# INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

# Handwritten Numeral Recognition Using Multiwavelets

Yueting Chen

A Major Report

In

The Department

Of

Computer Science

Presented in Partial Fulfilment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montreal, Quebec, Canada

August 2002

# ABSTRACT

Handwritten Numeral Recognition Using Multiwavelets

Yueting Chen

In this report, we review different techniques for handwritten numeral recognition. More importantly we develop and test a hand-written numeral recognition system using multiwavelets. Given a black-and-white numeral, we first trace the contour of the numeral. Secondly we normalize and resample the contour points. Thirdly we perform multiwavelet orthonormal shell expansion on the contour points and we get several resolution levels and the average. We use the multiwavelet coefficients as the features to recognize the hand-written numerals. We use the L1 distance as a measure and the nearest neighbour rule as classifier for the recognition. The experimental result shows that it is a feasible way to use multi-wavelet features in handwritten numeral recognition.

Table of Contents

# 1. Introduction

The recognition of handwritten numerals is an important problem in optical character recognition (OCR) with applications such as automatic ZIP-code reading or understanding annotations in technical drawings. Feature or descriptor extraction is probably the most important factor in achieving high recognition performance. We should use features that have small intra-class variance and large interclass separation, i.e. features derived from different samples of the same pattern class should be close and features derived from samples of different classes should differ significantly. One of the difficulties specific to handwriting recognition is that it has to deal with large intra-class variance due to the shape variations caused by the different writing styles of people. Since it is impossible to find features that are invariant with respect to writing style, one can only aim at finding features that experimentally prove reasonably insensitive to shape distortions caused by individual writing style and that at the same time maintain the ability to separate samples of different classes.

Experience shows that shapes that "look alike" have similar low-frequency components in the Fourier domain. Low-frequency components reflect the basic shape of a function, where as the high-frequency components represent the details. It is observed that low-frequency components of a character are less sensitive to writing style variations. This leads to the popularity of Fourier descriptors for handwritten character recognition. [1-4] However, Fourier descriptors are inherently global. The frequency information obtained from a Fourier transform is global. Intuitively, a more localized frequency representation should be more effective for handwritten character recognition. Wavelet transforms can

have variable time-frequency localization and orthonormal wavelets with finite support provide a powerful mathematical tool for decomposing a function into a multiresolution hierarchy of different localized frequency channels.

In this report, we present a novel set of shape descriptors that represents a digitized pattern in a very concise way and that is particularly well-suited for the recognition of handwritten numerals. The descriptor is derived from the multiwavelet shell expansion of an object's contour. The motivation to use multiwavelet basis is twofold. First, multiwavelets provide a localized frequency representation, which can reflect local properities much better than Fourier based method. Secondly and more importantly, orthonormal multiwavelets provide a natural hierarchical multiresolution representation, and there is substantial evidence that the human visual system use similar mutiscale representations. It should be mentioned that this work is in parallel with the work in [9].

The report is organized as follows. In Section 2, a background review is given on OCR - the process and various feature extraction methods, with emphasis on features using contours. Sections 3 and 4 introduce the foundations of our feature extraction method – discrete multiwavelet transform and the orthonormal shell expansion, respectively. Our new shape descriptors are presented in Section 5. The experimental results, using our shape descriptors and using nearest neighbour classifier, are provided in Section 6, and a conclusion is given in Section 7. Programming is considered as an important work of the report. Programs are listed as appendix. They are written in MatLab.

# 2. Background Review

Optical character recognition (OCR) is one of the most successful applications of automatic pattern recognition and has been a very active field for research and development since the mid 1950s. An OCR system is easily available to general public today. A home scanner under $100 may include an OCR package. But the less expensive OCR system can only recognize high quality printed text documents or neatly written handprinted text. The challenges for a good OCR system are to be able to recognize severely degraded printed text or unconstrained handwritten text with high recognition rate and low substitution/reject error rate.

An OCR system typically consists of the following steps (Figure 1):

1. Gray-level scanning at an appropriate resolution, typically 300-1000 dots per inch.
2. Preprocessing:
    - Binarization (two-level thresholding), using a global or a locally adaptive method.
    - Segmentation to isolate individual characters.
    - (Optional) Conversion to another character representation (e.g. skeleton or contour curve).
3. Feature extraction.
4. Recognition using one or more classifiers.
5. Contextual verification or postprocessing.

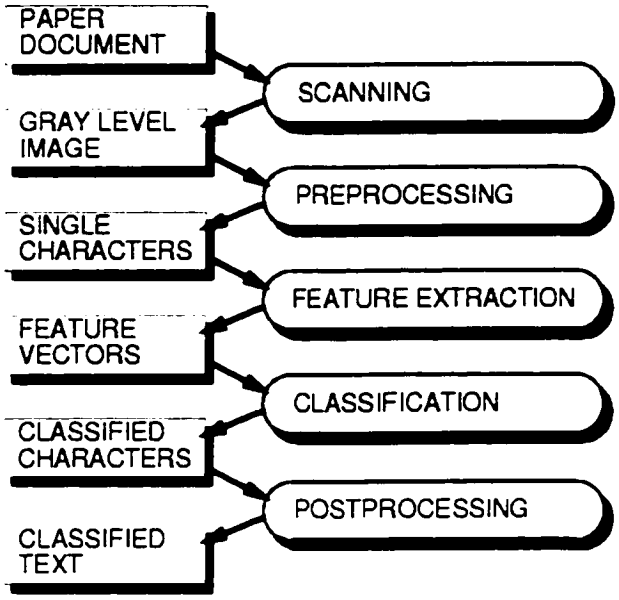


**Figure 1. Steps in a character recognition system.**

## 2.1. Feature Extraction in OCR

Feature extraction is an important step in achieving good performance of OCR systems. However, the other steps in the system also need to be optimized to obtain the best possible performance and these steps are not independent. The choice of feature extraction method limits or dictates the nature and output of the preprocessing step. Some feature extraction methods work on gray-level subimages of single characters (Figure 2), while others work on solid four- or eight-connected symbols segmented from the binary raster image (Figure 3), thinned symbols or skeletons (Figure 4), or symbol contours (Figure 5). Further, the type or format of the extracted features must match the requirements of the chosen classifier. Graph descriptions or grammar-based descriptions of the characters are well suited for structural or syntactic classifiers. Discrete features that may assume only 2 or 3 distinct values are ideal for decision trees. Real-valued

feature vectors are ideal for statistical classifiers. However, multiple classifiers may be used, either as a multi-stage classification scheme or as parallel classifiers, where a combination of the individual classification results is used to decide the final classification. In that case, features of more than one type or format may be extracted from the input characters.

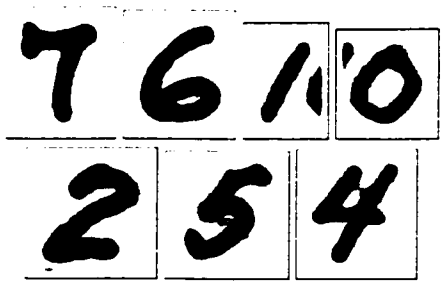

**Figure 2. Gray-scale subimages of segmented characters.**

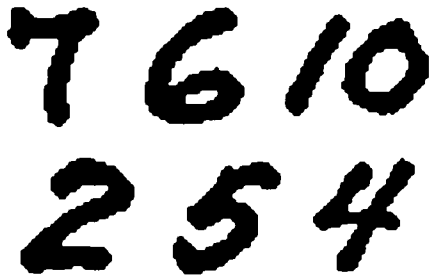

**Figure 3. Digits from the hydrographic map in the binary raster representation.**

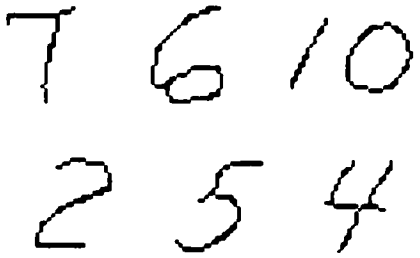

**Figure 4. Skeletons of digits (after thinning) in Figure 3.**

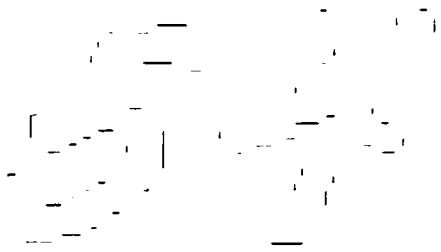

**Figure 5. Contours of two of the digits in Figure 3.**

In their extensive survey paper on feature extraction (reference [6]), Ø. D. Trier et al reviewed the following feature extraction methods (except the last one):

- Template matching;
- Deformable templates;
- Unitary image transforms;
- Graph descriptions;

- Projection histograms;
- Contour profiles ;
- Zoning;
- Geometric moment invariants ;
- Zernike moments;
- Spline curve approximation;
- Fourier descriptors;
- Wavelet descriptors.

Each of these methods may be applied to one or more of the following representation forms:

(1) Gray-level character image;

(2) Binary character image;

(3) Character contour;

(4) Character skeleton or character graph.

For each feature extraction method (with exception of Wavelet descriptors) and each character representation form above, they discussed the properties of the extracted features.

## 2.2. Feature Extraction Using Contour Information

As mentioned above, there are more than 10 types of feature extraction methods using contour information. But our interest is in developing a set of wavelet descriptors, which in turn are closely related to Fourier descriptors. Therefore only these two types of feature extraction methods are reviewed here.

### 2.2.1 Fourier descriptors

Zahn and Roskies [1] defined a normalized cumulative angular function $\phi^*(t)$ for a simple closed curve $\gamma$ and expand $\phi^*$ in a Fourier series to obtain descriptive coefficients (Fourier descriptors). A formula for reconstructing a curve from its Fourier descriptors and the explicit formulas for calculating the Fourier descriptors of a polygonal curve were given.

In Zahn and Roskies's method, the angular difference $\Delta\varphi$ between two successive line segments on the contour is measured at every pixel center along the contour. The contour is followed clockwise. Then the following descriptors can be extracted:

$$a_n = -\frac{1}{n\pi}\sum_{k=1}^{m}\Delta\varphi_k \sin\frac{2\pi n l_k}{L}, \tag{1}$$

and

$$b_n = \frac{1}{n\pi}\sum_{k=1}^{m}\Delta\varphi_k \cos\frac{2\pi n l_k}{L}, \tag{2}$$

where $L$ is the length of the boundary curve, consisting $m$ line segments, $l_k$ is the accumulated length of the boundary from starting point $p_1$ to the $k$th point $p_k$ and $\Delta\varphi_k$ is the angle between the vectors $[p_{k-1}, p_k]$ and $[p_k, p_{k+1}]$, $a_n$ and $b_n$ are size- and translation-invariant. Rotation invariance can be obtained by transforming to polar coordinates. Then the amplitudes:

$$A_n = \sqrt{a_n^2 + b_n^2} \tag{3}$$

are independent of rotation and mirroring, while the phase angles $\alpha_n = \tan(a_n/b_n)$ are not. However, mirroring can be detected via the $\alpha_j$ s. It can be shown that:

$$F_{kj} = j^{*}\alpha_k - k^{*}\alpha_j \tag{4}$$

is independent of rotation, but dependent on mirroring. Here, $j^{*} = j/\gcd(j,k)$, $k^{*} = k/\gcd(j,k)$ and $\gcd(j,k)$ is the greatest common divisor of $j$ and $k$.

Zahn and Roskies warn that $\alpha_k$ becomes unreliable as $A_k \to 0$ and is totally undefined when $A_k = 0$. Therefore, the $F_{kj}$ terms may be unreliable.

Granlund [2] described a pattern-recognition method using Fourier transformations to extract features which are significant for a pattern (contour of a character). He claims that the ordinary Fourier coefficients are difficult to use as input to categorizers because they contain factors dependent upon size and rotation as well as an arbitrary phase angle. But other more useful features can be easily derived from these Fourier coefficients. By using these derived property constants, a distinction can be made between genuine shape constants and constants representing size, location, and orientation.

Granlund uses a complex number $z(t) = x(t) + jy(t)$ to denote the points on the contour. Then the contour can be expressed as a Fourier series:

$$z(t) = \sum_{n=-\infty}^{\infty} a_n e^{j2\pi nt/T}, \tag{5}$$

where

$$a_n = \frac{1}{T}\int_0^T z(t) e^{-j2\pi nt/T} dt \tag{6}$$

are the complex coefficients. $a_0$ is the center of gravity and the other coefficients $a_n$, $n \neq 0$ are independent of translation. $T$ is the total contour length. The derived features

$$b_n = \frac{a_{1+n}a_{1-n}}{a_1^2}, \tag{7}$$

and

$$D_{mn} = \frac{a_{1+m}^{n\,k}a_{1-n}^{m\,k}}{a_1^{(m+n)\,k}}, \tag{8}$$

are independent of scale and rotation. Here, $n \neq 1$ and $k = \gcd(m,n)$ is the greatest common divisor of $m$ and $n$. Furthermore:

$$b_1^* = \frac{a_2|a_1|}{a_1^2}, \tag{9}$$

and

$$a_{m1}^* = \frac{d_{1+m}|a_1|^m}{a_1^{m+1}}, \tag{10}$$

are scale-independent, but depend on rotation, so they can be useful when orientation of the characters is known.

In Fuhl and Giardina [3], a classification and recognition procedure is presented that can be applied directly to classes of objects that cannot change shape and whose images are not subject to sensory-equipment distortions. The features used in the procedure are normalized Fourier coefficients derived from chain codes of the image contours. The normalization is performed according to various elliptic properties of the Fourier coefficients themselves.

In Fuhl and Giardina's approach, the closed contour, $(x(t), y(t)), t = 1,\ldots,m$, is approximated as:

$$\hat{x}(t) = A_0 + \sum_{n=1}^{N}\left[a_n \cos\frac{2n\pi t}{T} + b_n \sin\frac{2n\pi t}{T}\right], \tag{11}$$

$$\hat{y}(t) = C_0 + \sum_{n=1}^{N}\left[c_n \cos\frac{2n\pi t}{T} + d_n \sin\frac{2n\pi t}{T}\right], \tag{12}$$

where $T$ is the total contour length and with $\hat{x}(t) \equiv x(t)$ and $\hat{y}(t) \equiv y(t)$ in the limit when $N \to \infty$. The coefficients are:

$$A_0 = \frac{1}{T}\int_0^T x(t)\mathrm{d}t, \tag{13}$$

$$C_0 = \frac{1}{T}\int_0^T y(t)\mathrm{d}t, \tag{14}$$

$$a_n = \frac{2}{T}\int_0^T x(t)\cos\frac{2n\pi t}{T}\mathrm{d}t. \tag{15}$$

$$b_n = \frac{2}{T}\int_0^T x(t)\sin\frac{2n\pi t}{T}\mathrm{d}t, \tag{16}$$

$$c_n = \frac{2}{T}\int_0^T y(t)\cos\frac{2n\pi t}{T}\mathrm{d}t, \tag{17}$$

and

$$d_n = \frac{2}{T}\int_0^T y(t)\sin\frac{2n\pi t}{T}\mathrm{d}t. \tag{18}$$

The function $x(t)$ and $y(t)$ are piecewise linear and the coefficients can be obtained by summation instead of integration. It can be shown that the coefficients $a_n$, $b_n$, $c_n$ and $d_n$, which are the extracted features, can be expressed as:

$$a_n = \frac{T}{2n^2\pi^2}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left[\cos\phi_i - \cos\phi_{i-1}\right], \tag{19}$$

$$b_n = \frac{T}{2n^2\pi^2}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left[\sin\phi_i - \sin\phi_{i-1}\right], \tag{20}$$

$$c_n = \frac{T}{2n^2\pi^2}\sum_{i=1}^{m}\frac{\Delta y_i}{\Delta t_i}\left[\cos\phi_i - \cos\phi_{i-1}\right], \tag{21}$$

and

$$d_n = \frac{T}{2n^2\pi^2}\sum_{i=1}^{m}\frac{\Delta y_i}{\Delta t_i}\left[\sin\phi_i - \sin\phi_{i-1}\right], \tag{22}$$

where $\phi_i = 2n\pi t_i/T$, $\Delta x_i = x_i - x_{i-1}$, $\Delta y_i = y_i - y_{i-1}$, $\Delta t_i = \sqrt{\Delta x_i^2 + \Delta y_i^2}$, $t_i = \sum_{j=1}^{i}\Delta t_j$, $T = t_m = \sum_{j=1}^{m}\Delta t_j$, and $m$ is the number of pixels along the boundary. The starting point $(x_1, y_1)$ can be arbitrarily chosen and it is clear from equations (15)-(18) that the coefficients are dependent on this choice. To obtain features that are independent of the particular starting point, we calculate the phase shift from the first major axis as:

$$\theta_1 = \frac{1}{2}\tan^{-1}\frac{2(a_1b_1 + c_1d_1)}{\sqrt{a_1^2 + b_1^2 + c_1^2 + d_1^2}}. \tag{23}$$

Then, the coefficients can be rotated to achieve a zero phase shift:

$$\begin{bmatrix} a_n^* & b_n^* \\ c_n^* & d_n^* \end{bmatrix} = \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix}\begin{bmatrix} \cos n\theta_1 & -\sin n\theta_1 \\ \sin n\theta_1 & \cos n\theta_1 \end{bmatrix}. \tag{24}$$

To obtain rotation invariant descriptors, the rotation, $\psi_1$, of the semi-major axis can be found by:

$$\psi_1 = \tan^{-1}\frac{c_1^{\bullet}}{a_1^{\bullet}} \tag{25}$$

and the descriptors can then be rotated by $-\psi_1$, so that the semi-major axis is parallel with the x-axis:

$$\begin{bmatrix} a_n^{\bullet\bullet} & b_n^{\bullet\bullet} \\ c_n^{\bullet\bullet} & d_n^{\bullet\bullet} \end{bmatrix} = \begin{bmatrix} \cos\psi_1 & \sin\psi_1 \\ -\sin\psi_1 & \cos\psi_1 \end{bmatrix} \begin{bmatrix} a_n^{\bullet} & b_n^{\bullet} \\ c_n^{\bullet} & d_n^{\bullet} \end{bmatrix}. \tag{26}$$

This rotation gives $b_1^{\bullet\bullet} = c_1^{\bullet\bullet} = 0.0$, so these coefficients should not be used as features. Further, both these rotations are ambiguous, as $\theta$ and $\theta + \pi$ give the same axes, as do $\psi$ and $\psi + \pi$.

To obtain size-invariant features, the coefficients can be divided by the magnitude, $E$, of the semi-major axis, given by:

$$E = \sqrt{a_1^{\bullet 2} + c_1^{\bullet 2}} = a_1^{\bullet\bullet}. \tag{27}$$

Then $a_1^{\bullet\bullet}$ should not be used as a feature as well. In any case, the low-order coefficients that are available contain the most information (about the character shape) and should always be used.

Lin and Hwang [4] derived rotation-invariant features based on Kual and Giarddina's features:

$$I_k = a_k^2 + b_k^2 + c_k^2 + d_k^2, \tag{28}$$

$$J_k = a_k d_k - b_k c_k, \tag{29}$$

and

$$K_{j,k} = (a_j^2 + b_j^2)(a_k^2 + b_k^2) + (c_j^2 + d_j^2)(c_k^2 + d_k^2) + 2(a_j c_j + b_j d_j)(a_k c_k + b_k d_k). \quad (30)$$

As above, a scaling factor may be used to obtain size-invariant features.

### 2.2.2 Wavelet Descriptors

In Wunsch and Laine [5], a set of shape descriptors derived from the wavelet transform of a pattern's contour is presented. The approach is closely related to feature extraction methods by Fourier series expansion. The motivation to use orthonormal wavelet basis rather than the Fourier basis is that wavelet coefficients provide localized frequency information, and that wavelets make it possible to decompose a function into multiresolution hierarchy of localized frequency bands. They describe a character recognition system that relies upon wavelet descriptors to simultaneously analyze character shape at multiple levels of resolution.

A pattern contour can be represented as a closed parametric curve $c$ in the complex plane C, i.e.

$$c(t) = x(t) + jy(t), \qquad 0 \le t \le T, \quad (31)$$

where $j$ denotes the imaginary unit. The wavelet transform of the curve $c$ can be taken independently for each component

$$WTc(u) = WTx(u) + jWTy(u). \quad (32)$$

The continuous wavelet transform (CWT) of a function $f \in L^2(R)$ is defined as

$$Wf(a,b) = \int_{-\infty}^{\infty} f(t)\psi_{a,b}\,dt = < f, \psi_{a,b} >, \quad (33)$$

where wavelet basis $\psi_{a,b}$ is given by shifted and dilated version of a basic wavelet $\psi$

$$\psi_{a,b} = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right). \tag{34}$$

Hence the CWT decomposes a function $f$ using a family of functions that are dilations and translations of some basic wavelet $\psi$.

Orthonormal wavelets enable multiresolution decompositions. The concept of multiresolution analysis is mathematically formalized as a nested sequence of subspaces $V_j \in L^2(\mathrm{R})$. A function $f \in L^2(\mathrm{R})$ is represented as a limit of successive approximations, each of which is computed by projecting f onto some $V_j$. The sequence of subspaces $V_j$ must satisfy the following properties:

- Containment: $V_{j-1} \subset V_j, \forall j \in \mathbf{Z}$ i.e. sequence $\{V_j\}$ is nested. This implies that a function approximated at resolution $2^j$ contains all the information of its lower resolution approximations.
- Completeness: $\cap_{m\in Z} V_m = 0, \cup_{m\in Z} V_m = L^2(R)$. This implies that if the resolution is increased to $\infty$, the approximation converges to the original function, whereas the approximated function converges to zero as the resolution approaches zero.
- Scaling property: $f(x) \in V_{j-1} \Leftrightarrow f(2x) \in V_j$. This property states that the approximated functions are derived by scaling each other by the ratio of their resolution levels.

Given such a sequence of subspaces, the approximation of a function $f$ at resolution $2^j$ is defined as the projection of $f$ onto $V_j$. There exists a unique function $\phi \in L^2(R)$, called a

scaling function, whose translations at scale $2^j$ form an orthonormal basis of $V_{2^j}$. In general $\phi$ is a lowpass filter and the multiresolution approximation of a function $f$ is sequence of smoothed versions of $f$. Let $W_{2^j}$ be the orthogonal complement of $V_{2^j}$ in $V_{2^{j-1}}$. Each scaling function has an associated wavelet function $\psi \in L^2(R)$ whose dilations and translations provide a set of orthonormal bases of $W_{2^j}$. Each $f_{2^j} \in V_{2^j}$ can then be decomposed by

$$f_{2^j} = g_{2^{j-1}} + g_{2^{j-2}} + \cdots + g_{2^{j-k}} + f_{2^{j-k}}, \tag{35}$$

where $g_{2^j} \in W_{2^j}$ and $f_{2^j} \in V_{2^j}$. The dilations of $\psi$ can be regarded as bandpass filters and the coefficient sequences of the $g_{2^j}$ provide localized spectral information of $f$ within the frequency bands $\left[2^j(\mu_\psi - \sigma_\psi); 2^j(\mu_\psi + \sigma_\psi)\right]$. The coefficients of the $g_{2^j}$ can hence be interpreted as high-frequency details that distinguish the approximation of $f$ at two subsequent levels of resolution. On the other hand, $f_{2^{j-1}}$ represents a coarser approximation of $f_{2^j}$.

Let $A_{2^j} f$ denote the operator that computes the approximation of $f$ at resolution $2^j$, i.e. that projects $f$ onto $V_{2^j}$. Let $D_{2^j}$ be the operator that computes the projection of $f$ onto the sunspaces $W_{2^j}$. If $f$ is a discrete function, the decompositions described above can be achieved by successive convolutions of $f$ with discrete filters, followed by resampling the approximated function by a coarser grid.

The resulting set of coefficients

$$WT_{2^J}\{f\} = (A_{2^J} f, (D_{2^j} f)_{-J \le j \le -1}) \tag{36}$$

forms a pyramidal structure and is called a multiresolution representation of the discrete function $f$ ( $A_0 f = f$ ). If the original function has N samples, then $A_j f$ and $D_j f$ shall have $2^j N$ samples. Thus the above wavelet transform has the same total number of samples as the original function, i.e. the representation is non-redundant.

Experience shows that shapes that "look alike" have similar low-frequency components in the Fourier domain. Low-frequency Fourier coefficients reflect the basic shape of a function, whereas the high frequency components represent the details. As far as handprinted character recognition is concerned, low-frequency components of a character turn out to be less sensitive to writing style variations. The frequency information obtained from a Fourier transform is global. Intuitively, a more localized frequency representation should be more effective for handprinted character recognition. The above wavelet transform have variable time-frequency localization and decompose a function into a multiresolution hierarchy of different localized frequency channels. Such a decomposition allows to simultaneously analyse a function at several levels of resolution and favors coarse-to-fine recognition strategies similar to those known to exist in the human visual system.

Wunsch and Laine [5] 's results show:

- wavelet descriptors are a compact representation for digitized characters that contain sufficient shape information to allow for reliable recognition;
- wavelet descriptors are insensitive to individual writing style variations;

- although closely related to Fourier descriptors, wavelet descriptors are a better shape representation for handprinted characters than a complete Fourier descriptor set of the same dimensionality;
- Multiresolution recognition is a powerful methodology to increase recognition reliability. In contrast to most single scale techniques, it enable the system to reject patterns which could not be unambiguously classified, and thus considerably reduce the rate of substitution errors observed;
- although the multiresolution recognition scheme reduced the error rate at the expense of rejections, the rate of correct classifications remains satisfactory.

## 3. Discrete Multiwavelet Transform

Multiwavelet transform plays an important role in many applications. In this section, we will give a short introduction to multiwavelet transform. Multiwavelets are generalization of single wavelets. Multiwavelet basis uses translations and dilations of $M \geq 2$ scaling functions $\{\phi_k(x)\}_{1 \leq k \leq M}$ and M mother wavelet functions $\{\psi_k(x)\}_{1 \leq k \leq M}$. Let $\Phi(x) = (\phi_1(x), \phi_2(x), \ldots \phi_M(x))^T$ and $\Psi(x) = (\psi_1(x), \psi_2(x), \ldots \psi_M(x))^T$, then we have

$$\Phi(x) = 2\sum_{k=0}^{L-1} H_k \Phi(2x - k), \tag{37}$$

and

$$\Psi(x) = 2\sum_{k=0}^{L-1} G_k \Phi(2x - k), \tag{38}$$

where $\{H_k\}_{0 \leq k \leq L-1}$ and $\{G_k\}_{0 \leq k \leq L-1}$ are $M \times M$ filter matrices. The scaling functions $\phi_i(x)$ and associated wavelets $\psi_i(x)$ are constructed so that all the integer translations of $\phi_i(x)$

are orthogonal, and the integer translations and the dilations of factor 2 of $\psi_l(x)$ form an orthonormal basis for $L^2(R)$.

As an example, for $M = 2$, $L = 4$, we give the most commonly used multiwavelets developed by Geronimo, Hardin and Massopust [7]. Let

$$H_0 = \begin{pmatrix} 3/10 & 2\sqrt{2}/5 \\ -\sqrt{2}/40 & -3/20 \end{pmatrix}, \quad H_1 = \begin{pmatrix} 3/10 & 0 \\ 9\sqrt{2}/40 & 1/2 \end{pmatrix},$$

$$H_2 = \begin{pmatrix} 0 & 0 \\ 9\sqrt{2}/40 & -3/20 \end{pmatrix}, \quad H_3 = \begin{pmatrix} 0 & 0 \\ -\sqrt{2}/40 & 0 \end{pmatrix},$$

and

$$G_0 = \begin{pmatrix} -\sqrt{2}/40 & -3/20 \\ -1/20 & -3\sqrt{2}/20 \end{pmatrix}, \quad G_1 = \begin{pmatrix} 9\sqrt{2}/40 & -1/2 \\ 9/20 & 0 \end{pmatrix},$$

$$G_2 = \begin{pmatrix} 9\sqrt{2}/40 & -3/20 \\ -9/20 & 3\sqrt{2}/20 \end{pmatrix}, \quad G_3 = \begin{pmatrix} -\sqrt{2}/40 & 0 \\ 1/20 & 0 \end{pmatrix},$$

then the two functions $\phi_1(x)$ and $\phi_2(x)$ can be generated via (1). Similarly, the two mother wavelet functions $\psi_1(x)$ and $\psi_2(x)$ can be constructed by (2). Let $V_J$ be the closure of the linear span of $2^{J/2}\phi_l(2^J x - k)$, $l = 1, 2$; $k \in Z$. With the above constructions, it has been proved that $\phi_l(x - k)$, $l = 1, 2$; $k \in Z$ form an orthonormal basis for $V_0$, and moreover the dilations and translations $2^{j/2}\psi_l(2^j x - k)$, $l = 1, 2$; $j, k \in Z$ form an orthonormal basis for $L^2(R)$. In other words, the spaces $V_j$, $j \in Z$, form an orthogonal multiresolution analysis of $L^2(R)$. The two scaling functions $\phi_1(x)$ and $\phi_2(x)$ are supported in [0, 1] and [0, 2], respectively. They are also symmetric and Lipschitz continuous. This is impossible to achieve for single orthogonal wavelets.

## 4. The Orthonormal Shell Expansion

This section reviews briefly the concept of orthonormal shell developed in reference [8]. It is well known that the coefficients of orthogonal wavelet expansions are not shift-invariant. However, if all the wavelet coefficients of $n$ circulant shifts of a vector (signal) are computed, we may use them when shift invariance is important. Based on this observation, the notion of a shell (much more redundant than a frame) was introduced in reference [8] to obtain a redundant but shift-invariant family of functions.

In our applications, there is always the finest and the coarsest scales of interest and therefore the number of scales in finite and we can consider only shifts by multiples of certain fixed unit. Assuming that the finest scale is described by the $n$-dimensional subspace $V_0$ and consider only circulant shifts in $V_0$. Let $V_{j_0}$ be the subspace describing the coarsest scale ($1 \le j_0 \le J$) where $n = 2^J$, and let $\psi_{j,k}(x) = 2^{-\frac{j}{2}}\psi(2^j(x-k))$ and $\phi_{j,k}(x) = 2^{-\frac{j}{2}}\phi(2^{-j}(x-k))$. Therefore, the functions $\{\psi_{j,k}\}_{1 \le j \le j_0, 0 \le k \le 2^{J-j}-1}$ and $\{\phi_{j_0,k}\}_{0 \le k \le 2^{J-j_0}-1}$ generate the coefficients $s_k^j$ and $d_k^j$:

$$s_k^j = \int f(x)\phi_{j,k}(x)dx, \tag{39}$$

and

$$d_k^j = \int f(x)\psi_{j,k}(x)dx, \tag{40}$$

for $j = 1, 2, \ldots, j_0$ and $k = 0, 1, \ldots, 2^{J-j} - 1$. The coefficients $\{d_k^j\}_{1 \le j \le j_0, 0 \le k \le 2^{J-j}-1}$ are known as orthonormal wavelet coefficients.

In the case of orthonormal shell decomposition, Saito and Beylkin define the functions $\{\psi_{j,k}(x)\}_{1 \le j \le j_0, 0 \le k \le 2^J - 1}$ and $\{\phi_{j_0,k}(x)\}_{0 \le k \le 2^J - 1}$ as a shell of the orthonormal wavelets for shifts in $V_0$. As a consequence, the coefficients $\{d_k^j\}_{1 \le j \le j_0, 0 \le k \le 2^J - 1}$ and $\{s_k^{j_0}\}_{0 \le k \le 2^J - 1}$ are called the orthonormal shell coefficients. Clearly the set of coefficients in the orthonormal shell is much more abundant and overly redundant compared to the set of coefficients in the orthogonal wavelet transform. However, this redundancy is needed for our shift invariant property.

Assuming that the orthonormal wavelet coefficients of the finest scale $\{s_k^0\}_{0 \le k \le n-1}$ are given as an original signal and let us consider the function. The orthonormal shell coefficients of this function f are obtained from the quadrature mirror filter $H = \{h_l\}_{0 \le l \le L-1}$ and $G = \{g_l\}_{0 \le l \le L-1}$ (associated with the orthonormal basis of compactly supported wavelets)

$$s_k^j = \sum_{l=0}^{L-1} h_l s_{k+2^{j-1}l}^{j-1}, \tag{41}$$

and

$$d_k^j = \sum_{l=0}^{L-1} g_l s_{k+2^{j-1}l}^{j-1}, \tag{42}$$

for $j = 1, \ldots, j_0$, k = 0, …, $2^{J-j} - 1$. The complexity of (5) and (6) is $O(n \log n)$.

It is easy to show that the recurrence relations (41) and (42) compute the orthonormal wavelet coefficients of all circulant shifts of the function $f$. For $d_k^j$, the first scale is:

$$d_k^1 = \sum_{l=0}^{L-1} g_l s_{k-l}^0 = \sum_{l=0}^{L-1} g_l s_{2k-l}^0 + \sum_{l=0}^{L-1} g_l s_{2k-1-l}^0 = d_{2k}^1 + d_{2k-1}^1. \qquad (43)$$

for k = 0, .... $\frac{n}{2} - 1$.

It is clear that the sequence $\{ d_{2k}^1 \}$ contains all the orthonormal wavelet coefficients that appear if $f(x)$ is circularly shifted by even numbers and the sequence $\{ d_{2k-1}^1 \}$ contains all the orthonormal wavelet coefficients for odd shifts.

Similarly, at the $j$-th scale

$$s_{2^j k+m}^j = \sum_{l=0}^{L-1} h_l s_{2^{j-1}(2k+l)+m}^{j-1}, \qquad (44)$$

and

$$d_{2^j k+m}^j = \sum_{l=0}^{L-1} g_l s_{2^{j-1}(2k+l)+m}^{j-1}, \qquad (45)$$

for k = 0, ..., $2^{J-j}$, $m = 1, ..., 2^j - 1$.

The sequences $\{ d_{2^j k}^j \}$, $\{ d_{2^j k+1}^j \}$, ..., $\{ d_{2^j k+2^j-1}^j \}$ contain the orthonormal wavelet coefficients of the $j$-th scale of the signal shifted by 0, ..., $2^j - 1$, respectively. Therefore, the set $\{ d_k^j \}_{1 \le j \le J_0, 0 \le k \le 2^{J-j}-1}$ and $\{ s_k^{J_0} \}_{0 \le k \le 2^{J-J_0}-1}$ contains all the coefficients of the orthonormal wavelet expansion $f(x)$, $f(x+1)$, ..., $f(x+n-1)$. This set of coefficients defines the orthonormal shell decomposition. Figure 8 illustrates how these coefficients are obtained.

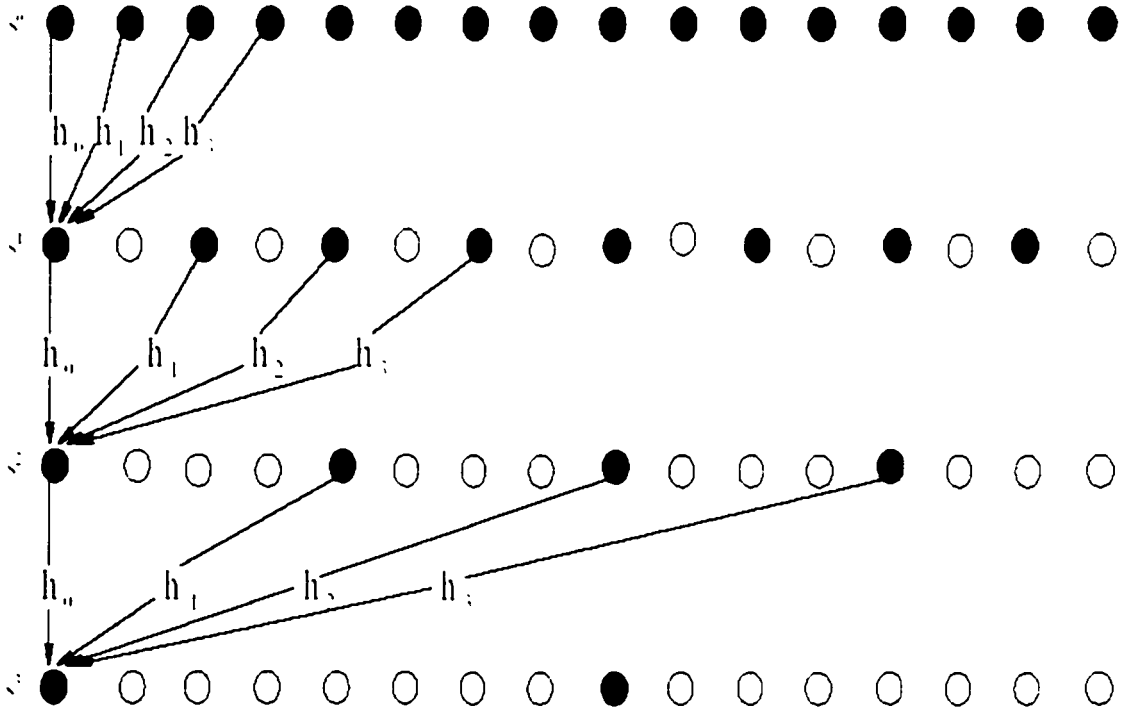

**Figure 6. Diagram illustrating the algorithm for expanding a signal into multiresolution scales using the quadrature mirror filter $H = (h_0, h_1, h_2, h_3)$.**

## 5. Orthonormal Multiwavelet Shell Descriptor

Feature selection is the critical step in the recognition process, and what distinguishes OCR methodologies from each other are the types of features selected for representation. In general, good features must satisfy the following requirements: First, intra-class variance must be small, which means that features derived from different samples of the same class should be close. Secondly, the interclass separation should be large, i.e. features derived from samples of different classes should differ significantly. Furthermore, features should be independent of the size and location of the character. This independence can be achieved by pre-processing or by extracting features that are

translation- and scale-invariant. General object recognition systems should also recognize objects regardless of their orientation, which requires rotation-invariant features. In hand-written recognition, however, this property is not necessary, on the contrary it may lead to confusion of characters like '6' and '9'.

In this report, we introduce a set of descriptors which uses orthonormal multiwavelet shell expansion to recognition handwritten numerals. In order to get invariant features, we have to normalize the start point of the contour. Suppose $O$ is the upper-left corner of the minimal bounding rectangle of the numeral, we select the start point as the point on the numeral that has the shortest distance from $O$. After finding the start point, we can trace the contour. We have to resample the contour so that the total number of contour points is fixed, say 64. We also need to normalize the contour so that the numeral falls into a unit circle. This finishes the normalization stage. We perform orthonormal multiwavelets shell expansion on this normalized contour and use the multiwavelets coefficients to query the database.

The algorithm can be summerize as follows:

1- Find the start point and trace the contour of the handwritten numeral.

2- Resample the contour and normalize the contour so that it falls inside a unit circle.

3- Apply orthonormal multiwavelet shell expansion to the contour (x,y).

4- Recognize the numeral by using the features extracted in the previous step.

In theory, the matching process can be done from coarse to fine scales. For each scale, we match the features of the target with those of the patterns in the database and we have three decisions to make:

1- Accept the target as a specific pattern.

2- Reject the target.

3- Mark those entries in the database that are similar to the target as to be determined and begin the next iteration.

If the target is accepted or rejected, the matching process in then terminated. If the target is undetermined, we continue the matching process to the next finer scale, but only apply to those entries that are marked as to be determined. We can use either $L_1$ or $L_2$ distance metric in each scale. Even though the above mentioned multi-resolution approach is similar to human's simultaneous interpretation of visual information, it is not trivial to apply it in real applications. There are two aspects we have to consider. First, we have to determine a threshold in order to give a guideline for acceptation, rejection, and to be determined. There is no way to give an optimal threshold mathematically. One has to choose it by doing a lot of experiments. Second, significant features are lost at very low resolution scales and it is likely that for very high resolution scales the intra-class variance becomes larger because of the deformation of the pattern and the accumulation errors during the transformations. Therefore, it is desirable to use only intermediate resolution scales during the classification phase.

## 6. Experiment Results

We use a file containing 1000 numerals as our training dataset. We apply the above mentioned multiwavelet orthonormal shell expansion to each numeral of this file and save the shells into a file. The testing data set contains about 100 hand-written numerals (as shown in Figure 7).

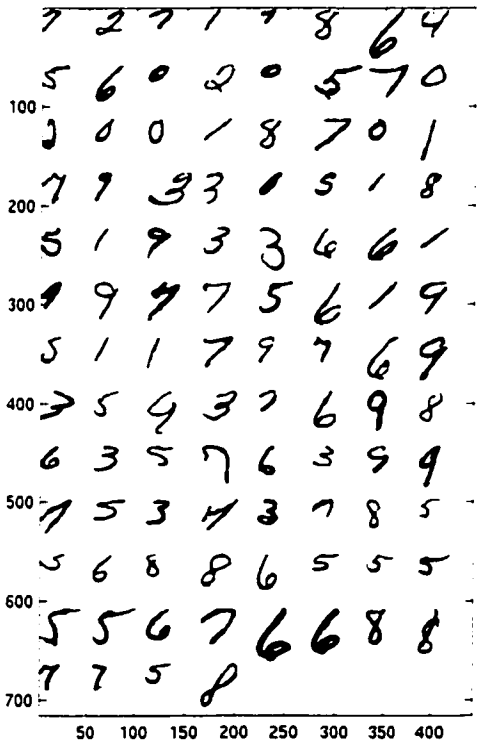


Figure 7. Numerals in the testing data set.

Some of the numerals are very difficult to recognize even with our human eyes. For each numeral, we apply our multiwavelet orthonormal shell descriptor on it (the computation process are depicted in Figure 8) and use nearest neighbour rule to query the database file to get the minimum difference between the coefficients. In our experiments, we only get

86% recognition rate. We are expecting a much higher recognition rate if we feed the multiwavelet coefficients into feed-forward neural network.

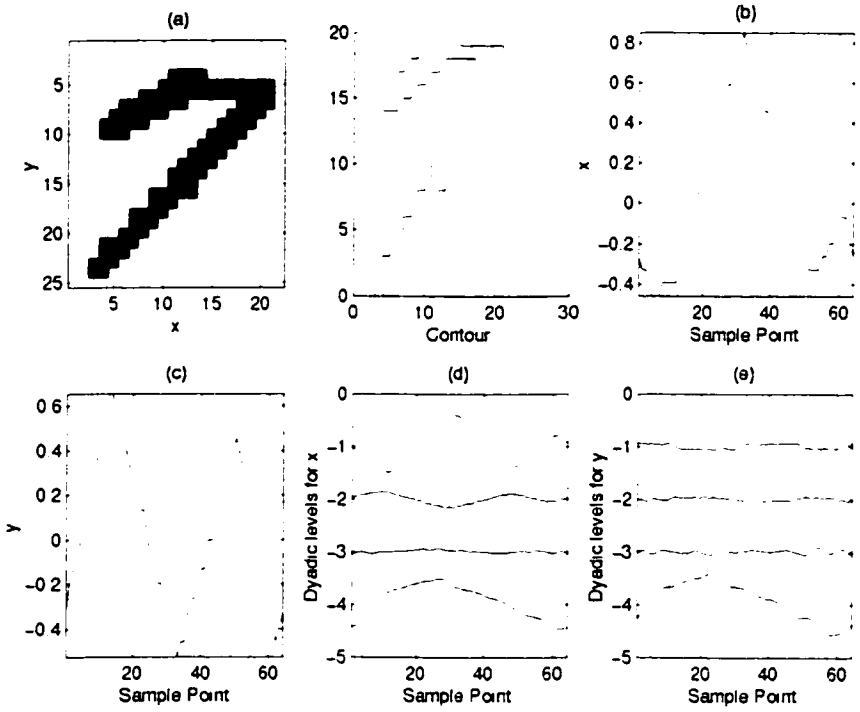


**Figure 8. Graphs depict the computation of our descriptors**

The confusion matrix of our test is as the following

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 9 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 15 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 14 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 15 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 11 |

## 7. Conclusion

In this report, we introduced a novel set of features that is well-suited for representing digitized hand written numerals. The features, which we term multiwavelet orthonormal shell descriptor, are derived from the multiwavelet shell expansion of the character contour. Since we only consider the low-frequency bands of the shell coefficients, the features are relatively insensitive to the shape variation caused by the writing styles of different persons. Multiwavelet shell coefficients depend on the scale and the parameterization starting point of the original function. Therefore, we presented normalization that allow us to derive a scale- and shift-invariant multiresolution for characters of known orientation.

## 8. References


[1] C. T. Zahn and R. Z. Roskies, Fourier Descriptors for Plane Closed Curves, IEEE Transactions on Computers, Vol. C-21, No. 3, pp. 269-281, March 1972.

[2] G. H. Granlund, Fourier Preprocessing for Hand Print Character Recognition, IEEE Transactions on Computers, Vol. C-21, No. 2, pp. 195-201, March 1972.

[3] F. P. Kuhl and C. R. Giardina, Elliptic Fourier Features of a Closed Contour. Computer Graphics and Image Processing, Vol. 18, pp. 236-258, 1982.

[4] C.-S. Lin and C. -L. Hwang , New Forms of Shape Invariants from Elliptic Fourier Descriptors, Pattern Recognition, Vol. 20, No. 5, pp. 535-545, 1987.

[5] P. Wunsch and A. F. Laine, Wavelet Descriptors for Multiresolution Recognition of Handprinted Characters, Pattern Recognition, Vol. 28, No. 8, pp. 1237-1249, 1995.

[6] O. D. Trier, A. K. Jain and T. Taxt, Feature Extraction Methods for Character Recognition – A Survey, Pattern Recognition, vol. 29, pp. 641-662, 1996.

[7] J. S. Geronimo, D. P. Hardin, and P. R. Massopust, Fractal Functions and Wavelet Expansions Based on Several Scaling Functions, *Journal of Approximation Theory*, Vol. 78, pp. 373-401, 1994.

[8] Naoki Saito and Gregory Beylkin, Multiresolution Representation Using the Auto-correlation Functions of Compactly Supported Wavelets, Proceedings of ICASSP, vol. 4, pp. 381-384, 1992.

[9] G. Y. Chen, T. D. Bui and A. Krzyzak, Contour-Based Handwritten Numeral Recognition using Multiwavelets and Neural Networks, *Pattern Recognition*, to appear.

# 9. Program Lists

## 9.1. Test.m

```
% Open the data file and get the character
num = 64;
cur_num = 0;
counter = 0;
confusion = zeros(10,10);

% Load the database (double arrays DBCONMWT_x and DBCONMWT_y)
load DBCONMWT.mat;

% Open the data file and get the log data
fid1 = fopen('alllog','r');
[a1,COUNT1] = fread(fid1);

% Close the file
fclose(fid1);

% Open the data file and get the character
fid = fopen('alltest','r');
[a,COUNT] = fread(fid);

% Close the file
fclose(fid);

% Deal with the whole data set
pos=1;
while pos < COUNT,

   % Get the numeral
   row = 0;
   col = 0;
   clear AA A contour_x contour_y c_x c_y len con_x con_y;

   % Get the row and column of the numeral
   while a(pos) ~= 10,
      if a(pos) ~= 32
         if row == 0
            row = a(pos) - 48;
         else
            if (a(pos-1) ~= 32) & (col == 0)
               row = row*10 + a(pos) - 48;
            else
               if (a(pos - 1) == 32)
                  col = a(pos) - 48;
               else
                  col = col * 10 + a(pos) - 48;
               end
            end
         end
      end
      pos = pos+1;
   end
   pos = pos+2;

   % Get the character from 'a'
   for ii = 1:row,
      for jj = 1:col,
          AA(ii,jj) = a(pos) - 48;
```

```
        if a(pos+1) == 10
           pos = pos + 3;
        else
           pos = pos + 1;
        end
    end
end
pos = pos + 1;

% add zero boundary
row = row + 2;
col = col + 2;
A = zeros(row,col);
for i = 1:row-2,
    for j = 1:col-2,
       A(i+1,j+1) = AA(i,j);
    end
end

% Calculate the centroid of the numeral
xsum  = 0;
ysum  = 0;
count = 0;
for i = 1:row,
    for j = 1:col,
       if A(i,j) == 1
          xsum  = xsum + i;
          ysum  = ysum + j;
          count = count + 1;
       end
    end
end
cen_x = floor(xsum / count);
cen_y = floor(ysum / count);

% Find the start point of the contour
first_y = 1;
quit_y  = 0;
for j = 1:col,
    for i = 1:row,
       if A(i,j) ~= 0
          first_y = j;
          quit_y  = 1;
          break;
       end
    end
    if quit_y ~= 0
       break;
    end
end

first_x = 1;
quit_x  = 0;
for i = 1:row,
    for j = 1:col,
       if A(i,j) ~= 0
          first_x = i;
          quit_x  = 1;
          break;
       end
    end
    if quit_x ~= 0
       break;
```

```
   end
end

distance = 0;
start_x = first_x;
start_y = first_y;
for i = first_x:row,
   for j = first_y:col,
      if A(i,j) ~= 0
         dis = sqrt((i-first_x)*(i-first_x) + (j-first_y)*(j-first_y));
         if distance == 0
            distance = dis;
         end
         if dis < distance
            distance = dis;
            start_x = i;
            start_y = j;
         end
      end
   end
end

% trace the contour
% chain code definition:
%    3  2  1
%    4     0
%    5  6  7

contour_x(1) = start_x;
contour_y(1) = start_y;
nnext=0;
i=0;
j=0;
pt=1;

for pt = 2:10000,
   start = nnext + 3;
   if start > 7
      start = start - 8;
   end

   con_x = contour_x(pt-1);
   con_y = contour_y(pt-1);
   [i,j,nnext] = next_point(A, start, con_x, con_y);

   contour_x(pt) = i;
   contour_y(pt) = j;

   if ((i == start_x) & (j == start_y))
      break;
   end
end

c_x = contour_x;
c_y = contour_y;
c_x(pt+1) = c_x(1);
c_y(pt+1) = c_y(1);

% display the contour
%figure;
%line(c_x,c_y);

% move coordinate origin to the centroid
```

```
contour_x = contour_x - cen_x;
contour_y = contour_y - cen_y;

% size normalization
leng1 = sqrt(contour_x(1)*contour_x(1) + contour_y(1)*contour_y(1));
for i = 2:pt,
   leng2 = sqrt(contour_x(i)*contour_x(i) + contour_y(i)*contour_y(i));
   if leng1 < leng2
      leng1 = leng2;
   end
end
contour_x = contour_x / leng1;
contour_y = contour_y / leng1;

% Calculate the length array
len(1) = 0;
for i = 2:pt,
   len(i) = len(i-1) + sqrt((contour_x(i) - contour_x(i-1)) * ...
                            (contour_x(i) - contour_x(i-1)) + ...
                            (contour_y(i) - contour_y(i-1)) * ...
                            (contour_y(i) - contour_y(i-1)));
end

% Resample the contour
con_x(1) = contour_x(1);
con_y(1) = contour_y(1);
for i = 2:num,
   curlen = (i-1) * len(pt) / num;
   for j = 1:pt-1,
      if curlen > len(j) & curlen <= len(j+1)
         break
      end
   end
   con_x(i) = contour_x(j+1);
   con_y(i) = contour_y(j+1);
end

% multiwavelet transform shell
[qmf_lo qmf_hi] = MMakeONFilter('GHM');
L = 2;
aver_out(1,1:num) = con_x(1:num);
aver_out(2,1:num) = con_y(1:num);
mwcoef = zeros(2,num);
shell_x = zeros(L+1,num);
shell_y = zeros(L+1,num);
for level = 0:L,
   for i = 1:num,
      aver_in(1,i) = aver_out(1,mod((i-1)*2^level,num)+1);
      aver_in(2,i) = aver_out(2,mod((i-1)*2^level,num)+1);
   end
   [aver_out, mwcoef] = MDownDyad(aver_in,qmf_lo,qmf_hi);
   shell_x(level+1,1:num) = mwcoef(1,1:num);
   shell_y(level+1,1:num) = mwcoef(2,1:num);
end

% Query the data base
for noch = 1:20,
   char_row = (noch-1)*(L+1);
   dist(noch) = 0;
   for ii = 1:(L+1),
      for jj = 1:num,
         dist(noch) = dist(noch) + abs(shell_x(ii,jj) - ...
                      DBCONMWT_x(ii+char_row,jj));
```

```
            dist(noch) = dist(noch) + abs(shell_y(ii,jj) - ...
                         DBCONMWT_y(ii+char_row,jj));
         end
      end
   end

   % Find the least difference
   min_loc = 1;
   min_val = dist(1);
   for ii = 1:20,
      if min_val > dist(ii)
         min_val = dist(ii);
         min_loc = ii;
      end
   end
   min_loc = floor((min_loc-1)/2);
   min_loc


            confusion(min_loc+1,a1(cur_num*16+5)-48+1) = ...
                      confusion(min_loc+1,a1(cur_num*16+5)-48+1) + 1;



   if min_loc == a1(cur_num*16+5) - 48
      counter = counter + 1;
      disp('OK');
   else
      disp('WRONG');
   end
   cur_num = cur_num+1;

end

disp(counter / cur_num);
save confusion_MT.mat confusion;
```

## 9.2. next_point.m

```
function [x,y,nnext] = next_point(A, start, prev_x, prev_y)

ii=1; xx=1; yy=1;
for i=start:-1:start-8,
  ii = i;
  if i<0
    ii=i+8;
  end

  switch ii
    case 0,
      xx=prev_x;
      yy=prev_y+1;
    case 1,
      xx=prev_x-1;
      yy=prev_y+1;
    case 2,
      xx=prev_x-1;
      yy=prev_y;
```

```
    case 3,
      xx=prev_x-1;
      yy=prev_y-1;
    case 4,
      xx=prev_x;
      yy=prev_y-1;
    case 5,
      xx=prev_x+1;
      yy=prev_y-1;
    case 6,
      xx=prev_x+1;
      yy=prev_y;
    case 7,
      xx=prev_x+1;
      yy=prev_y+1;
  end

  if A(xx,yy) ~= 0
    break;
  end
end

nnext=ii;
x=xx;
y=yy;
```

## 9.3. sample.m

```
% Open the data file and get the character
num = 64;
cur_num=0;

fid = fopen('sample.txt','r');

while cur_num < 20,

clear AA A contour_x contour_y c_x c_y len con_x con_y;

% get the label
for i=1:6,
  c = fscanf(fid,'%c',1);
end
label = fscanf(fid,'%d',1);
while c ~= 10
  c = fscanf(fid,'%c',1);
end
row=20;
col=20;

% get the numeral
for i=1:row,
  for j=1:col,
    AA(i,j) = fscanf(fid,'%c',1) - 48;
  end
  c = fscanf(fid,'%c',1);
  while c ~= 10
    c = fscanf(fid,'%c',1);
  end
```

```
end

% add zero boundary
row = row + 2;
col = col + 2;
A = zeros(row,col);
for i=1:row-2,
  for j=1:col-2,
    A(i+1,j+1) = AA(i,j);
  end
end


% Calculate the centroid of the numeral
xsum  = 0;
ysum  = 0;
count = 0;
for i=1:row,
  for j=1:col,
    if A(i,j) == 1
      xsum  = xsum + i;
      ysum  = ysum + j;
      count = count + 1;
    end
  end
end
cen_x = floor(xsum / count);
cen_y = floor(ysum / count);

% Find the start point of the contour
first_y = 1;
quit_y  = 0;
for j=1:col,
  for i=1:row,
    if A(i,j) ~= 0
      first_y = j;
      quit_y  = 1;
      break;
    end
  end
  if quit_y ~= 0
    break;
  end
end
first_x = 1;
quit_x  = 0;
for i=1:row,
  for j=1:col,
    if A(i,j) ~= 0
      first_x = i;
      quit_x  = 1;
      break;
    end
  end
  if quit_x ~= 0
    break;
  end
end
distance = 0;
start_x = first_x;
start_y = first_y;
for i=first_x:row,
  for j=first_y:col,
```

```
    if A(i,j) ~= 0
      dis = sqrt((i-first_x)*(i-first_x) + ...
                 (j-first_y)*(j-first_y));
      if distance == 0
        distance = dis;
      end
      if dis < distance
        distance = dis;
        start_x = i;
        start_y = j;
      end
    end
  end
end

% trace the contour
% chain code definition:
%    3  2  1
%    4     0
%    5  6  7

contour_x(1) = start_x;
contour_y(1) = start_y;
nnext=0;
i=0;
j=0;
pt=1;

for pt=2:10000,
  start=nnext+3;
  if start > 7
    start=start-8;
  end

  con_x=contour_x(pt-1);
  con_y=contour_y(pt-1);
  [i,j,nnext]=next_point(A, start, con_x, con_y);

  contour_x(pt)=i;
  contour_y(pt)=j;

  if ((i==start_x) & (j==start_y))
    break;
  end
end

c_x = contour_x;
c_y = contour_y;
c_x(pt+1) = c_x(1);
c_y(pt+1) = c_y(1);

% display the contour
%figure;
%line(c_x,c_y);
%pause
%close

% move coordinate origin to the centroid
contour_x = contour_x - cen_x;
contour_y = contour_y - cen_y;

% size normalization
lengl = sqrt(contour_x(1)*contour_x(1) + contour_y(1)*contour_y(1));
```

```
for i=2:pt,
  leng2 = sqrt(contour_x(i)*contour_x(i) + contour_y(i)*contour_y(i));
  if leng1 < leng2
    leng1 = leng2;
  end
end
contour_x = contour_x/leng1;
contour_y = contour_y/leng1;

% Calculate the length array
len(1) = 0;
for i=2:pt,
  len(i) = len(i-1) + sqrt((contour_x(i)-contour_x(i-1)) * ...
                           (contour_x(i)-contour_x(i-1)) + ...
                           (contour_y(i)-contour_y(i-1)) * ...
                           (contour_y(i)-contour_y(i-1)));
end

% Resample the contour
con_x(1) = contour_x(1);
con_y(1) = contour_y(1);
for i=2:num,
  curlen = (i-1) * len(pt) / num;
  for j=1:pt-1,
    if curlen>len(j) & curlen<=len(j+1)
      break
    end
  end
  con_x(i)  = contour_x(j+1);
  con_y(i)  = contour_y(j+1);
end

% multiwavelet transform shell
[qmf_lo qmf_hi] = MMakeONFilter('GHM');
L=2;
aver_out(1,1:num)=con_x(1:num);
aver_out(2,1:num)=con_y(1:num);
mwcoef = zeros(2,num);
shell_x = zeros(L+1,num);
shell_y = zeros(L+1,num);
for level=0:L,
  for i=1:num,
    aver_in(1,i) = aver_out(1,mod((i-1)*2^level,num)+1);
    aver_in(2,i) = aver_out(2,mod((i-1)*2^level,num)+1);
  end
  [aver_out, mwcoef] = MDownDyad(aver_in,qmf_lo,qmf_hi);
  DBCONMWT_x(cur_num*(L+1)+level+1,1:num) = mwcoef(1,1:num);
  DBCONMWT_y(cur_num*(L+1)+level+1,1:num) = mwcoef(2,1:num);
end

  cur_num = cur_num + 1;
end

% Close the file
fclose(fid);

% Save the database
save DBCONMWT.mat DBCONMWT_x DBCONMWT_y;
```

## 9.4. MMakeONFilter.m

```
function [qmf_lo,qmf_hi] = MMakeONFilter(Type)
% MMakeONFilter -- Generate Orthonormal Filters for MultiWavelet Transform
%  Usage
%    [qmf_lo,qmf_hi] = MMakeONFilter(Type)
%  Inputs
%    Type   string, 'GHM', 'Linear', ...
%  Outputs
%    qmf_lo   Low pass filter matrices  [H0 H1 H2 H3 ...]
%    qmf_hi   High pass filter matrices [G0 G1 G2 G3 ...]
%
%  Description
%    The "GHM" filters were the first multiwavelet developed by Geronimo,
%    Hardin and Massopust. Reference: "Fractal functions and wavelet
%    expensions based on several scaling functions," J. Approx. Theory,
%    1994.
%
%    The "Linear" filters were piecewise linear orthogonal multiwavelets
%    developped by Gilbert Strang and Vasily Strela. Refernece: "Orhtogonal
%    Multiwavelets with vanishing Moments," J. Optical Eng. 33(1994),
%    pp. 2104-2107.
%
%


       if strcmp(Type,'GHM'),
           sqrt2 = sqrt(2);
           qmf_lo(1,:)=[3/10         2*sqrt2/5    3/10        0        0
0         0       0];
           qmf_lo(2,:)=[-sqrt2/40    -3/20     9*sqrt2/40  1/2  9*sqrt2/40   -
3/20  -sqrt2/40  0];
           qmf_hi(1,:)=[-sqrt2/40    -3/20     9*sqrt2/40 -1/2  9*sqrt2/40   -
3/20  -sqrt2/40  0];
           qmf_hi(2,:)=[  -1/20   -3*sqrt2/20    9/20       0       -9/20
3*sqrt2/20  1/20    0];
       end

       if strcmp(Type,'Linear'),
           sqrt3 = sqrt(3);
           sqrt7 = sqrt(7);
           qmf_lo(1,:)=[   1     0        1        0        0    0];
           qmf_lo(2,:)=[sqrt3/2  0    -sqrt3/2    1/2      0  1/2];
           qmf_hi(1,:)=[   -1    0        1      2*sqrt3   0   0 ] ./ sqrt7;
           qmf_hi(2,:)=[sqrt3/2  0    -sqrt3/2    1/2      0 -7/2] ./ sqrt7;
       end
```

## 9.5. MdownDyad.m

```
function [aver_out, mwcoef] = MDownDyad(aver_in,qmf_lo,qmf_hi)
% MDownDyad -- Hi-Pass and Low-Pass Downsampling Operator (periodized)
%  Usage
%    [aver_out, mwcoef] = MDownDyad(aver_in,qmf_lo,qmf_hi)
%  Inputs
%    aver_in    2-d signal at fine scale
%    qmf_low    low pass filter matrix  (2 rows) [H0 H1 H2 H3 ...]
%    qmf_hi     high pass filter matrix (2 rows) [G0 G1 G2 G3 ...]
%  Outputs
%    aver_out   2-d signal at coarse scale
%    mwcoef     multiwavelet coefficients
%
```

```
% low pass (no shift)
 flen_lo  = length(qmf_lo(1,:));
 [s_pad]  = [aver_in aver_in(:,1:flen_lo)];
 s_len    = length(aver_in(1,:));
 aver_out = zeros(2,s_len);
 for ii=0:s_len-1,
    for nn=0:flen_lo/2-1,
       aver_out(1,ii+1) = aver_out(1,ii+1) ...
                        + qmf_lo(1,nn*2+1) * s_pad(1,ii+nn+1) ...
                        + qmf_lo(1,nn*2+2) * s_pad(2,ii+nn+1);
       aver_out(2,ii+1) = aver_out(2,ii+1) ...
                        + qmf_lo(2,nn*2+1) * s_pad(1,ii+nn+1) ...
                        + qmf_lo(2,nn*2+2) * s_pad(2,ii+nn+1);
    end
 end
 aver_out = aver_out .* sqrt(2);

 % high pass (no shift)
 flen_hi  = length(qmf_hi(1,:));
 [s_pad]  = [aver_in aver_in(:,1:flen_hi)];
 mwcoef   = zeros(2,s_len);
 for ii=0:s_len-1,
    for nn=0:flen_hi/2-1,
       mwcoef(1,ii+1) = mwcoef(1,ii+1) ...
                      + qmf_hi(1,nn*2+1) * s_pad(1,ii+nn+1) ...
                      + qmf_hi(1,nn*2+2) * s_pad(2,ii+nn+1);
       mwcoef(2,ii+1) = mwcoef(2,ii+1) ...
                      + qmf_hi(2,nn*2+1) * s_pad(1,ii+nn+1) ...
                      + qmf_hi(2,nn*2+2) * s_pad(2,ii+nn+1);
    end
 end
 mwcoef = mwcoef .* sqrt(2);
```