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

Application of the Fourier-Mellin transform to translation-, rotation- and scale-invariant plant leaf identification

by

John Graham le Maistre Pratt

A Thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Science

in the Department of Agricultural and Biosystems Engineering,

McGill University, Montreal, July 2000

Acknowledgments

I would like to thank my supervisor, Dr. Jacques-André Landry, for providing me with the opportunity to pursue graduate work in the first instance and also for allowing me the freedom to study and research the field of machine vision. I would also like to thank Thomas Bernier, friend, colleague and mentor, without whose selfless help this thesis would not have been possible.

Abstract

The Fourier-Mellin transform was implemented on a digital computer and applied towards the recognition and differentiation of images of plant leaves regardless of translation, rotation or scale. Translated, rotated and scaled leaf images from seven species of plants were compared: avocado (*Persea americana*), trembling aspen (*Populus tremuloides*), lamb's-quarter (*Chenopodium album*), linden (*Tilia americana*), silver maple (*Acer saccharinum*), plantain (*Plantago major*) and sumac leaflets (*Rhus typhina*). The rate of recognition was high among translated and rotated leaf images for all plant species. The rates of recognition and differentiation were poor, however, among scaled leaf images and between leaves of different species. Improvements to increase the effectiveness of the algorithm are suggested.

Résumé

La forme discrète de la transformé de Fourier-Mellin fut appliquée pour la reconnaissance et la différenciation d'images de feuilles dont l'orientation, la rotation ou l'échelle étaient variables. Des images de feuilles de sept espèces de plantes, soit l'avocat (*Persea americana*), le tremble (*Populus tremuloides*), le chou gras (*Chenopodium album*), le tilleul (*Tilia americana*), l'érable argenté (*Acer saccharinum*), le plantain (*Plantago major*) et des folioles de sumac (*Rhus typhina*), sous différentes orientations, rotations et échelles ont été comparées. Le taux de reconnaissance est élevé entre toute image d'une espèce ayant subi une rotation et un changement d'orientation. Cependant, le taux de reconnaissance entre des images de feuilles à plusieurs échelles ainsi que pour la différenciation entre des images de feuilles de différentes espèces sont peu concluant. Des modifications sont suggérées dans le but d'augmenter l'efficacité de l'algorithme présenté.

List of Symbols

$f(x), x(t), y(t), h(t), x(\tau)$	continuous arbitrary one-dimensional time- or space-domain functions
$f(x, y)$	continuous arbitrary two-dimensional time- or space-domain function
$x(n)$	discrete arbitrary one-dimensional time- or space-domain function
t	time
n	sampling index
$X(f)$	continuous one-dimensional Fourier transform of a function of variable x ; the original time- or space-domain function is now defined in terms of its continuous frequency content f
$X(m)$	discrete one-dimensional Fourier transform of a function of a variable x ; the original discretely defined time- or space-domain function is now defined in terms of its discrete frequency content index m
f	frequency
m	frequency index
e^x	exponential function, where $e = 2.71828182.....$
$\ln x$	natural logarithm of variable x
$\sin x$	sine of variable x
$\cos x$	cosine of variable x
j	imaginary portion of complex number; $j = \sqrt{-1}$
π	$\pi = 3.14159....$
N	uppermost limit of sampling interval
θ	arbitrary angle in radians; angular component of log-polar transform
$\mathrm{Re}\, X[m]$	real portion of discrete Fourier transform
$\mathrm{Im}\, X[m]$	imaginary portion of discrete Fourier tranform
$M(s)$	continuous one-dimensional Mellin transform
$M(ju, jv)$	continuous two-dimensional Mellin transform along the imaginary axis
ω	natural frequency; $\omega = 2\pi f$
r	radial component of log-polar transform
ρ	discrete exponential radial component of log-polar coordinate system
row	row index of image matrix
col	column index of image matrix
$d_E(f)$	Euclidean distance at frequency f
$\mathrm{Im}\, A$	Image A
$\mathrm{Im}\, B$	Image B

List of Equations

List of Figures and Tables

Figures **Page**

Tables

Table of Contents **Page**

Table of Contents (continued) **Page**

1. Introduction

Our ability to recognize objects in an image, regardless of how these objects are laid out before us, is one that we humans take for granted. When we view an image of a chair, for example, whether the chair is located in a dentist's office, barber salon or around the dinner table, each of us is still able to perceive that the object in question is a chair. Even when the chair is rotated upside down, very tiny or large, or moved to the upper left corner of the image, we are, again, still able to recognize the object as being a chair. For a machine, however, objects that have been translated to a new position in the image, rotated or scaled up or down represent completely new objects. In order for a machine to recognize the two objects as being similar, algorithms need to be developed that can successfully provide for object identification regardless of where they are located in the image or whether they are scaled or rotated. This thesis examines the use of such an algorithm, the Fourier-Mellin transform. In this particular application, the Fourier-Mellin transform is used to help identify the leaf of a plant, regardless of the leaf's scale or rotation, or location in the image.

It would be foolish at best to expect one algorithm alone to be successful at recognizing any and all variations of the same object. And it would be doubly foolish to then expect the same algorithm to be able to differentiate between different objects. This is especially true when the objects under consideration are natural (biological), rather than man-made, since: 1) biological objects exist in many shapes, sizes, colours and textures even within a single species, and 2) many different species resemble one another. It is hoped, then, that the Fourier-Mellin transform can serve as one among many analysis tools of a generalised machine vision system in order to both recognize biological objects within and outside their species.

2. Literature Review

The following assumes familiarity with the Fourier transform as well as the Fourier-Mellin transform. The reader is referred to section 3 of this thesis which provides the necessary background to these transforms.

The Fourier-Mellin transform is a variant of the Fourier transform; however, it certainly does not have as glorious or as predominant a history as the Fourier transform. Indeed, versions of the Fourier-Mellin transform do not seem to surface until the late 1960s. Among the first investigators to use a version of the Fourier-Mellin transform were Brousil and Smith [1967]. In retrospect, these investigators remain pioneers in the use of the transform. In their paper, "A Threshold Logic Network for Shape Invariance", they managed to demonstrate how a log-polar coordinate mapping and Fourier transform could be combined to achieve translation, rotation and dilation (TRD) invariance in image recognition. Though they did not mention it explicitly, the investigators essentially performed a Fourier-Mellin transform in their attempts at achieving TRD invariance. They did so by constructing a very primitive neural network incorporating the operations of the Fourier-Mellin transform and used binary images of either hand-printed or machine-generated characters, presented as 12x12 rasters of X's and O's as the input and output layers of the system. The network seems to respond well with a 70% recognition rate for the translated, rotated and scaled (dilated) machine-generated characters but recognises only 30% of the translated, rotated and scaled hand-printed characters.

The next pioneering attempt at applying the Fourier-Mellin transform was instigated by Robbins and Huang [1972]. In fact, these investigators are often credited as the first to use the discrete Fourier transform along with a logarithmic-polar sampling of an input image to produce an approximation of the Mellin transform [Schalkoff, 1989], though Brousil and Smith [1967] had clearly paved the way. Robbins and Huang described an implementation for the application of the Fourier-Mellin transform to correct various

optical distortions, including noise, in lenses. Though the achievement of translation-, rotation- and dilation-invariance via the Fourier-Mellin transform were implied, they were not specifically mentioned as goals the authors sought to demonstrate. They first applied their algorithm, in the general case, to the restoration of input images distorted by various forms of lens aberrations. Later, they implemented the algorithm on digital images that had been blurred as well as on images to which Gaussian noise was added. The results they choose to present, in pictorial form, seem convincing enough, though they indicate the outcome as being poor in images corrupted with noise. Specifically, in comparing the blurred input images to their outputs, they obtain a per point standard deviation error of $e = 0.7$; whereas they obtain a standard deviation error of $e = 1.77$ in the case of images distorted with Gaussian noise.

In the late 1970s, Casasent and Psaltis [1976, 1977] contributed substantially to the implementation of a digital form of the Fourier-Mellin transform in applications using physical lenses. They wished to design an optical matched filter correlator that responded well to scale changes in the input image. A matched filter correlator is a system in which an image is formed and correlated against another image. Such a system can be used to ascertain whether the images are similar, but performs poorly when the images differ in scale or rotation because these changes result in a severe loss in the signal-to-noise ratio that affects the correlation significantly. The investigators corrected for the scale difference in the two images by applying, via a Fourier lens system, the Fourier-Mellin transform to each image and correlating their output images. They first demonstrated scale invariance using lenses by taking simple images each of a vertical bar, a circle and a square, converting the image to a logarithmic scale in the vertical and horizontal components, projecting the log-transformed image through a Fourier transform lens that resulted in a Mellin transform of the image. If the input image is then scaled – Casasent and Psaltis illustrated this using an image of a square scaled to twice its original size – its Mellin-transformed output is similar to that of its unscaled counterpart. Later the investigators proposed a schematic for a scale-invariant Fourier-Mellin optical correlator.

They indicated it would be possible, using the aforementioned Fourier-Mellin lens system, to measure any scale differences in two input images and apply the resulting data to equalize the scale, enabling one to later correlate and compare the images for similarity. Finally, Casasent and Psaltis showed, step by step, how to implement the Mellin transform via the Fourier transform. This particular implementation is now recognized as today's Fourier-Mellin transform. Unfortunately, the authors continued to rely on a physical lens implementation of the Fourier-Mellin transform rather than one implemented by digital computer. Though the physical lens system more closely approximates the theory behind the continuous form of the Mellin transform, it has disadvantages in that one has to physically move and rotate the system of lenses in order to be able to match patterns.

The Fourier-Mellin transform has also been considered as a basis for two of the five mammalian senses. The log-polar transformation necessary in the implementation of the Fourier-Mellin transform and its relation to the sense of sight is discussed in section 4.5 of this thesis. It is important to mention here that the Fourier-Mellin transform has been considered in examining the sense of hearing. Altes [1978] noted similarities between the Fourier-Mellin transform and the cochlear transduction of signals in animals. In particular, he concluded that a biological model of the transform might be implemented in some bat species for the purposes of echolocation.

A further development in the use of the Fourier-Mellin transform was its application to the radar classification of ships. Zwicke and Kiss [1983] wished to use the shift- and scale-invariance derived from the Fourier-Mellin transform in order to identify ships from radar range profiles. A problem arises when the radar signal and length of the axis of the ship are no longer exactly aligned. If the ship is turned slightly, the radar profile dimension decreases in size and the radar profile must be expanded for the identification of the vessel. The investigators proposed that the application of the scale- and rotation-invariant Fourier-Mellin transform would enable the extraction of identifying factors in the radar signal regardless of the aspect angle of the radar. The authors also offered an alternate

implementation of the Fourier-Mellin transform called the modified direct Mellin transform (MDMT). This new implementation was based on directly expanding the Mellin integral rather than relying on the more traditional and indirect calculation of the Fourier-Mellin transform, that is, a rectangular to log-polar conversion in combination with the Fourier transform. The authors maintained the new implementation eliminated any errors introduced into the approximation of the Mellin transform by the traditional methodology in that no exponential sampling or interpolation was necessary. Though theoretically this might be the case, the scant experimental results, the investigators' caveat the MDMT is computationally more expensive than the traditional Fourier-Mellin implementation and lack of evidence in the literature since publication of this paper of MDMT usage, all suggest this newer implementation of the Fourier-Mellin transform is less than satisfactory.

Sheng *et al.* [1986, 1994] in the late eighties and early nineties dealt with what the investigators termed Fourier-Mellin moments. Essentially Fourier-Mellin moments are moments calculated from the radial component of a pseudo-Fourier-Mellin transformed image. Though the moments lend themselves to providing shift- and rotation-invariant descriptors of images, scale invariance was not completely achieved. Derrode [1999] suggested this was due to the fact that the radial base used in the moment calculation did not correspond to that of a true Mellin transform, and, as a result, discounted using moments as a substitute for the traditional Fourier-Mellin transform.

In 1995, the use of the Fourier-Mellin transform and a neural network was again combined. Raman and Desai [1995] calculated rotation-, translation- and scale- (RTS-) invariants – equivalent to TRD invariants – using the magnitude of the Fourier-Mellin transform. They used a multilayer neural network trained by backpropagation to recognize RTS-variants of input images. In total 6 images were used to train the network and 50 RTS-variants of these six images were presented to the network and were all correctly recognized as one of the six training set images. Partially occluded images – images in which a contiguous portion of their shape was removed – were also correctly recognized.

It is not clear whether occluded images were used to train the network, and, generally, the breadth of the experimental study is not great – only 6 images were used both for training and testing the network.

Recently, the Fourier-Mellin transform has seen a revival with the advent of watermarking. Watermarking is a method of preserving the copyright on a document or image. Techniques for hiding watermarks are becoming more sophisticated but one also wants a watermark to be able to withstand modifications to the image such as rotation, translation or scale changes so that the watermark, if searched for in the modified image, can still be found easily. Ó Ruanaidh and Pun [1997] and Lin *et al.* [2000] both investigated the use of the Fourier-Mellin transform in producing an RTS-invariant watermark. Ó Ruanaidh and Pun considered the phase of the Fourier-Mellin transform in their application so that they were able to recover the watermark from an RTSed image, even after lossy JPEG image compression of 75%. Lin *et al.*, on the other hand, studied the use of creating a unique signature of their watermark that was RTS-resilient. They preferred to use the term resilient over invariant because they did not believe it to be necessary to achieve complete RTS-invariance in order to detect the watermark. However, they decided to use correlation in order to search for the watermark signature – which is time consuming – and must rotate their searching pattern at all possible angles in order to find the watermark – which is also time consuming.

Though there is a fair amount of research on machine vision in agriculture, especially in the automated inspection of food items [Shatadal *et al.*, 1991], there has been very little research on frequency-domain analysis of leaves and even less research on specific applications of the Fourier-Mellin transform. Indeed, a search of the literature provided but one reference for the Fourier-Mellin transform in an agricultural domain. In this one reference, Franz *et al.* [1991] attempted to identify plant leaves at various life stages based on the leaves' shape or contour. The Fourier-Mellin transform was not applied directly to leaf identification but, rather, was used to compensate for occluded portions of leaves.

The authors extracted the leaf edges from their images – the extraction procedure is not firmly explained. They then represented the points in the leaf edge as a curvature function used to describe the leaf shape. The curvature function is invariant to object location and rotation. To compensate for partial occlusion of a leaf, the authors took the magnitude of the Fourier-Mellin transform of the leaf's curvature function, used correlation to compare the target leaf against the model and removed peaks that did not match in order to obtain a match – this also compensated for differences in scale since the Fourier-Mellin transform is resistant to scale changes. The authors then used a distance measure between the models' leaf shape and the unknown shapes. They generally obtained poor results especially when leaves were occluded or their shapes were similar. Finally, another set of investigators, Zhang and Chaisattapagon [1995], used the Fourier transform spectra of plants, among other methods, to differentiate between weeds and wheat species typically found in Kansas. The researchers were able to use Fourier spectra to describe the texture associated with a particular plant species and found that both the fineness and direction of a plant's texture pattern could be applied successfully to distinguish wheat from weeds. No attempts are made, however, to provide for differentiating between wheat and weeds when the textures of each are rotated and/or scaled.

3. The Fourier Transform

The Fourier transform is dealt with thoroughly in the literature (see, for instance, Lyons, 1997; Bracewell, 1986 and Brigham, 1988). A general background on the transform, however, is necessary in order to better appreciate its use in the Fourier-Mellin algorithm presented in this thesis. First, it should be mentioned that despite the Fourier transform's complex mathematics and conceptual difficulties, the transform is just that, a mathematical transformation. It transforms a function from its more easily understood time or spatial domain into a function existing in frequency space. The essence, and beauty, of the transform is that it demonstrates almost any function can be broken up into a sum of known periodic sinusoidal functions, each of which is characterized by its amplitude and frequency. This is more easily explained pictorially using Figure 3-1. Here a step function – which, incidentally, is a typical representation of an edge in an image – can be shown to be reproduced by summing various sine waves of different frequencies and amplitudes.

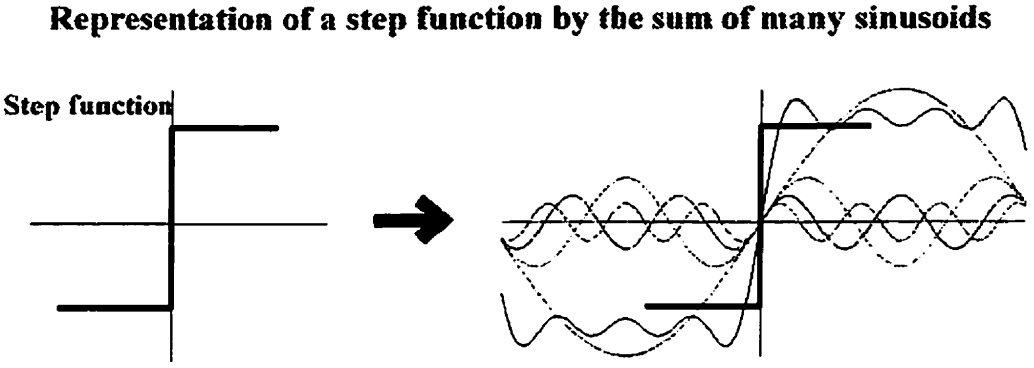


Figure 3-1: Step function as sum of many sinusoids

Equation 3-1 defines the mathematics of the one-dimensional Fourier transform, $X(f)$, of a function in time, $x(t)$, for the continuous case [Lyons, 1997]:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\,dt, \quad \text{eq. 3-1}$$

where $X(f)$ provides a complete representation of $x(t)$ except that the function is now expressed in terms of the amplitudes of frequencies, f, of sinusoidal waves instead of as a function of time. However, we wish to implement the discrete version of the continuous Fourier transform (CFT), that is, on a digital computer. As such, the infinite integral simply becomes a summation over a finite amount of time (or space). Equation 3-2 [Lyons, 1997] defines the discrete Fourier transform, $X(m)$, of a discrete sequence of time- or space-domain sampled values, $x(n)$, which represents the original continuous function:

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nm/N}, \text{ eq. 3-2}$$

Here n represents the sample number, spanning the zeroth to the last sample (0 to $N-1$) of the discrete function $x(n)$; m, like n, goes from 0 to $N-1$ and represents each of the possible frequencies of the function $X(m)$, which, in turn, defines the amplitude of each of these frequencies. Together all $X(m)$ represent a function that provides a complete frequency-domain representation of the original time- or space-domain function. The discrete Fourier transform (DFT) representation of the CFT is still rather complicated mathematically, especially for individuals not already possessing a graduate degree in electrical engineering. However, the complex notation of the transform can be further reduced to something a little more comprehensible via *Euler's relationship* defined in Equation 3-3:

$$e^{-j\theta} = \cos(\theta) - j\sin(\theta) \text{ eq. 3-3}$$

The transform now reduces to Equation 3-4 [Lyons, 1997]:

$$X(m) = \sum_{n=0}^{N-1} x(n)[\cos(2\pi nm/N) - j\sin(2\pi nm/N)] \text{ eq. 3-4}$$

The DFT now reveals itself to be made up explicitly of a series of cosine and sine waves which, when summed, provide a complete representation of the time or spatial domain

function. The equation is still complex; however, its use of the imaginary number, j equalling $\sqrt{-1}$, though invaluable as it is for the mathematical derivation of the transform, can be avoided in the discrete, digital computer implementation.

3.1 Calculating the DFT

There are three different ways to calculate the DFT: by simultaneous equations, by correlation and by the Fast Fourier Transform [Smith, 1997]. The standard way is by correlation. It is the simplest method and also provides some intuitive insight into the calculation of the DFT. What correlation entails is multiplying each individual indexed sample from the time- or space-domain signal for which we wish to obtain the DFT by its corresponding indexed sine and cosine frequencies and then adding the result [Lyons, 1997]. The result is then a reflection of whether that particular sine or cosine frequency is contained in the frequency domain of the time- or space-domain function. This is what is actually occurring when we apply equation 3-4 to obtain the DFT. That is, if we use correlation to find the DFT of a time- or space-domain function, $x(n)$, made up of samples indexed from $n = 0$ to N-1, we will obtain two frequency-domain functions, termed the Real part and the Imaginary part of the DFT. Each of these parts of the frequency domain also extends from 0 to N-1. These are just equation 3-4 divided into two portions, Equations 3-5 and 3-6 [Lyons, 1997]:

$$\mathrm{Re}\,X[m] = \sum_{n=0}^{N-1} x[n]\cos(2\pi mn / N) \quad \textbf{eq. 3-5}$$

$$\mathrm{Im}\,X[m] = \sum_{n=0}^{N-1} x[n]j\sin(2\pi mn / N) \quad \textbf{eq. 3-6}$$

where $\mathrm{Re}\,X(m)$ refers to the Real component and $\mathrm{Im}\,X(m)$ refers to the Imaginary component; index m again extends from 0 to N-1.

A particular point, $x(n)$, from the time or space domain, thus consists of two parts in the frequency domain: a Real component that represents the amplitude of the cosine wave of frequency m and an Imaginary component that represents the amplitude of the sine wave of frequency m. The sum of all the Real and Imaginary components from 0 to N-1 represent the transformation of the time- or space-domain signal into the frequency domain.

3.2 Polar Notation

When the results of the Fourier transform of a signal are reported as the "Real" and "Imaginary" parts, then these are being reported in rectangular notation. In the digital signal processing literature, however, frequency-domain results are typically reported in polar notation, as the Magnitude and Phase of each frequency m of the Fourier transform. If we graph one of the Fourier transformed points, $X[m]$, in the complex domain, it is easy to understand the conversion from rectangular to polar notation (see Figure 3-2, adapted from Lyons [1997]):

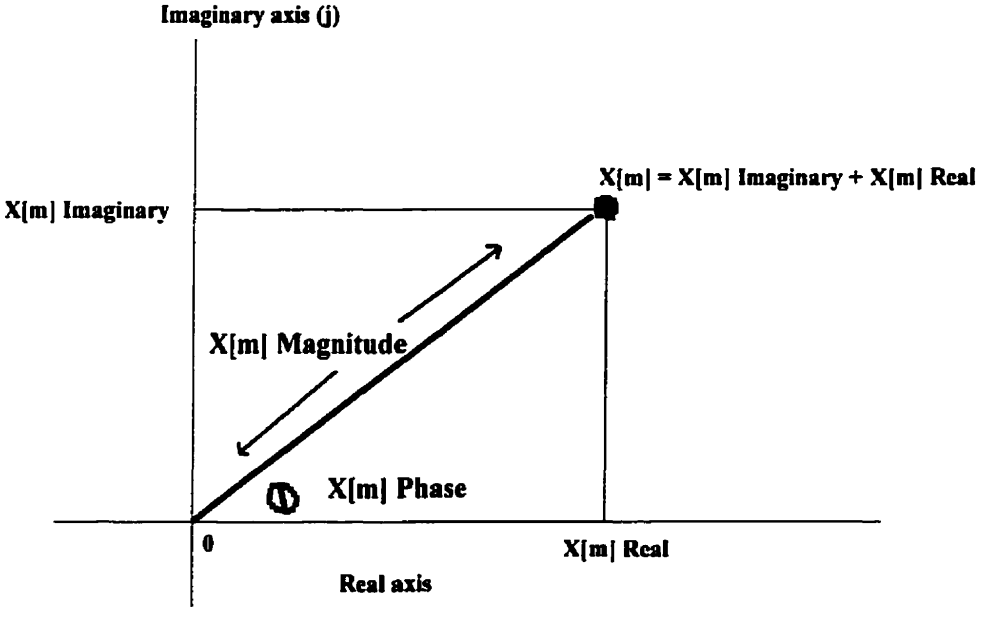


Figure 3-2: Determining polar notation (adapted from Lyons [1997])

Using the Imaginary/Real coordinate system of Figure 3-2 and the Pythagorean theorem, we see that the Magnitude of $X[m]$ is defined by Equation 3-7 as:

$$X_{mag} = |X[m]| = \sqrt{X[m]_{Real}^{2} + X[m]_{Imaginary}^{2}} \quad \textbf{eq. 3-7}$$

and the Phase is defined by Equation 3-8 as:

$$X_{phase} = X_{\phi}[m] = \arctan\left(\frac{X[m]_{Imaginary}}{X[m]_{Real}}\right) \quad \textbf{eq. 3-8}$$

3.3 Properties of the DFT

Bracewell [1986] and Brigham [1988] provide very detailed accounts of the Discrete Fourier transform's properties. A summary of those properties necessary for the understanding of the application used this thesis is presented here.

3.3.1 Periodicity

Unlike the CFT, the DFT views both its input (the time or space domain) and its output (the frequency domain) as being periodic and infinite. For a discrete signal, this means the DFT views its input as if the signal's beginning and ending were attached together and repeating from negative to positive infinity. Figure 3-3, adapted from Smith [1997], depicts this:

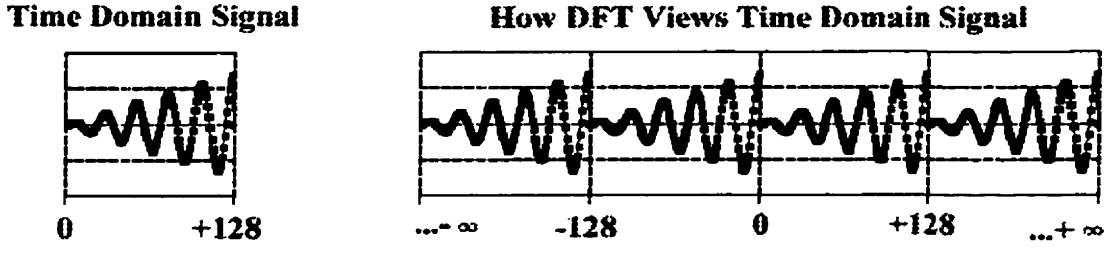


Figure 3-3: DFT Periodicity (adapted from Smith [1997])

In other words, the DFT views its time- or space-domain input as consisting of a single period of an infinitely repeating sequence. The same situation applies to the output of the DFT. This property of the DFT has some negative consequences. These are addressed

later under the topic of "Problems with the Discrete Fourier-Mellin Transformation" in section 5 of this thesis.

3.3.2 Scale

If a time- or space-domain function is scaled by a constant, then its Fourier transform will be inversely scaled by that same constant [Brigham, 1988].

3.3.3 Rotation

If a function is defined in a two-dimensional spatial domain (as is, for example, an image) and is rotated in this domain, then its corresponding DFT will also be rotated by the same angle [Castleman, 1996].

3.3.4 Translation

The shifting theorem of the Fourier transform states that if a time-or space-domain function is shifted by a constant, then this shift will be expressed as a constant shift only in the phase of the DFT [Lyons, 1997]. The magnitude of the DFT is not affected by a shift change; it is shift- or translation-invariant. The DFT magnitude's translation invariance is the cornerstone of the success of the Fourier-Mellin transform. In the Fourier-Mellin transform, scale and rotation changes are manifested as translations which are then removed by performing a second DFT and considering only the resultant translation-invariant magnitude.

3.3.5 Convolution

Though not specifically a Fourier transform property, convolution is often used in conjunction with frequency domain analysis because of the particularity that convolution in the time domain is equal to multiplication in the frequency domain and vice versa. In this thesis, convolution is used in filtering and windowing data before it is input to the DFT. As a result, a general explanation of convolution is necessary. According to Brigham [1988], convolution is defined mathematically by Equation 3-9 as follows:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau = x(t) * h(t) \text{ eq. 3-9 ,}$$

where * denotes convolution. Here, $y(t)$ is said to be the convolution of the functions $x(t)$ and $h(t)$. Equation 3-9, however, is not easy to visualize. The following is what actually occurs when two functions are convolved (based on Brigham, 1988): 1) The mirror image of $h(\tau)$ is taken. This simply results in $h(-\tau)$; 2) $h(-\tau)$ is then shifted by an amount t; 3) the shifted function, $h(t-\tau)$, is then multiplied by $x(\tau)$ and, finally, 4) this results in the area under the graph of the product of the two functions equalling the value of the convolution at time t (i.e. the integral of the product equals the convolution at time t).

3.4 Implementation of the DFT via the FFT

Though the correlation method of calculating the DFT is simple and makes intuitive sense, it is slow compared to the now famous Fast Fourier Transform (FFT) developed by Cooley and Tukey in 1965. For a one-dimensional signal of N points, the DFT via correlation requires N^2 calculations (for a two-dimensional signal, such as a square image, there are N^2 number of points; thus, it would require N^4 calculations). The FFT for a one-dimensional signal, on the other hand, requires $N \log_2 N$ calculations, which for a 32-point signal amounts to only 10 times the speed of the DFT via correlation; however, for a signal above roughly 4,000 points, the FFT is over a thousand times faster than correlation [Smith, 1997]. In obtaining the DFT of an image, we are often dealing with signals that are many thousands of points long (a typical 256x256 pixel image contains 65,536 signal points). Thus, the FFT is the algorithm of choice and has been used in this thesis.

A thorough description of the implementation of the FFT and why it is so much faster than the DFT correlation method is beyond the scope of this thesis. Suffice it to say that for an N-point signal the FFT gains a great speed advantage by performing N DFTs on single-

point signals [Smith, 1997]. A number of standard "programming tricks" are also involved in optimizing the FFT computation [Parker, 1997]. The FFT implemented in this thesis is a radix-2 type FFT that has been adapted from code provided in Smith [1997] (the radix-2 type FFT is so named because the number of points in the input signal must be equal to a power of two). The actual input to the FFT consists of two parts, the Real and Imaginary. Earlier, in section 3.2, it was mentioned that the Imaginary portion's use of the imaginary number, j, can be avoided in the implementation of the FFT. In fact, only the Real part holds the input data to the FFT. It consists of an array containing the real-valued sample points of the signal to be transformed into the frequency domain; the Imaginary portion's array, on the other hand, is empty, loaded only with zeroes. This is termed the "FFT of real-valued data" and successfully avoids the use of imaginary numbers in its implementation. It is also worthwhile to note that in order to obtain the DFT of an image via the FFT, a two-dimensional Fourier transform must be performed. Fortunately, the two-dimensional Fourier transform can be obtained by performing two one-dimensional transforms [Brigham, 1988]. In the case of the FFT, this is implemented by performing the following: 1) take the one-dimensional FFT of each of the columns of an image; 2) store the result; 3) take the one-dimensional FFT of each of the rows of the result. The final output will contain the two-dimensional Fourier transformation of the image.

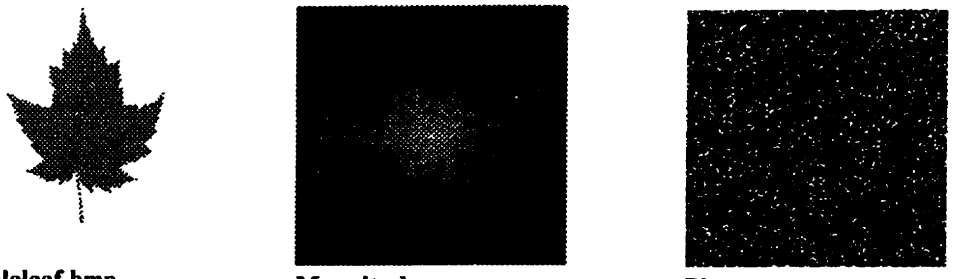
3.5 The Digital Image

This thesis presents the Fourier-Mellin analysis of digital images. It is necessary, therefore, to describe exactly what a digital image is. A natural image or object is viewed by the eye as a continuous array of various colours [Baxes, 1994]. In order to acquire a digital image of this continuous array of colours, the natural image or object is sampled so that each original colour becomes quantized to an integer value. In this thesis, we deal only with greyscale digital images, which are made up solely of 256 "colours" or greyscale intensity values – perhaps the term brightness value is more appropriate since the "colours" range only between intensity 0 (black) and intensity 255 (white). As such, the sampling and quantization of the continuous array of colours in the natural image or object will convert

each original colour to a brightness intensity level between 0 and 255. The two-dimensional digital image is thus a matrix of greyscale values. The position of each of these values in the matrix can be described on a Cartesian grid (x, y) so that we are able to describe the entire image as a two-dimensional function $f(x, y)$; that is, every point – every brightness value – in the digital image can be defined by: 1) its location in two-dimensional space by a coordinate pair (x, y) where x represents the vertical coordinate (or row), and y represents the horizontal coordinate (or column); and 2) by its intensity value on the greyscale. Each of these points is termed a pixel, short for "picture element". Thus, in a 256x256 pixel image there exist 65,536 pixels, each of which is defined by its greyscale intensity value and location (row, column indices) in the image.

3.6 The Fourier Transformation of Images

Displayed in Figure 3-4 is an original image of a maple leaf (a silver maple, *Acer saccharinum*, leaf) that was scanned and digitized resulting in a 256x256 pixel image made up of 256 shades of grey (a "greyscale image"). Also shown are the leaf image's DFT, represented by the magnitude and phase portions of the transform.



Mapleleaf.bmp **Magnitude** **Phase**

Figure 3-4: Fourier-transformed images of a maple leaf

The magnitude and phase images in Figure 3-4 do not seem to have much in common with the original leaf image. Indeed, for most images, the magnitude and phase seem to have no discernable relationship to the original image. What sort of information do the magnitude and phase of a Fourier-transformed image provide? This question is addressed in the following sections.

3.6.1 Magnitude Image

Images can be defined as changes in brightness in a two-dimensional space. That is, as we move, say, from one side of the image to the other side, we will encounter various brightness levels along our journey that define the digital image. The rate at which brightness levels change from light to dark and dark to light equals the spatial frequency of an image [Baxes, 1994]. When we Fourier-transform an image, we are measuring the spatial frequencies of the image. In particular, it is the magnitude rather than the phase of a Fourier-transformed image that best represents these spatial frequencies. Baxes [1994] provided examples using very simple, artificially-created images, that illustrate how the magnitude is useful in representing the spatial frequency content of an image. These examples are depicted in Figures 3-5 and 3-6:

Low-frequency image **Magnitude of low-frequency image**

Figure 3-5: Low-frequency image and its Fourier transform (adapted from Baxes [1994])

As we traverse the low-frequency image of Figure 3-5 from top to bottom, its brightness values change slowly, alternating from dark to light regions. The frequency of the brightness value change is also constant throughout the image, that is, only one frequency is necessary to describe the change from dark to light brightness values in the image. All of these features are easily seen in the magnitude of the low-frequency image's Fourier transform of Figure 3-5. Firstly, note that there are two bright points in the magnitude image. This results from the property of the Fourier transform being periodic. Indeed, the periodicity provides for the top half of the magnitude image being a distorted mirror image of the bottom half (the top right quadrant mirrors the bottom left and the top left quadrant mirrors the bottom right). Secondly, note that the magnitude image is usually displayed

such that the zeroth frequency component of the original image is located in the centre of the magnitude image and that frequencies increase from the centre in the horizontal and vertical directions. The amplitude of the frequencies is indicated by the brightness value of the pixels in the magnitude image. If we now consider just the top half of the magnitude image, we can see that our low-frequency image is depicted as being made up of a single bright spot located fairly near the image centre (low frequency) and on the vertical axis (representing the vertical direction of change in brightness level of the original image). The brightness value of the pixels representing the bright spot in the magnitude image are a measure of the amplitude of the frequency change in the original image.

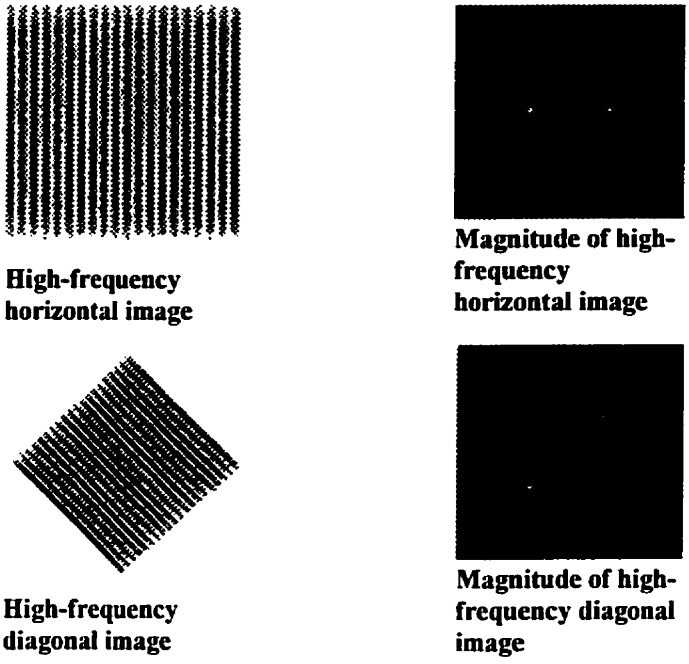


Figure 3-6: High-frequency images and their Fourier transforms (adapted from Baxes [1994])

The images in Figure 3-6 show how direction and frequency of brightness changes affect the Fourier-transformed magnitude image. Note the increased distance from the centre at which the bright spots are located in the magnitude images; the further distance represents the increase in the frequency of the changes from dark to bright in the original image. Also note the direction of brightness changes in the original images are reflected in the location of the bright spots on the horizontal/vertical axes of the magnitude images.

3.6.2 Phase Image

As was seen in Figure 3-4's phase image of the Fourier-transformed maple leaf, the phase does not bear much resemblance to the original image. The phase image does, however, contain much information about the original image. This was observed by Oppenheim and Lim [1981] who inverse-Fourier transformed images by assigning the magnitude a constant amplitude for each frequency and leaving the phase unchanged. In so doing they were able to observe the separate contributions of the phase and the magnitude in the original untransformed image. They concluded that much of the intelligibility of the original image is contained in its Fourier-transformed phase. This is due to the fact that typical images contain objects that are defined by their shape. Shape, in turn, is defined by edges and, in the frequency domain, edges are defined by the rise and fall of sinusoids the phases of which are coordinated [Smith, 1997].

3.6.3 Magnitude versus Phase of an Image

Compared to the phase, the magnitude holds less information about the beginning and ending of shapes in an image; rather, it is more a reflection of the energy content of an image [Lim, 1990], that is, compared to the phase which specifies where each sinusoidal component lies in an image, the magnitude specifies how much of each sinusoidal component is present [Castleman, 1996]. As a result, if image object discrimination is based solely on the Fourier-transformed images' magnitudes, there is a danger that two completely unrelated objects may produce the same magnitudes. This would be less likely to occur if the phases for the image objects alone were compared because the uniqueness of the phase corresponds to the uniqueness of the object's shape. The magnitude of an image, however, unlike its phase, can be manipulated to provide for an image object's translation-, rotation- and scale-invariance. This has been the overriding reason for the sole consideration of an image object's magnitude in the present thesis's application of the Fourier-Mellin transform. As well, this thesis's implementation of the Fourier-Mellin transform is meant to be considered as one among many analyses performed on an image for the purpose of image object recognition. In cases where dissimilar image objects share

similar energy content, it is expected that other image analyses might aid, in addition to the present transform, in discriminating between such image objects.

4. The Fourier-Mellin Transform

The Fourier-Mellin transform is used in this thesis to render descriptions of images that are translation-, rotation- and scale-invariant. In other words, if we wish to compare two images – each of which contains the same object – for similarity and one image is a translated and/or rotated and/or scaled image of the other, then the magnitudes of the Fourier-Mellin transforms of both images should be identical. The development of the Fourier-Mellin transform and explanation of its translational-, rotational- and scale-invariance properties will now be presented.

4.1 Mellin Transform

The Fourier-Mellin transform is based on the Mellin transform. A modified form of Bracewell's [1986] definition of the one-dimensional Mellin transform of any function $f(x)$ is given in Equation 4-1:

$$M(s) = \int_0^\infty f(x)x^{s-1}dx \quad \textbf{eq. 4-1}$$

We are, however, dealing with images that are functions defined in two dimensions. The two-dimensional Mellin transform of a function $f(x, y)$ is defined by Casasent and Psaltis [1977] along the imaginary axis by Equation 4-2:

$$M(ju, jv) = \int_0^\infty f(x, y)x^{-ju-1}y^{-jv-1}dxdy \quad \textbf{eq. 4-2}$$

4.2 Scale Invariance of the Mellin Transform

The Mellin transform, unlike the Fourier transform, possesses the property of being scale-invariant in its magnitude. Casasent and Psaltis [1977] demonstrated this scale invariance for the two-dimensional case by considering a simple rectangular input function $f(x, y)$ extending from x_1 to x_2 and y_1 to y_2, defined in Equation 4-3:

$$f(x,y) = \mathrm{Rect}\left[\frac{x-(x_1+x_2)/2}{x_2-x_1}\right]\mathrm{Rect}\left[\frac{y-(y_1+y_2)/2}{y_2-y_1}\right] \quad \textbf{eq. 4-3}$$

The magnitude of its Mellin transform is described by Equation 4-4:

$$|M(ju,jv)| = \left|\frac{4}{uv}\sin(u\ln\frac{x_1}{x_2})\sin(v\ln\frac{y_2}{y_1})\right| \quad \textbf{eq. 4-4}$$

Equation 4-4 demonstrates the magnitude is only dependent on the ratios x_2 / x_1 and y_2 / y_1 and is, therefore, invariant to scale changes in its input variables x and y.

4.3 Mellin Transform Derivation from the Fourier Transform

In 1976, Casasent and Psaltis demonstrated the Fourier-Mellin transform, showing that a Mellin transform could be attained via the Fourier transform by a change of variable (as first demonstrated by Brousil [1967]). For simplicity, the one-dimensional case is presented here. Recall from section 3, Equation 3-1, that the Fourier transform of a function $f(x)$ (where $f(x)$ now represents $x(t)$ from equation 3-1, and ω, the natural frequency, now represents $2\pi f$) can be defined by Equation 4-5:

$$X(\omega) = \int_{-\infty}^{\infty} f(x)e^{-j\omega x}dx \quad \textbf{eq. 4-5}$$

Recall, as well, the definition of the Mellin transform indicated earlier (Equation 4-1) :

$$M(s) = \int_{0}^{\infty} f(x)x^{s-1}dx \quad \textbf{eq 4-1}$$

Now, if the variable x in the Mellin transform, Equation 4-1, is replaced with e^{ξ} and $s = -j\omega$, then the Mellin transform of $f(x)$ on the imaginary axis is the Fourier transform of $f(e^{\xi})$, indicated in Equation 4-6:

$$M(j\omega) = \int_{-\infty}^{\infty} f(e^{\xi}) e^{(-j\omega\xi)} d\xi \qquad \text{eq. 4-6}$$

In other words, the Mellin transform can be realized by logarithmically scaling the coordinates of the input function and Fourier transforming the resultant scaled function.

4.4 Achieving Rotational Invariance in the Fourier Transform

The Fourier transform of a two-dimensional function will not normally be rotationally invariant. In fact, as was indicated earlier in section 3.3.3, a rotation of a function in two dimensions by an angle θ will also have its Fourier transform rotated by the same angle. Rotational invariance of the Fourier transform may, however, be achieved by converting a rotation into a shift (or translation) in a single coordinate of the function. This shift can then be eliminated by considering only the magnitude of the Fourier transform which is shift-invariant.

If a two-dimensional function is expressed on a polar coordinate system, then any rotation of the image will manifest itself in only one of the coordinates of the polar system. Casasent and Psaltis [1976] demonstrated such a polar coordinate transformation, showing that a function $f(x, y)$ defined in Cartesian coordinates (x, y) can be equally expressed in polar coordinates as $f(\theta, r)$. In other words, if an image is considered to be a matrix of points, each of which is defined by a vertical x component and a horizontal y component, then the image can be equally represented by a radial component, r, and an angular component, θ. Figure 4-1 demonstrates this. Here point **P** of the car object can be represented in Cartesian coordinates, (x, y), where $x = 4$ and $y = 4$, or Point **P** can be

equally represented in polar coordinates, (θ,r), where $\theta = \tan^{-1}\left(\frac{4}{4}\right) = 45^{o}$ and

$r = \sqrt{4^2 + 4^2} = \sqrt{32}.$

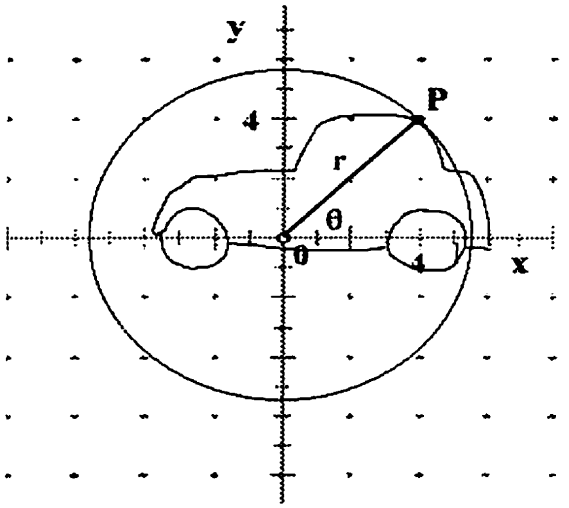


Figure 4-1: Cartesian to Polar coordinate conversion

Thus, if a function $f(x, y)$ is translated to a polar coordinate system as $f(\theta,r)$ then any rotation of the function will result in a change in only the angular coordinate θ. If the Fourier transform of the polar function is then taken, the rotational change in θ can be eliminated by considering only the magnitude which is shift (or translation) invariant. As a result, the magnitude of a Fourier-transformed image of some object that has been rotated will be identical to the magnitude of a Fourier-transformed image – containing the same object – that remained unrotated.

4.5 Log-Polar Transformation

The log-polar transformation is the means by which we are able to glean both the resultant scale-invariance of the Mellin transform as well as the resultant rotational-invariance provided by a polar conversion and Fourier transform. This transformation has been studied extensively by Araujo and Dias [1997], Wilson and Hodgson [1992], and Reitböck and Altmann [1984] among others, mostly in an effort to model the human visual system. The interest stems from the fact that in many primates and, likely, in human primates as

well, scenes projected onto the back of the eyeball (i.e. onto the retina) are mapped to the visual cortex in the brain in a logarithmically-polar fashion. We can describe this mathematically as an image function $f(x, y)$, appearing on the retina in a Cartesian plane, being approximately translated to appear at the visual cortex as if it had been sampled at the intersection of angular and exponentially-increasing radial points (i.e. as a function $f(\theta, \exp r)$). This is perhaps easier to understand through illustration. Shown in Figure 4-2 is a 256x256 pixel bitmapped image of the ever-radiant Lena, titled "Lena.bmp" (an image that has been used by the image processing and machine vision community since the 1960s). Also shown, in Figure 4-3, is an image representing how Lena would be re-sampled on a log-polar coordinate system, titled "Lena log-polar sampled". The white pixels indicate the sampling points which occur at the intersection of an angle and an exponentially increasing radius. Another image, Figure 4-4, adapted from Thornton [1998], shows the log-polar sampling grid with no underlying image, titled "Log-polar grid". Again, white pixels represent the points of log-polar sampling. Finally, the log-polar mapped image of Lena.bmp, adapted from Milanese and Cherbuliez [1999], is displayed in its log-polar coordinate system in Figure 4-5. This image is titled "Log-polar display of Lena.bmp".

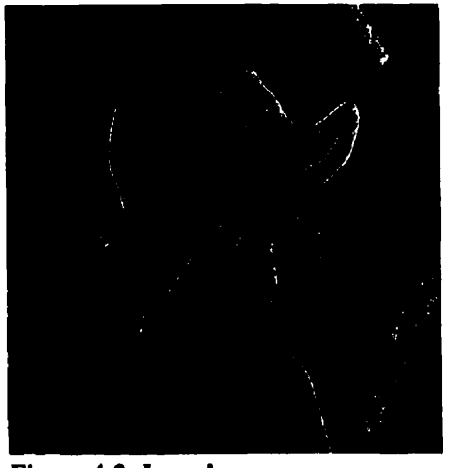


Figure 4-2: Lena.bmp

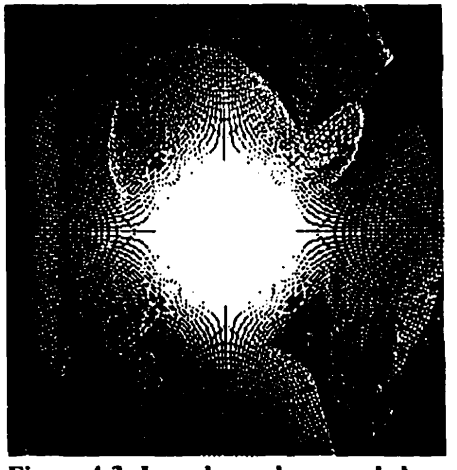


Figure 4-3: Lena log-polar sampled

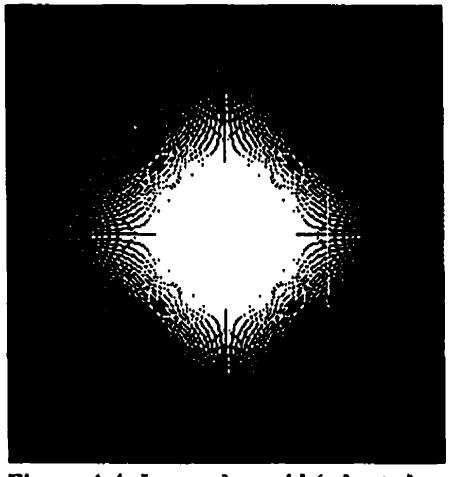


Figure 4-4: Log-polar grid (adapted from Thornton [1998])

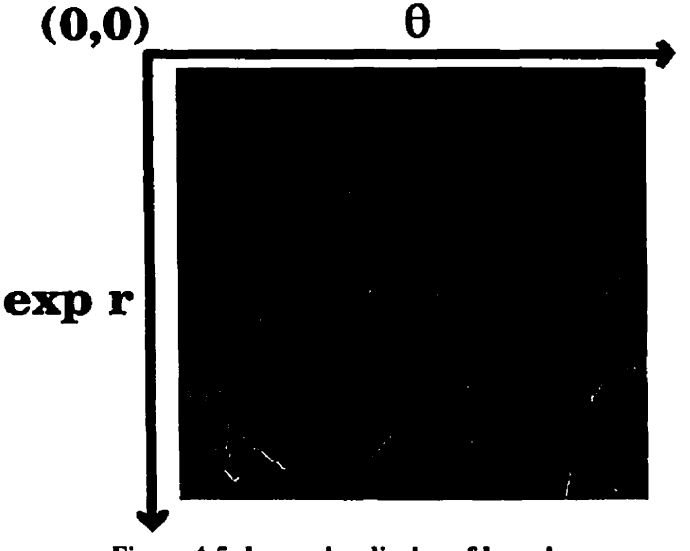


Figure 4-5: Log-polar display of Lena.bmp (adapted from Milanese and Cherbuliez [1999])

In Figures 4-3 and 4-4, there are 256x256 (65,536) log-polar sampled points. It is difficult to get an idea of how the log-polar sampling is occurring, except that it is obvious locations near the centre of the image are oversampled in comparison with points located further away on the periphery of the image. The grid in Figure 4-6 shows a 16x16 (256 pixel) log-polar sampling.

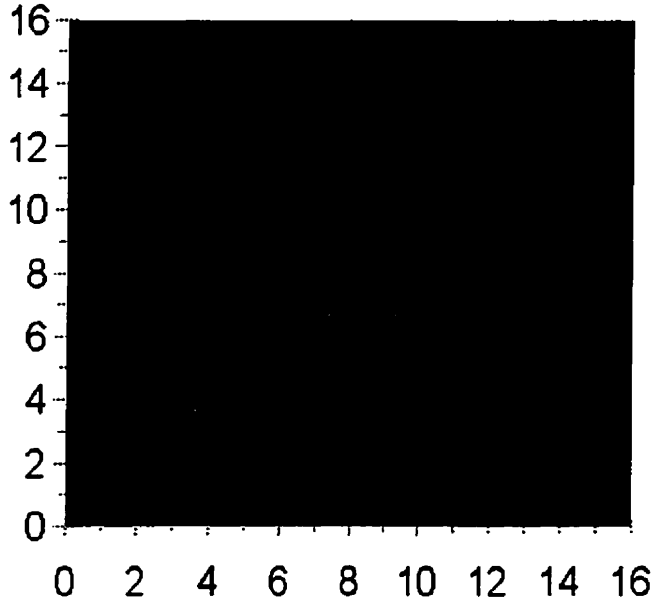


Figure 4-6: 16x16 Log-polar sampling grid

In this graphic it is easier to see that radial sampling intervals increase exponentially and that angular sampling intervals increase linearly by a constant.

The log-polar sampling of the image is the crux of the Fourier-Mellin transform in that this sampling is responsible for the scale and rotation invariance of the resultant Fourier-Mellin magnitude. To see why this is indeed the case, let's consider an image $f(x, y)$ and transform it onto a log-polar coordinate system. First, we assume the image centre is the starting point of the transformation. We can then represent each pixel in the image as occurring at a distance r from the image centre and at an angle θ (as represented earlier in Figure 4-1, the car on the polar coordinate system). If we now rotate the image, only θ will change, r will remain the same. Now, if instead of representing the second pixel

coordinate as a measure of r, we measured it on an exponential scale as $\log r$, then we can convert any scale changes in the image into shifts. Such would be the case if we scaled the image by a scale factor s. Now a pixel at Cartesian point $P(x, y)$ in the image, if scaled, would be represented as $P(\theta, \log(s*r))$. We exploit the fact that the log of a multiplication can be expressed as a sum of logarithms in order to convert scale changes into shifts. Thus, scaled point P, expressed on the log-polar coordinate system as $P(\theta, \log(s*r))$, can be equally represented as a shift: $P(\theta, \log s + \log r)$. Figures 4-7 and 4-8, adapted from Thornton [1998], and Araujo and Dias [1997], demonstrate how both changes in rotation and scale are converted to shifts by the log-polar transformation:

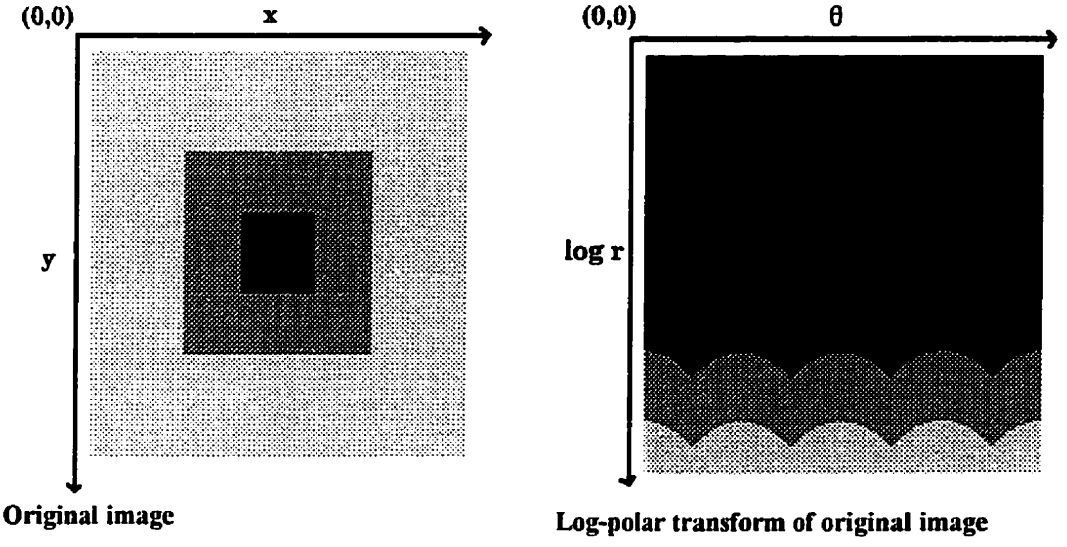
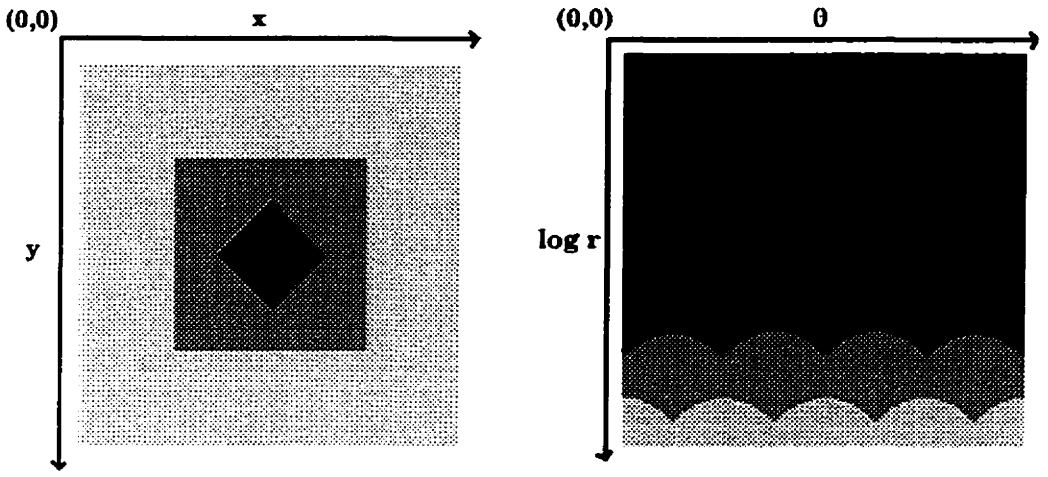


Figure 4-7: Scaled squares and log-polar transformations (adapted from Araujo and Dias [1997] and Thornton [1998])

Figure 4-7 shows how a scale change (larger and larger squares) in the original image on the Cartesian coordinate system, (x, y), results in a vertical shift along the log scale of the log-polar, $(\theta, \log r)$, coordinate system of the log-polar transformation of the original image.



Inner square rotation of original image

Log-polar transformation of inner square rotation of original image

Figure 4-8: Scaled – rotated central square – squares and log-polar transformations (adapted from Araujo and Dias [1997] and Thornton [1998])

Figure 4-8 shows how a change in rotation (the innermost square has been rotated by 45 degrees) in the original image results in a horizontal shift change of the angular (θ) component of the log-polar transformed image.

To summarise, the log-polar transformation permits us to express scale and rotational changes as shifts (translations). Shifts can then be removed by considering only the magnitude of the Fourier transform of the function expressed on a log-polar coordinate system. This is the basis for the Fourier-Mellin transform.

4.6 Implementation of the Log-Polar Transformation

The present implementation of the log-polar transformation is based on Thornton [1998] with some modifications. For a square image of size $N \times N$ pixels, we sample by determining what points in the original image correspond to each $(\theta, \log r)$ in the output image. In the output image, for both the angular coordinate, θ, and the radial coordinate, $\log r$, we can choose to divide the image into N divisions of each coordinate. Thus, θ is

sampled N times from 0 to $(2\pi - 1)$ angular divisions and $\log r$ is also sampled N times on an exponential scale from 0 to the maximum radius of the input image.

If the input image is mapped as a function of pixel locations indicated on a Cartesian grid (x, y), where x represents the row and y represents the column, the following equations are used to determine what points (x, y) from the original image will be used to provide for a log-polar sampling that results in a new image having a log-polar coordinate system $(\theta, \log r)$:

Let $\rho(row)$ represent the value of the exponentially increasing radial component of the input image. Equation 4-7 can then be used to determine each $\rho(row)$:

$$\rho(row) = \left[\frac{N}{2}\right]^{\left[\frac{row}{N-1}\right]} \textbf{eq. 4-7} \text{ , where } row = 0...N-1$$

Let $\theta(col)$ represent the value of the linearly increasing angular component of the input image. Equation 4-8 can then be used to determine each $\theta(col)$:

$$\theta(col) = col\left(\frac{2\pi}{col}\right) \textbf{ eq. 4-8} \text{ , where } col = 0...N-1$$

The centre of the input image is located at Cartesian coordinate $(\frac{N}{2}, \frac{N}{2})$. As a result, the x and y coordinates of the input image that correspond to each of the calculated $\theta(col)$ and $\rho(row)$ are determined using Equations 4-9 and 4-10 as follows:

$$x = \frac{N}{2} + \left[\rho(row)\cos(\theta(col))\right] \textbf{ eq. 4-9}$$

$$y = \frac{N}{2} - \left[\rho(row)\sin(\theta(col))\right] \quad \textbf{eq. 4-10}$$

The final output array will be of size $N \times N$. It will express θ in linear increments $(0...N-1)$ of its columnar data and an exponentially increasing r in linear increments $(0...N-1)$ of its row data.

5. Problems with the Discrete Fourier-Mellin transformation

Theoretically, the Fourier-Mellin transform should provide a truly translation-, rotation- and scale-invariant measure of an image. In practice, however, this is not the case. Problems arise in the digital implementation of the Fourier-Mellin transform that result in the discrete version of the transform merely approximating the continuous case [Altmann and Reitböck, 1984]. Chief among the problems leading to the divergence of the discrete and continuous Fourier-Mellin transforms are the negative effects of aliasing, leakage and interpolation, which will be addressed in the following sections.

5.1 Aliasing

In the digital implementation of the Fourier-Mellin transform, there is a likelihood of aliasing occurring whenever the signal is sampled. Aliasing is the process whereby a signal is converted into another signal – an aliased signal assumes or aliases the identity of the original signal – due to undersampling. This is demonstrated in Figure 5-1 (adapted from Smith [1997]):

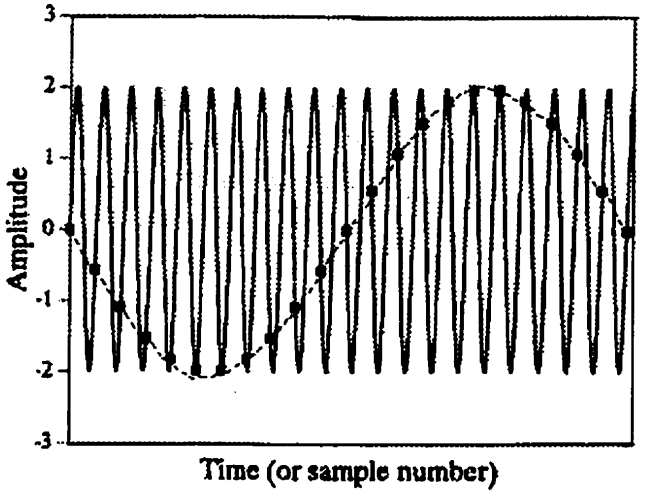


Figure 5-1: Aliased sampling (adapted from Smith [1997])

In Figure 5-1, a hypothetical signal (solid line) is being sampled (black squares) at a rate far less than the actual frequency of the hypothetical signal. The resulting signal is shown

as a dashed line (joining the sampling black squares); the original signal has become aliased and will now be falsely represented by the resulting signal due to undersampling.

The Nyquist theorem dictates that aliasing will not occur if the signal is sampled at least twice the rate of the signal's highest frequency component – termed the Nyquist frequency [Smith, 1997]. Another interpretation is that the highest frequency permitted in a signal must be less than or equal to one half the sampling rate. Adequate sampling is demonstrated in Figure 5-2 (adapted from Smith [1997]):

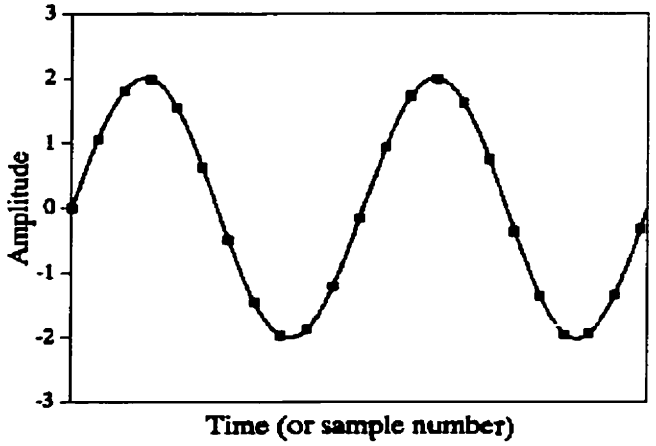


Figure 5-2: Non-aliased sampling (adapted from Smith [1997])

In Figure 5-2, the hypothetical signal is being sampled at over twice its highest frequency (the Nyquist frequency) component, and we can see that if we were to connect the sampling squares we would obtain the same underlying signal. Note that there is no danger in oversampling a signal. It simply creates more data. Though this may lead to more computation time in analysing the signal, no aliasing will occur.

5.1.1 Log-Polar Sampling May Result in Aliasing

One problem of the Fourier-Mellin transform is the possibility of aliasing occurring as a result of undersampling during the log-polar transformation. During a typical log-polar transformation, an image is sampled such that more samples are taken closer to the centre of the image than are taken near the periphery. To understand this, consider again Figure 4-6, the sampling grid for a 16x16 pixel image that was presented earlier in section 4.5:

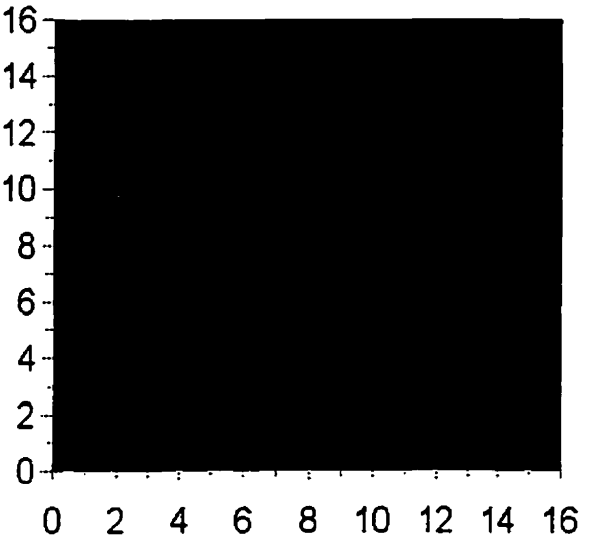


Figure 4-6: 16x16 Log-polar sampling grid

From Figure 4-6 we can see that as the radius of the sampling interval increases towards the periphery, more and more pixels are not sampled in between angular sampling intervals. As a result, pixels close to the centre of the image are oversampled and pixels near the periphery are undersampled. Though the oversampled pixels do not represent a problem, the undersampling at the periphery could lead to aliasing of the underlying image function.

5.1.2 Spatially-Variant Filter to Remove Possible Aliasing

In 1997, Thornton and Sangwine provided a means of removing the possibility of aliasing due to the undersampling inherent in a typical log-polar transformation. Essentially the implementation prevents aliasing from occurring in the first instance by sampling the entire image at or above twice the Nyquist frequency. Whereas in a typical log-polar transformation the outermost circumference is the most undersampled, in Thornton and Sangwine's implementation, the outermost circumference is sampled such that no pixels are missed. Consequently all lesser circumferences are oversampled, but, as mentioned earlier, this oversampling poses no danger of aliasing. In order to accomplish this, the angular sampling interval is increased approximately threefold such that a 256x256 image now results in a 805x256 array. The new log-polar sampling grid, adapted from Thornton

and Sangwine [1997] is represented in Figure 5-3 (compare this to the 256x256 log-polar sampling grid shown earlier in section 4.5 as Figure 4-4):

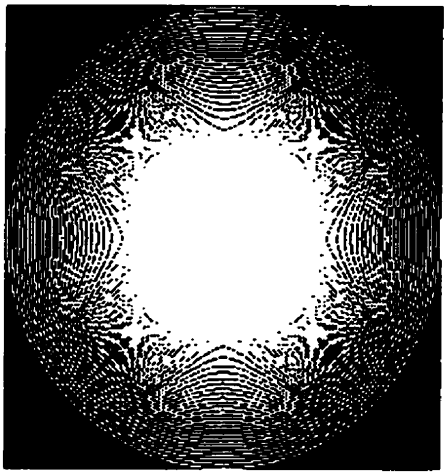


Figure 5-3: 805x256 log-polar sampling grid (adapted from Thornton and Sangwine [1997])

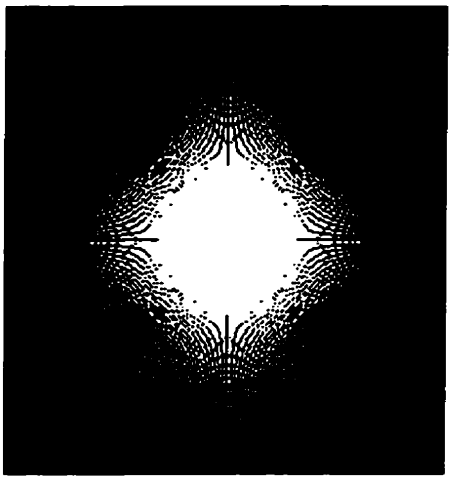


Figure 4-4: Log-polar grid (adapted from Thornton [1998])

This array is then filtered to average every two to three adjacent pixels on a circumference and downsampled back to the 256x256 image. Averaging the pixels in the oversampled image enables the downsampled image to be representative of the oversampled and non-aliased image function. The final result is a spatially-variant filtered image of the log-polar transformed original with any possible aliasing having been removed in the process.

5.2 **Leakage**

Another problem with the Fourier-Mellin transform is leakage. Leakage results from discontinuities in the input function to the DFT. In the application of the Fourier-Mellin transform in this thesis, there are two forms of discontinuity present in the input to the DFT: 1) there is discontinuity between the object (the foreground) in the image and its background, that is, at the object's edges; and 2) there is discontinuity at the borders of the entire image, that is, at the image's edges. The discontinuity between the image object (foreground) and background is self-evident; it is simply a sharp increase (if the background is darker than the foreground) in the pixel intensity at the edge of the object

with its background. The discontinuity resulting from the borders of the entire image, however, is not as obvious. This latter discontinuity results from the periodicity of the DFT, described in section 3.3.1 and can be explained as follows: the DFT views its input signal (and output signal) as being periodic and infinite. In the present implementation, the input happens to be a two-dimensional signal, that of an image. The consequences are that the DFT views the image as if its left and right sides as well as its bottom and top sides were joined together. In other words, it views the input image as being attached to itself on all sides and repeating infinitely. Figure 5-4, adapted from Baxes [1994], better illustrates this phenomenon:

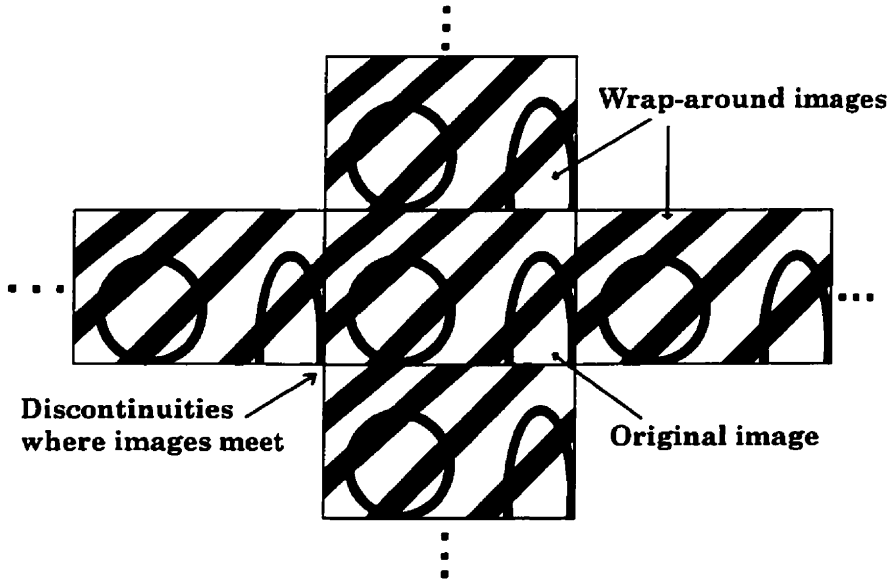


Figure 5-4: Periodicity of DFT input image (adapted from Baxes [1994])

The problem, of course, is that an image is not normally periodic or continuous – it does not repeat itself and its bottom and top, left and right sides do not normally match each other. As a result, the borders of an image are viewed by the DFT as discontinuities in the image function [Baxes, 1994; Pratt, 1991; Lim, 1990]. Both the discontinuities from the object edge and those from the image border will corrupt the results of the DFT of the image and will thus lead to errors when we attempt to compare otherwise translation-, rotation- and scale-invariant Fourier-Mellin image transforms [Reitböck and Altmann, 1984].

To see how the DFT of the image will be affected by these discontinuities and also how we can decrease their negative effects, we can model the phenomena mathematically. In essence, the discontinuities can be seen as resulting from the image function being multiplied by a rectangular function that has a value of 1 everywhere the image object and image exist and zero where they do not. The DFT of the rectangular function is the Sinc function [Smith, 1997]. This function has the general form $\sin(x)/x$ and is represented in Figure 5-5 (adapted from Smith [1997]).

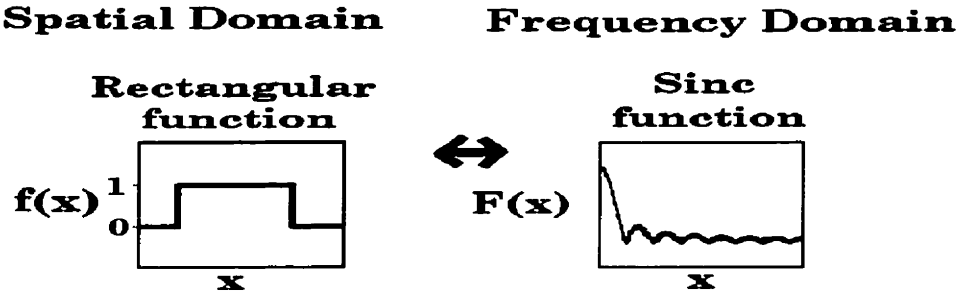


Figure 5-5: Rectangular and Sinc functions (adapted from Smith [1997])

The Sinc function can be seen as having one large mainlobe and smaller sidelobes. As pointed out earlier in section 3.3.5, multiplication in the spatial- or time-domain is equivalent to convolution in the frequency domain. As a result, when we multiply our image object and image by the rectangular function and take the DFT of the result, this is equivalent to a convolution of the DFT of our image function with the DFT of the rectangular function, which in the frequency domain is the Sinc function. Our image DFT becomes very corrupted by the Sinc function. Indeed, the frequencies that result from the DFT of our image become spread out due to the convolution with the sidelobes of the Sinc function. This spreading out of our image frequencies to other image frequencies is termed leakage, i.e. the frequencies from one particular location "leak" into another. Leakage degrades the results of our Fourier-Mellin transform significantly since we take not just one, but two DFTs during the Fourier-Mellin transform, making comparisons for similarity between image frequencies very difficult. Thus, we must limit the error due to leakage as much as possible.

5.2.1 Reducing Leakage by Windowing

In order to reduce leakage, we need to reduce the sidelobes of the Sinc function. One of the ways we can do this is to multiply our image function with a windowing or weighting function, the DFT of which has reduced sidelobes compared to those of the Sinc function [Harris, 1978]. This is the frequency domain solution to the problem. An alternative way to examine the problem, that is, in the spatial domain, is to find a windowing or weighting function that – instead of having a sharp cutoff from zero to 1 and then 1 to zero as does the rectangular function – has a gradual decrease towards a common value.
Multiplying such a windowing or weighting function with our image function would: 1) make the image object function's foreground gradually reach a common value with its background, eliminating the sharp edge between the object and its background; and likewise 2) make the image function's border gradually reach a common value with its opposite side, the side to which the DFT believes it is attached.

Figure 5-6, adapted from Pratt [1991], shows a very popular window, the Hamming window function in the one-dimensional case:

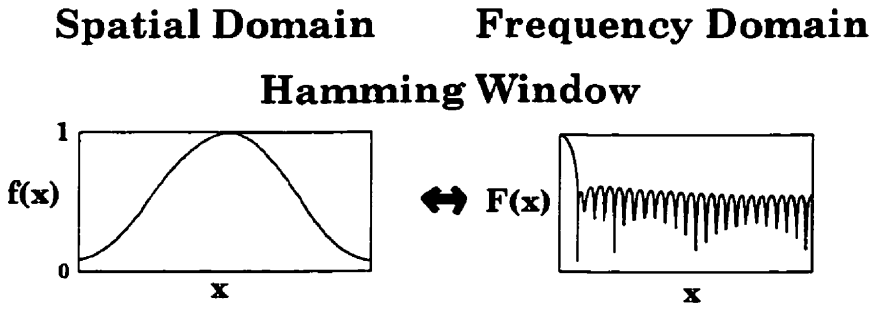


Figure 5-6: One-dimensional Hamming window (adapted from Pratt [1991])

And the following equation, Equation 5-1, modified from Pratt [1991], defines the one-dimensional Hamming window function:

$$f(x) = 0.54 - 0.46\cos\left(\frac{2\pi x}{N-1}\right) \quad \textbf{eq. 5-1}, \text{ where } 0 \le x \le N-1$$

and x represents the input sample point to be windowed; N is the total number of

sampling points in the discrete signal input to the DFT.

If we compare the spatial-domain representation of the Hamming window (Figure 5-6) to the previously illustrated rectangular window (Figure 5-5), we see that the ends of the Hamming window gradually taper to a common value near 0. If this function is multiplied by the image function at the correct position, then the object edge and borders of the image will also gradually taper to a common value near zero. Also, though it is not obvious in Figure 5-6, the Hamming window's frequency domain representation has much lower sidelobes than does the rectangular window. The tradeoff is that the Hamming window's mainlobe is broader than the rectangular window's mainlobe. As a result, it would be more difficult to distinguish lower frequency components from each other. This tradeoff is typical of all window functions; that is, a window function with very small sidelobes, producing little leakage at the higher frequencies, typically has a broader mainlobe, resulting in more leakage at the lower frequencies, than the rectangular function. In fact, windowing or weighting functions are usually rated on the performance tradeoffs between broad mainlobes and small sidelobes and are, therefore, application specific; in some applications, for instance, an investigator may be exclusively interested in the higher frequency components of the DFT and would, therefore, choose a windowing function that had very small sidelobes, unconcerned about the concomitant broad mainlobe. Some even argue the choice of a window is unimportant as long as one of them is used, indicating most windows achieve relatively the same results [Press *et al.*, 1986]. The Hamming window is a good compromise [Harris, 1978] and has been implemented in this thesis.

5.2.2 Implementation of the Hamming Window

Unfortunately, we are unable to apply the Hamming windowing function in the first case of leakage, that resulting from the object edge discontinuity. This is due to the fact that we do not know where the image object begins and ends and, thus, do not know where to apply the window. As an alternative, we implement the trivial solution of setting the

background intensity to the average pixel intensity of the object. Now the change in pixel intensity in going from background to object is less severe – producing less leakage – though not as gradual a change as could be achieved in the application of a window. In the second case of leakage, though, that resulting from the border effect of the image, we are able to apply the Hamming window function, since we do know where the image itself begins and ends. Note, however, that we need only apply the window to the image after it has been log-polar transformed and only in the radial component (vertical direction). The reasons for this are as follows: firstly, the input to the first DFT is a leaf image on a solid background, that is, the left, right, top and bottom borders all match with one another; there are, therefore, no discontinuities at the borders between the DFT-viewed periodic array of images and windowing becomes unnecessary. Secondly, though it is necessary to window the log-polar transformed image before inputting it to the second DFT, it is only necessary to window the non-continuous, exponential radial component, r, since the angular component, θ, is indeed continuous [Thornton, 1998]; that is, the angular component, θ, varies on the far left side of the log-polar transformed image from 0 to the far right side at 2π-1, which – in the DFT-viewed periodic array of images – again borders 0, i.e. the image data in the angular component form a continuous, periodic circle, one end matching the other. This is not the case, however, for the exponential radial component, r, the beginning and ending of which do not match (the top and bottom portions of the log-polar transformed image do not match). Thus, windowing becomes necessary.

To summarise, then, in order to reduce the leakage in the DFT caused by the discontinuities at the edges of the object in the image, the background is set to the average pixel intensity of the image object. To reduce leakage caused by the border effect of the image, we multiply the image input function's log-polar transformed radial component by the Hamming windowing function defined by Equation 5-1. In both cases, leakage will not have been completely removed, though it will have been reduced substantially.

5.3 Interpolation

In the digital implementation of the Fourier-Mellin transform, interpolation is required during the log-polar sampling of the image. The log-polar sampling will result in non-integer row and column image indices; however, all pixels in the image are located only at integer-valued indices. As a result, some form of interpolation is necessary and this will, of course, introduce some measure of error into the Fourier-Mellin transform. The simplest interpolation approach is to choose the closest integer-valued indices, termed nearest-neighbour, and assign these as points to be included in the log-polar re-sampled image. Nearest-neighbour interpolation, however, can result in a spatial offset error by as much as $\sqrt{2}/2$ pixel units [Pratt 1991]. A better method is to use bilinear interpolation. Bilinear interpolation provides for linearly interpolating along each row of an image and then, using the result, linearly interpolating along each column. The pixel value so found is an estimate of its four surrounding neighbours.

6. Implementation of the Fourier-Mellin Transform

Casasent and Psaltis [1977], Altmann and Reitböck [1984], Thornton and Sangwine [1997], Thornton [1998] and Milanese and Cherbuliez [1999] all implemented variations of the Fourier-Mellin transform. Elements of each of these implementations have been used in this thesis in order to obtain a Fourier-Mellin translation-, rotation- and scale-invariant descriptor of an image of a biological object. This is performed as follows:

Step 1: We set the background of each image to the average pixel intensity of the image object. This reduces leakage attributable to the image object edges as described in section 5.2.2. We then take the Fourier transform of the two images that are to be compared and retain only the magnitude. This step has two beneficial consequences: firstly, it removes any shift (or translational) differences between the two images. This means that if the two images contain identical objects but that each object is located in a different portion of the image, their Fourier-transform magnitudes will nevertheless be identical due to the shift invariance of the magnitude of the Fourier transform. Secondly, the Fourier-transformed magnitudes of the two images will now be centered [Thornton, 1998; Altmann and Reitböck, 1984]. This is beneficial because the magnitude will later undergo a coordinate change from Cartesian to log-polar that is more easily implemented if it is centered.

Step 2: The magnitudes of the two images are now shift- (or translation) invariant. The magnitude is then normalized by dividing each frequency component by its magnitude at the zeroth frequency. This cancels the multiplicative amplitude differences in pixel intensity that would have occurred if the image was scaled. [Altmann and Reitböck, 1984]. We now take the first step to enable rotation- and scale-invariance. To reiterate, we want any rotation and scaling in the images to be expressed as shifts that can be removed by taking the Fourier-transform and considering only the magnitude. The Mellin transform is scale-invariant. In order to obtain the Mellin transform, we can use the Fourier transform of a function whose input is expressed on an exponential scale. We also saw that rotation-

invariance can be achieved by converting the function coordinate system to a polar coordinate system and taking the Fourier-transform magnitude of that function. Both the scale-invariance of the Mellin transform and the rotation-invariance of a polar change and Fourier transform can be achieved, then, in one step by converting the input image function from a Cartesian coordinate system to a log-polar coordinate system via the log-polar transformation described in sections 4.5 and 4.6 and taking its Fourier transform. Thus, we take the log-polar transform of the image, not forgetting to then apply the spatially-variant filter, proposed by Thornton and Sangwine [1997], to reduce any aliasing due to undersampling as described in section 5.1.2. The result is the non-aliased function $f(\theta, \exp r)$.

Step 3: We have now converted all rotation and scale changes into shifts. As well, any translation changes were removed after the first Fourier-transform by considering only the magnitude. In order to remove the rotation and scale changes that were converted to shifts, we now, again, take another Fourier transform of the data and conserve the magnitude, remembering to first window the data in the exponential radial component with the Hamming window function to reduce the amount of leakage as described in section 5.2.2. This magnitude should now be completely translation-, rotation- and scale-invariant. Assuming the two images of the same object were different in some translation, rotation or scale, their final Fourier-Mellin magnitudes should now be identical.

7. Experimental Methodology

In the present application, the Fourier-Mellin transform has been used to compare images of different leaves that have been translated, rotated and scaled. There are three reasons why leaves were chosen as the objects used to demonstrate the effectiveness of the Fourier-Mellin transform. Firstly, it was desirable to apply the Fourier-Mellin transform to an application in agriculture; in this case, the Fourier-Mellin transform is being used as a machine vision analysis tool to aid in the identification of a biological object. Secondly, leaves are one of the indicators of plant speciation [Guyer, 1988]; therefore, being able to recognize a leaf aids us in identifying a plant of a particular species. Thirdly, leaves are relatively thin and flat such that two-dimensional images of leaves provide good representations of the actual object without having to deal with the encumbrances associated with variations in illumination, e.g. large shadows created by ridges or clefts in the object that would introduce additional difficulties into the recognition process.

Seven species of plant leaves were used in the experiment: avocado (*Persea americana*), trembling aspen (*Populus tremuloides*), lamb's-quarter (*Chenopodium album*), linden (*Tilia americana*), silver maple (*Acer saccharimum*), plantain (*Plantago major*) and sumac leaflets (*Rhus typhina*). All plant species, other than avocado, are native to the Montreal region; the specimens used were collected from plants located in Jeanne-Mance Park, Montreal, Quebec. The avocado leaves were collected from an avocado house plant. A leaf from each of the above-mentioned species, each between 4 and 6 cm in length, was placed on a flatbed scanner, scanned and digitized providing digital images of a resolution of 29 pixels per centimetre. All images contained the entire leaf. The final image size was maintained at 256x256 pixels. The following are examples of leaf images used in the experiment:

Figure 7-1: Aspen

Figure 7-2: Avocado

Figure 7-3: Lamb's-quarter

Figure 7-4: Linden

Figure 7-5: Maple

Figure 7-6: Plantain

Figure 7-7: Sumac

The following translated, rotated and scaled variations of leaves were taken: each leaf was rotated at approximately 12 degree intervals beginning at 12 degrees and ending at 72 degrees; each rotation was performed at a new location on the scanner. The leaves were rotated and translated by hand and re-scanned to avoid interpolation errors arising from attempting to rotate images using software. Leaves were then scaled between 40 percent of the original leaf size to 160 percent the original leaf size. These scaling limits were chosen for two reasons: firstly, on a practical note, scaling leaves more than 160 percent often resulted in the leaf exceeding the 256x256 pixel image size used in the experiments; and secondly, the literature indicates that the Fourier-Mellin transform is effective only when the object scale is less than 150 percent and greater than 50 percent. [Reddy and Chatterji, 1996; Raman and Desai, 1995; Chen *et al.* [1994], and Altmann and Reitböck, 1984]. Software (Jasc Software's Paint Shop Pro, version 5.03) was used to scale the leaf images since errors due to interpolation in scaling images are less severe. The leaf objects in the image were then measured for their approximate average pixel intensity. As described in section 5.2.2, this intensity value was used to produce a uniform background

around the leaf object foreground in order to reduce leakage. Leaf images were then input to the Fourier-Mellin transform. The transform was coded using Microsoft Visual C++ version 6.0. The radix-2 FFT implementation was adapted from code appearing in Smith [1997]. Salient components of the Fourier-Mellin algorithm's program code are listed in the appendices of this thesis.

In total, 203 leaf image comparisons were made. Each comparison required approximately 10 seconds in the debug software version of the programme, operating on an Intel Pentium I, 200 MHz system with 64 MB of RAM. The comparisons undertaken were as follows. Each leaf image within a species was compared to:

a) every translated and rotated version of the same leaf;

b) every rotated and scaled version of the same leaf;

c) every other plant species' leaf and three additional samples of the same species' plant leaf.

7.1 Euclidean Distance Measurement

The Fourier-Mellin transform of an image results in 0 to $N-1$ image frequencies. For a 256x256 image, $N = 256 \times 256 = 65536$; thus the amplitude of 65,536 frequencies from each image are available for comparison. A simple and fast method of evaluating frequency amplitude similarity is to use the Euclidean distance function [Milanese and Cherbuliez, 1999]. In our implementation, we define the Euclidean distance as (Equation 7-1):

$$\sum_{f=0}^{N-1} d_E(f) = \frac{1}{2}\sqrt{(\mathrm{Im}\, A_{(f)})^2 - (\mathrm{Im}\, B_{(f)})^2} \quad \textbf{eq. 7-1}$$

where $\mathrm{Im}\, A$ and $\mathrm{Im}\, B$ are the two images being compared and f represents the individual frequency index that runs from 0 to $N-1$. In using the Euclidean distance function as a measure of image similarity, images that are identical will have a Euclidean distance, d_E, of zero.

8. Results

The tables (1, 2 and 3) on page 49 of this thesis summarise the experimental results. All tables report measures of image similarity using the Euclidean distance function described in Equation 7-1. Table 1 reports the results of comparisons of original (untranslated, unrotated and unscaled) leaf images with their rotated and translated counterparts at 12 degree intervals, ranging from 0 degrees' to 72 degrees' rotation. Table 2 reports the results of comparisons of original (untranslated, unrotated and unscaled) leaf images with counterpart leaf images that have all been rotated 48 degrees and scaled between 40 percent and 160 percent of their original size. (48 degrees' rotation was chosen to ensure the whole leaf would still fit within the 256x256 pixel image when scaled maximally). Table 3 reports the results of comparisons of: 1) between-species leaf images, and 2) samples within the same species (last 3 columns).

From table 1, we observe that untranslated, unrotated and unscaled versions of the leaf images correspond very well with their translated and rotated versions. For instance, if we were to set the threshold for recognition at a Euclidean distance of 10, we would observe that all translated and rotated leaves would be recognized. There do not seem to be any noticeable trends associated with the extent of rotation, except that the Euclidean distance is at a minimum in all cases when the degree of rotation is at its minimum of 12 degrees.

From table 2, we see a marked difference when original images are compared with their scaled and rotated counterparts. Ideally, we should again set the recognition threshold at a Euclidean distance of 10 in order to include all the previous instances of recognition among rotated leaves. If we do this, we would find that only two out of the seven leaf species generally showed invariance to both scale and rotation, namely the aspen and sumac leaves. These two plant species' leaves are recognized at all scales within 50 percent to 130 percent, with the sumac plant leaves being recognized at a scale as low as 40 percent and as high as 140 percent. The same can not be said for leaves of other plant

species, where the rate of recognition is poor. In these cases, recognition is generally not achieved beyond a scale of 120 percent nor below a scale of 80 percent, and in one case, that of the linden plant, recognition is not achieved in any scaled version of the leaf. Unlike the observed results for rotation only, however, there does seem to be a trend in the recognition of scaled and rotated leaves: it is obvious that as the scale is increased or decreased, the rate of recognition falls off rapidly; Euclidean distances always increase with an increase or decrease in scale, reaching maximums at the highest and lowest scales.

From table 3, we observe that in most cases of comparing leaf images of one plant species to another plant species the Fourier-Mellin algorithm does not seem to allow us to differentiate among most of the various plants merely by comparing images of their leaves. The exceptions to this generality are: 1) the linden plant leaf, against which the system is able to differentiate all other plant species' leaves, and 2) the maple plant leaf, which records a minimum Euclidean distance difference of 20.34 when compared with the avocado plant's leaf. This Euclidean distance value is generally above most other Euclidean distance comparisons. Other between-species' plant leaf comparisons result in intermediate Euclidean distance values – all of which are less than 20 – and, as a result, do not indicate that the system found large differences in the images of the various plant species' leaves. Finally, when we examine the results for the same-species' additional leaf sample comparisons ("Within Species Comparison"– last three columns in table 3), it becomes evident that this thesis's implementation of the Fourier-Mellin algorithm was unable to recognize leaf images of the same species. This, however, is not surprising considering the poor results for the rotated and scaled leaf images; we would expect additional leaf samples from the same plant species to exhibit variations in scale and, therefore, large Euclidean distances are to be expected here among the within-species comparisons as well.

Measures of Image Similarity

	Translated and Rotated						
Leaf	0°	12°	24°	36°	48°	60°	72°
Aspen	0.00	4.31	4.92	5.32	4.91	5.16	4.99
Avocado	0.00	3.7	5.13	6.22	6.37	6.19	6.08
Lamb's-quarter	0.00	1.92	2.84	3.16	3.34	3.41	3.38
Linden	0.00	6.12	9.22	8.1	8.79	9.54	9.12
Maple	0.00	5.66	7.61	8.31	8.42	8.16	8.25
Plantain	0.00	2.31	3.14	3.68	4.01	2.51	2.82
Sumac	0.00	2.42	2.94	3.13	3.44	3.21	3.27

Table 1: Original (untranslated, unrotated, unscaled) versus translated and rotated leaf images

	Rotated and Scaled											
Leaf	48° 40%	48° 50%	48° 60%	48° 70%	48° 80%	48° 90%	48° 110%	48° 120%	48° 130%	48° 140%	48° 150%	48° 160%
Aspen	10.13	9.26	8.33	6.67	5.74	4.89	5.65	7.24	8.67	11.3	14.26	16.88
Avocado	23	20.42	17.09	14.89	10.86	8.05	7.11	9.74	14.02	17.46	21.86	26.8
Lamb's-quarter	16.44	14.43	12.05	9.87	6.5	4.6	5.89	8.22	12.8	16.14	20.49	24.35
Linden	114.37	98.18	78.83	60.27	40.18	20.57	20.89	37.64	54.14	67.19	79.5	91.58
Maple	29.4	23.72	18.05	13.78	10.95	9.12	10.27	13.78	16.97	19.49	21.6	23.49
Plantain	16.8	14.3	11.21	8.13	5.55	2.99	8.14	12.55	15.61	20.57	25.92	31.4
Sumac	7.26	6.08	4.91	3.37	2.94	2.67	4.56	6.58	7.41	9.42	10.86	12.51

Table 2: Original (untranslated, unrotated and unscaled) versus rotated and scaled leaf images

	Between Species Comparison							Within Species Comparison		
Leaf	Aspen	Avocado	Lamb's-quarter	Linden	Maple	Plantain	Sumac	Sample 2	Sample 3	Sample 4
Aspen	0.00	19.45	12.54	139.19	36.82	13.32	6.34	11.26	29.73	83.61
Avocado		0.00	9.29	121.89	20.34	8.31	18.14	11.22	43.7	43.62
Lamb's-quarter			0.00	128.65	26.83	4.77	11.01	38.68	14.58	18.9
Linden				0.00	104.06	127.59	137.55	21.79	11.27	49.36
Maple					0.00	25.35	35.64	20.01	21.73	14.3
Plantain						0.00	12.02	16.78	23.54	23.08
Sumac							0.00	32.81	47.96	60.25

Table 3: Comparison of original leaf images with: a) other species leaf images ("Between Species Comparison"), and b) additional samples of same species' plant leaf images ("Within Species Comparison")

9. Discussion

Overall, the application of the Fourier-Mellin algorithm used to recognize leaf images among their translated, rotated and scaled counterparts was not very robust. Though excellent results were reported when the leaf image was merely translated and rotated, generally poor results arose in the case of rotated and scaled leaf images as well as in comparisons with the between- and within-species groups. The system's failure to recognize scaled leaf images is particularly acute since this may be the reason why there was a lack of recognition among additional samples of same plant species' leaves, leaves that would not likely have been the same size as the original leaf. It is difficult to stipulate exactly why this failure to recognize scale differences occurred. One obvious reason is that the image frequencies may have been corrupted by leakage since, as was indicated in section 5.2.2, this could only be poorly controlled, especially at the object image edges. As well, further reducing object-edge leakage may not be possible. For instance, determining the image object boundaries to enable application of a windowing function is a large problem in its own right. While a trivial solution might entail using edge detection, perhaps enabling a rough mapping of the object based on the assumption that the object boundary would equate to the longest edge, even if we knew exactly where the object were located in the image, it would still be necessary to derive a windowing function that could be applied symmetrically to any and all variations of image objects such that they could blend gradually into their individual backgrounds – certainly not an easy feat and, more than likely, not possible. A more forgiving solution might be to limit the size of the allowable scaling of the object and apply various windowing functions according to each image object's approximate scale. This method, combined with setting the background to the object image's average pixel intensity, may provide better results than relying solely on the present implementation for leakage reduction. The results may also be further improved by applying a different windowing function to decrease the border effect leakage, perhaps one with a narrower mainlobe as most of the magnitude information of an image seems to be contained in the lower frequencies [Lim, 1990]. Though, as mentioned earlier in section

5.2.1, the literature indicates there are only small differences between various windows, this would indeed be the least difficult change to implement and may, nevertheless, be worthwhile attempting. Another alternative might even be to apply various windows and average the results. Doing so would reduce the efficiency of the algorithm, doubling the calculation time with every newly applied window, but the benefits might far outweigh the speed loss.

Of course another obvious possibility for the scale recognition failure is that the variation within each leaf species is simply too great at the whole-leaf image level resolution. The algorithm is perhaps unable to capture the light intensity frequency changes that all whole-leaf image specimens of a single species have in common. A solution might be to examine leaf images at a much greater resolution. In other words, an attempt could be made to digitize leaf images at a much higher scale of magnification. In doing so, we would seek to classify a common texture or pattern that would be particular to but one species of leaf. Analysing such a pattern or texture would not, however, be without its own set of difficulties. For instance, much experimentation would be required to determine the necessary image magnification that would provide this invariant pattern common to all leaf variations within a single plant species. Additionally, it is entirely possible, even likely, that the necessary image magnification would be species dependent and, therefore, would have to be determined experimentally for each and every plant species' leaf...but, then, such are the difficulties involved in machine vision.

Yet another reason for the recognition process's shortcomings might be the method used for evaluating similarity between the frequency domains of the two images; relying on the Euclidean distance function to provide an accurate measurement of the differences in image frequency domains may be flawed depending on the frequency representation of the image object we wish to identify. The Euclidean distance measurement is, after all, a weighted measure: it is very much dependent on all of the frequencies in the image, not just those that represent the object (in our case, the leaf). In the present application, it is

calculated by comparing each of the 65,536 different frequencies and adding any differences together. Though simple and fast, the Euclidean distance function would prove to be a poor estimate of image object content if the objects in the images – the leaves – were to represent only a small portion of the total frequency content of the image and their frequencies happened to be made up of low amplitudes relative to the rest of the image. It was not possible, however, to determine if this was indeed the case. Ideally, we would like to compare only those frequencies associated with the image object. However, at this point in the evolution of machine vision, there does not seem to be an accurate method of determining *a priori* the frequencies responsible for generating the image object alone. An alternative method of comparing images is to use correlation, a variant of convolution which, in the frequency domain, multiplies the conjugate of one image with the second image. Correlation is, however, much more complicated to implement, requires much longer computation and, like the Euclidean distance measure, requires the determination of a threshold at which objects are either recognized or not. Nevertheless, it may prove worthwhile to attempt to use correlation as a method of image frequency comparison.

Finally, as was mentioned in section 3.6.3, a comparison was made between the information contained in the phase to that contained in the magnitude of an image. In short, it seems the phase contains much of the spatial information contained in an image, that is, where edges of objects begin and end as well as where an object is situated in the image proper. Thus, phase information could serve us well in both differentiating between plant leaves of different species – based on shape differences – as well as in recognising plant leaves within the same species – based on shape similarities. The problem, of course, is that the phase is neither translation-, rotation- nor scale-invariant, unlike the Fourier-Mellin magnitude. A solution would be to calculate the rotation and scale differences of one image compared to the other, associate these differences somehow with the phase of the compared image and then correlate the phases of both images. Thornton [1998] demonstrates just this in her thesis on colour object recognition. Though the algorithm is much more complicated, time consuming and was not applied towards the recognition of

biological objects, the method may indeed prove more effective in comparing different species of plants' leaves because it does not discard valuable information contained in the objects' phase images.

10. Summary and Conclusion

In this thesis the Fourier-Mellin algorithm was implemented to attempt to recognise plant leaf images regardless of their translation, rotation or scale. The results indicated it was possible using the transform to recognise plant leaf images despite changes in translation and rotation; however, the rate of recognition was poor for scaled and rotated plant leaf images as well as for additional samples of same-species plant leaf images. Differentiation was also inadequate between images of various species of plant leaves.

Future efforts to improve the success of recognition and differentiation of the present implementation of the Fourier-Mellin algorithm include: 1) seeking better methods of reducing leakage resulting from the edges of the object image and the image borders; 2) applying the transform to leaf images at a magnified resolution in order to better represent the texture of the leaf; 3) using correlation instead of Euclidean distance to measure the similarity between leaf images, and 4) attempting to include the phase of the leaf image along with the Fourier-Mellin magnitude to better characterize the leaf image object. Barring the complete success of these measures, it is entirely possible the Fourier-Mellin tranform can not tolerate the variation found in images of the biological objects – the leaves – considered in this thesis; no previous implementation of the transform found in the literature, for instance, was ever applied towards the recognition of biological objects. If the biological variation of the leaves was indeed the reason for the transform's failure, then it is obvious much more research is required to provide for a more robust Fourier-Mellin implementation. Moreover, though a better implementation of the Fourier-Mellin algorithm may be possible, as mentioned in the introduction of this thesis it would be foolish to expect a single algorithm to provide for complete recognition of a biological object. The Fourier-Mellin algorithm should, therefore, be considered as one among many analysis tools in a general machine vision system used to recognise biological objects.

11. Bibliography and References

Bibliography

Bracewell, R.N., The Fourier Transform and its Applications, McGraw Hill, New York, 1986.

Brigham, E.O., The Fast Fourier Transform and its Applications, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

Lyons, R. G., Understanding Digital Signal Processing, Addison Wesley Longman, Reading, Massachusetts, 1997.

Smith, S. W., The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Publishing, San Diego, 1997.

References

Altes, R. A., "The Fourier-Mellin transform and mammalian hearing", *Journal of the Accoustical Society of America*, V. 63, No. 1, January, 1978, 174-183.

Altmann, J., Reitböck, H. J., "A Fast Correlation Method for Scale- and Translation-Invariant Pattern Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, V. PAMI-6, No. 1, January 1984, 46-57.

Araujo, H., Dias, J. M., "An Introduction to the Log-Polar Mapping", *Proceedings of the IEEE, 2nd Workshop on Cybernetic Vision 1996*, 1997, 139-144.

Baxes, G. A., Digital Image Processing. Principles and Applications, John Wiley & Sons, New York, 1994, p. 37, p. 99, pp. 105-106.

Bracewell, R.N., The Fourier Transform and its Applications, McGraw Hill, New York, 1986, p. 254.

Brigham, E. O., The Fast Fourier Transform and its Applications, Prentice Hall, Englewood Cliffs, New Jersey, 1988, p. 32, pp. 50-51, p. 240.

Brousil, J. K., Smith, D. R., "A Threshold Logic Network for Shape Invariance", *IEEE Transactions on Electronic Computers*, V. EC-16, No. 6, December 1967, 818-828.

Casasent, D., Psaltis, D., "Scale Invariant Optical Transform", *Optical Engineering*, V. 15, No. 3, May-June 1976, 258-261.

Casasent, D., Psaltis, D., "New Optical Transforms for Pattern Recognition", *Proceedings of the IEEE*, V. 65, No. 1, 1977, 77-84.

Castleman, K., Digital Image Processing, Prentice Hall, Upper Saddle River, New Jersey, 1996, p. 197, p. 201.

Chen, Q., Defrise, M. Deconinck, F., "Symmetric Phase-Only Matched Filtering of Fourier-Mellin Transforms for Image Registration and Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, V. 16, No. 12, December 1994, 1156-1168.

Cooley, J., Tukey, J., "An Algorithm for the Machine Calculation of Complex Fourier Series", *Mathematics of Computation*, V. 19, No. 90, 1965, 297-301.

Derrode, S. "Représentation de formes planes à niveaux de gris par différentes approximations de Fourier-Mellin analytique en vue d'indexation de bases d'images", PhD Thesis, Rennes University, December 1999, p. 19.

Franz, E., Gebhardt, M. R., Unklesbay, K. B., "Shape Description of Completely Visible and Partially Occluded Leaves for Identifying Plants in Digital Images", *Transactions of the ASAE*, V. 34(2), March-April, 1991, 673-681.

Guyer, D. E., "Application of Machine Vision to Human Shape Analysis Techniques in Leaf and Plant Identification: An Intelligent Vision Structure", PhD Thesis, Purdue University, December 1988, p. 21.

Harris, F. J. "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform", *Proceedings of the IEEE*, V. 66, No. 1, January 1978, 51-83.

Lim, J. S., Two-Dimensional Signal and Image Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1990, pp. 39-40.

Lin, C-Y, Wu, M., Bloom, J. A., Cox, I. J., Miller, M. L., Lui, Y. M., "Rotation, scale, and translation resilient public watermarking for images", *Proceedings of SPIE Conference on Security and Watermarking of Multimedia Contents II*, V. 3971, May 2000, 90-98.

Lyons, R. G., Understanding Digital Signal Processing, Addison Wesley Longman, Reading, Massachusetts, 1997, p. 49, p. 63, p. 68.

Milanesse, R., Cherbuliez, M., "A rotation, translation and scale-invariant approach to content-based image retrieval", *Journal of Visual Communication and Image Representation*, V. 10, No. 2, 1999, 186-196.

Oppenheim, A. V., Lim, J. S., "The Importance of Phase in Signals", *Proceedings of the IEEE*, V. 69, No. 5, May 1981, 529-541.

Ó Ruanaidh, J. J. K., Pun, T., "Rotation, Scale and Translation Invariant Digital Image Watermarking", *Proceedings of the IEEE International Conference on Image Processing*, V. 1, 1997, 536-539.

Parker, J. R., Algorithms for Image Processing and Computer Vision, John Wiley & Sons, New York, 1997, p. 226.

Pratt, W. K., Digital Image Processing, 2nd ed., John Wiley & Sons, New York, 1991, p. 199, pp. 441-442.

Press, W. H., Flannery, B. P., Vetterling, W. T., editors, Numerical Recipes. The Art of Scientific Computing., Cambridge University Press, Cambridge, 1986, p. 425.

Raman, S. P., Desai, U. B., "2-D Object Recognition using Fourier Mellin Transform and a MLP Network", *Proceedings of the IEEE International Conference on Neural Networks*, V. 4, 1995, 2154-2156.

Reddy, B. S., Chatterji, B. N., "An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration", *IEEE Transactions on Image Processing*, V. 5, No. 8, August 1996, 1266-1271.

Reitböck, H. J., Altmann, J., "A Model for Size- and Rotation-Invariant Pattern Processing in the Visual System", *Biological Cybernetics*, V. 51, 1984, 113-121.

Robbins, G. M., Huang, T. S., "Inverse Filtering for Linear Shift-Variant Imaging Systems", *Proceedings of the IEEE*, V. 60, No. 7, July 1972, 862-872.

Schalkoff, R. J. Digital Image Processing and Computer Vision, John Wiley & Sons, New York, 1989, p. 285.

Shatadal, P., Jayas, D. S., Bulley, N., R., "Fourier and Spatial Domain Analysis of Image Texture" [in: Automated Agriculture for the 21st Century. Proceedings of the 1991 Symposium,American Society of Agricultural Engineers, Michigan, 1991.] 36-41.

Sheng, Y., Arsenault, H. "Experiments on pattern recognition using invariant Fourier-Mellin descriptors", *Journal of the the Optical Society of America A*, V. 3, No. 6, June 1986, 771-776.

Sheng, Y, Shen L., "Orthogonal Fourier-Mellin moments for invariant patern recognition", *Journal of the Optical Society of America A*, V. 11, No. 6, June 1994, 1748-1757.

Smith, S. W., The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Publishing, San Diego, 1997, p. 42, p. 157, p. 195, p. 237, p. 192, p. 212-213.

Thornton, A.L., "Colour object recognition using a complex colour representation and the frequency domain", PhD Thesis, University of Reading, May 1998, pp. 48-49, p. 52, p. 97.

Thornton, A. L., Sangwine, S. J., "Log-Polar Sampling Incorporating a Novel Spatially Variant Filter to Improve Object Recognition", *IEE Conference on Image Processing and its Applications*, IPA97, 15-17 July 1997, 776-779.

Wilson, J. C., Hodgson, R. M., "A Pattern Recognition System Based on Models of Aspects of the Human Visula System", *IEE 4th International Conference on Image Processing and its Applications*, 1992, 282-285.

Zhang, N., Chaisattapagon, C., "Effective Criteria for Weed Identification in Wheat Fields Using Machine Vision", *Transactions of the ASAE*, V. 38, No. 3, 1995, 965-974.

Zwicke, P.E., Kiss, I. Jr., "A New Implementation of the Mellin Transform and its Application to Radar Classification of Ships", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, V. PAMI-5, No. 2, March 1983, 191-199.

12. Appendices: Software Programme Code

All of the software was written and compiled using Microsoft Visual C++ version 6.0. Included here are salient portions of the programme code that relate specifically to the: FFT implementation (appendix A); the log-polar transformation, bilinear interpolation and application of the spatially-variant anti-aliasing filter (appendix B); the application of the Hamming window (appendix C), and the sequence of steps to perform the Fourier-Mellin transformation (appendix D). Also included in appendix E is a display of the programme interface.

Appendix A: FFT implementation

```
//Step 1 of FFT: decomposition of N point spatial domain signal
//(represented by Real and Imag arrays) into N signals in frequency domain
//each containing one point. Real and Imag array elements must be placed in an order
//that corresponds to swapping indices that are bit reversals of each other - repeat for
//columns

void Fourier::Bit_reverse_rows(unsigned int row)
{
    unsigned int Bit_reversed_index;

    //loop through each column
    for (unsigned int Original_index=0;Original_index<width;Original_index++)
    {
        Bit_reversed_index = Reverse_bit(Original_index);

        //Only swap if indexes are not equal and were not swapped earlier (elements would
        //have been swapped earlier if the Bit_reversed index returned were smaller than
        //the Real_array_index)
        if (Bit_reversed_index>Original_index)
        {
            swap(Real_biD_base[row][Original_index],
                Real_biD_base[row][Bit_reversed_index]);
            swap(Imag_biD_base[row][Original_index],
                Imag_biD_base[row][Bit_reversed_index]);
        }
    }
}


//Fxn to perform bit reversal for 2D data

unsigned int Fourier::Reverse_bit(unsigned int array_index)
{
    unsigned int bit_rev=0, upperlimit=height-1;
    //We want to limit the values from 0->size-1 (lower bound->upper bound of input array)
    //The lower bound -> upper bound must be a power of 2 (i.e. 2 4 8 16 32 64 128 256)
    while(upperlimit)
    {
        bit_rev = (bit_rev << 1) | (array_index & 1);
        array_index >>= 1;
        upperlimit >>= 1;
    }
    return bit_rev;
}
```

```
//Step 2 of FFT: synthesis of frequency signals: combines sqrt(image_size)
//bit-reversed points into their frequencies
//Do rows then repeat for columns

void Fourier::Butterfly_rows(unsigned int row)
{
    //we do log2(sqrt(image_size)) number of outer loops==sqrt(image size) in bits
    //ie. we do number of cycles equal bit number of elements in each row/col
    unsigned int cycles = (log10(height)/log10(2));

    //declare indices:
    unsigned int x, y, z, y_minus_one, index;

    //declare local variables:
    double Real_mult, Imag_mult, Real_ofs, Imag_ofs, Real_cos, Imag_sin;
    unsigned int powerbit;

    //loop from 1 to height or width in bits
    for(x=1;x<=cycles;x++)
    {
        powerbit = (unsigned int) pow(2, x);//values from 2->height or width
        Real_cos = cos((double)PI/(double)(powerbit/2));
        Imag_sin = -sin((double)PI/(double)(powerbit/2));
        Real_mult = 1.0;
        Imag_mult = 0.0;

        for(y=1;y<=(powerbit/2);y++)//loop for each sub DFT
        {
            y_minus_one=y-1;
            //loop for each "butterfly"
            for(z=y_minus_one;z<width;z+=powerbit)
            {
                index = z + (powerbit/2);
                Real_ofs = Real_biD_base[row][index]*Real_mult -
                    Imag_biD_base[row][index]*Imag_mult;
                Imag_ofs = Real_biD_base[row][index]*Imag_mult +
                    Imag_biD_base[row][index]*Real_mult;
                Real_biD_base[row][index]=Real_biD_base[row][z]-Real_ofs;
                Imag_biD_base[row][index]=Imag_biD_base[row][z]-Imag_ofs;
                Real_biD_base[row][z] = Real_biD_base[row][z] + Real_ofs;
                Imag_biD_base[row][z] = Imag_biD_base[row][z] + Imag_ofs;
            }
            Real_ofs = Real_mult;
            Real_mult = Real_ofs * Real_cos - Imag_mult * Imag_sin;
            Imag_mult = Real_ofs * Imag_sin + Imag_mult * Real_cos;
        }
    }
}
```

Appendix B: Code for log-polar transformation, bilinear interpolation and application of anti-aliasing filter

```
/*Funtion to perform:

    1)log-polar transformation on FFT magnitude:
    Takes FFT amplitude spectrum (magnitude) values from Cartesian coordinate system and
    transforms them on to a log-polar coordinate system.
    2)Oversampling is performed on each radius by increasing the angular
    sampling intervals to SamplingSize (==height or width of image) X PI.
    3)Oversampled input is then filtered using moving average filter mask
    of approximately PI width (3) on each radius.
    4)Final output is downsampled back to SamplingSize X SamplingSize-sized
    matrix for another FFT run.

    Oversampling is used to remove alias that would otherwise be caused
    by regular LPT sampling. (implemented according to Thornton and Sangwine [1997]).
*/

void COpsDoc::FM_Log_Polar_Transform(float **tempPtr, float **FMresult, WORD height)
{
    //Declare and assign height, width of image to height, width variables
    WORD Input_ImageHeight, Input_ImageWidth;
    Input_ImageHeight = Input_ImageWidth = height;
    double Oversample = PI;//define times more than image height size we wish to sample

    //Declare, allocate memory for a temp matrix for the Oversampled input FFT magnitude
    float **FMtempPtr=biD_Matrix_uneven(SampleSize, ((int)(SampleSize*Oversample)));

    /*Perform the log-polar transformation.
    1. We have a matrix to hold the log-polar transformed data, passed to this fxn
    as FM_tempPtr. This array has 0 to SampleSize-1 number of rows and 0 to SampleSize-1
    number of columns.
    2. The log-polar transform samples points from the power spectrum (magnitude) of
    the FFT for an image. In so doing it will sample points at the intersection of:
    column number of 2PI divisions with row number of exponential FFT power spectrum
    samplings.
    3. Angular division samplings will occur every: 2PI*col0/column-1, 2PI*col1/column-1,
    2PI*col3/column-1.....2PI*column-1/column (i.e. 0->2PI-1). As such, this is a linear
    sampling.
    4. Exponential division samplings will occur at exponential radial increments in the
    original FFT power spectrum data. We determine these divisions as follows:
    For an NxN FFT power spectrum dataset, we will sample from 0...rho-1.
    Max rho = maximum number of radii in the output log-polar transformation.
    rho represents the exponential radial sampling of the original FFT power spectrum.
    If Rht = maximum radius possible in the original FFT power spectrum, then it will
    be equal to (N/2)-1 in pixel height.
    To determine which exponential radius, r, will be used as a sample intersection
    point, we have the following relationship:

    rho(row) = Rht ^ (row/Max_rho - 1)

    where:

      row varies from 0....Max rho -1
      Rht = N/2-1
      Max_rho = maximum number of radii in output log-polar array

    N.B.
    1) the centre point is never sampled, but it will contribute to the sample if values
    arecalculated by bilinear interpolation.
    2) the first column and row (column 0 and row 0) are not sampled.
    No. 2 occurs because in order to have the centrepoint (0 theta, 0 rho)
    at N/2, N/2 be a true centrepoint , we must have a maximum radius of (N/2)-1 and
    in an NxN matrix it would be impossible to have a true centre without eliminating
    or adding one row and one column (because otherwise the centrepoint occurs at
    the joining of the four quadrants, which is not a pixel location). We could
    have added a row and a column, reproducing data in it from their opposite sides,
    but then the matrix size would be larger than the original power of 2 and no
    further FFT could be performed without deleting the added row and column.
```

```
The loss of the data from row 0, column 0 is not substantial because row 0
and column 0 represent data of the highest frequencies from the FFT and are
usually so small in amplitude as to be negligible.

3) Bilinear interpolation is used for indices that occur between integer indices
Bilinear interpolation formula according to Pratt [1991].
Bilinear interp. is default unless user changes to nearest neighbour.
*/

//Define variables:
double rho, theta;//rho = input radius, theta = input angle
double Rht = (Input_ImageHeight/2)-1;//maximum radius input in pixels -- regular LPT
double Max_rho = SampleSize - 1;//maximum number of radii for output (0->SampleSize-1)
int Cx,Cy;//Cint x, y input matrix indices
float a, b, InterpX, InterpY;//b = decimal portion of x, a = decimal portion of y
int IntX, IntY;//IntX, IntY integer portions of calculated x, y indices

//rho holds exponential radius value
//theta holds angle
//x, y indices to values in FFT power spectrum array


//Sample using bilinear interpolation
if(BilinearInterpolationFlag)
{
//Calculate indices and perform log-polar sampling
for(WORD u = 0; u<SampleSize;u++)//u is the row index and maps to rho
    for(WORD v = 0;v<((long)(SampleSize*Oversample));v++)//v is the column index and
                                //maps to theta
    {
        rho = pow(Rht,(u/Max_rho));//for exponential sampling of radius
        theta = (2*PI*(double)v)/((long)(SampleSize*Oversample));//calculate theta:
                    //runs from 0 to (2PI-1 angular division)

        //Nearest-neighbour interpolation (used w/in Bilinear interpolation formula):

        //Calculate rounded-to-nearest-integer indices to input FFT power spectrum
        //Cint global fxn. (defined in StdAfx.cpp) rounds a double to nearest int
        Cx = Cint((Input_ImageHeight/2) + (rho*cos(theta)));//round to nearest
                                    //integer
        Cy = Cint((Input_ImageHeight/2) - (rho*sin(theta)));//round to nearest
                                    //integer

        //Bilinear interpolation:

        //First get the actual calculated floating point indices
        InterpX = (Input_ImageHeight/2) + (rho*cos(theta));
        InterpY = (Input_ImageHeight/2) - (rho*sin(theta));

        //Get the int portions of x, y indices
        IntX = (int) InterpX;
        IntY = (int) InterpY;

        //Get the decimal portions of the x, y indices
        b = (InterpX - IntX);
        a = (InterpY - IntY);

        //Bilinear Interpolated pixel value of log-polar sampling
        //Only use interpolation if indices are valid (i.e. do not attempt
        //to address an index that does not exist in the matrix)
        if((IntY >= 0) && (IntX >= 0) && ((IntY + 1) < Input_ImageHeight) &&
             ((IntX+1) < Input_ImageHeight))
            FMtempPtr[u][v] = ((1-a)*(((1-b)*tempPtr[IntY][IntX]) +
                 (b*tempPtr[IntY][IntX+1]))) +
                 (a*(((1-b)*tempPtr[IntY+1][IntX]) +
                (b*tempPtr[IntY+1][IntX+1])));

        //if interpolated index lies outside of image realm then use either
```

```
            //nearest-neighbour index (LPT sampling within image) or
            //assign output matrix index a value of zero (LPT sampling outside image
            //realm)
            else
            {
                //If sampling outside image realm, assign 0 value
                if(SampleEntireImage) FMtempPtr[u][v] = 0;//for LPT sampling that
                        //encompasses entire image

                //Index within image realm (regular LPT sampling)
                else FMtempPtr[u][v] = tempPtr [Cy][Cx];

                //Count number of times indices occur outside of image realm
                InvalidIndexCounter++;
            }
        };
    }

    //Sample using nearest neighbour interpolation
    else
    {
    //Calculate indices and perform log-polar sampling
    for(WORD u = 0; u<SampleSize;u++)//u is the row index and maps to rho
        for(WORD v = 0;v<((long)(SampleSize*Oversample));v++)//v is the column index and
                                //maps to theta
        {
            rho = pow(Rht,(u/Max_rho));//for exponential sampling of radius
            theta = 2*PI*v/((long)(SampleSize*Oversample));//calculate theta: runs from
                            //0 to (2PI-1 angular division)

            //Nearest-neighbour interpolation:

            //Calculate rounded-to-nearest-integer indices to input FFT power spectrum
            //Cint global fxn. (defined in StdAfx.cpp) rounds a double to nearest int
            Cx = Cint((Input_ImageHeight/2) + (rho*cos(theta)));//round nearest integer
            Cy = Cint((Input_ImageHeight/2) - (rho*sin(theta)));//round nearest integer

            //Assign indexed value from FFT power spectrum to log-polar transform array
            FMtempPtr[u][v] = tempPtr [Cy][Cx];
        };
    }

    //Filter the oversampled input along each radius
    Filter(FMtempPtr);

    //Downsample matrix back to original input matrix size
    for(WORD row = 0;row<SampleSize;row++)
        //Sample every PI samples
        for(WORD col = 0; col<SampleSize;col++)
            FMresult[row][col] = FMtempPtr[row][((long)(col*Oversample))];


    //Deallocate memory for temporary oversample matrix holder
    DeAllocate(FMtempPtr,SampleSize);
}


//Fxn to filter the oversampled input in the radial direction
//An adaptation of the implementation for a "recursive" moving average filter according to
//Smith [1997]

void COpsDoc::Filter(float **originalPtr)
{
    //For a size filter of n, the mask size is (2*n)+1, so set n
    //Mask size should be 3, because you want to average the oversampled pixels
    //-- and it was oversampled approx. three times -- to produce one output pixel
    WORD n = 1;
```

```
    //Filter implementation is in one dimension so we must convert 2D data to 1D arrays

    //Create a temporary 1D input array
    float *inputPtr = oneD_float_Matrix(SampleSize * ((long)(SampleSize*PI)));

    //Remember the base address of inputPtr
    float *remember_Ptr = inputPtr;

    //Create a temporary 1D output array
    float *outputPtr = oneD_float_Matrix(SampleSize * ((long)(SampleSize*PI)));

    //Load up the 1D input array with the original data
    for(WORD row = 0;row<SampleSize;row++)
        for(WORD col = 0;col<((long)(SampleSize*PI));col++)
            *inputPtr++=originalPtr[row][col];

    //Restore base address of inputPtr
    inputPtr = remember_Ptr;

    //Define and initialise an accumulator
    double Add=0;

    //Find the first point for the "recursive" filter (middle point of the n-point mask)
    //(2*n)+1 is the total mask size
    for(DWORD index=0;index<((2*n)+1);index++)
        Add += (double)(inputPtr[index])/((2*n)+1);

    //Assign first result of "recursive" filter to its output
    outputPtr[n] = (float)Add;

    //Assign rest of points based on first point ("recursive" filtering)
    //start at next point and proceed until last point-midpoint of mask
    for(index = n+1;index<((SampleSize*((long)(SampleSize*PI)))-n);index++)
    {
        Add+= (double)(((inputPtr[index+n])-(inputPtr[index-(n+1)]))/((2*n)+1));
        outputPtr[index] = (float) Add;
    }

    //The first and last n points are missed in the implementation
    //Assign these the result from the nth point and the (last point-n) respectively

    //...first the beginning points
    for(index=0;index<n;index++)
        outputPtr[index] = outputPtr[n];

    //...then the last points
    for(index=((SampleSize*((long)(SampleSize*PI)))-n);
    index<(SampleSize*((long)(SampleSize*PI)));index++)
        outputPtr[index] = outputPtr[((SampleSize*((long)(SampleSize*PI)))-(n+1))];

    //Remember the base address of the outputPtr
    remember_Ptr = outputPtr;

    //Load up the 2D original array with the output data
    for(row = 0;row<SampleSize;row++)
        for(WORD col = 0;col<((long)(SampleSize*PI));col++)
            originalPtr[row][col] = *outputPtr++;

    //Restore the base address to the outputPtr
    outputPtr = remember_Ptr;

    //Return memory back to the free store
    delete [] outputPtr;
    delete [] inputPtr;
}
```

Appendix C: Code to build and apply Hamming window

```
//Fxn to build 1D Hamming window

void COpsDoc::Build_1DWindow()
{
    //Only build window function if user does NOT select rectangular window
    //(Get_SelectWindow returns NULL if user selected rectangular window)
    if (!Get_SelectWindow()) return;

    //Define height (== width) of matrix used to calculate window
    WORD height = Get_SampleSize();

    //Build the Window according to user's selection:

    //Build indices for Hamming window
    if (Get_SelectWindow() ==1)
    {
        for(WORD index = 0;index<height;index++)
            //Define 1 dimensional Hamming Window function
            Windowfxn_1D[index] = 0.54-(0.46*(cos((2*PI*(double)index)/(height-1))));
    }
}


//Fxn to apply 1D Hamming window

void COpsDoc::Window_1D(float **tempPtr)
{
    //Only apply window function if user does NOT select rectangular window
    //(Get_SelectWindow returns NULL if user selected rectangular window)
    if (!Get_SelectWindow()) return;

    //Otherwise build and apply the window
    else
    {
        //Allocate the memory for the window
        Windowfxn_1D = oneD_float_Matrix(Get_SampleSize());

        //Build the window indices
        Build_1DWindow();

        //Apply the window to the exponential portion of the log-polar sampling ONLY
        for(WORD row=0; row<Get_SampleSize();row ++)
            for(WORD col=0;col<Get_SampleSize();col++)
                tempPtr[row][col] = tempPtr[row][col]*Windowfxn_1D[col];

        //Deallocate the memory from the free store
        DeAllocate(Windowfxn_1D,Get_SampleSize());
    }
}
```

Appendix D: Code for sequence of steps to perform the Fourier-Mellin transformation

```
//Function to perform the Fourier-Mellin transform on an image
//Only available when FFT has NOT first performed
//Fourier-Mellin proceeds as follows:
//1) FFT 2) log-polar transform of FFT magnitude
//3) FFT of 2) produces RTS-invariant FFT magnitude

void COpsView::OnOperationsFouriermellin()
{
    //Get a pointer to the Document
    COpsDoc* pDoc = GetDocument();

    //Create holding matrices for FM magnitudes
    pDoc->FM_MatrixCreate(pDoc->Get_SampleSize());

    //Attempt to construct the image-holding matrix and
    //load up the matrix array with the image bits

    //If LoadBits fails, clean everything up for user to load a new image
    if(!pDoc->LoadBits())
    {
        pDoc->DeleteContents();//clean up the Document
        OnInitialUpdate();//clean up the client area
        return;//exit to await new command
    }

    //Show user this may take some time -- display the hourglass cursor
    CWaitCursor wait;

    //Instantiate a Fourier object to perform FFT
    Fourier FFT1(pDoc->Get_biD_Real(),pDoc->Get_biD_Imag(), pDoc->Get_height(),
        pDoc->Get_width(), pDoc->Get_Image_Size());

    //Perform FFT
    FFT1.biD_FFT();

    //Convert the real and imaginary portions of FFT to magnitude and phase
    FFT1.Polar_convert();

    //Centre the zeroth frequency in the matrix
    FFT1.CentreZeroFrequency();

    //Normalise the data to cancel out scale differences (if argument true),
    //transfer magnitude values to real matrix and zero imaginary matrix
    FFT1.Normalize(true);

    //Only scale the FFT and generate mag, phase BMPs if user does NOT perform multiple
    //FM-transforms
    if(!pDoc->Get_Automatic())
    {
        //Scale the magnitude and phase to displayable pixel values
        FFT1.FFT_Scale();

        //Assign the magnitude HBITMAP handle the magnitude HBITMAP
        //returned after creating the magnitude bitmap if it does not exist
        if(!pDoc->Get_MagBitmapHandle())
        pDoc->Get_MagBitmapHandle() = pDoc->MakeBitmap(pDoc->Get_MagDisplay());

        //Assign the phase HBITMAP handle the phase HBITMAP
        //returned after creating the phase bitmap if it does not exist
        if(!pDoc->Get_PhaseBitmapHandle())
        pDoc->Get_PhaseBitmapHandle()= pDoc->MakeBitmap(pDoc->Get_PhaseDisplay());
    }
```

```
    //Record the Document has now been modified
    pDoc->SetModifiedFlag(true);

    //Do a log polar transform on the FFT's magnitude
    pDoc->FM_Log_Polar_Transform(pDoc->Get_biD_Real(), pDoc->Get_FM_Original(),
         pDoc->Get_height());

    //Window the exponentially-sampled component of the log-polar transformed data
    //before applying the FFT: exponentially-sampled component is not continuous so this
    //should reduce leakage
    pDoc->Window_1D(pDoc->Get_FM_Original());

    //Do the FFT again:

    //Instantiate a second Fourier object to perform FFT
    Fourier FFT2(pDoc->Get_FM_Original(), pDoc->Get_FM_Imaginary(),pDoc->Get_SampleSize(),
        pDoc->Get_SampleSize(), ((pDoc->Get_SampleSize())*(pDoc->Get_SampleSize())));

    //Perform FFT
    FFT2.biD_FFT();

    //Convert the real and imaginary portions of FFT to magnitude and phase
    FFT2.Polar_convert();

    //Centre the zeroth frequency in the matrix
    FFT2.CentreZeroFrequency();

    //Transfer mag values to the real matrix (without normalisation) and
    //zero the imaginary matrix
    FFT2.Normalize(false);

    //Show user we're finished -- restore the default cursor
    wait.Restore();

    //Compare results from Fourier-Mellin transform of first image and display to user
    pDoc->FM_Compare();
}
```

Appendix E: Programme Interface