.0005
16×16	0.5862	0.0592	0.0362	0.0224	-0.0002	-0.0031	0.0006
MEAN	0.5427	0.0252	0.0262	0.0269	-0.0004	-0.0028	0.0005
STD DEV	0.0195	0.0154	0.0047	0.0023	0.0001	0.0003	0.0000

Table 5.6: The Hu's 7 moment invariants of the image "C" with spatial resolutions from 512×512 to 16×16.

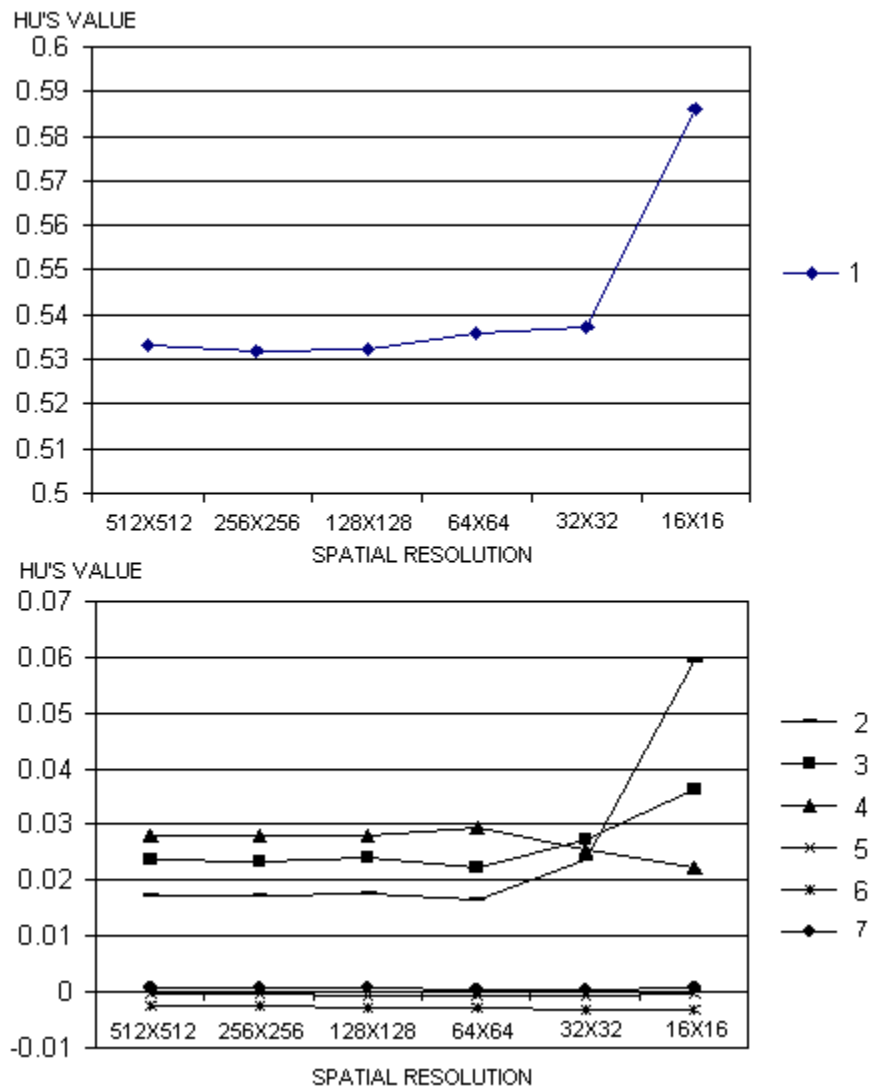


Figure 5.9: The Hu's moment invariants of the image "C" with spatial resolutions from 512×512 to 16×16.

T	1	2	3	4	5	6	7
512×512	0.4261	0.0532	0.0355	0.0002	0.0000	0.0000	0.0000
256×256	0.4265	0.0536	0.0355	0.0002	0.0000	0.0000	0.0000
128×128	0.4289	0.0533	0.0358	0.0001	0.0000	0.0000	0.0000
64×64	0.4534	0.0631	0.0394	0.0001	0.0000	0.0000	0.0000
32×32	0.4155	0.0569	0.0244	0.0001	0.0000	0.0000	0.0000
16×16	0.3395	0.0472	0.0051	0.0002	0.0000	0.0000	0.0000
MEAN	0.4150	0.0546	0.0293	0.0002	0.0000	0.0000	0.0000
STD DEV	0.0356	0.0048	0.0118	0.0000	0.0000	0.0000	0.0000

Table 5.7: The Hu's 7 moment invariants of the image "T" with spatial resolutions from 512×512 to 16×16.

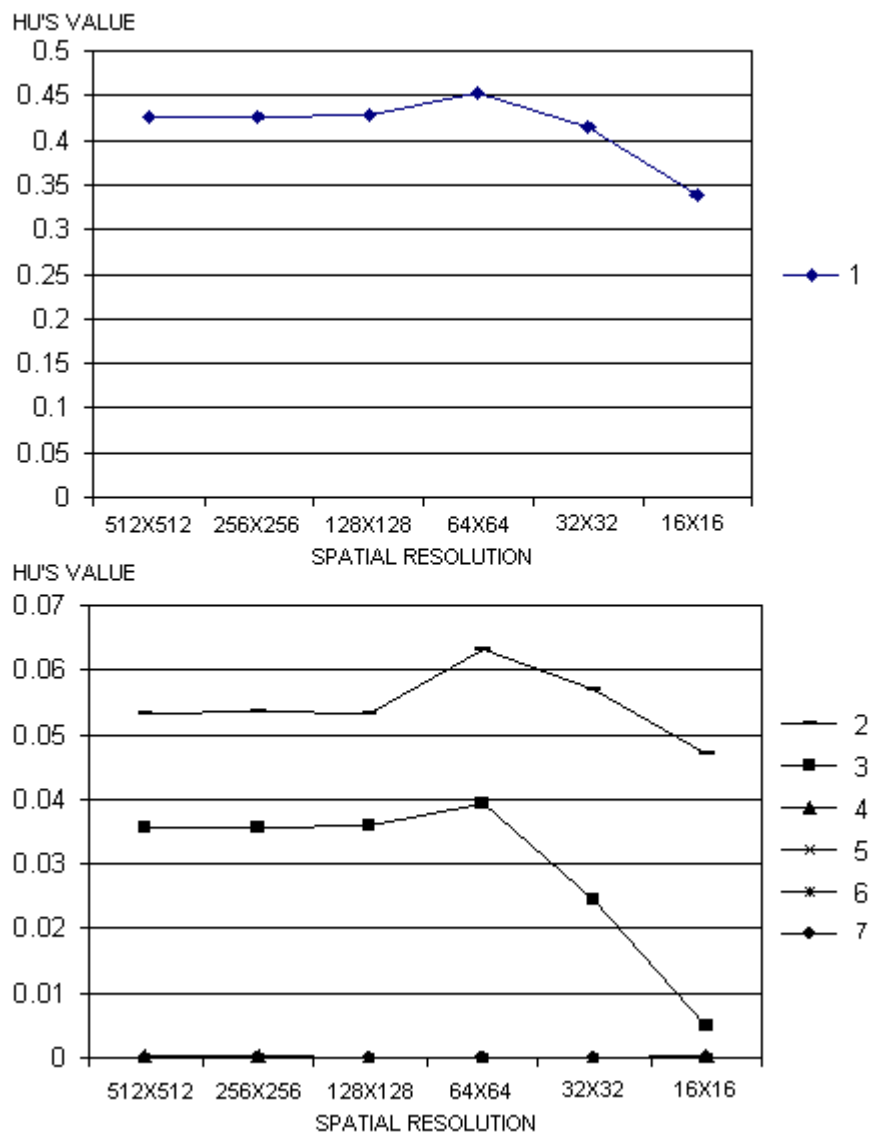


Figure 5.10: The Hu's moment invariants of the image "T" with spatial resolutions from 512×512 to 16×16.

V	1	2	3	4	5	6	7
512×512	0.4156	0.0124	0.0463	0.0043	0.0001	-0.0003	0.0000
256×256	0.4152	0.0126	0.0461	0.0043	0.0001	-0.0003	0.0000
128×128	0.4141	0.0129	0.0452	0.0042	0.0001	-0.0003	0.0000
64×64	0.4117	0.0136	0.0437	0.0040	0.0000	-0.0003	0.0000
32×32	0.4080	0.0143	0.0432	0.0041	0.0000	-0.0003	0.0000
16×16	0.4111	0.0176	0.0481	0.0028	0.0000	-0.0003	0.0000
MEAN	0.4126	0.0139	0.0454	0.0039	0.0000	-0.0003	0.0000
STD DEV	0.0027	0.0018	0.0017	0.0005	0.0000	0.0000	0.0000

Table 5.8: The Hu's 7 moment invariants of the image "V" with spatial resolutions from 512×512 to 16×16.

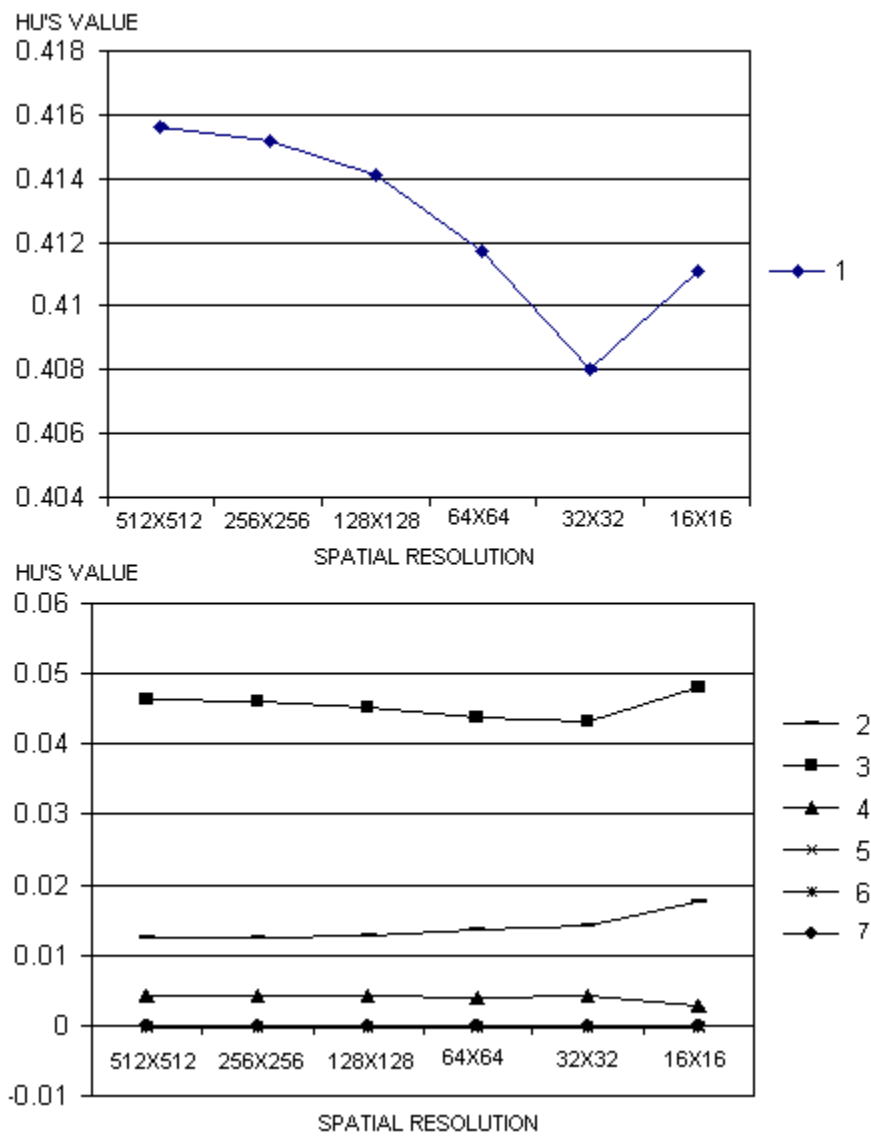


Figure 5.11: The Hu's moment invariants of the image "V" with spatial resolutions from 512×512 to 16×16.

5.3.3 Invariance to Salt and Pepper Noises

Hu's seven moment invariants provide good properties under noise-free condition. However, in the presence of noise, the computed moment invariants are not expected to be strictly invariant.

Table 5.9, Table 5.10 and Table 5.11 list the values of Hu's seven moment invariants of the images "C", "T" and "V" corrupted by salt and pepper noise of intensities from 0.01 to 0.07. Figure 5.12, Figure 5.13 and Figure 5.14 plot the diagrams. From the figures we see that the first Hu's moment invariant goes up approximately linearly as the noise intensity increases. On the other side, the second and third moment invariant goes down when the intensity increases.

Compared with Fourier descriptors' vulnerability to salt and pepper noise, Hu's seven moment invariants are more robust. We can still use the image features based on Hu's seven moment invariants to recognize salt and pepper noise-corrupted images without any implementation modifications. But we can expect the recognition rate will decrease as the noise intensity goes up.

C	1	2	3	4	5	6	7
0	0.5330	0.0173	0.0239	0.0280	-0.0004	-0.0025	0.0006
0.01	0.5511	0.0147	0.0195	0.0292	-0.0004	-0.0025	0.0005
0.02	0.5615	0.0128	0.0162	0.0287	-0.0004	-0.0023	0.0005
0.03	0.5743	0.0100	0.0128	0.0290	-0.0004	-0.0022	0.0004
0.04	0.5843	0.0091	0.0107	0.0288	-0.0004	-0.0020	0.0003
0.05	0.5883	0.0074	0.0080	0.0270	-0.0003	-0.0018	0.0002
0.06	0.5984	0.0064	0.0068	0.0265	-0.0003	-0.0017	0.0002
0.07	0.6009	0.0056	0.0063	0.0252	-0.0003	-0.0014	0.0002

Table 5.9: The Hu's 7 moment invariants of the image "C" affected by salt and pepper noises with intensities from 0.01 to 0.07.

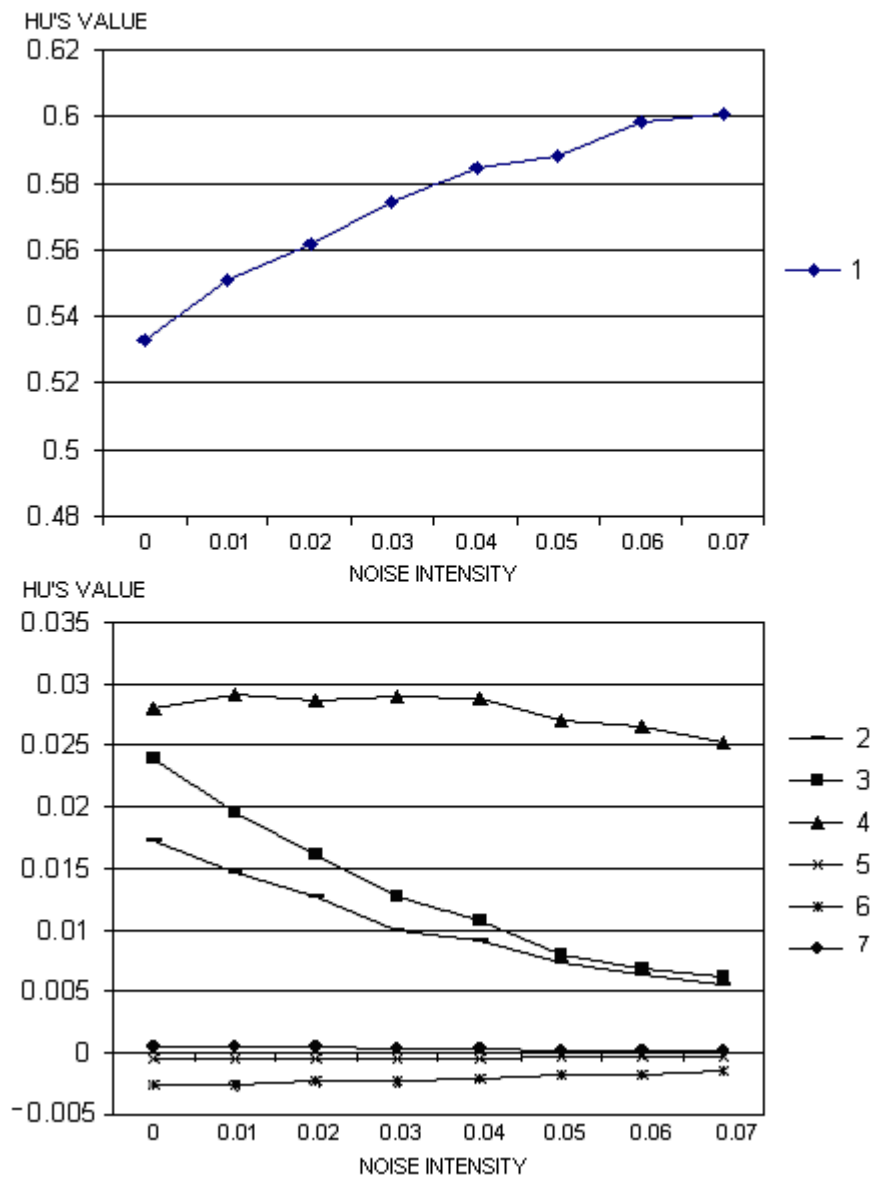


Figure 5.12: The Hu's moment invariants of the image "C" affected by salt and pepper noises with intensities from 0.01 to 0.07.

T	1	2	3	4	5	6	7
0	0.4261	0.0532	0.0355	0.0002	0.0000	0.0000	0.0000
0.01	0.4485	0.0466	0.0282	0.0004	0.0000	0.0001	0.0000
0.02	0.4683	0.0410	0.0236	0.0006	0.0000	0.0001	0.0000
0.03	0.4847	0.0353	0.0197	0.0008	0.0000	0.0002	0.0000
0.04	0.4973	0.0327	0.0165	0.0010	0.0000	0.0002	0.0000
0.05	0.5126	0.0289	0.0126	0.0013	0.0000	0.0002	0.0000
0.06	0.5224	0.0245	0.0101	0.0012	0.0000	0.0002	0.0000
0.07	0.5317	0.0216	0.0094	0.0012	0.0000	0.0002	0.0000

Table 5.10: The Hu's 7 moment invariants of the image "T" affected by salt and pepper noises with intensities from 0.01 to 0.07.

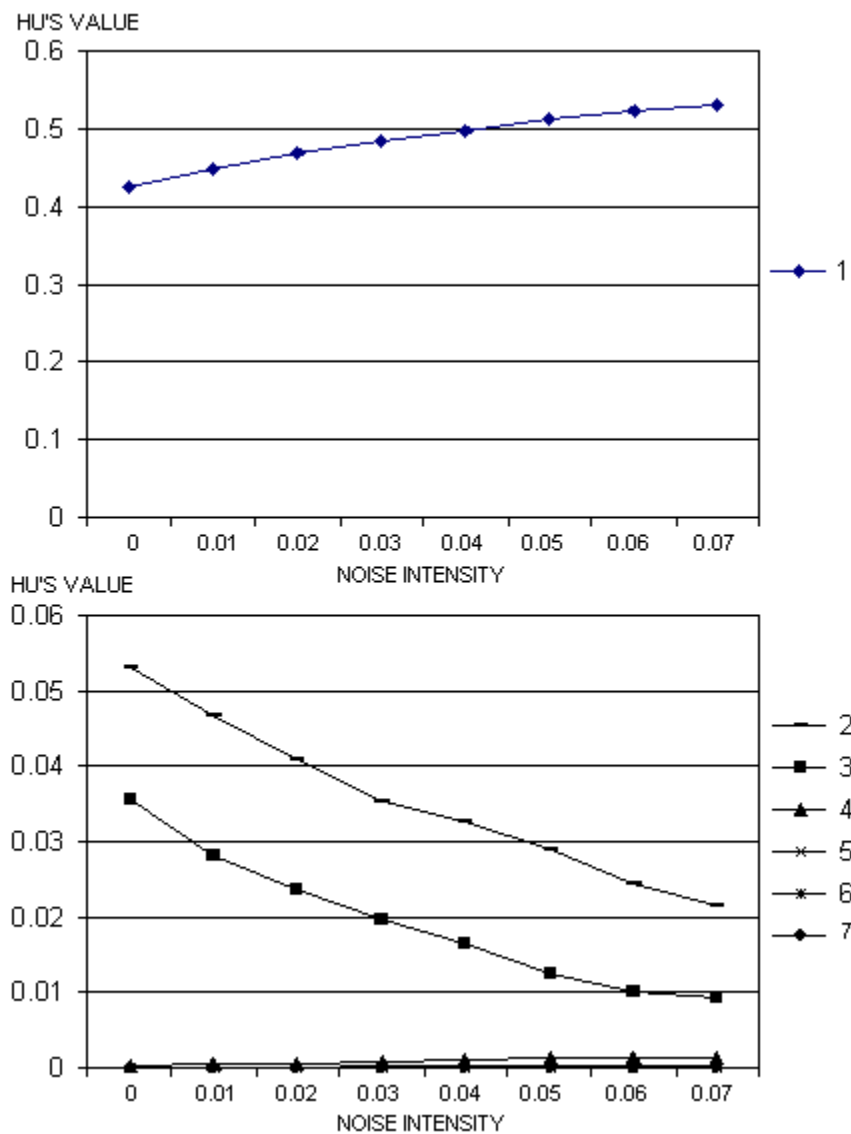


Figure 5.13: The Hu's moment invariants of the image "T" affected by salt and pepper noises with intensities from 0.01 to 0.07.

V	1	2	3	4	5	6	7
0	0.4156	0.0124	0.0463	0.0043	0.0001	-0.0003	0.0000
0.01	0.4472	0.0110	0.0377	0.0037	0.0000	-0.0004	0.0000
0.02	0.4719	0.0095	0.0314	0.0040	0.0000	-0.0004	0.0000
0.03	0.4943	0.0088	0.0254	0.0047	0.0000	-0.0004	0.0001
0.04	0.5163	0.0079	0.0227	0.0048	0.0000	-0.0003	0.0000
0.05	0.5304	0.0065	0.0193	0.0056	0.0000	-0.0003	0.0001
0.06	0.5457	0.0066	0.0153	0.0054	0.0000	-0.0002	0.0000
0.07	0.5538	0.0055	0.0134	0.0055	0.0000	-0.0001	0.0000

Table 5.11: The Hu's 7 moment invariants of the image "V" affected by salt and pepper noises with intensities from 0.01 to 0.07.

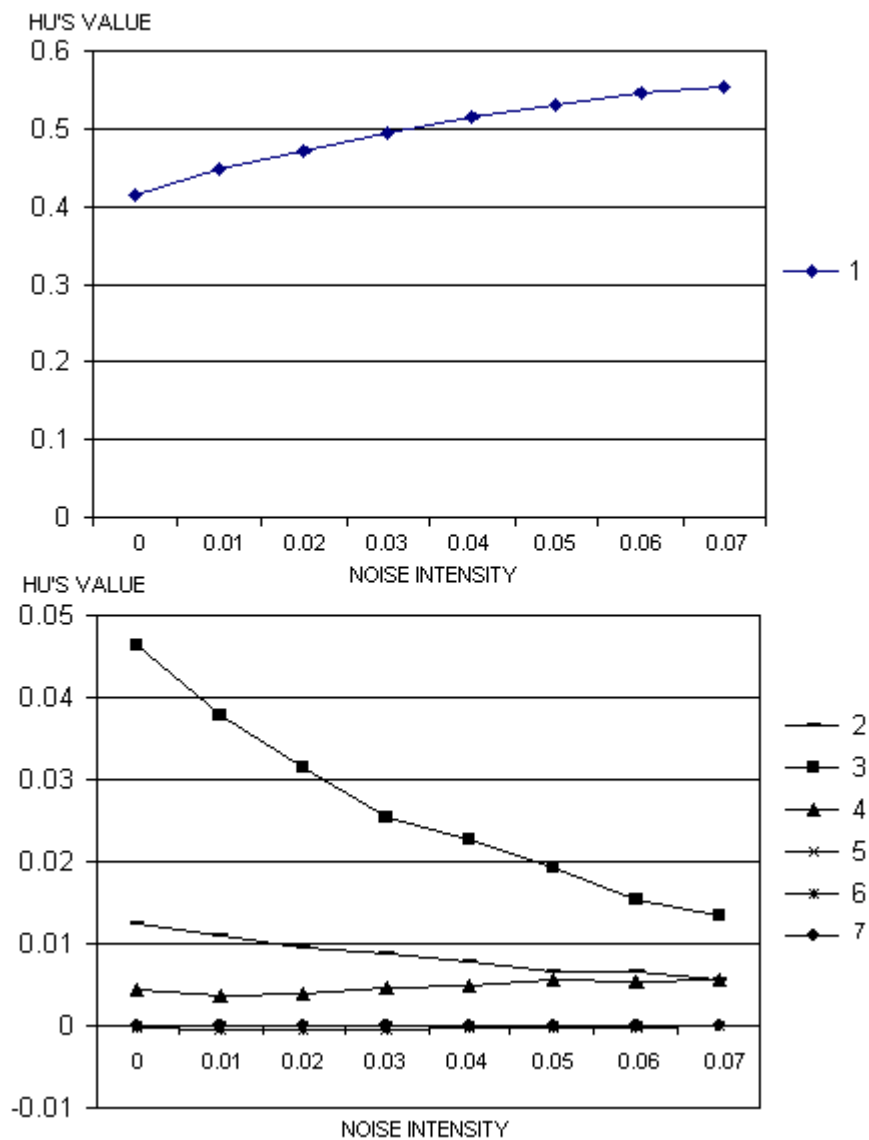


Figure 5.14: The Hu's moment invariants of the image "V" affected by salt and pepper noises with intensities from 0.01 to 0.07.

5.3.4 Overall Performance Evaluation

We implemented the image recognition using the OCR engine. The evaluation includes two steps. First, we evaluated the performance of Hu's moment invariants with the 26 capital English letters' images with different resolutions from 512×512 to 32×32. Second, we evaluated the performance of Hu's moment invariants with salt and pepper noise corrupted images with resolution of 512×512. The noise intensities spans from 0.01 to 0.07. For both parts, the model library we used are built up with the features collected from the clean images with resolution of 256×256.

Table 5.12 lists the recognition result with images of different resolutions. We see that for images with resolutions from 512×512 to 128×128, the recognition rate keeps at 100%. It drops to 53.8% at resolution of 64×64 and 69.2% at resolution of 32×32, which verifies the spatial resolution invariance analysis in section 5.3.2.

SPATIAL RESOLUTION	IMAGES MISSED	RECOGNITION RATE
512×512	0/26	100%
256×256	0/26	100%
128×128	0/26	100%
64×64	4/26	84.6%
32×32	8/26	69.2%

Table 5.12: The recognition result using Hu's moment invariants for the images of 26 capital English letters with resolutions from 512×512 to 32×32.

Table 5.13 is the recognition results with images corrupted by salt and pepper noises. The results show the noises have a very severe impact on the recognition rate. When the noise intensity gets to 0.03, the recognition rate drops to 15.4%.

NOISE INTENSITY	IMAGES MISSED	RECOGNITION RATE
0.01	10/26	61.5%
0.02	15/26	42.3%
0.03	22/26	15.4%

Table 5.13: The recognition result using Hu's moment invariants for the images of 26 capital English letters corrupted by salt and pepper noises with intensities from 0.01 to 0.04.

5.4 Results of Zernike Moments

5.4.1 Reconstruction Property Evaluation

Compared with Hu's seven moment invariants, the advantage of Zernike moments is that it can be calculated easily to an arbitrarily high order. Due to this property, one question need to be answered is: up to what order of Zernike moments are needed to compose the feature vector so that we can achieve good image recognition result? An easy method to answer this question is to utilize Zernike moments' orthogonal property which can reconstruct the image from a finite series.

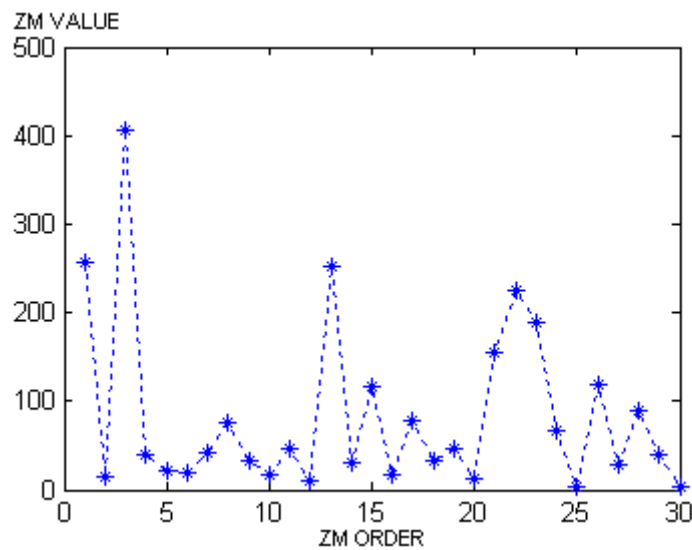


Figure 5.15: The distribution of the Zernike moments' magnitudes up to order 30 of image "B" with resolution of 64×64.

The results of Zernike moments are a series of complex numbers. Figure 5.15 is the diagram of the magnitudes of the first 30 Zernike moments of image "B" with resolution

of 64×64 . The magnitudes of the complex series do not show any regular properties as the Fourier descriptors do.

The difference between an image and its reconstructed version from a set of Zernike moments up to a finite order is a good measure to answer above question. In the experiment, we implemented the reconstruction process of Zernike moments with the image samples of the resolution of 64×64 . Figure 5.16 shows some reconstruction result for different capital English letter images with a set of Zernike moments up to order 40.

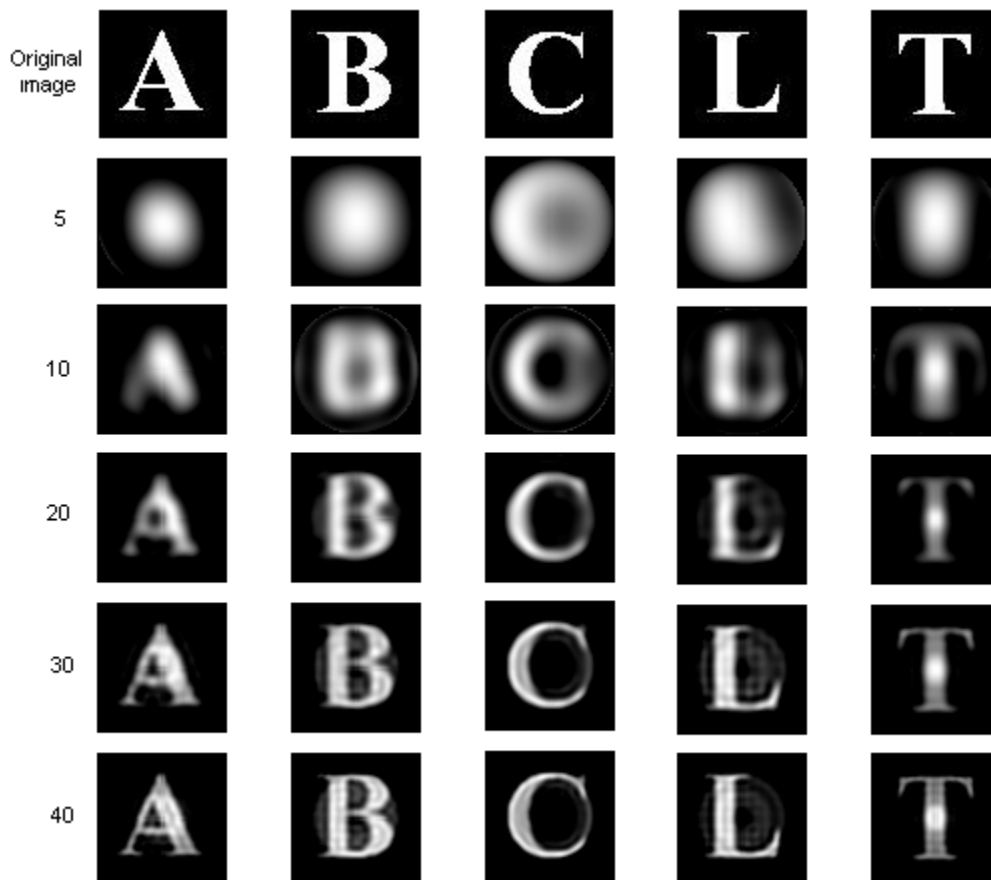


Figure 5.16: Reconstruction results with Zernike moments of order 5, 10, 20, 30, 40.

From Figure 5.16, we see that Zernike moments up to order 10 can catch the gross shapes of the images with resolutions of 64×64 . When the order reaches to 20, the reconstructed images can be recognized by human eyes without confusions. The reconstructed images are more and more clear and include most of the details of the original images when the order gets to 40. Similar to the property of Fourier descriptors, the lower order Zernike moments capture the gross shape information, and the higher order moments capture the fine details of the image.

Based on the reconstruction result, we recommend using Zernike moments up to order 20 to compose the feature vectors so that good recognition result can be achieved.

Chapter 6

Conclusions

6.1 Conclusions

This thesis compared and evaluated the performance of the shape-based image invariants in OCR from the perspective of their invariance property to image transformations including scale, translation, rotation, different spatial resolutions as well as salt and pepper noises of different intensities. Here are the conclusions:

1. both Fourier descriptors and Hu's seven moment invariants can be used in OCR for 26 capital English letters image samples of different resolutions, and display very stable invariance properties against scaling, translation and rotation. However, a resolution threshold exists for each method. For Fourier descriptors, with feature vectors composed of the 10 most significant Fourier descriptors, the recognition rate keeps at 100% when the images resolutions are at 64×64 . It drops sharply to 53.8% when the spatial resolution decrease to 32×32 . For Hu's seven moment invariants, the resolution threshold is 128×128 . The recognition rate drops to 84.6% when the image resolution drops to 64×64 .
2. for images affected by salt and pepper noise, Fourier descriptors are more vulnerable than Hu's seven moment invariants. A small variation along the image boundary pixels can result to a very different Fourier descriptors series. It is

necessary to eliminate the salt and pepper noise with median filter before the noise-corrupted images can be recognized with Fourier descriptors. Hu's seven moment invariants are more robust against salt and pepper noise as they take all image pixels into account. The same implementation can still be used towards the salt and pepper noise corrupted images without any changes. However, the salt and pepper noise do affect the recognition result obviously. For 512×512 images corrupted by salt and pepper noise of intensity 0.01, the recognition rate drops to 61.5%.

3. the computation time of Hu's seven moment invariants is much longer than Fourier descriptors for the same image. With the same computer configuration, the average computation time of Hu's seven moment invariants is about as three times as Fourier descriptors.
4. Zernike moments themselves are only invariant to image rotations. The invariance to different spatial resolutions can be achieved by enlarging or reducing each image so that the image's 0th regular moment m'_{00} equals to a predetermined value. Compared with Hu's seven moment invariants, Zernike moments possess the priority of easy image reconstruction ability. Base on the analysis of reconstructed images with Zernike moments of different orders, we recommend using the first 20 orders of the series to compose the feature vectors to achieve good image recognition result.
5. a generic MATLAB OCR Engine is developed which can be used for gray level images in either "JPEG" or "TIFF" format.

6.2 Future Work

In this thesis, we mainly focused our research on the image invariants from the perspective of the image transformations of scale, translation and rotation. The evaluation of the invariant property of these image invariants under the transformation of shear will be of great interest for the future work; because it is very common that the angle of the camera can be changed during the image acquisition. Find out the angle range of the camera to keep the property of the image invariants will make the AGV and 3D model-base object recognition system more flexible and reliable.

Another research interest will be on the images degraded or blurred by various reasons. In this thesis, we researched the salt and pepper affected images. But as the image acquisition process can be affected by other factors such as diffraction, lens aberration, wrong focus and atmospheric turbulence, the degraded image can be of various formats. Exploration of the blur-invariant image features based on these image invariants or some other algorithms will make the recognition systems more reliable and robust.

Appendix

MATLAB Scripts

A.1 ocr

```

function ocr(op)
% the ocr engine

global H I L HU FD H_index F_index H_temp F_temp s hcount fcount htest ftest H_dist
      F_dist

% if no input argument, draw the GUI
if nargin == 0
op = 0;
end

width = 800;
height = 600;

switch op

%-----0
case 0 % Draw figure
    hcount = 0;
    fcount = 0;
    s = get(0,'ScreenSize');

%-----FIGURE & MENUS-----

H.fig = figure('Position',[(s(3)-width)/2 (s(4)-height)/2 width height],...
    'NumberTitle','off',...
    'MenuBar','none',...
    'Color',[.8 .8 .8],...
    'Name','OCR ENGINE');

H.menu(1) = uimenu(H.fig,'Label','&Image');

H.menu(2) = uimenu(H.menu(1),'Label','&Open',...

```

```

        'Callback','ocr(1)'); %-----Case 1

H.menu(3) = uimenu(H.fig,'Label','&Data');

H.menu(5) = uimenu(H.menu(3),'Label','&Load',...
    'Callback','ocr(7)'); %-----Case 7

H.menu(4) = uimenu(H.menu(3),'Label','&Save',...
    'Callback','ocr(8)'); %-----Case 8

% -----IMAGE FRAME -----

% "Current Image" Frame
H.ax(1) = axes('position',[20/width 20/height 0.5+25/width 1-60/height],...
    'XTick',[],'YTick',[],'Box','on');

uicontrol('Style','text',...
    'BackgroundColor',[.8 .8 .8],...
    'Units','normalized',...
    'Position',[20/width (height-30)/height 0.5+25/width 20/height],...
    'String','Current Image',...
    'HorizontalAlignment','center',...
    'FontSize',12);

% -----FEATURE TRAINING FRAME-----

% Feature Training Frame
uicontrol('Style','frame',...
    'Units','normalized',...
    'Position',[480/width 320/height 300/width 1-360/height]);

uicontrol('Style','text',...
    'Units','normalized',...
    'Position',[580/width (height-120)/height 180/width 50/height],...
    'String','Feature Extraction',...
    'HorizontalAlignment','left',...
    'FontSize',10);

% H.button(1):'Get Hu's Invariants'
H.button(1) = uicontrol('Style','pushbutton',...
    'Units','normalized', ...
    'Position',[540/width (height-240)/height 180/width 30/height],...
    'FontSize',10,...

```

```

'String','Get Hu's Invariants',...
'CallBack','ocr(2)'); %-----Case 2

% H.button(2):'Get Fourier Descriptors'
H.button(2) = uicontrol('Style','pushbutton',...
'Units','normalized', ...
'Position',[540/width (height-190)/height 180/width 30/height],...
'FontSize',10,...
'String','Get Fourier Descriptors',...
'CallBack','ocr(3)'); %-----Case 3

uicontrol('Style','text',...
'Units','normalized',...
'Position',[550/width (height-140)/height 120/width 30/height],...
'String','The current letter is:',...
'HorizontalAlignment','left',...
'FontSize',10);

% H.edit(1): edit box
H.edit(1) = uicontrol('Style','edit',...
'Units','normalized',...
'Position',[670/width (height-130)/height 30/width 20/height],...
'HorizontalAlignment','center',...
'FontSize',10,...
'CallBack','ocr(4)'); %-----Case 4

% -----CHARACTER RECOGNITION-----

% H.button(3): 'Recognition by HU'
H.button(3) = uicontrol('Style','pushbutton',...
'Units','normalized', ...
'Position',[540/width (height-390)/height 180/width 30/height],...
'FontSize',10,...
'String','Recognition by HU',...
'CallBack','ocr(5)'); %-----Case 5

% H.button(4): 'Recognition by FD'
H.button(4) = uicontrol('Style','pushbutton',...
'Units','normalized', ...
'Position',[540/width (height-340)/height 180/width 30/height],...
'FontSize',10,...
'String','Recognition by FD ',...
'CallBack','ocr(6)'); %-----Case 6

% Text Axe

```

```

H.ax(2) = axes('position',[0.5+155/width (height-570)/height 150/width 150/height],...
             'XTick',[],'YTick',[],'Box','on');

%-----1

case 1 % Read and display an image

H.ax(2) = axes('position',[0.5+155/width (height-570)/height 150/width 150/height],
             'XTick',[],'YTick',[],'Box','on');

[filename,pathname] = uigetfile({'*.tif;*.jpg','Image files'});

if filename ~= 0

    % Clear the old data
    clear global I
    global I

    (axes(H.ax(2)))
    % Read the image and convert to intensity
    [I.Image,map] = imread([pathname filename]);
    cd(pathname)

    if ~isempty(map)
        I.Image = ind2gray(I.Image,map);
        I.Image = gray2ind(I.Image,256)
    else
        if size(I.Image,3)==3
            I.Image = rgb2gray(I.Image);
        end
    end
end

% Resize the first axes accordingly
w_max = 425;
h_max = 540;
[h,w] = size(I.Image);
[im_width,im_height] = fit_im(w_max,h_max,w,h);
left = 20 + (w_max - im_width)/2;
bott = 20 + (h_max - im_height)/2;

% Display the image in the first axes
colormap(gray(256))
axes(H.ax(1))
set(H.ax(1),'position',[left/width bott/height im_width/width...
im_height/height])
image(I.Image)

```

```

        set(H.ax(1),'XTick',[],'YTick',[])

    end

%-----2

case 2 % Collect Hu's 7 moments of the current image

    hcount = hcount + 1;
    bw = im2bw(I.Image, .5);
    HU(hcount,:) = hu(bw);
    H_index(hcount) = s;

%-----3

case 3 % Collect Fourier Descriptors of the current image

    fcount = fcount + 1;
    bw = im2bw(I.Image, .5);
    FD(fcount,:) = getfdi(bw, 64);
    F_index(fcount) = s;

%-----4

case 4 % Save the input letter from the edit box

    s = get(H.edit(1),'String');

%-----5

case 5 % Recognize the image using hu's moments

    wb = im2bw(I.Image,.5);
    H_temp = hu(wb);
    [b c] = size(HU);

    for i = 1:b
        H_dist(i) = edist (H_temp, HU(i,:));
    end

    mt = min(H_dist);
    [s t] = size(H_dist);

    for j = 1:t
        if H_dist(j) == mt;
            htest = j;
        end
    end

```

```

    end
end

if gca ~= H.ax(2)
    axes(H.ax(2))
end
L = char(H_index(htest));
H.text(1) = text(.29, .51, L,'FontSize',76);

%-----6

case 6 % Recognize the image using Fourier Descriptors

wb = im2bw(I.Image,.5);

F_temp = getfdi(wb, 64);

[b c] = size(FD);

for i = 1:b
    F_dist(i) = edist (F_temp, FD(i,:));
end

mf = min(F_dist);
[s t] = size(F_dist);

for j = 1:t
    if F_dist(j) == mf;
        ftest = j;
    end
end

if gca ~= H.ax(2)
    axes(H.ax(2))
end
L = char(F_index(ftest));
H.text(1) = text(.29, .51, L,'FontSize',76);

%-----7

case 7 % Load image feature data from a file

[filename,pathname] = uigetfile({'*.mat','Data files'});

if filename ~= 0

```

```

        % Clear the old data
        clear I stats count
        global H I stats count

        cd(pathname);
        load(filename,'I','HU','FD', 'H_index', 'F_index')

    end

%-----8

case 8 % Save Image and feature data to a file

    [filename,pathname] = uiputfile({'*.mat','Data files'});

    if filename ~= 0

        % Save the stats to a mat file
        cd(pathname);
        save(filename,'I','HU','FD', 'H_index', 'F_index');

    end
end

% Sub-functions

%-----

function [im_width,im_height] = fit_im(w_max,h_max,w,h)
%Resize the image accordingly

    w_ratio = w/w_max;
    h_ratio = h/h_max;

    if (w_ratio > 1) | (h_ratio > 1)
        if w_ratio > h_ratio
            im_width = w_max;
            im_height = h/w_ratio;
        else
            im_height = h_max;
            im_width = w/h_ratio;
        end
    else
        im_width = w;
        im_height = h;
    end
end

```

```

end

%-----

function d = edist(x,y);
% finds the euclidean distance d between two vectors
% d = edist(x,y)

    d = sum((x-y).^2).^0.5;

%-----

function fd = getfdi(img, n);
%
    ie = edge(img, 'zerocross', 0);
    [c, d] = trace_boundary(ie);
    [x, y] = resample( c(1,:), c(2,:), n);
    contour = x*j + y;
    fda = abs(fft(contour));
    normaliz_factor = fda(2);
    fd6 = fda(3:10);
    fd = fd6/normaliz_factor;

```

A.2 getfd

```

%function fd = getfd(img, Nfft);
% get a fourier descriptor given coordinates

% img - binary image
% Nfft - number of coefficients in DFT

ie = edge(img, 'zerocross', 0);
[c, d] = trace_boundary(ie);
[x, y] = resample( c(1,:), c(2,:), n);
contour = x*j + y;
fda = abs(fft(contour));
normaliz_factor = fda(2);
fd6 = fda(3:12);
fd7 = fda(55:64);
fd66 = fd6/normaliz_factor;
fd77 = fd7/normaliz_factor;
fd = [fd66 fd77];

```

A.3 trace_boundary

```

function [coords, code] = trace_boundary(bw, N);
% get the coordinates of the image's boundary pixels

% bw - binary image
% N - connectivity (4 or 8 (default))
% coords - coordinates of border pixels
% code - chaincode

warning off
if nargin < 2, N = 8;
end

if N == 8,
    table = [0 -1; -1 -1; -1 0; -1 1; 0 1; 1 1; 1 0; 1 -1];
elseif N == 4,
    table = [0 -1; -1 0; 0 1; 1 0];
else
    error('connectivity can be either 8 or 4');
end

offset = N/2;          %offset between adjacent pixels

error = 0;             %indicates an error condition
height = size(bw,1);
width = size(bw,2);

%find the starting pixel - 'most up-left'
c = 0;                %column
r = [];               %row

while isempty(r)
    c = c+1;
    r = find(bw(:,c));
end

if c == size(bw,2), error('could not find valid image'); end

r = r(1);             %if there is more than one pixel in a column, get the upper one
start = [r,c];        %set the coordinates of the starting pixel
current = start;
count = 1;            %perimeter length
b = current - [0, 1]; %set the 'previous' pixel to be the one to the left
index = 1;            %chaincode of b is 0, so next possible guess is 1

```

```

while 1 == 1,          %loop until there are no more pixels
    xy = current + table(index+1,:);

    if ( current(1)<2 | current(2)<2 ) | ( current(1) == height | current(2) == width )
        warning('found occluded image, setting to zero');
        error = 1;
        break;
    end

    while bw(xy(1),xy(2)) ~= 1,
        index = mod( index + 1, N );
        xy = current + table(index+1,:);
    end

    coords(1, count) = current(1);
    coords(2, count) = current(2);
    code(count) = mod( index + offset, N);    %record chaincode
    current = current + table(index+1,:); %move to the next pixel
    index = mod( code(count) + 1, N ); %index will equal current chaincode + 1

    if current == start,
        break;
    end
    count = count + 1;
end

if error == 1 | count < 32,
    temp = bwselect(bw, current(2), current(1), N);
    bw = XOR(bw, temp);
    [coords, code] = getchaincode(bw, N);
end

```

A.4 resample

```

function [vector_x, vector_y] = resample(x, y, num)
%resample the boundary pixels' coordinates uniformly to the predefined number

% x - the original X coordinates vector
% y - the original Y coordinates vector
% num - the predefined number: how many pixels to resample

ori_len = length(x);
dis_arr = distance(x,y);

```

```

arc_len = sum(dis_arr);
x = x./arc_len;
y = y./arc_len;
dis_arr = distance(x, y);

vector_x = x(1);
vector_y = y(1);

x = [x, x(1)];
y = [y, y(1)];

cnt = 1;
for i = 2:num
    remaining = 1/num;
    while(1)
        if remaining > dis_arr(cnt)
            remaining = remaining - dis_arr(cnt);
            cnt = cnt + 1;
        else
            next_x = x(cnt+1);
            cur_x = x(cnt);
            next_y = y(cnt+1);
            cur_y = y(cnt);
            d = dis_arr(cnt);

            new_x = cur_x + (next_x-cur_x)/d*remaining;
            new_y = cur_y + (next_y-cur_y)/d*remaining;
            dis_arr(cnt) = d - remaining;
            x(cnt) = new_x;
            y(cnt) = new_y;
            break;
        end
    end

    vector_x = [vector_x, new_x];
    vector_y = [vector_y, new_y];
end
return

function [distance_vector] = distance(x, y)

tmp = x;
tmp(1) = [];
x_shift = [tmp, x(1)];

tmp = y;

```

```

tmp(1) = [];
y_shift = [tmp, y(1)];

distance_vector = sqrt((x-x_shift).^2 + (y-y_shift).^2);
return

```

A.5 moment

```

function [m] = moment(fig, p, q)
% Calculate the regular moment of a binary image with order p, q

% fig - binary image
% p, q - order

fig = double(fig);
[y x] = size(fig);
m = zeros(size(p));

for i = 1:y
    for j = 1:x
        m = m + fig(i,j).*(i.^p).*(j.^q);
    end
end
return

```

A.6 centroid

```

function [cx, cy] = centroid(fig)
% calculate the centroid of a binary image

% fig - binary image
% cx, cy - centroid

m00 = sum(sum(fig));
m = moment(fig, [1 0], [0 1]);

cx = m(1)/m00;
cy = m(2)/m00;
return

```

A.7 central_moment

```
function [m] = central_moment(fig, p, q)
% calculate the regular the central moment of a binary image with order p, q

% fig - binary image
% p, q - order

fig = double(fig);
[ny nx] = size(fig);
[cx cy] = centroid(fig);
m = zeros(size(p));

for i = 1:ny
    for j = 1:nx
        m = m + fig(i,j).*(i - cx).^p.*(j-cy).^q;
    end
end
return
```

A.8 normalized_moment

```
function [m] = normalized_moment(fig, p, q)
% calculate the regular the normalized moment of a binary image with order p, q

% fig - image
% p, q - order

m00 = sum(sum(fig));
mpq = central_moment(fig, p, q);
m = mpq./(m00.^((p+q+2)/2));
return
```

A.9 hu

```
function [m] = hu(fig)
% calculate Hu's seven moment invariants

% fig - binary image
```

```

% m - the vector of Hu's 7 moment invariants

nm = normalized_moment(fig, [2 0 1 3 1 2 0], [0 2 1 0 2 1 3]);

n20 = nm(1);
n02 = nm(2);
n11 = nm(3);
n30 = nm(4);
n12 = nm(5);
n21 = nm(6);
n03 = nm(7);

h1 = n20 + n02;

h2 = sq(n20-n02) + 4*sq(n11);

h3 = sq(n30-3*n12) + sq(3*n21-n03);

h4 = sq(n30+n12) + sq(n21+n03);

h5 = (n30-3*n12)*(n30+n12)*(sq(n30+n12)-3*sq(n21+n03))+(3*n21-
n03)*(n21+n03)*(3*sq(n30+n12)-sq(n21+n03));

h6 = (n20-n02)*(sq(n30+n12)-sq(n21+n03))+4*n11*(n30+n12)*(n21+n03);

h7 = (3*n21-n03)*(n30+n12)*(sq(n30+n12)-3*sq(n21+n03))+(3*n12-
n30)*(n21+n03)*(3*sq(n30+n12)-sq(n21+n03));

m = [h1 h2 h3 h4 h5 h6 h7];

return

```

A.10 bi

```

function I = bi(fig)
% binarize a gray level image

%fig - input gray level image
% I - return binary image

A = imread(fig);
level = graythresh(A);
I = im2bw(A, level);
return

```

A.11 z_zmoment

```

function [A_nm,zmlist,cidx,V_nm] = z_zmoment(img, n, m)
% compute Zernike moments of a binary image

%   img   binary image           (matrix)
%   n     order                   (vector)
%   m     repetition              (integer)

%   A_nm  Zernike moments         (row vector)
%   zmlist keeps order and repetition information (Mx2 matrix)
%         M = total # of moments
%   cidx  1-D indices of retained unit circle region (vector)
%   V_nm  Zernike polynomials size of img          (col vectors)

if nargin>0
%REGULAR
if nargin==1
    n = 0;
end

d = size(img);           % dimension
img = double(img);

%normalize co-ordinates to [-sqrt2,sqrt2] (centroid assumed @ origin)
xstep = 2/(d(1)-1);
ystep = 2/(d(2)-1);
[x,y] = meshgrid(-1:xstep:1,-1:ystep:1);

%compute unit disc mask
circle1 = x.^2 + y.^2;
inside = find(circle1<=1);
mask = zeros(d);
mask(inside) = ones(size(inside));

%mask pixels lying inside/on unit circle
[cimg,cidx] = z_clipimg(img,mask);
z = z_clipimg(x+i*y,mask);
p = abs(z);
theta = angle(z);

%compute Zernike moments
c = 1;

```

```

for order = 1:length(n)
    n1 = n(order);
    if nargin<3
        m = z_zpossible(n1);           % repetition unspecified
    end

    for r = 1:length(m)
        V_nmt = z_zpoly(n1,m(r),p,theta);
        zprod = cimg.*conj(V_nmt);
        A_nm(c) = (n1+1)*sum(sum(zprod))/pi;
        zmlist(c,1:2) = [n1 m(r)];
        if nargin == 4
            V_nm(:,c) = V_nmt;
        end
        c = c+1;
    end
end
end
%REGULAR ends
else
%DEMO
clf;clc;
disp('running zmoment.m in demo mode');

dim = 16;
c = round(dim/2);
l = 4;
r = 3;
gimg = zeros(dim);
gimg(c-l:c+l,c-l:c+l) = ones(2*l+1);
gimg = gimg*255;

gimg(c-r:c+r,c-r:c+r) = zeros(2*r+1);

%plot scale/translation normalized image
clf;
iimg=z_invariant(gimg,'-scale 0 -translation 0');
image(iimg);
colormap gray;
centroid= z_centroid(double(iimg));
xcen = centroid(1);
ycen = centroid(2);
hold on;
plot(xcen,ycen,'r+');
axis square;
title('Original Image');
drawnow;

```

```

%compute Zernike Moments up to an order
order = 0:30;
disp('Computing Zernike moments of order 30');
[A_nm,zmlist,cidx,V_nm] = z_zmoment(iimg,order);

%compute/plot reconstructed image
disp('Reconstructing image from Zernike moments (order 1, 2, ..., 30)');
[reimg] = z_zmrecon(A_nm, V_nm, zmlist, cidx, [dim dim], 50,1);
%DEMO ends
end

```

A.12 z_invariant

```

function [iimg,x,y] = z_invariant(bimg,para)
%center and scale image to fixed mass

%   bimg  binary image
%   para  parameter string                (string)
%   -scale      0 or parameter, usu. m00
%   -translation  0

%   iimg  normalized binary image        (matrix)
%   x     invariant co-ordinates (horizontal) (matrix)
%   y     invariant co-ordinates (vertical) (matrix)

if nargin<2
    para = [];
end

bimg = double(bimg);
beta = z_paraget('l-scale',para);
center = z_paraget('l-translation',para);

[row,col] = size(bimg);
[x,y] = meshgrid(1:row,1:col);

%compute centroid
if center
    thecen = z_centroid(bimg);
else
    thecen = zeros(2,1);
end

```

```

%compute scale for given beta
if beta
    m00 = z_gmoment(bimg,0,0);
    inva = 1/sqrt(beta/m00);
else
    inva = 1;
end

%interpolate using new co-ordinates
if beta | center
    xi = x*inva+thecen(1);
    yi = y*inva+thecen(2);
    iimg = interp2(x+inva*col/2+.5,y+inva*row/2+.5,bimg,xi,yi,'*nearest');

    %eliminate undefined values to zero (black)
    wnan = find(isnan(iimg)==1);
    iimg(wnan) = zeros(size(wnan));
else
    iimg = bimg;
end

```

A.13 z_zmrecon

```

function [reimg] = z_zmrecon(A_nm, V_nm, zmlist, cidx, dim, thold, showi)
% reconstruct image from Zernike Moments

%   A_nm Zernike moments (vector)
%       length of A_nm denotes maximum desired order
%   V_nm Zernike polynomials size of img (vector)
%   zmlist keeps order and repetition information (Mx2 matrix)
%       M = total # of moments
%   cidx 1-D indices of retained unit circle region (vector)
%   dim  original dimension of image (2-vector)
%   thold threshold for discarding pixels 1-255 (scalar)
%       defaults = 128
%   showi show image of intermediate orders (set to 1)

%   rimg  reconstructed image

if nargin<7
    showi = 0;
    if nargin<6
        thold = 128;

```

```

end
end

ring = zeros(size(cidx));

for c=1:length(A_nm)
    n = zmlist(c,1);
    m = zmlist(c,2);
    if n>zmlist(max(c-1,1),1)&showi
        eimg = z_equalize(real(ring));
        if thold>1
            timg = eimg;
            wcut = find(eimg<thold);
            timg(wcut) = zeros(size(wcut));
            wgood = find(timg~=0);
            if ~isempty(wgood)
                timg(wgood) = z_equalize(timg(wgood));
            end
        else
            timg = eimg;
        end

        hold off;
        pimg = zeros(dim);
        pimg(cidx) = timg;
        image(pimg);axis square;
        ts = sprintf('Reconstructed Image using up to Order %d Zernike...
                    Moments',n);
        title(ts);
        axis off;
        drawnow
    else
        pimg = zeros(dim);
    end
    ring = ring+A_nm(c)*V_nm(:,c);
end
reimg = zeros(dim);
reimg(cidx) = timg;

```

References

- [1] Y. S. Abu-Mostafa and D. Psaltis, "Recognitive aspects of moment invariants," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-6, pp.698-706, November 1984.
- [2] Y. S. Abu-Mostafa and D. Psaltis, "Image normalization by complex moments," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No.1, pp.46-55, January 1985.
- [3] K. Arbeter, W. E. Snyder, H. Burkhardt, and G. Hirzinger, "Application of affine-invariant Fourier descriptors to recognition of 3-D objects," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.12, pp 640-647, July 1990.
- [4] A. B. Bhata and E. Wolf, Proc. Camb. Phil. Soc. Vol.50, pp.40-48, 1954.
- [5] S. A. Dudani, K. J. Breeding, and R. B. McGhee, "Aircraft identification by moment invariants," IEEE Transaction on Comput.. Vol.C-26, No.1, pp. 39-45, January 1983.
- [6] J. Flusser and T. Suk, "Degraded image analysis: an invariant approach," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.20, No.6, June 1998.
- [7] H. Freeman, "Computer processing of line drawing images," Computer Survey, Vol.6, pp.57-97, 1974.
- [8] K. S. Fu, Syntactic Pattern Recognition and Application, Prentice Hall, 1982.

- [9] K. S. Fu, R. C. Gonzalez, and G. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, 1987.
- [10] R. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, 2002.
- [11] A. Hero, J. O'Neill, and W. Williams, "Moment matrices for recognition of spatial patten in noise images," *J. Royal Statistical Society, Ser. B*, Vol.51, No.2, pp.271-279, 1989.
- [12] M. K. Hu, "Visual pattern recognition by moment inariants," *IRE Trans. Inform. Theory*, Vol.IT-8, pp.179-187, February 1962.
- [13] S. Impedove, L. Ottaviano, and S. Occhinegro, "Optical character recognition – a survey," *Internal Journal of Pattern Recognition and Artificial Intelligence*, Vol. 5 (1-2), pp.1-24, 1991.
- [14] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw-Hill, 1995.
- [15] H. Kalhalfallah, E. M. Petriu, and F. C. A. Groen, "Visual position recovery for an automatic guided vehicle," *IEEE Transactions on Instrumentation and Measurement*, Vol.41, No.6, pp.906-910, December 1992.
- [16] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.12, No.5, May 1990.
- [17] S. Kuo and G. R. Cross, "A two-step string-matching procedure," *Pattern Recognition*, Vol. 24, No.7, pp.711-716, 1991.

- [18] S. Mori, C. Y. Suen, and K. Yanamoto, "Historical review of OCR research and development," *Proceedings of the IEEE*, Vol.80, pp.1029-1058, July 1992.
- [19] S. Pakchalakis and P. Lee, "Pattern recognition in gray level images using moment based invariant features," *Image Processing and its Applications, IEE Conference Publication No.465*, pp.245-249, 1999.
- [20] E. A. Patrick, *Fundamentals of Pattern Recognition*, Prentice Hall, 1972.
- [21] T. Pavdilis and Z. Jiangying, "Page segmentation and classification," *CVGIP: Graphical models and image processing*, Vol.54, No.6, 1992.
- [22] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Transactions on Systems, Man Cybernetics*, Vol.7, No.2, pp.170-179, 1977.
- [23] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Vision*, Vol.22, No.1, January 2002.
- [24] A. P. Reeves, R. J. Prokop, S. E. Andrews and F. Kuhl, "Three-dimensional shape analysis using moments and Fourier descriptors," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.10, No.6, pp.937-943, November 1988.
- [25] T. Reiss, *Recognizing Planar Objects Using Invariant Image Features*, Springer-Verlag, 1993.
- [26] Y. Rui, A. C. She, and T. S. Huang, "Modified Fourier descriptors for shape representation – a practical approach," <http://citeseer.nj.nec.com/296923.html>.

- [27] R. Sivaramakrishna and N. S. Shashidhar, "Hu's moment invariants: how invariant are they under skew and perspective transformations," IEEE Conference on Communications, Power and Computing, pp.292-295, 1997.
- [28] T. Suk and J. Flusser, "Blur and affine moment invariants," ICPR2002 16th International Conference on Pattern Recognition, Vol.4, pp.339–342, 2002.
- [29] C. C. Tappert, C. Y. Suen, and T. Wakahara, "State of the art in online hand-writing recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.12, No.8, August 1990.
- [30] G. Tauschek, "Reading machine," U.S. Patent 2026329, December 1935.
- [31] M.R.Teague, "Image analysis via the general theory of moments," J. Opt. Soc. Amer. 70, pp.920-930, August 1980.
- [32] C-H Teh and R. T. Chin, "On image analysis by the methods of moments," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.10, No.4, July 1988.
- [33] O. D. Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition – a survey, " Pattern Recognition, Vol.29, No.4, pp 641-662, 1996.
- [34] E. Trucco and A. Verri, Introductory Techniques for 3-D Computer Vision, Prentice Hall, 1998.
- [35] T. P. Wallace and P. A. Wintz, "An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors," Computer Graphics and Image Processing, 13, pp.99-126, 1980.

- [36] S. K. Yeung, W. S. McMath, E. M. Petriu and N. Trif, "Three dimensional object recognition using integrated robotic vision and tactile sensing," IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS'91, pp.1370-1373, Osaka, Japan, November 3-5, 1991.
- [37] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," IEEE Transaction on Computer, C-21 (1), pp.269-281, 1972.
- [38] F. Zernike, Physica, Vol.1, pp.689, 1934.
- [39] D. Zhang and G. Lu, "A comparative study of three region shape descriptors," DICTA 2002: Digital Image Computing Techniques and Applications, pp21-22, Melbourne, Australia, January 2002.
- [40] D. Zhang and G. Lu, "A comparative study on shape retrieval using Fourier descriptors with different shape signatures,"

<http://www.gscit.monash.edu.au/~dengs/resource/publications.html>.
- [41] "Code: a system in which arbitrary values are given to letters, words, numbers or symbols to ensure secrecy or brevity,"

<http://homepages.cwi.nl/~dik/english/codes/charrec.html>.