8-1-1990

Prediction of screener-induced moire in digital halftone pattern generation

Richard Comeau

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
PREDICTION OF SCREENER-INDUCED MOIRE IN
DIGITAL HALFTONE PATTERN GENERATION

by

Richard G. Comeau

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in the
Center for Imaging Science of the
College of Graphic Arts and Photography
of the Rochester Institute of Technology

August 1990

Signature of the Author ________________________ Center for Imaging Science

Accepted by ______________________________ Dr. M. Vaez-Iravani, Coordinator, M.S. Degree Program
The M.S. Degree Thesis by Richard G. Comeau has been examined and approved by the thesis committee as satisfactory for the thesis requirement for the Master of Science degree.

Dr. Joseph Delorenzo, Thesis Advisor

Dr. Roger Easton

Mr. Andrew Masia

August 20, 1990

Date
Prediction of Screener-Induced Moire in Digital Halftone Pattern Generation

I, Richard Comeau, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date August 20, 1990
PREDICTION OF SCREENER-INDUCED MOIRE IN DIGITAL HALFTONE PATTERN GENERATION

by

Richard G. Comeau

Submitted to the Center For Imaging Science in partial fulfillment of the requirements for the Master of Science degree at the Rochester Institute of Technology

ABSTRACT

In the graphic arts, objectionable moire patterns are often observed on films or printed products due to the interaction of various periodic structures of halftone images. A particular type of moire pattern that results from digital halftoning at arbitrary angles and frequencies using a virtual screen function has been studied. A computer program was developed that produces uniform digital halftone patterns using a virtual screen approach and that calculates the corresponding amplitude spectra. It was found that aliasing due to the sampling of the virtual screen causes low frequency components in the amplitude spectrum. Moire patterns with fundamental vector frequencies equal to those of the strong aliased components were observed in halftone images reconstructed on a film recorder. Moire was also observed at frequencies not represented or under represented in the amplitude spectrum. It is shown that this moire effect is due to the additive beating of two or more higher frequency components that differ by the frequency of the observed moire. It is suggested that the non-linearities of the film recording process amplify this effect.

The effects on the resulting moire patterns of varying the halftone parameters of dot size, dot shape, screen angle, and screen frequency were examined. In general, the amplitude spectra are complex, indicating many overlapping patterns. Screener induced moire was found to behave in a nearly identical manner to that induced by digital scanning of an existing halftone.
ACKNOWLEDGEMENTS

Sincere thanks are extended to the members of the thesis committee, Dr. Joseph DeLorenzo (thesis advisor), Dr. Roger Easton, and Andrew Masia, for their advice and participation. The research was performed at the facilities of EKTRON Applied Imaging, Inc. The computer and lab equipment, film recorder, film, and supplies used for the thesis were provided by EKTRON. The help of Rob Tello, who prepared a number of the graphs, is greatly appreciated. Sincere thanks are extended to Barbara Hill who provided encouragement throughout the research and assisted by proofreading the document.
TABLE OF CONTENTS

LIST OF FIGURES ........................................ iii
LIST OF TABLES ........................................ iv
1.0 INTRODUCTION ....................................... 1
2.0 THEORETICAL DEVELOPMENT ............................. 10
   2.1 The Photographic Halftoning Process ................. 10
   2.2 Digital Halftoning .................................. 14
   2.3 The Virtual Screen Halftoning Process ............... 17
   2.4 Fourier Transform of the Uniform Virtually-Screened Halftone .......... 23
   2.5 A View in Two Dimensions ............................ 29
   2.6 The Virtual Screen Font ............................ 32
3.0 EXPERIMENTAL ......................................... 33
   3.1 Experimental Approach .............................. 33
   3.2 Software Development ................................ 34
   3.3 Reporting the Amplitude Spectrum .................... 39
   3.4 Physical Reconstruction of Digital Images ............ 40
   3.5 Generating the Optical Transform ..................... 43
   3.6 Selection of Test Images ............................ 44
4.0 RESULTS AND DISCUSSION .............................. 49
   4.1 Initial Results ...................................... 49
   4.2 Interpretation of the Spectrum ....................... 57
   4.3 Results of Parameter Change Series .................. 63
   4.4 Relationship to Scanned Halftone Images ............. 71
5.0 CONCLUSIONS ........................................ 72
6.0 LIST OF REFERENCES .................................. 75
APPENDIX A - DESCRIPTION OF USEFUL FUNCTIONS, PROPERTIES AND OPERATORS .................... 78
APPENDIX B - SOURCE CODE LISTINGS ........................ 83
APPENDIX C - MOIRE PROGRAM RESULTS SUMMARIES ............ 84
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>The Halftoning Process</td>
<td>12</td>
</tr>
<tr>
<td>2-2</td>
<td>Example of a 5 x 5 Dither Matrix for a 0° Screen</td>
<td>16</td>
</tr>
<tr>
<td>2-3</td>
<td>Example of an 8 x 8 Dither Matrix for a 45° Screen</td>
<td>16</td>
</tr>
<tr>
<td>2-4</td>
<td>Illustration of Equation 2-14</td>
<td>18</td>
</tr>
<tr>
<td>2-5</td>
<td>Virtual Screen Halftoning: Spatial Domain</td>
<td>21</td>
</tr>
<tr>
<td>2-6</td>
<td>Virtual Screen Halftoning: Frequency Domain</td>
<td>27</td>
</tr>
<tr>
<td>2-7</td>
<td>Plot of Spectrum of an Unsampled Virtual Screen</td>
<td>31</td>
</tr>
<tr>
<td>2-8</td>
<td>Effect of Sampling on the Spectrum</td>
<td>31</td>
</tr>
<tr>
<td>3-1</td>
<td>Flow of Program moire.c</td>
<td>38</td>
</tr>
<tr>
<td>3-2</td>
<td>Tile Format on the 6444R Film Recorder</td>
<td>42</td>
</tr>
<tr>
<td>3-3</td>
<td>Setup for Producing the Optical Transform</td>
<td>45</td>
</tr>
<tr>
<td>4-1a</td>
<td>Image 1: Film Recorder Image</td>
<td>52</td>
</tr>
<tr>
<td>4-1b</td>
<td>Image 1: Magnified View</td>
<td>52</td>
</tr>
<tr>
<td>4-1c</td>
<td>Image 1: Illustration of Moire Patterns</td>
<td>53</td>
</tr>
<tr>
<td>4-1d</td>
<td>Image 1: Amplitude Spectrum to 1000 cpi</td>
<td>54</td>
</tr>
<tr>
<td>4-1e</td>
<td>Image 1: Amplitude Spectrum to 75 cpi</td>
<td>54</td>
</tr>
<tr>
<td>4-1f</td>
<td>Image 1: Optical Transform</td>
<td>55</td>
</tr>
<tr>
<td>4-2</td>
<td>Contrast Sensitivity of the Human Visual System</td>
<td>59</td>
</tr>
<tr>
<td>4-3a</td>
<td>Sum of Two Cosine Waves with Relative Frequencies of 9 and 10.</td>
<td>61</td>
</tr>
<tr>
<td>4-3b</td>
<td>Amplitude Spectrum Corresponding with Part (a)</td>
<td>61</td>
</tr>
<tr>
<td>4-4a</td>
<td>Product of Two Cosine Waves with Relative Frequencies of 9 and 10.</td>
<td>62</td>
</tr>
<tr>
<td>4-4b</td>
<td>Amplitude Spectrum Corresponding with Part (a)</td>
<td>62</td>
</tr>
<tr>
<td>4-5</td>
<td>Matrix of the 16 Test Images</td>
<td>64</td>
</tr>
<tr>
<td>4-6a</td>
<td>Image 3: Film Recorder Image</td>
<td>65</td>
</tr>
<tr>
<td>4-6b</td>
<td>Image 3: Illustration of Moire Patterns</td>
<td>65</td>
</tr>
<tr>
<td>4-6c</td>
<td>Image 3: Amplitude Spectrum to 75 cpi</td>
<td>66</td>
</tr>
<tr>
<td>4-7a</td>
<td>Film Recorder Image of Screen Frequency Series:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Image 5, 6, 7, and 8.</td>
<td>66</td>
</tr>
<tr>
<td>4-7b</td>
<td>Image 5: Illustration of Moire Patterns</td>
<td>67</td>
</tr>
<tr>
<td>4-7c</td>
<td>Image 5: Amplitude Spectrum to 75 cpi</td>
<td>67</td>
</tr>
<tr>
<td>4-8a</td>
<td>Image 9: Film Recorder Image</td>
<td>68</td>
</tr>
<tr>
<td>4-8b</td>
<td>Image 9: Amplitude Spectrum to 75 cpi</td>
<td>68</td>
</tr>
<tr>
<td>4-8c</td>
<td>Image 9: Illustration of Moire Patterns</td>
<td>69</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Required Arguments to moire.c</td>
</tr>
<tr>
<td>2</td>
<td>List of Experimental Images</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

Many methods for producing images are limited to binary output. All of the four major types of impact printing (lithography, screen printing, relief printing, and intaglio) and most types of non-impact printing, such as electrophotography and standard ink jet, are essentially binary (Adams and Faux 1982, 10; Johnson 1986, 12). With each of these systems some form of colorant (ink, toner, dye, etc.) is either applied to the receiving medium, or the medium contains a colorant that is activated by a chemical or physical process. The end result is that the colorant is either present or absent at any particular point on the medium. Many display devices are also binary. In order to render a continuous-tone (i.e., intensity modulated) image like a photograph on a binary system, the image must first be binarized. Any process that achieves this conversion by using area modulation of the binary imaging elements in order to represent tonal amplitude is defined as halftoning.

Although the actual implementation may vary, nearly all halftoning methods involve the interaction of the continuous-tone image with a non-image-related, continuous-tone "screen function," then a fixed-level threshold is applied to form the halftone image (Roetling 1977a). Halftoning in this manner is often referred to as "screening." The screen function modulates the continuous-tone image with high spatial frequency structure in order to introduce a carrier for the binary colorant. Due to the bandpass characteristics of the human visual system, the high-frequency transitions that result cannot be resolved; the
viewer has the illusion of continuous tone. The resulting tone depends on the average area covered with the colorant. Image quality is related to how well the halftone image maintains the visible frequency content (i.e., detail) and the tonal quality of the original image without introducing objectionable visual artifacts such as screen texture, noise, and moire patterns.

Halftoning has traditionally been accomplished by photographing a continuous-tone picture through a screen onto high contrast film (Brynhahl 1978). The basic process has changed little since its invention by W.H. Fox Talbot in 1852, although improvements have been made (Wesner 1974). Early methods used woven fabric placed near the film as the screen. The next improvement was the "crossline screen", a sandwich of a pair of rulings (line screens) aligned orthogonally. The crossline screen was placed near the film. Each of these methods utilizes a combination of absorption and diffraction to introduce modulation (Streifer et al. 1974). A significant improvement was realized with the introduction of the contact screen in the 1940's. This screen is placed in direct contact with the film; the screen function is defined directly by the transmittance profile of the contact screen.

A "classical screen" has a two-dimensional, periodic arrangement of identical square halftone "cells" (Ulichney 1987). The value of the transmittance function of the basic cell is usually at a maximum near the center of the cell. It then drops off monotonically in all directions such that a smooth periodic transmittance function is formed when the cell is replicated.
The screen is usually rotated with respect to the continuous-tone image. There exist a number of conventions for defining the amount of rotation, or "screen angle." In this paper screen angle is defined as the degree of clockwise rotation of the horizontal screen axis from the horizontal axis of the continuous-tone image. Since the screen axes are orthogonal, the horizontal axis is defined as that which results in a screen angle between 0 and 90 degrees. "Screen frequency" is the reciprocal of the screen cell spacing.

The halftone image that results from using a classical screen consists of clustered dots, arranged periodically according to the square grid of the screen. The dots vary in size and shape depending on image intensity and detail. Some other halftoning methods result in "dispersed dots," which are small dots dispersed in some manner in the halftone cell. Aperiodic screens also produce dispersed dots. There is a fundamental trade off between the two approaches. In general, dispersed dots have a higher perimeter-to-area ratio. Brynghdahl (1978) showed that image detail increases with dot perimeter because all information is encoded at the dot periphery (i.e., where the screen function threshold was crossed). But, imaging systems differ in their ability to render small spots. An indicator of this ability is the system spread function, which is the output image produced when a point image is input into the system. Most printing methods have spread functions that depend on a number of process parameters such as ink type, paper type, press conditions, etc. Since the spreading primarily affects the
dot periphery, it is difficult to reproduce dispersed dots in a controllable manner with such systems. The principal advantage of the clustered dot approach over the dispersed dot approach is that the dots are less sensitive to dot spreading. The reason is seen by noting that a circular dot has the lowest perimeter-to-area ratio of any shape covering the same area (Brynhdahl 1978).

The periodic nature of clustered-dot halftones, which is so helpful in controlling tone reproduction, can also create low-frequency structures, or beats. These structures are referred to as moire patterns for their resemblance to moire fabric, which has a wavy appearance, (Wesner 1974). Moire patterns can occur whenever two periodic structures are combined. One cause of moire is the intermodulation of the screen function with periodic image detail (such as a striped shirt or tweed jacket). Another type of moire occurs in process color halftone printing, in which four halftone images printed with different inks and screened at different angles, are overlaid. If these screens were printed at the same angle, small registration errors would produce very noticeable moire patterns. To avoid this, three of the four screens are oriented at 30 degree angles from each other resulting in the high-frequency "rosette" pattern that is a common characteristic of process color printing (Cox and Hillam 1936; Cox 1957; Tollenaar 1957; McKinney 1957; Wurzburg 1961). Usually this pattern is not objectionable. The fourth screen (for yellow ink, which forms the lightest pattern) is angled halfway between two of the others. Yet another type of moire
occurs when a halftone image is formed by "rescreening," which is the screening of an existing halftone.

Digital halftoning is a class of digital image processing methods for the conversion of continuous-tone digital imagery to binary digital imagery. The inherent flexibility of digital processing has led to numerous halftoning algorithms, some of which simulate the traditional halftoning methods (Stoffel and Moreland 1981; Bayer 1973; Kekolahti 1982). Ulichney (1988) has organized the algorithms according to the type of dot that is formed (dispersed or clustered), the type of processing used (area or point), and whether the pattern is periodic or aperiodic.

This paper will consider periodic clustered dot methods that simulate the classical screen. The simplest and most common approach to this type of digital halftoning is to generate a "dither matrix" that represents a single period of the sampled screen function with each sample corresponding directly to an addressable pixel of the binary output device. This addressable pixel will be referred to as a microdot. The dither matrix need not be square but must contain an integer number of microdots.

As explained previously, color printing requires that screens be oriented at specific angles. Various configurations of ordered dither have been devised for simulating different screen angles and frequencies, but the screen angles are restricted to those with rational tangents (Chapman 1974; Gast 1974). Screen frequency is restricted similarly; the halftone cell has an integer number of microdots so not all
frequencies are possible. Holladay (1980) invented a method for increasing the number of available rational tangent angles by allowing non-orthogonal screen axes, and similar approaches have been devised for dispersed dot methods (Rao and Arce 1988; Ulichney 1987). However, non-orthogonal axes introduce moire when screens are overlapped as is necessary for color printing. Each of these methods uses an integer number of microdots per halftone cell.

Arbitrary control of screen angle and frequency can be achieved using an alternate approach that is termed here as "virtual screen halftoning." Algorithms in a number of electronic halftoning patents use this process in some form and they refer to the screen as "virtual," "aerial," or "imaginary" (Rosenfeld 1982; Hammes 1983; Rosenfeld 1984; Winrich and Klausdorf 1984). The basic idea is that the virtual screen function is scaled and rotated (in software) with respect to the array of microdots so that there is no integer relationship between the continuous-tone image and the screen function. Because of this, the screen angle is said to be irrational. The virtual screen is, in effect, sampled by the array of microdots. Generally there is not an integer number of samples per halftone cell. This inevitably leads to beats between the halftone and microdot patterns that are often visible as an objectionable periodic two-dimensional artifact called "screener-induced moire."
The moire patterns described previously are similar in that they are caused by the intermodulation of two or more periodic structures and that they occur with both spatially continuous and with quantized image methods. Screener-induced moire differs in that it can only occur in a sampled imaging system. Another such moire occurs when an existing halftone image is digitally scanned.

Moire phenomena have been extensively studied both for their undesirable aspects such as patterns in halftoning, and also as a useful tool in metrology (Oster 1965; Nishijima and Oster 1964). Many analyses model moire in the spatial domain using geometry and trigonometry to show how the patterns are formed. Oster et al. (1964) used this approach in a general moire analysis. Tollenaar (1957) used trigonometry to predict the frequencies and angles of moire patterns that result from process color printing. He extended the considerations to show the effect of using dots placed on a hexagonal instead of an orthogonal grid. Toor and George (1983) also analyzed graphic arts related moire in the spatial domain.

Often, more insight can be obtained by examining moire in the frequency domain. Bryngdahl (1974) showed that when one or more periodic patterns are overlaid, the fringe pattern that results can be described according to three parameters: spatial frequency, orientation (angle), and profile. For simple structures, the spatial frequency and angle of the pattern can be predicted simply by adding vectors in the frequency domain. In another paper, Bryngdahl (1975) extended his analysis to include
the moire intensity profile by calculating the amplitude spectrum (amplitude of the frequency spectrum) for overlapping structures. The frequency spectrum of a periodic function is an infinite series of weighted sine and cosine functions that, when summed, form the periodic function. The frequency spectrum of a function is found by taking its Fourier transform (defined in Section 2).

A number of researchers have analyzed halftone patterns in the frequency domain. An expression for the frequency spectrum of the spatially continuous halftone image had been previously developed by Kermisch and Roetling (1975). This analysis was later extended to account for ordered-dither digital halftoning methods (Allebach and Liu 1977). Bestenreiner and Freund (1975) predicted the multi-screen moire amplitude spectrum and showed that by using patterns arranged on a rhombic grid the amplitude of the most dominant components can be reduced. The phenomenon of moire patterns in scanned halftone pictures was researched by Huang (1974) and then more thoroughly by Steinbach and Wong (1982).

The proposal for this thesis stated that a major objective was to derive a Fourier-domain description for the virtually screen halftone in order to identify the amplitudes of moire components (Comeau 1986). However, a complete derivation similar to that of Allebach and Liu (1977) would be quite complicated because of the effects of aliasing. Granger (1988) suggested a basic model to the author that assumed uniform input. This approach seemed reasonable, since moire is most objectionable in image areas with constant intensity, and thus has been used.
The goal of this thesis was to derive, implement, and evaluate a simple virtual screening process and its corresponding amplitude spectrum in order to predict the magnitude of screener-induced moire and its dependence on screening parameters. It was expected that this approach would allow prediction of moire visibility, to a first approximation, to aid in the design and analysis of virtual-screening methods.

In Section 2, an analytic expression of the virtual-screening process and the amplitude spectrum is derived. Section 3 describes the experimental procedures devised for implementing that analytic expression in the form of a computer program, plots, and film samples. A set of sample images and spectra were produced in order to identify how the moire changes with screening parameters. In Section 5 the results are evaluated.
2.0 THEORETICAL DEVELOPMENT

2.1 The Photographic Halftoning Process

Today, most halftones that are produced photographically use a contact, rather than a cross-line screen. The contact-screen process works by exposing a high-contrast film to the image of a continuous-tone picture through a non-image related, continuous-tone screen that is placed in direct contact with the film. If no change in magnification is required, then all three layers can be sandwiched and contact printed with a uniform source. When a very high contrast film (i.e. high gamma) is used, the film can be assumed to have a binary transfer characteristic, i.e. those areas receiving exposure above a given level are black, and those below that level are clear. It is also possible to have positive working films that have a high negative gamma, in which case the threshold rule is reversed.

To mathematically analyze the halftoning process, consider the sandwich arrangement discussed above in a cartesian coordinate system so that all functions have the same scale. Let \( f(x,y) \) and \( g(x,y) \) represent the transmittance functions of the original continuous-tone image and of the contact screen respectively. The exposure profile on the film is defined as

\[
e(x,y) = c \cdot f(x,y)g(x,y),
\]

(2-1)

where the constant \( c \) represents the incident exposure, the product of the exposure time and the irradiance incident on the
sandwich from the uniform light source. The exposure function is mapped to density according to the density vs. log exposure (D-log H) curve of the film. If the gamma of the film is very high, this transfer function is assumed to be a binary threshold operator \( \Gamma \), defined as

\[
\Gamma(\cdot) = \begin{cases} 
0, & \cdot < t \\
1, & \cdot \geq t 
\end{cases} \tag{2-2}
\]

where \( t \) is the threshold exposure level. The complete halftone process can now be described as

\[
h(x,y) = \Gamma \left[ c \cdot f(x,y)g(x,y) \right] \tag{2-3}
\]

where \( h(x,y) \) is the transmittance profile (which has a range from 0.0 to 1.0) of the resulting halftone. The definition of the threshold operator assumes, for convenience, that a positive (negative gamma) film is used. If a negative (positive gamma) film is used then the threshold would be reversed. A one-dimensional view of this halftoning process is illustrated in Figure 2-1.

The contact screen transmittance profile acts directly as the screen function, and its design is an important determinant of halftone image quality. Consider the use of a random "white noise" screen function. Such a function adds non-image related low-frequency structure, which causes the image to appear grainy.
Figure 2-1  The Halftoning Process
(a) Contact screen transmittance. (b) Continuous-tone image transmittance. (c) Exposure profile. (d) Halftone image transmittance.
Also it adds unstructured power at high frequencies that is significantly affected by the spread function of the image rendering process. Tiny dots smear or bleed together due to ink flow so that it is impossible to control tone reproduction.

These difficulties explain why white-noise screen functions are almost never used and they give insight into why periodic, clustered dot methods are used (Roetling 1977a). As stated previously, clustered dots have a low perimeter-to-area ratio so they are less affected by the spread function. Ideally, the screen frequency is high enough so that the dot texture is not perceived, yet low enough so that the dots do not smear or clump together. In the printing process, the effect of the spread function is referred to as dot gain. The upper limit on screen frequency depends on the printing method, paper, and ink properties.

The contact screen profile affects tone reproduction and the nominal dot shape. The actual dot shape depends on the original image detail as well. The characteristic of the halftone process that encodes image detail in the dot shape is referred to as partial dot structure. This structure indicates that the halftone process is not a sampled imaging system in the classical sense. Roetling (1977a) demonstrated that partial dot structure can encode edge detail at higher than one-half the halftone frequency.
2.2 Digital Halftoning

In the ordered dither approach to digital halftoning, the continuous tone image and screen functions are sampled. The functions are quantized in amplitude as well, but for the purposes of this discussion it will be assumed that there are enough levels so that the functions can be considered continuous. Each discrete point corresponds to an addressable location (microdot) on the output device. As with a contact screen, the screen function profile controls tone reproduction and the nominal shapes of the halftone dots. For ease of computation, the screen function and image are usually combined additively rather than multiplicatively. The ordered dither operation can be described as

\[ h(x_i, y_j) = \Gamma [f(x_i, y_j) + g(x_i, y_j)] . \] (2-4)

The subscripts i and j are pixel indices. They indicate that the functions are spatially discrete and are mapped similarly. Addition followed by thresholding can be shown to be equivalent to a comparison operation:

\[ h(x_i, y_j) = \begin{cases} 
1, & f(x_i, y_j) \geq g(x_i, y_j) \\
0, & (x_i, y_j) < g(x_i, y_j) 
\end{cases} \] (2-5)

Since ordered dither is periodic, it is only necessary to store (in computer memory) one period of the screen function. The simplest digital screen functions are square matrices. The
size of the matrix and the arrangement of its values determine the effective screen frequency, screen angle, and halftone dot shape. For example, the 5 x 5 square matrix in Figure 2-2 will form a 0 degree halftone with 25 microdots per halftone cell. The dither matrix values range from 0 to 250. The 8 x 8 square matrix in Figure 2-3 will form a 45 degree halftone with 32 microdots per cell. Both matrices, as described by Roetling (1977b) were for use with an 8-bit image (values range from 0-255). This matrix was also designed for use with an 8-bit digital image. It contains the information for two identical halftone cells. The number of microdots per cell also affects tone reproduction.

When using square matrices, one is limited to screen angles with rational tangents, e.g. \( \tan(18.43^\circ) = 1/3 \). The rational tangent approach is used by Hell in its DC-300 series color separating scanner (Gast 1974). Other methods have been developed that allow practical implementations of non-square matrices to achieve more screen angles, but the axes of the halftone pattern are not orthogonal (Holladay 1980).

In virtual-screen halftoning the screen function is sampled much more finely than the microdot spacing, or it is not sampled at all. The term "virtual" is used because the screen function does not have integer boundaries with respect to the microdot grid. The screen function can be arbitrarily scaled and rotated in software or firmware. At this position, a sample of the screen function is taken and used for the threshold operation.
Figure 2-2  Example of a 5 x 5 Dither Matrix for a 0° Screen

<table>
<thead>
<tr>
<th>40</th>
<th>60</th>
<th>150</th>
<th>90</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>170</td>
<td>240</td>
<td>200</td>
<td>110</td>
</tr>
<tr>
<td>140</td>
<td>210</td>
<td>250</td>
<td>220</td>
<td>130</td>
</tr>
<tr>
<td>120</td>
<td>190</td>
<td>230</td>
<td>180</td>
<td>70</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>160</td>
<td>50</td>
<td>30</td>
</tr>
</tbody>
</table>

0° Halftone Screen

Figure 2-3  Example of an 8 x 8 Dither Matrix for a 45° Screen

<table>
<thead>
<tr>
<th>52</th>
<th>44</th>
<th>36</th>
<th>124</th>
<th>132</th>
<th>140</th>
<th>148</th>
<th>156</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>4</td>
<td>28</td>
<td>116</td>
<td>220</td>
<td>228</td>
<td>236</td>
<td>164</td>
</tr>
<tr>
<td>68</td>
<td>12</td>
<td>20</td>
<td>108</td>
<td>212</td>
<td>252</td>
<td>244</td>
<td>172</td>
</tr>
<tr>
<td>76</td>
<td>84</td>
<td>92</td>
<td>100</td>
<td>204</td>
<td>196</td>
<td>188</td>
<td>160</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>132</th>
<th>140</th>
<th>148</th>
<th>156</th>
<th>52</th>
<th>44</th>
<th>36</th>
<th>124</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>228</td>
<td>236</td>
<td>164</td>
<td>60</td>
<td>4</td>
<td>28</td>
<td>116</td>
</tr>
<tr>
<td>212</td>
<td>252</td>
<td>244</td>
<td>172</td>
<td>68</td>
<td>12</td>
<td>20</td>
<td>106</td>
</tr>
<tr>
<td>204</td>
<td>196</td>
<td>188</td>
<td>180</td>
<td>76</td>
<td>84</td>
<td>92</td>
<td>100</td>
</tr>
</tbody>
</table>

45° Halftone Screen
Ulichney (1987) referred to virtual screening as "angle dithering" because sampling causes a piecewise linear approximation to the screen angle from dot to dot. However, the virtual screen angle is fixed during the screening operation. The apparent dithering is actually the effect of aliasing that is caused by undersampling the screen function. The visual effect of such aliasing is screener-induced moire.

2.3 The Virtual Screen Halftoning Process

If the contact screen function \( g(x,y) \) is periodic, it may be fully described by a single cell \( p(x/s, y/s) \). The screen is generated by replicating \( p(x/s, y/s) \) in both dimensions with a spacing equal to the cell width \( s \). From the replicating property of the comb function the screen function is described as

\[
g(x,y) = p\left(\frac{x}{s'}, \frac{y}{s}\right) \ast \ast \text{comb}\left(\frac{x}{s'}, \frac{y}{s}\right).
\]  

(2-6)

The derivation of Equation 2-6 is illustrated in Figure 2-4.

If \( f(x,y) \) is assumed constant and if we substitute Equation 2-6 into Equation 2-3 then fold all constants into the thresholding operation, the expression for uniform halftoning is as follows:

\[
h(x,y) = \Gamma \left[ p\left(\frac{x}{s'}, \frac{y}{s}\right) \ast \ast \text{comb}\left(\frac{x}{s'}, \frac{y}{s}\right) \right].
\]  

(2-7)

Define the binary "dot" function as

\[
dot\left(\frac{x}{s'}, \frac{y}{s}; a\right) = \Gamma \left[ p\left(\frac{x}{s'}, \frac{y}{s}\right) \right],
\]  

(2-8)

which is the dot shape that results from applying a threshold to \( p(x/s, y/s) \), such that it covers area \( a \). Given that the
A continuous-tone image $f(x,y)$ is a constant equal to $t$, combine Equations 2-7 and 2-8 to yield an expression for the uniform spatially continuous halftone as follows:

$$h(x,y) = \text{dot} \left( \frac{x}{S}, \frac{y}{S}; a \right) \ast \ast \frac{1}{s^2} \text{comb} \left( \frac{x}{S}, \frac{y}{S} \right) .$$

(2-9)
The threshold operation has been eliminated. In this paper, three basic dot shapes will be examined: round, square, and diamond. The derivation of Equation 2-9 is illustrated in Figure 2-5, parts (a) through (c).

In the virtual screen process, the screen function is sampled at each addressable point of the output device (i.e. the microdot grid). The sample value is then compared to $f(x,y)$ to determine whether the microdot is to be set to a '1' or a '0'. For uniform input, the process reduces to sampling $h(x,y)$, (which can be treated as a virtual, spatially continuous function) by the microdot grid. From the sampling property of the comb function, the sampled virtual halftone is defined as follows:

$$h(x_i, y_i) = h(x, y) \frac{1}{w^2} \text{comb} \left( \frac{x}{w'}, \frac{y}{w} \right) . \quad (2-10)$$

where $w$ is the microdot spacing. In practical implementations of virtual screening, the microdot frequency is typically 10 to 20 times that of the screen. Combining Equations 2-9 and 2-10, we can form

$$h(x_i, y_i) = \left[ \text{dot} \left( \frac{x}{s}, \frac{y}{s} \right) \times 1 \frac{1}{s^2} \text{comb} \left( \frac{x}{s}, \frac{y}{s} \right) \right] \cdot \frac{1}{w^2} \text{comb} \left( \frac{x}{w'}, \frac{y}{w} \right) . \quad (2-11)$$

This expression applies when the screen angle is 0 degrees. In general, the screen is rotated with respect to the microdot grid, and the microdot grid is aligned to the reference coordinate system. This is shown as

$$h(x_i, y_j) = \left[ \text{dot} \left( \frac{x'}{s'}, \frac{y'}{s'} \right) \times 1 \frac{1}{s^2} \text{comb} \left( \frac{x'}{s}, \frac{y'}{s} \right) \right] \cdot \frac{1}{w^2} \text{comb} \left( \frac{x}{w'}, \frac{y}{w} \right) . \quad (2-12)$$
where $x'$ and $y'$ are coordinates in the rotated screen coordinate system. They are related to $x$ and $y$ by the rotation transformation

$$
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = 
\begin{bmatrix}
  \cos \theta & \sin \theta \\
  -\sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix},
$$

(2-13)

where $\theta$ is the screen angle.

Equation 2-12 defines the digital halftone. A continuously varying image may be formed by convolving with a physical reconstruction function (prf). This function is defined by the effective spot of the device on which the image is displayed. The general prf is defined as $\text{prf}(x/w, y/w)$. The size, $w$, has been included to show that the spot size is of the order of the microdot spacing; the actual spot size depends on the functional form of the prf. The prf form depends on the spot shape and size of the output device, processing parameters (e.g. film gamma, etc.). The exposure profile of the reconstructed halftone is the convolution of the prf with the digital halftone image:

$$
e_r(x, y) = \left[ \text{dot}\left(\frac{x'}{s}, \frac{y'}{s}\right) \ast \frac{1}{s^2} \text{comb}\left(\frac{x'}{s}, \frac{y'}{s}\right) \right] \cdot \frac{1}{w^2} \text{comb}\left(\frac{x}{w}, \frac{y}{w}\right) \ast \text{prf}\left[\frac{x}{w}, \frac{y}{w}\right].
$$

(2-14)
Figure 2-5 Virtual Screen Halftoning: Spatial Domain
(a) Comb function with a period equal to the screen spacing. (b) Dot function. (c) Uniform virtual halftone. (d) Sampling function at microdot spacing w. (e) Sampled digital halftone. (f) Rectangle function as the prf. (g) Reconstructed halftone image.
After the film is processed, the reconstructed halftone function is defined as

\[
h_r(x, y) = \Gamma \left[ \left[ \text{dot} \left( \frac{x'}{s}, \frac{y'}{s} \right) ** \frac{1}{s} 2 \text{comb} \left( \frac{x'}{s}, \frac{y'}{s} \right) \right] \cdot \frac{1}{w} 2 \text{comb} \left( \frac{x}{w}, \frac{y}{w} \right) \right] ** \text{prf} \left[ \frac{x}{w} \frac{y}{w} \right]
\]  
(2-15)

If the physical reconstruction function is defined as \( \text{rect}(x/w, y/w) \), then the thresholding operation can be eliminated. The rectangle ensures that the output is already binary. The choice of this prf is a simplifying assumption that is made at this time. The effect of real prf's will be considered later. Equation 2-15 is now re-written to account for the rectangle function.

\[
h_r(x, y) = \left[ \left[ \text{dot} \left( \frac{x'}{s}, \frac{y'}{s} \right) ** \frac{1}{s} \text{comb} \left( \frac{x'}{s}, \frac{y'}{s} \right) \right] \cdot \frac{1}{w} \text{comb} \left( \frac{x}{w}, \frac{y}{w} \right) \right] ** \text{rect} \left[ \frac{x}{w} \frac{y}{w} \right]
\]  
(2-16)

Figure 2-5 illustrates each stage of this derivation in one-dimension. The virtual halftone is derived by convolving the function in Figure 2-5 (a) with that of Figure 2-5 (b) and is shown in Figure 2-5 (c). The effect of the sampling process (Figure 2-5 (d) is apparent in the digital halftone (Figure 2-5-(e)). Some of the dots have three samples "on" and some have two. Figure 2-5 (g), produced from convolving the prf, shown in Figure 2-5 (f) as a rectangle function, shows that
the reconstructed halftone has dots (indicated by the shaded areas) of varying size. There are, on average, 5.33 samples per halftone cell. A structure has been introduced that repeats every 16 microdots or every three halftone dots. The low-frequency portion of this pattern, with a fundamental period of three halftone dots, is interpreted visually as a moire pattern. Note that in this simple example, the pattern repeats exactly every 16 microdots, but in general the pattern is not perfectly periodic. One can see that numerous overlapping patterns are possible.

2.4 Fourier Transform of the Uniform Virtually Screened Halftone

The Fourier transform of a function \( f(x) \) is defined as

\[
F(\xi) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi \xi x} \, dx,
\]

(2-17)

where \( \xi \) is the frequency coordinate. The two-dimensional Fourier transform is defined as

\[
F(\xi, \eta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)e^{-i2\pi (\xi x + \eta y)} \, dx \, dy
\]

(2-18)

Functions in the spatial domain are denoted by lower case letters; their transforms are denoted by it upper case letter. The Fourier transform of the reconstructed exposure profile expressed by Equation 2-16 will now be derived. The derivation is straightforward, given several theorems of the Fourier
transform that are stated here without proof. The derivations can be found in most any Fourier transform text (e.g. Bracewell 1978).

The convolution theorem states that the Fourier transform of the convolution of two functions is the product of the Fourier transforms of the two functions. This is stated in Equation 2-19. Per Bracewell’s notation, the "\( \Rightarrow \)" symbol denotes the Fourier transform operation.

\[
f(x) * g(x) \Rightarrow F(\xi)G(\xi) \tag{2-19}
\]

Conversely, the Fourier transform of a product of two functions is the convolution of the Fourier transforms of the two functions as shown in Equation 2-19.

\[
f(x)g(x) \Rightarrow F(\xi) * G(\xi) \tag{2-20}
\]

The scaling property of the Fourier transform is:

\[
f(x/b) \Rightarrow |b| F(b\xi) \tag{2-21}
\]

Finally, the transform of a comb function is a comb function. By applying the scaling property, the transform of the scaled comb function is:

\[
\text{comb}(x/b) \Rightarrow |b| \text{comb}(b\xi) \tag{2-22}
\]
The Fourier transform of \( f(x) \) is often referred to as its frequency spectrum, which can be expressed as

\[
F(\xi) = A(\xi) e^{-i\Phi(\xi)},
\]  

(2-23)

where \( A(\xi) \) is known as the amplitude spectrum and \( \Phi(\xi) \) is the phase spectrum. As in Gaskill (1978), we allow \( A(\xi) \) to take on negative values. Each of the functions in Equation 2-15 is real-valued and even, as is the resultant. Since products and convolutions of real-valued even functions are themselves real-valued and even, \( h_r(x,y) \) is also real and even. The Fourier transform of a real-valued, even function is also real-valued and even. When \( F(\xi) \) is real-valued, Equation 2-23 reduces to

\[
F(\xi) = A(\xi),
\]  

(2-24)

thus the signed amplitude spectrum is a complete and valid description for the Fourier transform of a real-valued, even function.

By Fourier transforming the individual functions in Equation 2-15 and applying the scaling and convolution theorems, the expression for the amplitude spectrum of the exposure profile is found to be

\[
E_r(\xi, \eta) = [s^2 \left[ \text{DOT}(s_{\xi'}, s_{\eta'}) \text{comb}(s_{\xi'}, s_{\eta'}) \right] ** \text{comb}(w_{\xi}, w_{\eta}) ] w^2 \text{PRF}(w_{\xi}, w_{\eta}).
\]  

(2-25)
The values $\xi'$ and $\eta'$ are coordinates along frequency axes that have been rotated by the screen angle, $\theta$, from the reference coordinate system. This rotation is only valid when $\xi$ and $\eta$ represent orthogonal axes, as is the case here.

The simplifying assumption is made that the physical reconstruction function is a rectangle function with width $w$, and thus its spectrum is a sinc function as in Equation 2-26.

$$\text{rect}(x/w, y/w) \Rightarrow w^2 \text{sinc}(w\xi, w\eta)$$

(2-26)

The Fourier transform of the complete halftoning process is expressed as

$$H_r(\xi, \eta) = [s^2 \left[ \text{DOT}(s\xi', s\eta') \right. \text{comb}(s\xi', s\eta') \left. \right] \newline \text{comb}(w\xi, w\eta) \right) \w^2 \text{sinc}(w\xi, w\eta) \right). \quad (2-27)$$

Figure 2-6 illustrates a one-dimensional development of Equation 2-27. The fundamental component of the moire is observed in Figure 2-6 (g) as two impulses located at 1/3 of the halftone frequency. This corresponds exactly to the results derived in the spatial domain in Section 2.3. In this one-dimensional example, the period of the moire pattern was as easy to identify in the spatial domain as in the frequency domain, but the situation is more difficult in two dimensions as shown in the next section. The frequency domain view also shows the moire harmonics.
Figure 2-6 Virtual Screen Halftoning: Frequency Domain
(a) Comb function with period equal to 1/s. (b) DOT function, shown as sinc(2sξ). (c) Product of functions in (a) and (b). (d) Amplitude spectrum of sampling function, a comb function with period equal to 1/w.
Figure 2-6 Virtual Screen Halftoning: Frequency Domain (continued). (e) Amplitude spectrum of sampled uniform virtual halftone. The subscript D indicates a discrete image. (f) Sinc(\(w\xi\)) as PRF. (g) Spectrum of reconstructed halftone which is the product of parts (e) and (f). The subscript r indicates reconstructed image.
Finally, Equations 2-15 and 2-27 are combined into an analytical expression relating the uniform virtually screened halftone spatial model with its Fourier transform as

\[
\left[ \text{dot} \left( \frac{x'}{s}, \frac{y'}{s} \right) \right. \left. \ast \frac{1}{s^2} \text{comb} \left( \frac{x'}{s}, \frac{y'}{s} \right) \right] \ast \frac{1}{w^2} \text{comb} \left( \frac{x}{w'}, \frac{y}{w'} \right) \ast \text{rect} \left[ \frac{x}{w}, \frac{y}{w} \right] = \\
\left[ s^2 \left[ \text{DOT} \left( \frac{s_1'}{s'}, \frac{s_2'}{s'} \right) \ast \text{comb} \left( \frac{s_1'}{s'}, \frac{s_2'}{s'} \right) \right] \ast \text{comb} \left( \frac{w_1}{w'}, \frac{w_2}{w'} \right) \right] \ast \text{sinc} \left( \frac{w_1}{w'}, \frac{w_2}{w'} \right) .
\]

(2-28)

This equation applies when the PRF is \( \text{rect} \left( \frac{x}{w}, \frac{y}{w} \right) \). The screen spectrum has been replicated to infinity with a comb function spaced at the microdot frequency. The replicated screen spectra are weighted by a sinc function that has its first zero value at the microdot frequency. Note that \( \text{comb} \left( \frac{w_1}{w'}, \frac{w_2}{w'} \right) \) is a series of delta functions weighted by a factor of \( \frac{1}{w^2} \) and \( \text{comb} \left( \frac{s_1}{s'}, \frac{s_2}{s'} \right) \) is a series of delta functions weighted by \( \frac{1}{s^2} \). Taking into account all factors, the DC (frequency \( 0,0 \)) amplitude is equal to \( \text{DOT} \left( 0,0 \right) \). From the Central Ordinate theorem (Gaskill 1978), this value is the area of \( \text{dot} \left( x, y \right) \), which by definition is the "dot area." Therefore, the normalized amplitude spectrum is the product of \( \frac{1}{\text{DOT} \left( 0,0 \right)} \) and the amplitude spectrum.

2.5 A View in Two-Dimensions

The two-dimensional view of the screening process is much more complex. Figure 2-7 represents the amplitude spectrum of a continuous, uniform, 150 dots per inch (dpi) screen with round
dots that cover ten percent of the halftone cell area.
It is a quasi three-dimensional plot similar to that used by
Bestenreiner (1975). The frequency space is plotted in two
dimensions. Each impulse is drawn as a circle centered at the
impulse coordinate and with area proportional to the absolute
value of its amplitude. The proportionality constant was chosen
empirically to obtain circles of reasonable sizes. The key in
the lower left provides a reference to absolute amplitude. This
plot corresponds roughly with Figure 2-6 (c).

Figure 2-8 illustrates the effect of sampling on the
spectrum. The screen spectrum has been replicated by convolution
with a comb function at 1411 dpi. This is the frequency of the
film recorder (device for converting digital images to hardcopy)
that was used for this research. This film recorder is described
in Section 3.4. For clarity, only the first +3 components of the
screen spectra are shown. The scale of the plot has been changed
in order to show a full order of the sampling frequency. This
plot is analogous to Figure 2-6 (e). It shows the result of the
sampling operation before reconstruction. Those components of
the replicated screen spectrum that remain after reconstruction
are "aliased." These components cause moire.
Figure 2-7 Amplitude Spectrum of an Unsampled Virtual Screen

Figure 2-8 Effect of Sampling on the Amplitude Spectrum
2.6 The Virtual Screen Font

In practical implementations, the virtual screen is not continuous, but rather is a sampled function. A digital representation of the profile of a single cell is stored in computer memory. For a uniform, single-level cell the profile is stored as a binary "font". It is stated without derivation that the effect on the spectrum would be an additional convolution with a group of impulses located at the font frequency (the frequency at which a continuous screen is sampled to produce the font) and its harmonics. The screen spectrum is repeated about each font impulse. The sampled font is reconstructed in software by convolution with a rectangle with width equal to the font period. The reconstructed function is sampled by the a comb function with spacing w. In the frequency domain, a comb with spacing 1/w is convolved about each font spectrum impulse. More moire patterns can result.

In order to limit the scope of this work, it has been assumed that the virtual screen is continuous. In a practical implementation, this could be approximated by storing fonts with a large number of samples.
3.0 **EXPERIMENTAL**

3.1 **Experimental Approach**

The experimental approach was to implement, test, and evaluate the analytic expression for the uniform input virtual screen halftoning process described by Equation 2-28. The implementation is a computer program that produces a digital halftone image that, when physically reconstructed, represents the left hand side of the equation. The program also produces a results file that includes a description of the amplitude spectrum (essentially the right hand side of the equation).

Previously developed methods were used to physically reconstruct the digital image onto graphic arts film using a film recorder. Hardcopy at an enlarged scale was obtained from a Seikosha thermal printer connected to a display processor to show halftone image detail. Methods were developed for presenting the spectra in graphical form.

To aid in testing the validity of the results, a method was devised for producing and recording the optical transform of the films produced using the film recorder.

Starting with a somewhat arbitrary screen, the parameters for a set of test images were chosen to investigate how moire changes with dot shape, screen frequency, screen angle, and dot size. The experimental procedures are detailed in the sections that follow. The results are evaluated in Section 4, Results and Discussion.
3.2 Software Development

The computer program "moire.c" and a group of supporting modules were written in the C Language for implementation on a Digital Equipment Corporation MicroVAX II computer running the VMS operating system. The program performs two functions: (1) it calculates the amplitude spectrum of a uniform virtual screen halftone image and (2) it produces the corresponding bit-mapped digital halftone image.

The program "xmath.c" and its corresponding header file "xmath.h" extend the basic C language math functions to include some of Gaskill's special functions such as the rectangle, cylinder, sombrero, and sinc functions. Each can be called as either a one-dimensional or a two-dimensional function. The program also includes a four quadrant arctangent function, atan2d.

The source code for all C language computer modules written for this project is given in Appendix B. Only two modules are called from other sources: (1) "write_picture_hdr," which writes an image buffer to a binary file using low-level, and therefore fast, VAX system calls and (2) "besjl," from the SLATEC library which calculates the first order Bessel function of the first kind. The modules "project.h" and "portdefs.h" are general header files containing definitions, typedefs, and macros to help write structured and more portable code. Parts of these files were taken from sources in the public domain. Credit is given in the code listing where this was done.
To use the program on a VAX computer running the VMS operating system, the executable file "moire.exe" should be set up as a foreign command by typing the following command:

```
moire := $ [fullpathname]moire.exe
```

where fullpathname is the directory where moire.exe is located. This allows the program to be executed in a UNIX-like manner by typing the command "moire" followed by the four required arguments listed in Table 1.

The input file contains one or more "specifications groups" that form the main program input. An example of a specifications group follows:

```
out.011
pic.011 512 512
Font___ 0000.0000   0.000 1.000 2   0.000
Writer  1411.1111  0.000 1.000 4   50000.000
Screen   150.000   0.000 0.100 1   75.000
Visibi  00000.0000  000.000 0.000 5  75.000
```

This somewhat cryptic form is interpreted as follows: the first line contains the name of a new file to which results are written; the second line contains the name of the new digital image file, x-direction image size in pixels and y-direction image size in pixels; and the next four lines contain the font, film recorder (writer), screen, and visibility function specifications respectively. Listed in order, the specifications are: specs type, frequency, angle, area of coverage, shape code, and cutoff frequency. Shape codes from 0 to 4 indicates that the spatial domain functional form is respectively an impulse, round, square, diamond, or gaussian. A shape code of 5 indicates that the shape is circular in the frequency domain.
### Table 1

**Required Arguments to moire.c**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_file</td>
<td>string</td>
<td>Name of a file containing one or more specifications groups.</td>
</tr>
<tr>
<td>font_quantizer</td>
<td>integer</td>
<td>The font frequency is set equal to this value multiplied by the halftone frequency.</td>
</tr>
<tr>
<td>font_order_limit</td>
<td>float</td>
<td>This value limits the order of the font amplitude spectrum that is used for the convolution with the film recorder (writer) amplitude spectrum.</td>
</tr>
<tr>
<td>contrast_min</td>
<td>float</td>
<td>Only those amplitude spectrum components with an absolute amplitude greater than this value are reported.</td>
</tr>
</tbody>
</table>

This manner of entering data was intended to provide flexibility and symmetry, but for the examples used in this study only the screen parameters and cutoff frequencies were varied from the values given in the example above. Also, the font specifications are derived from the screen parameters and the command line arguments as follows:

\[
\text{font frequency} = (\text{font_quantizer})(\text{screen frequency})
\]
\[
\text{font angle} = \text{screen angle} \quad (3-1)
\]
\[
\text{font cutoff frequency} = (\text{font_order_limit})(\text{font frequency})
\]
As per the simplifying assumption made in Section 2.6, the parameters were set so that the font was essentially continuous. As a result, the font produced no moire components in the amplitude spectrum. The font was not even implemented as part of the "screen_pic" module, which produces the digital image. Another simplification is that the visibility function was applied only as an ideal low pass filter (i.e. a cylinder function in the frequency domain) with the cutoff frequency as defined in the visibility specifications. For most plots, the cutoff frequency was set to 75 dots per inch (dpi). In Figure 3-1, pseudo-code is used to describe the flow of the moire program.

A possible source of confusion is the use of the linked list to store convolution results. Rather than declaring large, multi-dimensional arrays in advance to store the data for each convolution, the linked list approach simply stores the information in a single line of similar structures that point to the next structure in line. The memory for each structure is declared during run time so it is not necessary to know in advance how many impulses will result from the convolutions. Each structure has a tag that indicates what kind of impulse (font, writer, or screen) it is.
parse the command line arguments
open the input file
loop
  read in specifications group or exit if end of file or error
  make corrections to font specs as necessary
  open a linked list.
  convolve the font spectrum with the origin
to form the "a" spectrum, add component
  to the linked list. Limit the results to those components within the font cutoff frequency.
  convolve the writer spectrum with the "a"
spectrum to form a new "a" spectrum, add each component to the linked list. Limit the results to those components that fall within the writer cutoff frequency.
  convolve the screen spectrum with the "a"
spectrum to form a new "a" spectrum, add each component to the linked list. Limit the results to those components that fall within the screen cutoff frequency.
  calculate the amplitude of each component in the final "a" spectrum.
  print the results to the results file.
  close the linked list.
  generate a screened image
  write the screened image in bit-mapped form to a file.
endloop
close input file
end

Figure 3-1 Flow of program moire.c

Each font impulse is generated by convolving the font spectrum with the origin, a unit impulse located at the origin of the frequency domain. This forms a spectrum called A. Each
writer impulse results from the convolution of the writer spectrum with spectrum A. This forms a new spectrum called B. The screen spectrum is then convolved with B to form the final spectrum, C. The coordinates of each impulse are stored in the data structure. The coordinates can also be derived from the vector sum of the components that caused the impulse and that of its "parents." Only those components from the C convolution are part of the spectrum. The amplitudes for these components are calculated using the information stored in their parent structures.

3.3 Reporting the Amplitude Spectrum

For each specifications group the program moire.c prints a "Moire Program Results Summary" that lists the program arguments, halftone specifications, and the amplitude spectrum (a collection of discrete impulses). For each impulse, the frequency, angle, vector description, and normalized amplitude are given. Refer to Appendix C for results summary examples. The "vector description" is an alternate way of describing the location of a component. There are four values in the description corresponding with two vectors. The first vector gives the harmonic of the film recorder (writer) spectrum that contributed to the component. The second vector gives the harmonic of the rotated screen that contributed to the component. The program also generates an "XYZ" file that contains a list of the frequency, angle, and normalized amplitude for each component in the frequency domain.
A method for producing quasi three-dimensional plots (as in Figures 2-6 and 2-7) was developed. The plots were made on a Hewlett Packard ColorPro plotter driven by a Tandy 3000 PC running a customized version of the ASYST language. ASYST is a general purpose data manipulation, acquisition, and control software package. An ASYST program was written to read data from the XYZ file via a computer network and produce the graph. The ASYST code is not reproduced here since it was somewhat ad hoc and many of the commands were given interactively.

The DC component was excluded from plots scaled to 75 cycles per inch (cpi) because its amplitude, equal to 1.0, is overwhelming compared to the moire component amplitudes.

3.4 Physical Reconstruction of Digital Images

The bit-mapped digital images were transferred via network to another MicroVAX computer that is the host for a Model 6444R Film Recorder manufactured by EKTRON Applied Imaging, Inc. The Model 6444R is capable of exposing film of size up to 64 x 44 inches. with an addressable microdot spacing of either 1411 or 2419 microdots per inch. The film is wrapped (emulsion side out) around a drum that spins at a high rate. An optical head with eight laser beams exposes a swath of eight microdots with each revolution of the drum. The optics head steps along the drum a distance equal to eight microdot spacings to expose the next swath. The laser beams are modulated in a binary fashion by an acousto-optic modulator.
In the film plane, each laser beam has a Gaussian profile with a 50% diameter approximately equal to the microdot spacing. This was confirmed by measuring the beam size with a Photon, Inc. BeamScan Model 1080 beam profile instrument. Some blur in the around-the-drum direction is added by the motion of the spinning drum.

During the film recording process, each bit of the byte determines the status of one of the eight laser beams. Normally, a '1' causes the beam to be turned "on" which results in a dense spot on the film. A '0' leaves the beam turned off so that the film remains clear. The user has the option of reversing the beam status polarity. Two sets of test images were recorded on Eastman Kodak Film Type LPF7. This film, designed for use with film recorders and for camera exposures of line originals, is moderately high contrast, orthochromatic, and negative working. The film recorder exposure level was set based on obtaining an optical density of 1.0 in a particular screened area that is part of a special test target provided by the film recorder manufacturer. When used in this manner, the film recorder produces microdots of a size appropriate for exposing a wide range of halftone patterns and line structures.

The normal film recorder polarity was used to generate halftone patterns with low density dots on high density backgrounds. These images (not presented here) were observed using a diffuse light box.
Reversed polarity was used to generate halftone patterns with high density dots on low density backgrounds. The films were then contact printed onto negative-working photographic paper (Eastman Kodak Paper Type Polycontrast III RC). The halftone patterns on the prints have low density dots on high density backgrounds. The contact prints were exposed in the image plane of a Besseler enlarger with a tungsten source that was filtered by a number four polycontrast filter. The paper was developed in undiluted Dektol at 70°F.

Figure 3-2 Tile Format on the Film Recorder
A test apparatus that allows any pixel-mapped image up to a maximum of 4096 x 4096 pixels to be replicated or "tiled" across the entire film area was used. At 1411 dpi, 4096 microdots results in a tile size of 2.9 inches. The tiling format is shown in Figure 3-2. The film recorder images presented in this paper were written with 4096-pixel tiles, but the screens themselves are either 3096 pixels, with a ± 50 pixel border, or they are grouped in sets of four 2000-bit images with a border in order to separate the groups.

3.5 Generating the Optical Transform

The simple arrangement shown in Figure 3-3 was set up in order to produce the optical transforms of the hardcopy halftone patterns. A source of nearly plane waves of coherent light was produced using a 5-mW Helium-Neon laser and a beam expander. The test film was placed just to the left of an objective lens of 1800 mm focal length. With this arrangement, the optical spectrum is formed one focal length away from the objective lens in the Fourier, or Fraunhofer, plane. With plane wave field illumination, the magnification in the transform plane is entirely dependent on the focal length of the transform lens (Gaskill 1978, 416). The long focal length was chosen in order to obtain a high magnification. The image irradiance at a particular point in the transform plane is proportional to the square of the amplitude of a particular spatial frequency component in the test film. The fact that all phase information has been lost is not of concern here and is actually an advantage.
since translations in the x, y, or z axes do not affect the results.

The image in the transform plane was recorded by exposing photographic paper to the transform for 20 seconds. Since the laser emits at 632.8 nm, a panchromatic paper is necessary. A black-and-white, silver-halide, negative working material (3M Dry Silver Paper Type 7774) was used. The paper was dry processed using a thermal processor for 8 seconds at 273° F. According to the manufacturer, this material has a sensitivity of 39 ergs/cm² at 632.8 nm and a gamma of 2.2 when processed as stated above. The exposure was determined empirically to obtain transform images that showed low-amplitude components without causing over-exposure of the high-amplitude components. A .9 neutral density filter was used to increase exposure time to 20 seconds so that it could be controlled repeatably.

3.6 Selection of Test Images

As a starting point, a halftone pattern with a 0-degree screen angle and 150 dots per inch (dpi) screen frequency was chosen. The choice of 0 degrees was made in order to simplify the analysis. The use of 150 dpi screens is common in the graphic arts.
The dot area (specified as the fractional area of the halftone cell covered by the dot) was set to .10. This dot area results in a low tonal level (a "shadow"). It was reasoned that screener-induced moire would be more significant at low tonal levels because there are a small number of microdots forming each
dot. Also, since the dot area is low, a small change in dot area (such as a single transition of a microdot) will produce a larger change in optical density (density) than at higher tonal levels. Density is defined as

\[ D = -\log_{10}(t), \quad (3-2) \]

where \( t \) is the transmittance. From the derivative of Equation 3-2, a differential change in density \( dD \) is

\[ dD = \left( \frac{c}{t} \right) dt \quad (3-3) \]

where \( c \) is a constant and \( dt \) is the differential transmittance. For the simple model of the halftone described in Section 2 in which the pattern is binary with transmittance either 1.0 or 0.0, the transmittance is equal to the dot area. Therefore, at low tonal levels, a given change in dot area results in a larger density change than would occur at higher tonal levels. Each test image was generated with 1411 samples per inch (the microdot frequency of the film recorder). The prf was assumed to be \( \text{rect}(x/w, y/w) \), where \( w \) is the microdot spacing. This assumption is not valid for the film recorder; the effect of this error on the accuracy of the prediction is discussed in the next section. Table 2 lists the parameters for the test images.

The plan was to observe the effects on the observed moire of making small changes in each the screen characteristics of dot shape, screen frequency, dot size, and screen angle,
independently. It is predicted from Equation 2-28 that small changes in these parameters can produce significant alterations in the moire structure.

Most screens have round dots in low tonal areas. They progress into a diamond (square rotated by 45°) or ellipse near the mid-tones. Highlights are inverse circles. Images 1, 2, and 3 form a series in which dot shape is respectively round, square, and diamond. All other characteristics are kept the same.

Images 5, 6, 7, and 8 form a screen frequency series. The four images were grouped into a single 4096 pixel tile. Images 1, 9, 16, and 17 form a screen angle series. The practical graphics arts angles of 0, 75, 60, and 45 degrees are presented. Images 12, 13, 14, and 15 form a dot size series. These images are also shown as a group of four on one 4096 pixel tile. Note that Images 1, 7, and 13 are the same screen.

A special case was tried with Image 4: the frequency is 141.1111 dpi, an exact submultiple of the microdot frequency. There are exactly 10 microdots per linear halftone cell.
Table 2

List of Experimental Images

<table>
<thead>
<tr>
<th>Image No.</th>
<th>Frequency (dpi)</th>
<th>Angle (deg.)</th>
<th>Dot Area</th>
<th>Dot Shape</th>
<th>Image Size (Pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dot shape series:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>150.000</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>4096</td>
</tr>
<tr>
<td>2</td>
<td>150.000</td>
<td>0.000</td>
<td>10.0</td>
<td>Square</td>
<td>4096</td>
</tr>
<tr>
<td>3</td>
<td>150.000</td>
<td>0.000</td>
<td>10.0</td>
<td>Diamond</td>
<td>4096</td>
</tr>
<tr>
<td>Screen frequency series:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>149.000</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>6</td>
<td>149.5000</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>7</td>
<td>150.0000</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>8</td>
<td>150.5000</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>Screen Angle Series</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>150.000</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>4096</td>
</tr>
<tr>
<td>9</td>
<td>150.000</td>
<td>75.000</td>
<td>10.0</td>
<td>Round</td>
<td>4096</td>
</tr>
<tr>
<td>16</td>
<td>150.000</td>
<td>60.000</td>
<td>10.0</td>
<td>Round</td>
<td>4096</td>
</tr>
<tr>
<td>17</td>
<td>150.000</td>
<td>45.000</td>
<td>10.0</td>
<td>Round</td>
<td>4096</td>
</tr>
<tr>
<td>Dot Size Series</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>150.000</td>
<td>0.000</td>
<td>8.5</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>13</td>
<td>150.000</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>14</td>
<td>150.000</td>
<td>0.000</td>
<td>10.7</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>15</td>
<td>150.000</td>
<td>0.000</td>
<td>11.4</td>
<td>Round</td>
<td>2000</td>
</tr>
<tr>
<td>Special Cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>141.1111</td>
<td>0.000</td>
<td>10.0</td>
<td>Round</td>
<td>4096</td>
</tr>
<tr>
<td>11</td>
<td>150.000</td>
<td>45.000</td>
<td>10.0</td>
<td>Diamond</td>
<td>4096</td>
</tr>
</tbody>
</table>

Note: Images 1, 7, and 13 have an identical screen, but 11 is larger.
4.0 RESULTS AND DISCUSSION

4.1 Initial Results

Initial results were obtained for Image 1, which has screen parameters as follows: 0 degree screen angle, 150 dots per inch (dpi), and round dots with .10 area. The moire program was used to generate a 4096 x 4096 binary pixel digital image and its amplitude spectrum. Figure 4-1 shows a series of results that are specific to Image 1.

Figure 4-1 (a) is a contact print of the image written on the 6444R Film Recorder at the 1411 microdot per inch resolution and reversed polarity. The around-the-drum direction is from left to right. The along-the-drum direction is from the bottom of the image to the top. By examining the image with a microscope scale or using a graphic arts tool such as the RIT Graphic Arts Research Center Screen Pattern Analyzer and Rescreening Key (SPARK), it can be seen that the halftone pattern has a 0-degree screen angle and a 150-dpi screen frequency. The halftone pattern axes are orthogonal. The dots, which are low density on a high density background, have various shapes that are approximations to a circle.

There are on average 9.4 microdots per length of halftone cell (1411/150.) or about 88 microdots per halftone cell. Since the virtual screen was set up as a circle with area of .10, 8.8 microdots per cell, on average, can be expected. In other words, either 8 or 9 microdots form the halftone dot. The actual number and average may vary due to the sampling process. Some aperiodic
structure is visible in Figure 4-1 (a). This is most likely due to position noise on the film recorder optical head.

Figure 4-1 (b) is a photocopy of a magnified version of Image 1 produced on a Seikosha thermal printer. Each block corresponds to one pixel in the digital image (one microdot on the film recorder). There are only about eight or nine pixels per halftone dot, as expected, and the approximation to a circle is poor. The dot shapes and spacings change, but it is difficult to recognize a pattern.

Note that the thermal printer images have an aspect ratio of 1.2 to 1. Therefore, the halftone grid shown in Figure 4-1 (b) is not square. This is not of significance since the thermal printer images are presented to illustrate the halftone pattern and the dot structure. Note that the 6444R film recorder has a square aspect ratio so the patterns produced with it have square halftone grids.

When the image is observed at a normal reading distance of about 11 inches, the basic halftone pattern is not visible, but two moire structures are apparent. Since the halftone pattern is dark and the moire patterns subtle, it is suggested that the image be viewed with a bright, diffuse source. It may be necessary to view the image from a shorter distance.

Both moire patterns are two-dimensional and aligned on an orthogonal grid oriented at 0 degrees. The lower frequency pattern was measured to have a frequency of 5.5 cycles per inch (cpi). The higher frequency, measured at about 28 cpi, appears
to be the fifth harmonic of the lower. Measurements were made by observing the normal polarity film using a diffuse light box. To increase accuracy, the total period of a number of moire cycles (usually 10) was measured.

Figure 4-1 (c) illustrates the appearance of the moire patterns as observed by the author. It is shown at 10X actual size. The actual moire patterns in Figure 4-1 (a) make smoother transitions and have lower apparent contrast, but the basic structures are shown.

The moire frequency values listed in Figure 4-1 (c) were determined by comparing the Moire Results Summary for "out.001" in Appendix C to the measured values. The measured frequency values approximated those of components in the results summary.

The above approach merely identifies fundamental components. A moire pattern formed by one frequency would have a sinusoidal profile. The observed moire profile, a complex pattern of square and circular shapes, indicates that there is significant power at the harmonic frequencies. This is confirmed by the plot of the amplitude spectrum (Figure 4-1 (d)). To avoid confusion caused by large numbers of low amplitude impulses, only those components with amplitudes above .01 were shown. The amplitudes have been normalized so that the DC component has unit amplitude. For this plot, the circles were not filled to avoid obscuring important components underneath.
Figure 4-1 (a)  Image 1: Film Recorder Image

Figure 4-1 (b)  Image 1: Magnified View
Figure 4-1 (c) Image 1: Illustration of Moire Patterns
Figure 4-1 (d) Image 1: Amplitude Spectrum to 1000 cpi

Figure 4-1 (e) Image 1: Amplitude Spectrum to 75 cpi
Figure 4-1 (f) Image 1: Optical Transform
The underlying halftone spectrum can be seen by observing the large circles near the center and the regular grid formed by their harmonics. Each is spaced at 150 cycles per inch (cpi). The other components have been aliased from the film recorder (writer) spectrum, primarily from its first and second orders. Note that at around 700 cpi the pattern of the original screen and that aliased from the fundamental writer components are of nearly equal strength. At higher frequencies, the filtering effect of the physical response function (prf) tends to reduce the amplitude. The four aliased components positioned at 0, 90, 180, and 270 degrees have a frequency of 27.78 cpi. This matches the higher frequency moire that is observed in Figure 4-1 (a).

Figure 4-1 (e) shows the same spectrum with the cutoff frequency changed to 75 cpi and the threshold amplitude to .001. All screen components aliased from film recorder components within 0 to 50000 cpi were considered. All components in this plot are listed in the results summary in Appendix C. It is clear that there is a strong group of components at integer multiples of 27.78 cpi. The vector description for the component 0 degree, 27.78 cpi component is: -2, 0, 19, 0. This is interpreted as the 19th component of the original screen spectrum, aliased from the second negative component of the film recorder spectrum.

The aliased components form an even function. Since the components were aliased from other symmetrical components, they tend to have similar (but enlarged) shapes and patterns.
The circles and squares in the moire profile are related to the dot shape (round) and the sampling grid (square).

The optical transform was generated using the film version of Image 1. The recording of the optical transform is shown in Figure 4-1 (f). A direct comparison can be made between this transform and Figure 4-1 (d). Close examination reveals components of nearly equal strength grouped with a difference frequency of 5.56 cpi. The fundamental 5.56 components are not observed. This may be because they are obscured by the strong DC component.

The comparison of the optical transform with the predicted spectrum is used as evidence of the accuracy of the prediction. In fact, during early tests, a programming error was identified by careful comparison of results. The model predicted the strength of the 61 dpi component as about 4 times as that of the 27.78 dpi component, but it was barely visible on film. After correcting the program error, the predicted spectrum appeared similar to the optical transform.

4.2 Interpretation of the Spectrum

Each pair of components centered about the origin indicate a cosinusoid with an amplitude equal to twice that of the individual components. Nasanen (1984) defined the contrast of an image with a sinusoidal luminance distribution as

\[ C = \frac{\text{luminance amplitude}}{\text{average luminance}}. \]
Since the negative halftone dots are clear on black backgrounds, the luminance amplitudes are proportional to the absolute amplitudes of the spectral sinusoid. The average luminance is proportional, by the same factor, to the average area of coverage. For a device with a rectangle prf, the average area of coverage is approximately equal to the fractional area parameter input into the moire program. Using Equation 4-1, it is seen that an estimate of sinusoidal contrast is equal to twice the amplitude of a single component in the normalized amplitude spectrum. Each opposing pair of coordinates forms a cosinusoid in one of the two dimensions.

A prediction of the visibility of patterns must account for several important factors. Algie (1983) reproduced a plot showing the contrast resolution of the human visual system to sine wave spatial modulation as determined by Ginsburg, and the information is reproduced here in Figure 4-2. The curve applies to a normal viewing distance of about 280 mm (11 inches). The peak response is at .5 cy/mm (12.7 cpi) and has a contrast resolution of about 350, indicating a threshold response to a sinusoid with amplitude of .003.

Before the results of this thesis were analyzed, the author thought that the appearance of a moire pattern required the presence of a component of sufficient amplitude at the correct frequency and angle in the amplitude spectrum. But it was found that two sinusoidal components can add to produce a beat at their
difference frequency. Consider Figure 4-3 (a) in which the sum of two sinusoids of relative frequencies 9 and 10 is plotted. A beat occurs with a frequency of 1. It can be seen from the amplitude spectrum given in Figure 4-3 (b) that the difference frequency is not present. Compare this type of beat with that formed by the product of the two sinusoids shown in Figure

Figure 4-2 Contrast Sensitivity of the Human Visual System Given a 280 mm (11 inch) Viewing Distance.
4-4 (a), and its amplitude spectrum in Figure 4-4 (b). This spectrum is predicted by the convolution theorem. The fundamental difference between the two types of beats is that, in a linear system, the amplitudes of the high frequencies that add to form the a beat are attenuated by the system pass band. The beat in Figure 4-3 (a) would not be present if the function was filtered by an ideal low pass filter with a cutoff frequency less than 9.

Clearly, additive beating plays a significant role in the formation of moire patterns. The amplitude of the Image 1, 5.56 cpi pattern is .0042, but it has a high visibility. The amplitude spectrum shows that difference frequencies of 5.56 are formed from the strong 27.78 and 33.33 components as well as higher frequency pairs. The significance of high frequency pairs is made stronger still by considering that the nature of the physical reconstruction includes a severe non-linearity and that the human visual system also responds non-linearly. Eschbach (1988) proved that power is produced at intermodulation frequencies whenever two sinusoids are added and then a non-linearity is applied. This suggests that low frequency patterns are reinforced both by the additive beating effect and by intermodulation frequencies due to non-linearities. A determination of the effect on pattern visibility was beyond the scope of this work, but the assertion is supported by the results. For all of the test images, a moire was observed at the difference frequency of two high amplitude two components whenever those components were of similar frequency.
Figure 4-3 (a) Sum of Two Cosine Waves of Similar Frequencies

Figure 4-3 (b) Amplitude Spectrum Corresponding with part (a).
Figure 4-4 (a) Product of Two Cosine Waves of Similar Frequencies

Figure 4-4 (b) Amplitude Spectrum Corresponding with part (a).
4.3 Results of Parameter Change Series

A thermal printer image matrix of the 16 test images identified in Table 2 is given in Figure 4-5. This section will analyze results for Images 3, 5, and 9. Images 12 and 15 are discussed but not shown. Similar analyses were applied for the remainder of the images but they have been omitted for brevity.

Figure 4-6 gives the film recorded image (a), moire illustration (b) and amplitude spectrum (c) for Image 3. The dot shape has been changed from round to diamond. Comparison with Figure 4-1 shows clearly that alterations in dot shape dramatically affect the appearance of moire. Since the same sampling function was used for both images, the locations of the aliased components are exactly the same. The amplitudes of the components are different because aliased components are weighted by dot function transform. There are components of significant amplitude in Image 3 that were not significant in Image 1 and vice-versa. The spectrum shows significant 45 degree components at 39.28 cpi and 47.14 cpi. The difference frequency is a 45 degree component at 7.86 cpi. In Figure 4-6 (b), the 47.14 and 7.86 cpi moire patterns have been illustrated to match the observed patterns. The crosswise nature of the components indicates a strong square pattern as would be obtained by crossing two lined grids.
Figure 4-5 Matrix of Test Images:  
(Refer to Table 2)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4-6 (a) Image 3: Film Recorder Image

\[ m_2 = 0.021 " (0.53 \text{ mm}) \]
\[ \zeta_2 = 47.14 \text{ cpi (1.86 c/mm)} \]

Moire 2

\[ m_1 = 0.127" (3.2 \text{ mm}) \]
\[ \zeta_1 = 7.86 \text{ cpi (31 c/mm)} \]

Moire 1

Scale = 10X Actual Size

\[ \theta = 45^\circ \]

Figure 4-6 (b) Image 3: Illustration of Moire Patterns
Figure 4-6 (c) Image 3: Amplitude Spectrum to 75 cpi

Figure 4-7 (a) Film Recorder Image of Screen Frequency Series.
Images: 5 6 7 8
Figure 4-7 (b) Image 5: Illustration of Moire Patterns.

![Moire Patterns Illustration](image)

Figure 4-7 (c) Image 5: Amplitude Spectrum to 75 cpi.

![Amplitude Spectrum](image)
Figure 4-8 (a) Image 9: Film Recorder Image

Figure 4-8 (b) Image 9: Amplitude Spectrum to 75 cpi
Figure 4-8 (c) Image 9: Illustration of Moire Patterns
The dot size series was used to show the effect of small changes in dot size. Just as with dot shape, dot size changes merely alter the amplitudes of the components and not their locations in the frequency domain. The dot size values for Images 12 and 15 (not shown) were chosen so that the sombrero function envelope of the screen pattern function was a zero at 27.78 cpi when the screen spectrum was aliased from the second component of the writer function (vector -2 0 19 0 and all its other permutations). The same type of pattern was observed in all of the images of the series, but the strength was muted in Image 15 and somewhat less so in Image 13. This suggests the possibility with odd shaped dots could be designed as means to reduce the amplitude of specific offending components while maintaining the same sampling geometry.

The four images in the screen frequency series (Images 5, 6, 7, and 8) are shown in 4-7 (a). The frequency values for the series are 149.0, 149.5, 150.0 and 150.5 respectively. Note that Image 7 is the same pattern as Image 1. It is seen that these small changes in frequency dramatically affect the location of aliased components. Examination of the results summaries shows that moire components range, in these examples, from the 10th to the 50th harmonic of the screen spectrum. As described previously the 27.78 component in Image 1 can be described as vector -2, 0, 19, 0. In Image 5, the screen frequency is changed 1.0 cpi from that of Image 1. Therefore, the moire pattern is moved 19 cpi from that of the component in Image 1 which has the
same vector description. In image 11 that component aliases to 27.78 cpi. Subtracting 19 cpi from 27.78 results in 8.78 cpi. A moire pattern is observed at that frequency in Image 5 as illustrated in Figure 4-7 (b). The frequency of the fine structure in that pattern, was too high to be measured visually.

The screen angle series is important in order to understand the true two-dimensional nature of moire. Figure 4-8 shows the results for Image 9. The moire pattern spectrum is clearly aligned along a different direction than that of the halftone pattern. The lower frequency moire in Image 9 is quite subtle. Its form has been exaggerated in Figure 4-8 (c). Note that the scale of the lower frequency moire, in this picture is actually 13X rather than 10X. The scale of the higher frequency pattern is correct.

4.4 Relationship to Scanned Halftone Images

The derivation of Equation 2-28, which relates the uniform virtual screen halftone spatial model to its Fourier transform, was found to be similar to an analysis of moire patterns in scanned halftone pictures by Steinbach and Wong (1982). The scanning process causes the image to be convolved with the scanning aperture function and then sampled. To view the image it is reconstructed on a physical device. Sampling of the virtual screen is conceptually the same except that the aperture
function is a delta function. This realization suggests that a simulated aperture function could be applied as part of the virtual screening algorithm in order to reduce the significance of the side-lobes that are aliased as moire. This is left as a suggestion for future work.
5.0 CONCLUSIONS

A model of the uniform screen virtual screen digital halftoning process and its Fourier transform has been derived and implemented in software. The feature that distinguishes virtual screen methods from other digital halftoning approaches is that the screen function is not related to the grid of addressable locations (microdots) in an integer manner. The screen function is scaled and rotated virtually with respect to this sampling grid. During the screening process, the screen function is sampled at each microdot location.

In deriving the model and amplitude spectrum the following assumptions were made: input to the halftoning process is uniform; the screen function and sampling grid are real even functions, each with orthogonal axes; and the virtual screen is spatially continuous. A further assumption is made that the reconstruction function is a rectangle function with width equal to the sample spacing. This assumption is not valid for real image output devices, but its use did allow a solution in closed form that is quite useful as an initial estimation of the final moire pattern.

The predicted amplitude spectrum has been shown to include low frequency components that combine to form moire patterns. The impression of moire can be seen at frequencies not represented on the amplitude spectrum or not in accordance with the expected amplitude. This is due to the additive beating of two higher frequency components and due to intermodulation of
these components caused by non-linearities of the physical reconstruction processes and the human visual system.

For all of the test images, the amplitude spectrum that was calculated by the moire program could be used to predict the fundamental angles and frequencies of the most significant moire components provided that difference frequencies were taken into account.

The significance of the screen parameter series was to show the variety of patterns that are possible and how they can be affected by changes in these parameters. This understanding can aid in the design of halftoning algorithms. For example, for a given screen angle, screen frequency, and sampling rate, there are dot shapes that will result in less moire than others. One could design a dot shape "tailored" to a set of screen parameters. This is left as an area for future research.

The similarities between screener induced moire and that produced by sampling an existing halftone image have been suggested. The scanning spot function acts as a low pass filter before the sampling takes place. This suggests a possible improvement in the virtual screening algorithm may be to filter the screen function prior to its sampling.

Hardcopy film samples were produced on a film recorder. The optical transform of each film sample was produced. Comparisons of the actual images, optical transforms, and calculated spectra show that the spectra can be predicted. Given a square sampling
grid and a square halftone pattern, all moire patterns are themselves square and symmetric about the 0,0 axis and the amplitude spectrum is symmetric about the origin.

Results were obtained that indicated that small changes in screen frequency or dot size can significantly change the moire result. Changing dot shape or size alters the shape of the screen function spectral envelope but not the location of components. Altering screen angle and frequency changes the entire spectrum.
6.0 LIST OF REFERENCES


APPENDIX A

DESCRIPTION OF USEFUL FUNCTIONS, PROPERTIES, AND OPERATORS

Some useful functions, properties of functions, and operators used in this paper are described. In general, conventions described by Gaskill (1978) are followed.

The rectangle function is defined as follows:

\[
\text{rect}\left(\frac{x-x_0}{b}\right) = \begin{cases} 
1, & |(x-x_0)/b| > 1/2 \\
1/2, & |(x-x_0)/b| = 1/2 \\
0, & |(x-x_0)/b| < 1/2 
\end{cases} \quad (A-1)
\]

It can be seen that \(x_0\) and \(b\) are shifting and scaling parameters respectively. Equation A-2 describes a rectangle, centered at the point \(x_0\), of unit height, and with width and area equal to \(b\).

The Gaussian function is as follows:

\[
\text{Gaus}\left(\frac{x-x_0}{b}\right) = e^{-\pi \left(\frac{x-x_0}{b}\right)^2}, \quad (A-2)
\]

which describes is centered at the point \(x_0\), of unit height, and with total area, \(b\). The "sinc" function is defined as

\[
\text{sinc}\left(\frac{x-x_0}{b}\right) = \frac{\sin(\pi (x-x_0)/b)}{\pi (x-x_0)/b} \quad (A-3)
\]
A single point may be located with a delta (impulse) function, defined by taking the limit as the width of a rectangle function approaches zero as in Equation A-4.

$$\delta((x-x_0)) = \lim_{b \to 0} \frac{1}{|b|} \text{rect}((x-x_0)/b)$$  \hspace{1cm} (A-4)

This function has unit area, infinite height, and zero width. Bracewell (1978, 78) stresses that the "impulse symbol $\delta(x)$ does not represent a function in the sense in which the word is used in analysis." Instead, it must always be interpreted in the limit as in Equation A-4. Note that Equation A-4 is not a unique definition of the delta function. The Gaussian function or a number of other distribution functions could be used in place of the rectangle function.

Bracewell (1978, 76) shows that, given a function $f(x)$ that is continuous at $x = 0$,

$$f(x)\delta(x) = f(0)\delta(x),$$  \hspace{1cm} (A-5)

which is a weighted delta function.

An infinite series of equally spaced delta functions is called a "comb" function. The function

$$\text{comb}((x-x_0)/b) = |b| \sum_{n=-\infty}^{\infty} \delta(x-x_0-nb)$$  \hspace{1cm} (A-6)
defines a series of weighted delta functions spaced $b$ units apart. Each weighted delta function has area $b$. The entire
function is shifted to the right by an amount $x_0$.

In two-dimensional rectangular coordinates, the rectangle, sinc, Gaussian, delta, and comb functions are all "separable." This means that the two-dimensional functions can be described as the product of two one-dimensional functions, each of which depends only on one of the coordinates. For example, the two-dimensional comb function is defined as

$$\text{comb}\left(\frac{x-x_0}{b}, \frac{y-y_0}{d}\right) = \text{comb}\left(\frac{x-x_0}{b}\right) \cdot \text{comb}\left(\frac{y-y_0}{d}\right),$$

where $b$ and $d$ are the spacings of the $x$-direction and $y$-direction delta functions respectively.

The "cylinder" function is defined conveniently in polar coordinates as

$$\text{cyl}(r/b) = \begin{cases} 1, & 0 < r < b/2 \\ 1/2, & r = b/2 \\ 0, & r > b/2 \end{cases}$$

where $r$ is the radial coordinate, $\theta$ is the angular coordinate, and $b$ is the diameter of the cylinder. The cylinder function is not separable in rectangular coordinates, but it is rotationally symmetric since it depends only on $r$, and not $\theta$. 
The sombrero function is defined as

\[ \text{somb}(r/b) = 2J_1(\pi r/b)/(\pi r/b) \quad (A-9) \]

where \(J_1(\cdot)\) is the first-order Bessel function of the first kind.

A function \(f(x)\) is even if

\[ f(x) = f(-x) \quad (A-10) \]

Each function defined in this appendix is real and even when \(x_0\) is equal to zero. A two-dimensional function is even if

\[ f(x, y) = f(-x, -y) \quad (A-11) \]

Given two functions \(f(x)\) and \(g(x)\), the convolution operation is defined as

\[ f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)\,d\alpha \quad (A-12) \]

In this paper, the symbol "*" denotes the convolution operation and "**" denotes two-dimensional convolution.

Bracewell (1978, 77-79) describes two important properties of the comb function. The sampling property is defined as

\[ \text{comb}(x)f(x) = \sum_{n=-\infty}^{\infty} f(n)\delta(x-n) \quad (A-13) \]
The result of the sampling operation is a series of weighted delta functions. The replicating property is defined as

\[
\text{comb}(x) \ast f(x) = \sum_{n=-\infty}^{\infty} f(x-n).
\]  

(A-14)

The function \( f(x) \) has been replicated at unit intervals in both directions to infinity.
APPENDIX B

SOURCE CODE LISTINGS

This Appendix contains the source code for all computer modules written for this project. The files names and a brief description of contents are listed below.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>moire.c</td>
<td>This is the main program for calculation of the amplitude spectrum and for producing digital halftone images.</td>
</tr>
<tr>
<td>xmath.c</td>
<td>Implements some of Gaskill’s functions such as rect(x), somb(r), etc. and special math functions.</td>
</tr>
<tr>
<td>ansi.h</td>
<td>Includes all ANSI standard header files. Will prevent double includes. This is necessary since not all of the standard header files have the #define check mechanism for preventing double includes.</td>
</tr>
<tr>
<td>moire.h</td>
<td>Contains definitions and typedefs for moire.c</td>
</tr>
<tr>
<td>pointer.h</td>
<td>Concise macros for pointer checking and casting.</td>
</tr>
<tr>
<td>portdefs.h</td>
<td>Contains definitions for portability of code. This routine is environment dependent. It should be adjusted to the local system.</td>
</tr>
<tr>
<td>project.h</td>
<td>General project definitions, macros, etc.</td>
</tr>
<tr>
<td>xmath.h</td>
<td>This is the header file for xmath.c. Declares math functions and macros that extend the standard &quot;C&quot; library. Also implements Gaskill’s functions as macros.</td>
</tr>
</tbody>
</table>
Module: moire.c

Overview: This program performs two major functions:
(1) calculates the amplitude spectrum of a uniform, virtual screen halftone image.
(2) produces the corresponding bit-mapped digital halftone image.

Usage: To use the program on a VAX computer the executable file, moire.exe, should be set as a foreign command. To do this type the following at the DCL level:

moire := $ [fullpathname]moire.exe

where fullpathname is the directory where moire.exe is located.

Input: The moire command has four required arguments as follows:

(*char)input_file Name of a file containing one or more specifications groups.
(int)font_quantizer The font frequency is set equal to this value * halftone frequency.
(float)font_order_limit The font has its own spectrum. This value limits the convolution with the writer spectrum to this font harmonic.
(float)contrast_min Only those amplitude spectrum components with normalized amplitude greater than this value are reported.

The input file contains one or more "specifications groups" with the following form:

name of new file to which results summary is reported
name of new digital image file, x-direction picture size, in bits, y-direction picture size in bits.
font specifications
writer (film recorder) specifications
screen specifications
visibility function specifications

The specifications consist of:

specs name: this should be entered exactly
frequency: frequency of periodic structure
angle: angle of periodic structure
area of coverage: for screen this is % of tone /100
shape code: basic shape of function
  0 = impulse
  1 = round
  2 = square
  3 = diamond
  4 = gaussian
  5 = round in frequency domain
cutoff frequency: limits the convolution to this frequency

Example of specifications group:

out.011
pic.011 512 512
Font_ 00000.0000 0.000 1.000 2 0.000
Writer 1411.1111 0.000 1.000 4 50000.000
Screen 150.000 0.000 0.100 1 75.000
Visibi 00000.0000 000.000 0.000 5 75.000

As it turn out many of the specifications not used or are
set later:

font frequency = font quantizer * screen frequency
font angle = screen angle
font cutoff freq = font order limit * font frequency
visibility for now, an ideal cylinder function
low pass filter is assumed. The only parameter that is used is the cutoff frequency.

Description:

The program flow is as follows:

parse the command line arguments
open the input file
loop
  read in specifications group
  make corrections to font specs as necessary
  open a linked list.
  convolve the font spectrum with the origin
     to form the "a" spectrum, add each component
     to the linked list. Limit the results to
     those components within the font cutoff
     frequency.
  convolve the writer spectrum with the "a" spectrum
     to form a new "a" spectrum, add each component
     to the linked list. Limit the results to
     those components within the writer cutoff
     frequency.
  convolve the screen spectrum with the "a" spectrum
     to form a new "a" spectrum, add each component
     to the linked list. Limit the results to
     those components within the screen cutoff
     frequency.
  calculate the amplitude of each component in the
     final "a" spectrum.
  print the results to the results file.
  close the linked list.
  generate a screened image
  write the screened image in bit-mapped form to
  a file.
endloop
end

The use of a linked list adds some confusion. It was
originally intended as a way of avoiding the declaration
of large arrays in advance since there was no way of knowing
how many impulse locations would result. The space for each
point on the linked list is "malloced" (i.e. it is allocated
at run time).

The linked list is just a long list of all of the components
from all stages of the multiple convolutions. Each component,
referred to as an impulse, contains information regarding its
coordinates, amplitude, etc. and it contains a tag that
what convolution it came from. Only those components from the
final "screen" convolution represent the actual spectrum. In
order to calculate amplitude the program effectively looks back
in the list to see what other earlier convolution components it
was convolve from.

The beginning of the linked list might look like this:

origin : font component 0,0 : writer component -1,1 :
screen component -2,-2 : screen component -2,-1 :
screen component -2 0 : ... writer component -1 0 :
screen component -2 -2 ... etc.
The coordinate of the spectrum are located at the vector sum of the font, writer, and screen components.

**Results**

The print_results routine has been modified so that font related information is not printed. This is because the current study assumes that the font is not quantized. The screen-pic routine is not currently set-up to account for font quantization but the font quantization has not been disabled. To convolve with an essentially continuous font the argument for font quantizer and font_order_limit should be passed in as 50,000 and 0.0 respectively.

**Notes:**

Various macros from xmath.h, some of which in turn call functions from xmath.c. Note that the SOMB macro calls the somb function, which calls:

besj1 from the SLATEC library. This routine calculates the first order bessel function of the first kind.

write_picture hdr from IOPIC library:
writes picture to file. This routine is VAX specific. It writes an image with a 512 header to a file. It was used here to provide a header format consistent with the Model 6444 Film Recorder special test software. This routine is also faster than standard C routines because it calls low level system functions. This routine could be replaced with the standard fwrite function.

**Calls:**

Contains:

main main controlling function
parse_input parses argv into options structure: opt
fget_allspecs calls get_specs to read in all specs
fget_specs reads a specifications line from a file
cnv1_laib convolves impulse set 'a' with impulse set 'b'
cnv1_1b convolves a single impulse with impulse set 'b'
calc_amplitudes loops through impulse list; calls amp_spec
amp_spec calculates amplitude of a single impulse
print_results loops through impulse list; prints results
fprint_spec prints a specifications line to a file
update_coordinate used by screen_pic to get new position.
set-up_descriptor inserts elements into descriptor structure for passing string arguments to FORTRAN.

Includes:

"project.h" includes and defines for general "c" projects
"moire.h" defines, structures, function prototypes for this program.
"xmath.h" defines, structures, function prototypes for xmath.

descrip.h VAX: This is found in the system library. It defines structures for calling FORTRAN routines among other things.

**Linking:**

The linking instructions are as follows:

link moire,xmath,iopic.olb,slatec.olb

**Author:**

Richard G. Comeau

**Rev. History:**

da-mon-yr rev name reason
27-mar-1989 1.0 rgc original
28-mar-1989 1.1 rgc got rid of specstype
2.0 rgc add screen_pic
2.1 rgc add .xyz file
2.2 rgc improve documentation

******************************************************************************/*

#include "project.h" /* includes and defines for general "c" projects */
#include "moire.h" /* defines, structures, function prototypes for moire */
#include "xmath.h"  /* defines, structures, function prototypes for xmath */
#include descrip.h /* VAX: This is found in the system library. */
/* It defines structures for calling FORTRAN */
/* routines among other things. */

OPTIONS opt = {"", ",", 0, 0, 0, 0, 0, 0, 0}; /* program options: treated as global */
static IOIC_PIC pic = {"", ",", LUN1, ONE BYTE, 0, 0, 0};
static SPECS font = { 0 }; /* halftone font function */
static SPECS writer = { 0 }; /* laser film writer function */
static SPECS screen = { 0 }; /* halftone screen function */
static SPECS vis = { 0 }; /* visibility function */
static ALLSPECS aspecs = { &font, &writer, &screen, &vis }; static IMPULSE origin = { NULL, NULL, NULL, 0, 0, 0.0, 0.0 };

struct dsc$descriptor picname; struct dsc$descriptor pichdrmsg;
main (int argc, char *argv[])
{
    /* define these variables here to assure non-global but make static */
    /* so that will be stored in the heap rather than the stack */
    static FILE *pifile = NULL;  /* ptr to input file */
    static FILE *pofile = NULL;  /* ptr to output file */
    static ALLSPECs *pasp = &aspecs; /* ptr to all specs */
    static IMPULSE *head = &origin; /* open impulse list */

    int nbytes=1; /* number of bytes per pixel, used for IOPIX */

    /* parse the parameters that were passed in through argv */
    /* exit if they are not correctly passed in */
    EXITIF(parse_input(argc, argv), NO, "cannot parse input", -1);

    /* open the input data file using the name parsed in */
    /* parse input, if there is a problem then exit. */
    EXITIF(pifile = fopen(opt.infile,"r"), NULL, "cannot open input file", -1);

    /* read groups of specifications from the input file */
    /* if EOF, read error, or other error occurs then close files, exit */

    /* this is the major data handling loop */
    /* read in groups of specifications from the input file, */
    /* each group consists of the font, writer, screen, and visibility */
    /* specifications to be used to calculate the spectrum and produce */
    /* the screened image file */

    while (fget_allspecs(pifile, &aspecs, &pic) == YES) {
        pic.ns = pic.ny/8; /* number of swaths in picture */

        /* based on options and data read in from file, calculate the */
        /* remaining font specifications */
        aspecs.pfont->freq = opt.font_quantizer * aspecs.pscreen->freq;
        aspecs.pfont->freq_limit = opt.font_order_limit*aspecs.pfont->freq;
        aspecs.pfont->angle = aspecs.pscreen->angle;
        cnvl_iaib(head, NULL, pasp->pfont);
        /* origin.pspecs = NULL */
        cnvl_iaib(head, pasp->pfont, pasp->pwriter);
        cnvl_iaib(head, pasp->pwriter, pasp->pscreen);
        calc_amplitudes(head, pasp);
        print_results(head, pasp);
        close_list(&origin.next); /* but keep origin as first element */
        screen_pic(pic.buffer, pic.nx, pic.ny, &aspecs);
        setup_descriptor(&picname,pic.filename);
        setup_descriptor(&picname,pic.buffer,&(pic.nx),&(pic.ny),&(nbytes),&pic.hdrmsg);
    }
}
fclose(pifile);
Function:     parse_input
Description:  parses command line arguments.
Parameters:   (int) argc  argument count
              (char *) argv[]  pointer to array of input strings
Returns:      (bool) NO = error, YES = ok
Side Effects: modifies global opt structure
Special Notes: The command line arguments must all be entered, no more, no
              less. They must be entered in the correct order.
*******************************************************************************

bool parse_input(int argc, char *argv[])
{
    if (argc != 5) { /* incorrect number of arguments on comand line */
        printf("\n%s\n%s\n", "parse_input: error, the program should be executed as follows:", "moire inputfile font_quantizer font_order_limit contrast_min");
        if (VAX) {
            printf("\n%s\n", "parse_input: VAX users must install moire as foreign command");
        }
        return (NO);
    }
    else { /* correct number of arguments so make assignments */
        opt.infile = argv[1];
        opt.font_quantizer = atoi(argv[2]);
        opt.font_order_limit = atof(argv[3]);
        opt.contrast_min = fabs(atof(argv[4]));
    }
    return(YES);
}
Function: fget_allspecs
Description: Reads in a complete specifications group.
Parameters: (FILE *) pfile ptr to file
            (ALLSPECS *) pasp ptr to allspecs structure
            (IOPIC_PIC *) pic ptr to picture data structure
Returns: (bool) NO = error or EOF, YES = ok
Side Effects: Updates opt.outfile (opt structure is global).
Special Notes: See the description of specifications group in the introduction.

在北京

bool fget_allspecs(FILE *pfile, ALLSPECS *pasp, IOPIC_PIC *pic)
{
    int a;
    if ((a = fscanf(pfile, " %s
",opt.outfile)) != 1)
        return (NO);
    if ((a = fscanf(pfile, " %d %d %d
",pic->filename,pic->nx,pic->ny)) != 3)
        return (NO);
    if (fget_specs(pfile, pasp->pfont) == NO)
        return (NO);
    if (fget_specs(pfile, pasp->pwriter) == NO)
        return (NO);
    if (fget_specs(pfile, pasp->pscreen) == NO)
        return (NO);
    if (fget_specs(pfile, pasp->pvis) == NO)
        return (NO);
    return (YES);
}

Function: fget_specs
Description: Get specifications from pfile, parse, put into *psp
Parameters: (FILE *) pfile ptr to file
            (SPECS *) psp ptr to specs structure
Returns: (bool) NO = error or EOF, YES = ok
Side Effects: fills the (SPECS *psp) structure.
Special Notes: See the description of specifications group in the introduction.

bool fget_specs(FILE *pfile, SPECS *psp)
{
    short it1;
    if (fscanf(pfile," %s %lf %lf %lf %hd %lf",
                &psp->name,&psp->freq,&psp->angle,&psp->area,&it1, &psp->freq_limit) != 6) {
        return (NO);
    }
    psp->domain_and_shape = (PROFILE)it1;
    return (YES);
}
/*******************************************************************************
Function: cnvl_iaib
Description: Convolves impulse group 'a' (with specs pointed to by pspa) with the impulse group 'b' (with specs pointed to by pspb).
Group 'a' is stored within the singly-linked list pointed to 'p'
Parameters: (IMPULSE *) p ptr to current impulse, starts at top
(SPECS *) pspa ptr to specs of impulse group a
(SPECS *) pspb ptr to specs of impulse group b
Returns: (void)
Side Effects: may add type 'b' impulses to list after each type 'a' impulse.
Special Notes:*******************************************************************************

void cnvl_iaib(IMPULSE *p, SPECS *pspa, SPECS *pspb)
{
    for (; p != NULL ; p = p->next) {
        if (p->pspecs == pspa) /* found a point to convolve with */
            (cnvl_ib(&p, pspb)); /* do so */
    }
}

/*******************************************************************************
Function: cnvl_ib
Description: Convolves a single impulse (from group 'a') with the impulse group 'b' (with specs pointed to by psb).
IMPULSE nodes for the resulting components (that are within the frequency limit defined in *psb) are singly-linked after the impulse pointed to by *ppa.
Parameters: (IMPULSE **) ppa pointer to pointer of group 'a' impulse
(SPECS *) ps specs of new group of impulses
Returns: (bool)
Side Effects: 
Special Notes:*******************************************************************************

void cnvl_ib(IMPULSE **ppa, SPECS *ps)
{
    IMPULSE *p;
    IMPULSE *pn;
    double uba0,vba0;
    double uai,vai,ua;j,vaj;
    double ua,va;
    short i,j,iclow,ichigh,jclow,jchigh;
    /* a-space: standard space */
    /* b-space: impulse as origin, rotated by angle of new impulse group */
    /* c-space: b-space, normalized to frequency of new impulse group */
    /* Find b-space coordinates of a-space origin. */
    uba0 = ROTX(-(*ppa)->u,(-(*ppa)->v),RAD(ps->angle));
    vba0 = ROTY(-(*ppa)->u,(-(*ppa)->v),RAD(ps->angle));

    /* find the c-space coordinates of the "box" that is square with the */
    /* b-space axes and encloses circle: r = SQRT(SQ(u-uba0),SQ(v-vba0)), */
    /* convert to integer coordinates by moving outward */
    iclow = (short)floor((uba0 - ps->freq_limit)/ ps->freq);
    ichigh = (short)ceil ((uba0 + ps->freq_limit)/ ps->freq);
    jclow = (short)floor((vba0 - ps->freq_limit)/ ps->freq);
    jchigh = (short)ceil ((vba0 + ps->freq_limit)/ ps->freq);

    /* find the a-space coordinates of the c-space basis vectors */
    uai = ROTX(ps->freq,0.0,-RAD(ps->angle));
    vai = ROTY(ps->freq,0.0,-RAD(ps->angle));
uaj = ROTX(0.0, ps->freq, -RAD(ps->angle));
vaj = ROTY(0.0, ps->freq, -RAD(ps->angle));

/* loop over component in box, if within freq_limit then link in list */
for (p = *ppa, i = iclow; i<=ichigh; i++) {
    for (j = jclow; j<=jchigh; j++) {
        ua = (*ppa)->u + (uai*i) + (uaj * j);
        va = (*ppa)->v + (vai*i) + (vaj * j);
        if (hypot(ua, va) <= ps->freq_limit) {
            if ((pn = (IMPULSE *)malloc(sizeof(IMPULSE))) == NULL) {
                /* remark("Storage is full",""); */
                /* return(NO); */
            } else {
                pn->next = p->next;            /* assign impulse structure */
                pn->pspecs = ps;
                pn->pamp = NULL;
                pn->i = i;
                pn->j = j;
                pn->u = ua;
                pn->v = va;
                p->next = pn;
                p = pn;
            }
        }
    }
}
}
Function: calc_amplitudes
Description: This function calculates the amplitude of all of the doubly convolved components stored in the link list. Only those components with a font, writer, and screen associated with them are the doubly convolved ones. The amplitude is a function of both the component coordinates, but also its coordinates with respect to the writer component from which it was convolved and in turn from the font component from which the writer component was convolved. The amp_spec routine does the actual amplitude calculation. Finally, the AMPLITUDE structure is filled in. Several intermediate steps are stored but the most important value is contrast, since that is the final amplitude normalized by the DC value. All contrast spectra have a 1.0 value at DC (contrast of sinusoid is twice the value of component).

Parameters: (IMPULSE *) p ptr current impulse, starts at top
(ALLSPECS *) pasp ptr to all specs

Returns: (bool) YES or NO

Side Effects:

Special Notes:

bool calc_amplitudes(IMPULSE *p, ALLSPECS *pasp)
{
    IMPULSE *pf, *pw, *ps;
    double u, v;

    for (pf = pw = ps = NULL; p != NULL; p = p->next) {
        if (p->pspecs == pasp->pfont) {
            pf = p;
            pw = ps = NULL;
        } else if (p->pspecs == pasp->pwriter) {
            pw = p;
            ps = NULL;
        } else if (p->pspecs == pasp->pscreen) {
            ps = p;
        }

        if ((pf != NULL) && (pw != NULL) && (ps != NULL)) {
            if ((ps->pamp = (AMPLITUDE *)malloc(sizeof(AMPLITUDE))) == NULL) {
                fprintf(stderr, "Storage is full");
                return (NO);
            } else {
                u = ps->u - pw->u;
                v = ps->v - pw->v;
                ps->pamp->font_wt = amp_spec(ps->pspecs, u + pf->u, v + pf->v);
                ps->pamp->writer_wt = amp_spec(pw->pspecs, ps->u, ps->v);
                ps->pamp->screen_wt = amp_spec(ps->pspecs, u, v);
                ps->pamp->visible_wt = amp_spec(pw->pspecs, u, v);
                ps->pamp->product = ps->pamp->font_wt * ps->pamp->writer_wt * ps->pamp->screen_wt * ps->pamp->visible_wt;
                ps->pamp->contrast = (ps->pamp->product / ps->pspecs->area / pw->pspecs->area);
            }
        }
    }

    return (YES);
}
Function: amp_spec
Description: Calculates amplitude of a component as the value of a point
Parameters: (SPECS *) psp pointer to specifications structure (double) u,v spatial frequency coordinates
Returns: (double) calculated amplitude
Side effects:
Special Notes:
*******************************************************************************
double amp_spec(SPECS *psp, double u, double v)
{
    double b;
    double up, vp;
    EXITIF(IN_RANGE(psp->area, MIN_AREA, MAX_AREA), FALSE, "amp specs: area range error error", -1);
    up = ROTX(u, v, RAD(psp->angle));
    vp = ROTY(u, v, RAD(psp->angle));
    switch (psp->domain_and_shape) {
    case sp_impulse:
        return(1.0);
        break;
    case sp_round:
        b = (sqrt(psp->area * 4.0 / PI) / psp->freq);
        return(psp->area * SOMB(b*hypot(up, vp)));
        break;
    case sp_square:
        b = sqrt(psp->area) / psp->freq;
        return(psp->area * SINC_XY(b*up, b*vp));
        break;
    case sp_diamond:
        b = sqrt(psp->area) / psp->freq;
        return(psp->area * SINC_XY(b*ROTX(up, vp, RAD(45.0)), b*ROTY(up, vp, RAD(45.0))));
        break;
    case sp_gaussian:
        /* correction, was missing */
        b = sqrt(psp->area) / psp->freq;
        return(psp->area * GAUS_XY(b*up, b*vp));
        break;
    case fr_cylinder:
        return(CYL(hypot(up, vp)/psp->freq_limit/2.0));
        break;
    default:
        return(1.0);
    }
}
Function: print_results

Description:
Parameters:
  (IMPULSE *) p    ptr to impulse list, start at top
  (ALLSPECS *) pasp  ptr to all specs

Returns: b^l

Side effects: LINES PERTAINING TO FONT INFO HAVE BEEN COMMENTED OUT
SINCE ONLY 0 ORDER OF FONT IS BEING CONSIDERED AT THIS TIME

Special Notes: LINES PERTAINING TO FONT INFO HAVE BEEN COMMENTED OUT
SINCE ONLY 0 ORDER OF FONT IS BEING CONSIDERED AT THIS TIME

bool print_results(IMPULSE *p, ALLSPECS *pasp)
{
    char filename[FILENAME_SIZE], shape[20];
    double freq, angle, x[5000], y[5000], z[5000];
    int i=0;
    IMPULSE *pf, *pw, *ps;
    FILE *pofile, *pxyzfile;

    /* set up shape string */
    switch (pasp->pscreen->domain_and_shape) {
        case sp_impulse:
            strcpy(shape,"Impulse");
            break;
        case sp_round:
            strcpy(shape,"Round");
            break;
        case sp_square:
            strcpy(shape,"Square");
            break;
        case sp_diamond:
            strcpy(shape,"Diamond");
            break;
        default:
            strcpy(shape,"Unknown Shape");
    }

    /* open outfile and xyz file (xyz file is used for plotting) */
    EXITIF(pofile = fopen(opt.outfile,"w"), NULL,
           "cannot open output file", -1);
    sprintf(filename,"%s%s", opt.outfile, "xyz");
    EXITIF(pxyzfile = fopen(filename,"w"), NULL,
           "cannot open xyz output file", -1);

    /* print program arguments */
    fprintf(pofile,"MOIRE PROGRAM RESULTS SUMMARY

    PROGRAM ARGUMENTS:
    Results File Name = %s
    Corresponding Picture Name = %s
    Filename = %s
    Normalized Amplitude Threshold = %6.4f
    Font Order Limit = %3.1f
    Font Quantizer Value = %d

    HALFTONE SPECIFICATIONS:
      Name Frequency Angle %Area Shape Frequency",

    /* print specifications */
    fprintf(pofile, "HALFTONE SPECIFICATIONS:

    Name Frequency Angle %Area Shape Frequency",

    /* */
/* fpnnt_specs(pofile, pasp->pfont); */
fprint_specs(pofile, pasp->pfont);

fprint_specs(pofile, pasp->pwriter);
fprint_specs(pofile, pasp->pscreen);
fprint_specs(pofile, pasp->pvis);

/* print amplitude spectrum header */
fprintf(pofile, "%s\n%" AMPLITUDE SPECTRUM:\n


%" Frequency Angle Vector Description Normalized",
%" (c/in) (deg) Writer Screen Amplitude",
%" i j i j");

/* print amplitude spectrum */
/* search through linked list */
/* print those components that are convolved impulses of */
/* screen with writer with font */
/* and have contrast above contrast threshold */
for (pf = pw = ps = NULL ; p != NULL ; p = p->next) {
  if (p->pspecs == pasp->pfont) {
    pf = p;
    pw = ps = NULL;
  } else if (p->pspecs == pasp->pwriter) {
    pw = p;
    ps = NULL;
  } else if (p->pspecs == pasp->pscreen) {
    ps = p;
  }
  if ((pf != NULL) && (pw != NULL) && (ps != NULL)) {
    if (fabs(ps->pamp->contrast) >= opt.contrast_min) {
      freq = hypot(ps->u, ps->v);
      angle = DEG(atan3(ps->v, ps->u));
      printf(pofile, "%7.2f %7.2f %3d %3d %3d %3d %7.4f %7.4f %7.4f\n",
        freq, angle,
        pf->i, pf->j, pw->i, pw->j, ps->i, ps->j,
        ps->pamp->font_wt, ps->pamp->screen_wt,
        ps->pamp->product, ps->pamp->contrast);
    }
  }
}
#endif

fprint_specs(pofile, "%7.2f %7.2f %7.2f %7.4f\n",
  freq, angle, pw->i, pw->j, ps->i, ps->j,
  ps->pamp->product, ps->pamp->contrast);
fprint(pofile, "%7.2f %7.2f %7.2f %7.4f\n",
  ps->u, ps->v, fabs(ps->pamp->contrast));

fclose(pofile);
fclose(pxyzfile);
return(YES);
}
Description: Print specifications to pfile
Parameters: (FILE *) pfile ptr to file
           (SPECS *) psp ptr to specs structure
Returns: (bool) NO = error, YES = ok
Side Effects: none
Special Notes: 
******************************************************************************

bool fprint_specs(FILE *pfile, SPECS *psp)
{
    if (fprintf(pfile,"%s %10.4f %9.4f %9.3f
",
               psp->name, psp->freq, psp->angle, psp->area,
               (int)psp->domain_and_shape, psp->freq_limit) != 6)
        return (NO);
    else
        return (YES);
}
Function: screen_pic
Description: generates the screened picture
Parameters: (utiny) buffer buffer of picture data
(int) nx number of pixels in x-direction
(int) ny number of pixels in y-direction
(ALLSPECS *) pasp pointer to all specs
Returns: (bool) YES or NO
Side effects:
Special Notes:
*******************************************************************************

bool screen_pic(utiny buffer[], int nx, int ny, ALLSPECS *pasp)
{
    double u,v; /* current coordinates of pixel in halftone cell */
    double u0,v0; /* starting coordinates */
    double du, dv; /* delta coordinates for each pixel */
    double b; /* dot size value */
    long ltemp; /* temporary long variable */
    int polarity; /* indicates whether ... */
    int jbit; /* variable that indicates which bit to use */
    int jbyte; /* variable that indicates which byte to start with */
    int i,j; /* counters */
    int ns; /* number of swaths */
    int bord; /* border size */

    /* initialization */
    ns = ny/8;
    ltemp = nx*ns;
    if ( (nx == 4096) && (ny == 4096) )
        bord = 50;
    else
        bord = 0;
    for (i = 0; i<ltemp; i++)
        buffer[i] = 0;

    du = (pasp->pscreen->freq/pasp->pwriter->freq)*
         cos(RAD(pasp->pscreen->angle));
    dv = (pasp->pscreen->freq/pasp->pwriter->freq)*
         sin(RAD(pasp->pscreen->angle));
    u0 = 0.0;
    v0 = 0.0;

    switch (pasp->pscreen->domain_and_shape) {
      case sp_round:
        if (fabs(pasp->pscreen->area <= .50))
            b = sqrt(pasp->pscreen->area * 4.0 / PI);
        else
            b = sqrt((1.0-pasp->pscreen->area) * 4.0 / PI);
        for (j=bord; j<(ny-bord); j++)
            if (j%100 == 0)
                printf("Line number %d finished\n",j);
        u = u0;
        v = v0;
        jbit = j%8;
        jbyte = (j/8)*nx;
        for (i=bord; i<(nx-bord); i++)
            if (CYL XY(u/b,v/b) < 1.0)
                Buffer[jbyte+i]=SET_BIT(buffer[jbyte+i],jbit);
        u = update_coordinate(u,du);
        v = update_coordinate(v,-dv);
        }
    u0 = update_coordinate(u0,du);
    v0 = update_coordinate(v0,du);
    }
break;
case sp_square:
    if (pasp->pscreen->area <= .50)
        b = sqrt(pasp->pscreen->area);
    else
        b = sqrt(1.0-pasp->pscreen->area);
    for (j=bord; j<(ny-bord); j++) {
        u = u0;  
        v = v0;  
        if (j%100 == 0) 
            printf("Line number %d finished\n", j);  
        jbit = j%8;  
        jbyte = (j/8)*nx;  
        for (i=bord; i<(nx-bord); i++) {
            if (RECT_XY(u/b,v/b) < 1.0)
                buffer[jbyte+i]=SET_BIT(buffer[jbyte+i],jbit);
            u = update_coordinate(u,du);
            v = update_coordinate(v,-dv);
        }
        u0 = update_coordinate(u0,du);  
        v0 = update_coordinate(v0,du);
    }
break;
case sp_diamond:
    if (pasp->pscreen->area <= .50)
        b = sqrt(pasp->pscreen->area);
    else
        b = sqrt(1.0-pasp->pscreen->area);
    for (j=bord; j<(ny-bord); j++) {
        u = u0;  
        v = v0;  
        if (j%100 == 0) 
            printf("Line number %d finished\n", j);  
        jbit = j%8;  
        jbyte = (j/8)*nx;  
        for (i=bord; i<(nx-bord); i++) {
            if (RECT_XY(ROTX(u,v,RAD(45.0))/b,
                        ROHY(u,v,RAD(45.0))/b) < 1.0)
                buffer[jbyte+i]=SET_BIT(buffer[jbyte+i],jbit);
            u = update_coordinate(u,du);
            v = update_coordinate(v,-dv);
        }
        u0 = update_coordinate(u0,du);  
        v0 = update_coordinate(v0,du);
    }
break;
default:
    printf(" error in screen_pic: screen dot shape invalid");
}
Function: close_list
Description: close the list accessed via *headp
Parameters: (IMPULSE **) headp to list pointer
Returns: (void)
Side effects: closes list
Special Notes: Based on: Thomas Plum, "Reliable Data Structures in C"
******************************************************************************
void close_list(IMPULSE **headp)
{
    IMPULSE *p;
    IMPULSE *pnext;

    for (p = *headp; p != NULL; p = p->next) {
        pnext = p->next;
        free(p); /* p is (momentarily) undefined */
    }

    *headp = NULL; /* prevent dangling ptr */
}
******************************************************************************
Function: double update_coordinate
Description: Updates to position of next coordinate for screening algorithm.
The coordinate wraps around a cell that ranges from -.5 to .5.
Parameters: (double) x coordinate
            (double) dx delta to next coordinate
Returns: halftone cell coordinate (-.5 : .5) of next microdot
Side effects: Special Notes: assumes that -.5 <= x passed in <= .5 and that
              - .5 < dx < .5 . No explicit check made.
******************************************************************************
double update_coordinate(double x, double dx)
{
    x = x+dx;
    if (x < -.5)
        x = x+1.0;
    if (x > .5)
        x = x-.5;
    return(x);
}
******************************************************************************
Function: void setup_descriptor
Description: Sets up character string descriptor structure for passing
              string to a FORTRAN routine.
Parameters: struct dsc$descriptor *a_descriptor string descriptor
            char *a_string the string
Returns: nothing
Side effects: Special Notes:******************************************************************************
void setup_descriptor(struct dsc$descriptor *a_descriptor, char *a_string)
{
    a_descriptor->dsc$w_length = strlen(a_string);
    a_descriptor->dsc$b_dtype = DSC$K_DTYPE_T;
    a_descriptor->dsc$b_class = DSC$K_CLASS_S;
    a_descriptor->dsc$a_pointer = a_string;
}
Module: xmath.c

Overview: Declares math functions that extend the standard "C" math library.

All of the functions are of type (double) except as noted. Most of the functions have a corresponding macro with the same name but uppercase, for consistancy with other macros. The macros explicitly cast float arguments to double.

Contains:
atan3
step
rect
ramp
somb

Includes: xmath.h

Author: Richard G. Comeau

Rev. History:

<table>
<thead>
<tr>
<th>da-mon-yr</th>
<th>rev</th>
<th>name</th>
<th>reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-mar-1989</td>
<td>1.0</td>
<td>rgc</td>
<td>original</td>
</tr>
</tbody>
</table>

*******************************************************************************

#include "xmath.h"

*******************************************************************************

Function: atan3

Description: Computes the arctangent in radians of y/x given y and x.

Parameters:
  y (double) range: (-DBL_MIN : DBL_MAX)
  x (double) range: (-DBL_MIN : DBL_MAX)

Returns: (double) value of atan3(y, x) in radians, range: {-180 : 180}

Side effects: none

Special Notes: Special case if x = 0.0: if y < 0.0, return(-PI/2.0)
  y = 0.0, return(0.0)
  y > 0.0, return(+PI/2.0)

  Doesn't handle EDOM or ERANGE errors that could be caused
  by y/x.

*******************************************************************************

double atan3 (double y, double x)
{
    if (x == 0.0) {
        return(sgn(y, 0., 1.0)*PI / 2.0); /* sgn is defined in Gaskill functions*/
    }
    else {
        return(atan2(y, x)); /* DEG computes degrees from radians */
    }
}
gaskill's special functions:


In the discussion that follows, the term 'function' is used for describing mathematical functions as well as C language routines. The meaning should be obvious from the context.

One-dimensional functions:

These are real valued functions of the real independent variable x. These functions have either one parameter, x, or three parameters x, x0, and b.

In Gaskill's notation, "x0 is a real constant that essentially determines the 'position' of the function along the x-axis, and the real constant b is a scaling factor that regulates the orientation of the function about the point x=x0 and is usually proportional to its 'width.' " If x0 is positive then the function is shifted to the right. If b is negative then the function is reflected about the line x = x0.

The value of the symmetrical functions (i.e. those with symmetrical shapes) depends only on \(|(x-x0)/b|\). These functions have only one formal parameter, x. The calling routine should pass \((x-x0)/b\) as its argument.

Two-dimensional functions in rectangular coordinates:

These are real valued functions of the real independent variables x and y. These functions have two parameters, x and y. All of these functions are symmetrical. The calling routine should pass \((x-x0/bx),(y-y0/by)\) as arguments.

Two-dimensional functions in polar coordinates:

These functions have one parameter, r. The calling routine should pass the argument \((r/d)\) where d is a diameter scaling value.
/*******************************
Function: step
Description: Computes the value of step function.
Parameters: x, x0, b
Returns: (double) value of step(x, x0, b)
Side effects: none
Special Notes: The value at the discontinuity is set to .5 by definition. The computation below does not parallel the book definition to avoid wasting time by dividing both x and x0 by b. The sign of b determines which direction the step is made.
******************************************************************************/

double step(double x, double x0, double b)
{
    double f;
    if (b < 0.0) {
        if (x > x0)
            f = 0.0;
        else if (x == x0)
            f = 0.5;
        else
            f = 1.0;
    }
    else if (b > 0.0) {
        if (x < x0)
            f = 0.0;
        else if (x == x0)
            f = 0.5;
        else
            f = 1.0;
    }
    else {            /* if b == 0, step((x-x0)/b) not defined, so set to 0.0 */
        f = 0.0;
    }
    return(f);
}
Function: sign

Description: Computes the value of sign function.
Parameters: x, x0, b
Returns: (double) value of sign(x, x0, b)
Side effects: none
Special Notes: The value at the discontinuity is set to 0.0 by definition. The computation below does not parallel the book definition to avoid wasting time by dividing both x and x0 by b. The sign of b determines which direction the step is made.

```c
double sign(double x, double x0, double b)
{
    double f;
    if (b < 0.0) {
        if (x > x0)
            f = -1.0;
        else if (x == x0)
            f = 0.0;
        else
            f = 1.0;
    }
    else if (b > 0.0) {
        if (x < x0)
            f = -1.0;
        else if (x == x0)
            f = 0.0;
        else
            f = 1.0;
    }
    else { /* if b == 0, sign((x-x0)/b) not defined, so set to 0.0 */
        f = 0.0;
    }
    return(f);
}
```

Function: rect

Description: Computes the value of rectangle function.
Parameters: x
Returns: (double) value of rect(x)
Side effects: none
Special Notes: 

```c
double rect(double x)
{
    double f;
    if (fabs(x) > .5)
        f = 0.0;
    else if (fabs(x) == .5)
        f = 0.5;
    else
        f = 1.0;
    return (f);
}
```
/* Function: ramp */

Description: Computes the value of ramp function.

Returns: (double) value of ramp(x,x0,b)

Parameters: x, x0, b

Side effects: none

Comments: The value at the discontinuity is set to 0.0 by definition. The computation below does not parallel the book definition to avoid wasting time by dividing both x and x0 by b. The sign of b determines which direction the step is made.

double ramp(double x, double x0, double b)
{
    double f;
    if (b < 0.0) {
        if (x >= x0)
            f = 0.0;
        else
            f = (x-x0)/b;
    }
    else if (b > 0.0) {
        if (x <= x0)
            f = 0.0;
        else
            f = (x-x0)/b;
    }
    else {
        /* if b == 0, ramp((x-x0)/b) not defined, so set to 0.0 */
        f = 0.0;
    }
    return(f);
}

/* Function: sombrero */

Description: Computes the value of sombrero function.

Returns: (double) value of sombrero(x)

Parameters: x

Side effects: none

Comments: This routine calls the SLATEC FORTRAN library routine: besjl which calculates the first-order Bessel function of the first kind. The argument and return value are type (float). The maximum input value of besjl is MAX_BESSEL_INPUT = 62.5. To be consistent with other routines, the argument and return value of sombrero(x) are of type (double).

double sombrero(double x)
{
    float temp_x;
    double f;

    temp_x = (float)(PI * fabs(x));

    if (temp_x > MAX_BESSEL_INPUT)
        f = 0.0;
    else if (temp_x == 0.0)
        f = 1.0;
    else
        f = (double) (2.0 * besjl(&temp_x) / temp_x);
return (f);
Module: ansi.h

Overview: includes ANSI header files, will prevent double includes - this is done since not all of the standard header files have mechanism (i.e. unique #define check) for preventing double includes.

Author: Richard G. Comeau

Rev. History: da-mon-yr rev name reason
20-mar-1989 1.0 rgc

/*******************************************************************************
Module: ansi.h
Overview: includes ANSI header files, will prevent double includes - this is done since not all of the standard header files have mechanism (i.e. unique #define check) for preventing double includes.
Author: Richard G. Comeau
Rev. History: da-mon-yr rev name reason
20-mar-1989 1.0 rgc
*******************************************************************************/

#ifndef ANSIH_H
#define ANSIH_H
#endif

#ifndef STDIO_H
#define STDIO_H
#include <stdio.h>
#endif

#ifndef CTYPE_H
#define CTYPE_H
#include <ctype.h>
#endif

#ifndef MATH_H
#define MATH_H
#include <math.h>
#endif

#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif

#ifndef STDIO_H
#define STDIO_H
#include <stdio.h>
#endif
/* standard input output functions */

#ifndef CTYPE_H
#define CTYPE_H
#include <ctype.h>
#endif
/* character functions */

#ifndef MATH_H
#define MATH_H
#include <math.h>
#endif
/* math functions */

#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
/* (ANSI) standard definitions */

#ifndef LIMITS_H
#define LIMITS_H
#include <limits.h>
#endif
/* (ANSI) machine parameters */

#ifndef STRING_H
#define STRING_H
#include <string.h>
#endif
/* (ANSI) string functions */

#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
/* (ANSI) string miscellaneous standard functions */

#endif
/*************** Header: moire.h *********************/
_header for program: moire
Author: Richard G. Comeau
Rev. History: da-mon-yr rev name reason
27-mar-1989 1.0 rgc original
28-mar-1989 1.1 rgc simpler getspecs
23-jun-1990 2.0 rgc changed options
29-jun-1990 2.1 rgc modified specs, options

ifndef MOIRE_H
#define MOIRE_H

/* don't allow header file to be read in twice */

/* defines */
#define MIN_AREA 0.0
#define MAX_AREA 1.0

/* defines for storing images using IOPIC procedures */
#define IOPIC_HDR_LEN 512
#define IOPIC_HDRMSG_LEN 440
#define FILENAME_SIZE 81
#define ONE_BYTE 1
#define LUN1 1
#define LUN2 2
#define BUFFER_SIZE 4096*512

/* structure typedefs */

typedef enum profile { /* code for profile: domain_shape { 0 : 5 } */
    sp_impulse, /* "sp" indicates that the shape is of an object */
    sp_round,  /* in the spatial domain */
    sp_square, /* "fr" indicates that the shape is of an object */
    sp_diamond, /* in the frequency domain */
    sp_gaussian,
    fr_cylinder
} PROFILE;

typedef struct options { /* program options: infile and outfile required */
    char *infile;
    char *outfile[FILENAME_SIZE]; /* number of quantization cells per halftone cell */
    int font_quantizer;
    double font_order_limit; /* factor of font frequency to be used */
    double contrast_min; /* ignore impulses with contrast less, default = 0.0 */
    bool inp_metric; /* input spatial units: 0 = english, 1 = metric */
    bool out_metric; /* output spatial units: 0 = english, 1 = metric */
    bool verbose; /* verbose user info printed: 0 = NO, 1 = YES */
} OPTIONS;

typedef struct specs { /* name of specs group */
    char name[20]; /* fundamental frequency */
    double freq;
    double angle; /* angle, measured counterclockwise from horiz. */
    double area; /* profile area of coverage: { 0.0 : 1.0 } */
    PROFILE domain_and_shape; /* profile shape code { 0 : 4 } */
    double freq_limit; /* highest frequency to save after convolution */
} SPECS;

typedef struct allspecs { /* contains all specs structures: */
    SPECS *pfont; /* font specifications */
    SPECS *pwriter; /* film recorder (writer) specifications */
    SPECS *pscreen; /* screen function specifications */
    SPECS *pvis; /* visibility function specifications */
} ALLSPECS;
typedef struct amplitude  /* contains amplitude info for a spectrum impulse */
  double font_wt;
  double writer_wt;
  double screen_wt;
  double visib_wt;
  double product;
  double contrast;  /* this is normalized amplitude, DC value is = 1.0 */
} AMPLITUDE;

typedef struct impulse  /* contains all info for a single spectrum impulse */
  struct impulse *next; /* pointer to next impulse structure (linked list */
  AMPLITUDE *pamp;  /* this impulse has specs pointed to here */
  short i;  /* relative coordinates of impulse in integer */
  short j;  /* multiples of the specs frequency */
  double u;  /* absolute frequency coordinates of impulse */
  double v;
} IMPULSE;

typedef struct iopic_pic  /* contains picture info */
  char filename[FIELMEN SIZE];
  char hdrmsg[IOPIC_HDRMSG_LEN];
  int lun;  /* logical unit number used by FORTRAN */
  int nb;  /* number of bytes per pixel */
  int nx;  /* number of pixels in x direction */
  int ny;  /* number of pixels in y direction */
  int ns;  /* number of swaths in picture (for bit-mapped images) */
  unsigned char buffer[BUFFER_SIZE];  /* bit-mapped data is here */
} I0PIC_PIC;

/* function prototypes */
/* This helps to avoid errors when passing arguments to functions */
/* All of these functions are declared in moire.c */

main (int argc, char *argv[]);  
bool parse_input(int argc, char *argv[]);  
bool fget_allspecs(FILE *pfile, ALLSPECS *pasp, I0PIC_PIC *pic);  
bool fget_specs(FILE *pfile, SPECS *psp);  
void cnvl_iaib(IMPULSE *head, SPECS *pspa, SPECS *pspb);  
void cnvl_ib(IMPULSE **ppa, SPECS *ps);  
bool calc_amplitudes(IMPULSE *p, ALLSPECS *pasp);  
double amp_spec(SPECS *psp, double u, double v);  
void close_list(IMPULSE **phead);  
bool screen_pic(utiny buffer[], int nx, int ny, ALLSPECS *pasp);  
double update_coordinate(double x, double dx);  
void setup_descriptor(struct dsc$descriptor *a_descriptor, char *a_string);  
#endif
#include <stdio.h>

#define PNN(p) ((p) != NULL ? (p) : PNN_BAD(p))
#define PNNC(p, t) ((p) != NULL ? (t)(p) : (t)PNN_BAD(p))
#define PNN_BAD(p) (fprintf(stderr, "NULL\n"), p)

#ifdef NDEBUG

  #define PNN(p) (p)
  #define PNNC(p, t) ((t)(p))

#endif

#ifdef NDEBUG

  #define PNN(p) test that p is not NULL, return p */
  #define PNNC(p, t) test that p is not NULL, return p cast to type t*/

  #define PNN(p) ((p) != NULL ? (p) : PNN_BAD(p))
  #define PNNC(p, t) ((p) != NULL ? (t)(p) : (t)PNN_BAD(p))
  #define PNN_BAD(p) (fprintf(stderr, "NULL\n"), p)

#endif
Module: portdefs.h

Overview: definitions for portability
ENVIRONMENT DEPENDENT  ADJUST TO LOCAL SYSTEM

Source: Thomas Plum, "Reliable Data Structures in C," Plum Hall Inc.,

Rev. History: da-mon-yr  rev  name  reason  VAX version
14-mar-1989  1.0  rgc

/******************************************************************************
Module:  portdefs.h
Overview: definitions for portability
ENVIRONMENT DEPENDENT  ADJUST TO LOCAL SYSTEM
Source: Thomas Plum, "Reliable Data Structures in C," Plum Hall Inc.,
Rev. History: da-mon-yr  rev  name  reason  VAX version
14-mar-1989  1.0  rgc
*******************************************************************************/
ifndef PORTDEFS_H
#define PORTDEFS_H
	/* adjust these names to local machine/compiler environment */
typedef unsigned short ushort;  /* or "unsigned" if short-size== int-size */
typedef unsigned char utiny;  /* to get unsigned byte */
typedef unsigned index_t;  /* may be chosen ad-lib locally */
typedef void *data_ptr;  /* use ANSI "generic ptr" if available */

	/* next 5 names require no local changes, will work anywhere */
typedef char tbits;  /* one byte, for bitwise uses */
typedef char tbool;  /* one byte: \{0:1\} */
typedef ushort bits;  /* 16 bits (or moire), for bitwise uses */
typedef int bool;  /* for functions returns: \{0:1\} */
typedef short metachar;  /* return from getchar: \{EOF,0:UCHAR_MAX\} */

	/* modulo function giving non-negative result */
#define IMOD(i,j) ((l((i) % (j)) < 0 ? ((i) % (j)) + (j) : ((i) % (j))))

	/* portably convert unsigned number to signed */
#define UI_TO_I(ui) (int)(ui)  /* more complicated on ones complement */

	/* structure offsets and bound; adjust to local system */
#define STRICT_ALIGN int  /* adjust to local alignment requirement */
#define OFFSET(st, m)  
	((char *)(struct_addr)->m - (char *)&struct_addr)
#define BOUNDOP(t)  
	((char *)(struct char byte0; t byten; ) *)&struct_addr->byten - 
	(uchar *)&struct_addr)
static STRICT_ALIGN struct_addr = 0;
#define STRUCTASST(a,b) memcpy(&a, &b, sizeof(a))

	/* define constants that are system dependent */
#define FAIL 1  /* failure exit */
#define SUCCEED 0  /* normal exit */
#define STDIN 0  /* standard input */
#define STDOUT 1  /* standard output */
#define STDERR 2  /* standard error output */
#if 0  /* the next three are defined in VAX stdio.h */
#define SEEK_SET 0  /* seek relative to start of file */
#define SEEK_CUR 1  /* seek relative to current position */
#define SEEK_END 2  /* seek relative to end */
#endif
#endif
Module: project
Overview: general project definitions and includes
Author: Richard G. Comeau
Source: Thomas Plum, "Reliable Data Structures in C," Plum Hall Inc.,
Rev. History: da-mon-yr rev name reason
20-mar-1989 1.0 rgc VAX version
1.1 rgc bit function byte order changed
******************************************************************************
#ifndef PROJECT_H
#define PROJECT_H
#include "ansih.h"
#include "portdefs.h"
/* defined constants that are not system dependent */
#if 0 /* the next two lines are defined in VAX stdio.h */
#define FALSE 0 /* Boolean value */
#define TRUE 1 /* Boolean value */
#endif
#define NO 0 /* Boolean value */
#define YES 1 /* Boolean value */
/* useful macros that are not system dependent */
#define FOREVER for (;;) /* endless loop */
#define getln(s, n) ((fgets(s, n, stdin)==NULL) ? EOF : strlen(s))
#define ABS(x,y) ((x) < (0)) ? -(x) : (x)
#define MAX(x,y) ((x) < (y)) ? (y) : (x)
#define MIN(x,y) ((x) < (y)) ? (x) : (y)
#define DIM(a) (sizeof(a) / sizeof(a[0]))
#define IN_RANGE(n,lo,hi) ((lo) <= (n) && (n) <= (hi))
#define EXITIF(a,b,c,d) if((a) == (b)) { 
  fprintf(stderr,"\n%s\n",(c)); 
  exit(d); } /* rgc */
 ifndef NDEBUG
#define asserts(cond, str) \ 
  (if (!cond) fprintf(stderr,"Assertion 's' failed\n",str));
#else
#define asserts(cond,str)
#endif
#define SWAP(a,b,t) ((t) = (a), (a) = (b), (b) = (t))
#define LOOPDN(r,n) for ((r) = (n)+1; --(r) >0; )
#define STREQ(s, t) strcmp(s, t) == 0
#define STRLT(s, t) strcmp(s, t) < 0
#define STRGT(s, t) strcmp(s, t) > 0
/* macros and definitions from C-Language Programming Standard, July 28, 1986, Advanced System Group, Government Systems Division, Eastman Kodak Company; that are not already defined above */
#define LURSHIFT(n,b) (((long)(n)) >> (b)) & (0x7FFFFFFF >> (b-1)))
/* bit definitions */
#define B0 0x01
#define B1 0x02
#define B2 0x04
#define B3 0x08
#define B4 0x10
#define B5 0x20
#define B6 0x40
#define B7 0x80
#define B8 0x100
#define B9 0x200
#define B10 0x400
#define B11 0x800
#define B12 0x1000
#define B13 0x2000
#define B14 0x4000
#define B15 0x8000

/* Other macros: bit shifting order modified by rgc
   to work with 6444 convention */

#define BJTEST(a,x) ((a) & (0x01 << (int)(x)))
#define BITEST(a,x) ((unsigned char)(a) & (0x01 << (int)(x)))
#define IIEOR(a,x) ((short)(a) ^ (short)(x))
#define MOD(a,x) ((int)(a) % (int)(x))
#define SET_BIT(a,x) ((unsigned char)(a) | (0x01 << (int)(x)))

#undef
Module: xmath.h
Overview: Declares math functions and macros that extend the standard "C" math library.

All of the macros will cast float arguments to double and return a double.
Some of the macros call one or more functions in xmath.c.
Most of the functions have a corresponding macro that will first cast float arguments to double before calling the function, but if the functions are used directly any float arguments should first be casted to double in the calling statement.

Author: Richard G. Comeau

History: da-mon-yr rev name reason
8-mar-1989 1.0 rgc original

#ifndef XMATH_H
#define XMATH_H
#endif

#ifndef MATH_H
#define MATH_H
#include <math.h>
#endif
#define PI 3.141592653589793
#define RADIANS_PER_DEGREE (PI/180.)
#define MAX_BESSEL_INPUT 62.5

/*******************************************************************************
* declare function prototypes for xmath.c */
double atan3 (double y, double x);
double step (double x, double x0, double b);
double sgn (double x, double x0, double b);
double rect (double x);
double ramp (double x, double x0, double b);
double somb (double r);
float besjl (float *ptr_r);
/*******************************************************************************
* general math macros:
*
* RAD convert angle from degrees to radians
* DEG convert angle from radians to degrees
* ROTX rotation transformation x,y,angle to xprime
* ROTY rotation transformation x,y,angle to yprime
* (angle = angle_prime measured counterclockwise)
*******************************************************************************/
#define RAD(x) ( (double)(x) * RADIANS_PER_DEGREE )
#define DEG(x) ( (double)(x) / RADIANS_PER_DEGREE )
#define ROTX(x,y,a) ( (x)*cos(a) + (y)*sin(a) ) /* UNSAFE */
#define ROTY(x,y,a) ( -(x)*sin(a) + (y)*cos(a) ) /* UNSAFE */
gaskill's special functions:


ستراتيجize the following macros:

```c
#define STEP(x,xO,b) (step((double)(x),(double)(xO),(double)(b)))
#define SGN(x,xO,b) (sgn ((double)(x),(double)(xO),(double)(b)))
#define RECT(x) (rect((double)(x)))
#define RAMP(x,xO,b) (ramp((double)(x),(double)(xO),(double)(b)))
/* next three macros are UNSAFE */
#define TRI(x) ((fabs((double)(x))>1.0) ? 0. : 1.0-(fabs((double)(x))))
#define SINC(x) (((double)(x) == 0.0) ? 1.0 : sin(PI*(x))/(PI*(x)))
#define GAUS(x) (exp(-PI * (x) * (x)))
#define CYL(r) (rect((double)(r)))
#define SOMB(r) (somb((double)(r)))
#define RECT_XY(x,y) (rect((double)(x)) * rect((double)(y)))
#define TRI_XY(x,y) (TRI ((double)(x)) * TRI((double)(y)))
#define SINC_XY(x,y) (SINC((double)(x)) * SINC((double)(y)))
#define GAUS_XY(x,y) (GAUS((double)(x)) * GAUS((double)(y)))
#define CYL_XY(x,y) (CYL (hypot((double)(x),(double)(y))))
#define SOMB_XY(x,y) (somb (hypot((double)(x),(double)(y))))
```
APPENDIX C

MOIRE PROGRAM RESULTS SUMMARIES

This Appendix contains the Moire Program Results Summary listings for test Images 1, 3, 5, and 9. The listings are identified by the output file name.

<table>
<thead>
<tr>
<th>Image</th>
<th>Output File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>out.001</td>
</tr>
<tr>
<td>3</td>
<td>out.003</td>
</tr>
<tr>
<td>5</td>
<td>out.005</td>
</tr>
<tr>
<td>9</td>
<td>out.009</td>
</tr>
</tbody>
</table>
PROGRAM ARGUMENTS:

Results File Name = out.001
Corresponding Picture Name = pic.001
Normalized Amplitude Threshold = 0.0010
Screen Dot Shape = Round

HALFTONE SPECIFICATIONS:

<table>
<thead>
<tr>
<th>Name</th>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>%Area</th>
<th>Shape Code</th>
<th>Frequency Limit (c/in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writer</td>
<td>1411.1111</td>
<td>0.0000</td>
<td>1.000</td>
<td>2</td>
<td>50000.000</td>
</tr>
<tr>
<td>Screen</td>
<td>150.0000</td>
<td>0.0000</td>
<td>0.100</td>
<td>1</td>
<td>75.000</td>
</tr>
<tr>
<td>Visibi</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.000</td>
<td>5</td>
<td>75.000</td>
</tr>
</tbody>
</table>

AMPLITUDE SPECTRUM:

<table>
<thead>
<tr>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>Vector Description</th>
<th>Normalized Amplitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.79</td>
<td>-99.46</td>
<td>-5 -3 47 28</td>
<td>-0.0025</td>
</tr>
<tr>
<td>28.33</td>
<td>101.31</td>
<td>-5 -2 47 19</td>
<td>-0.0018</td>
</tr>
<tr>
<td>61.36</td>
<td>-95.19</td>
<td>-5 -1 47 9</td>
<td>0.0021</td>
</tr>
<tr>
<td>5.56</td>
<td>180.00</td>
<td>-5 0 47 0</td>
<td>0.0042</td>
</tr>
<tr>
<td>61.36</td>
<td>95.19</td>
<td>-5 1 47 -9</td>
<td>0.0021</td>
</tr>
<tr>
<td>28.33</td>
<td>-101.31</td>
<td>-5 2 47 -19</td>
<td>-0.0018</td>
</tr>
<tr>
<td>33.79</td>
<td>99.46</td>
<td>-5 3 47 -28</td>
<td>-0.0025</td>
</tr>
<tr>
<td>64.79</td>
<td>-30.96</td>
<td>-4 -3 38 28</td>
<td>0.0039</td>
</tr>
<tr>
<td>62.11</td>
<td>26.57</td>
<td>-4 -2 38 19</td>
<td>0.0013</td>
</tr>
<tr>
<td>55.56</td>
<td>0.00</td>
<td>-4 0 38 0</td>
<td>-0.0048</td>
</tr>
<tr>
<td>62.11</td>
<td>-26.57</td>
<td>-4 2 38 -19</td>
<td>0.0013</td>
</tr>
<tr>
<td>64.79</td>
<td>30.96</td>
<td>-4 3 38 -28</td>
<td>0.0039</td>
</tr>
<tr>
<td>33.79</td>
<td>-170.54</td>
<td>-3 -5 28 47</td>
<td>-0.0025</td>
</tr>
<tr>
<td>64.79</td>
<td>120.96</td>
<td>-3 -4 28 38</td>
<td>0.0039</td>
</tr>
<tr>
<td>47.14</td>
<td>-135.00</td>
<td>-3 -3 28 28</td>
<td>-0.0019</td>
</tr>
<tr>
<td>43.39</td>
<td>140.19</td>
<td>-3 -2 28 19</td>
<td>-0.0035</td>
</tr>
<tr>
<td>69.61</td>
<td>-118.61</td>
<td>-3 -1 28 9</td>
<td>0.0059</td>
</tr>
<tr>
<td>33.33</td>
<td>180.00</td>
<td>-3 0 28 0</td>
<td>-0.0065</td>
</tr>
<tr>
<td>69.61</td>
<td>118.61</td>
<td>-3 1 28 -9</td>
<td>0.0059</td>
</tr>
<tr>
<td>43.39</td>
<td>-140.19</td>
<td>-3 2 28 -19</td>
<td>-0.0035</td>
</tr>
<tr>
<td>47.14</td>
<td>135.00</td>
<td>-3 3 28 -28</td>
<td>-0.0019</td>
</tr>
<tr>
<td>64.79</td>
<td>-120.96</td>
<td>-3 4 28 -38</td>
<td>0.0039</td>
</tr>
<tr>
<td>33.79</td>
<td>170.54</td>
<td>-3 5 28 -47</td>
<td>-0.0025</td>
</tr>
<tr>
<td>28.33</td>
<td>-11.31</td>
<td>-2 -5 19 47</td>
<td>-0.0018</td>
</tr>
<tr>
<td>62.11</td>
<td>63.43</td>
<td>-2 -4 19 38</td>
<td>0.0013</td>
</tr>
<tr>
<td>43.39</td>
<td>-50.19</td>
<td>-2 -3 19 28</td>
<td>-0.0035</td>
</tr>
<tr>
<td>39.28</td>
<td>45.00</td>
<td>-2 -2 19 19</td>
<td>-0.0085</td>
</tr>
<tr>
<td>67.13</td>
<td>-65.56</td>
<td>-2 -1 19 9</td>
<td>-0.0100</td>
</tr>
<tr>
<td>27.78</td>
<td>0.00</td>
<td>-2 0 19 0</td>
<td>0.0161</td>
</tr>
<tr>
<td>67.13</td>
<td>65.56</td>
<td>-2 1 19 -9</td>
<td>-0.0100</td>
</tr>
<tr>
<td>39.28</td>
<td>-45.00</td>
<td>-2 2 19 -19</td>
<td>-0.0085</td>
</tr>
<tr>
<td>43.39</td>
<td>50.19</td>
<td>-2 3 19 -28</td>
<td>-0.0035</td>
</tr>
<tr>
<td>62.11</td>
<td>-63.43</td>
<td>-2 4 19 -38</td>
<td>0.0013</td>
</tr>
<tr>
<td>28.33</td>
<td>11.31</td>
<td>-2 5 19 -47</td>
<td>-0.0018</td>
</tr>
<tr>
<td>61.36</td>
<td>-174.81</td>
<td>-1 -5 9 47</td>
<td>0.0021</td>
</tr>
<tr>
<td>69.61</td>
<td>-151.39</td>
<td>-1 -3 9 28</td>
<td>0.0059</td>
</tr>
<tr>
<td>67.13</td>
<td>155.56</td>
<td>-1 -2 9 19</td>
<td>-0.0100</td>
</tr>
<tr>
<td>61.11</td>
<td>180.00</td>
<td>-1 0 9 0</td>
<td>0.0042</td>
</tr>
<tr>
<td>67.13</td>
<td>-195.56</td>
<td>-1 2 9 -19</td>
<td>-0.0100</td>
</tr>
<tr>
<td>69.61</td>
<td>151.39</td>
<td>-1 3 9 -28</td>
<td>0.0059</td>
</tr>
<tr>
<td>61.36</td>
<td>174.81</td>
<td>-1 5 9 -47</td>
<td>0.0021</td>
</tr>
<tr>
<td>5.56</td>
<td>-90.00</td>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>55.56</td>
<td>90.00</td>
<td>0</td>
<td>-4</td>
</tr>
<tr>
<td>33.33</td>
<td>-90.00</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>27.78</td>
<td>90.00</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>61.11</td>
<td>-90.00</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>61.11</td>
<td>90.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>27.78</td>
<td>-90.00</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>33.33</td>
<td>90.00</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>55.56</td>
<td>-90.00</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5.56</td>
<td>90.00</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>61.36</td>
<td>-5.19</td>
<td>1</td>
<td>-5</td>
</tr>
<tr>
<td>69.61</td>
<td>-28.61</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>67.13</td>
<td>24.44</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>61.11</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>67.13</td>
<td>-24.44</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>69.61</td>
<td>28.61</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>61.36</td>
<td>5.19</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>28.33</td>
<td>-168.69</td>
<td>2</td>
<td>-5</td>
</tr>
<tr>
<td>62.11</td>
<td>116.57</td>
<td>2</td>
<td>-4</td>
</tr>
<tr>
<td>43.39</td>
<td>-129.81</td>
<td>2</td>
<td>-3</td>
</tr>
<tr>
<td>39.28</td>
<td>135.00</td>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>67.13</td>
<td>-114.44</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>27.78</td>
<td>180.00</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>67.13</td>
<td>114.44</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>39.28</td>
<td>-135.00</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>43.39</td>
<td>129.81</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>62.11</td>
<td>-116.57</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>28.33</td>
<td>168.69</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>33.79</td>
<td>-9.46</td>
<td>3</td>
<td>-5</td>
</tr>
<tr>
<td>64.79</td>
<td>59.04</td>
<td>3</td>
<td>-4</td>
</tr>
<tr>
<td>47.14</td>
<td>-45.00</td>
<td>3</td>
<td>-3</td>
</tr>
<tr>
<td>43.39</td>
<td>39.81</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>69.61</td>
<td>-61.39</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>33.33</td>
<td>0.00</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>69.61</td>
<td>61.39</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>43.39</td>
<td>-39.81</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>47.14</td>
<td>45.00</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>64.79</td>
<td>-59.04</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>33.79</td>
<td>9.46</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>64.79</td>
<td>-149.04</td>
<td>4</td>
<td>-3</td>
</tr>
<tr>
<td>62.11</td>
<td>153.43</td>
<td>4</td>
<td>-2</td>
</tr>
<tr>
<td>55.56</td>
<td>180.00</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>62.11</td>
<td>-153.43</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>64.79</td>
<td>149.04</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>33.79</td>
<td>-80.54</td>
<td>5</td>
<td>-3</td>
</tr>
<tr>
<td>28.33</td>
<td>78.69</td>
<td>5</td>
<td>-2</td>
</tr>
<tr>
<td>61.36</td>
<td>-84.81</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>5.56</td>
<td>0.00</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>61.36</td>
<td>84.81</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>28.33</td>
<td>-78.69</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>33.79</td>
<td>80.54</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
MOIRE PROGRAM RESULTS SUMMARY

PROGRAM ARGUMENTS:

Results File Name = out.003  
Corresponding Picture Name = pic.003  
Normalized Amplitude Threshold = 0.0010  
Screen Dot Shape = Diamond

HALFTONE SPECIFICATIONS:

<table>
<thead>
<tr>
<th>Name</th>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>%Area Code</th>
<th>Shape Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writer</td>
<td>1411.1111</td>
<td>0.0000</td>
<td>1.000</td>
<td>2</td>
</tr>
<tr>
<td>Screen</td>
<td>150.0000</td>
<td>0.0000</td>
<td>0.100</td>
<td>3</td>
</tr>
<tr>
<td>Visibi</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.000</td>
<td>5</td>
</tr>
</tbody>
</table>

AMPLITUDE SPECTRUM:

<table>
<thead>
<tr>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>Vector Description</th>
<th>Normalized Amplitude</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Writer Screen</td>
<td>i j</td>
</tr>
</tbody>
</table>

<p>| 7.86  | 45.00       | -22 -22 207 207   | 0.0033               |
| 7.86  | -45.00      | -22 22 207 -207   | 0.0033               |
| 70.71 | -135.00     | -18 -18 169 169   | -0.0041              |
| 70.71 | 135.00      | -18 18 169 -169   | -0.0041              |
| 15.71 | 45.00       | -17 -17 160 160   | -0.0044              |
| 15.71 | -45.00      | -17 17 160 -160   | -0.0044              |
| 62.85 | -135.00     | -13 -13 122 122   | 0.0057               |
| 62.85 | 135.00      | -13 13 122 -122   | 0.0057               |
| 55.00 | -135.00     | -8 8 75 75        | -0.0094              |
| 55.00 | 135.00      | -8 -8 75 -75      | -0.0094              |
| 31.43 | 45.00       | -7 7 66 66        | -0.0108              |
| 31.43 | -45.00      | -7 7 66 -66       | -0.0108              |
| 64.79 | -30.96      | -4 -4 38 38       | 0.0014               |
| 55.56 | 0.00        | -4 0 38 0         | 0.0014               |
| 64.79 | 30.96       | -4 3 38 -28       | 0.0014               |
| 64.79 | 120.96      | -4 -4 28 28       | 0.0014               |
| 47.14 | -135.00     | -3 -3 28 28       | 0.0253               |
| 69.61 | -118.61     | -3 -1 28 9        | 0.0015               |
| 33.33 | 180.00      | -3 0 28 0         | 0.0014               |
| 69.61 | 118.61      | -3 1 28 -9        | 0.0015               |
| 47.14 | 135.00      | -3 3 28 -28       | 0.0253               |
| 64.79 | -120.96     | -3 4 28 -38       | 0.0014               |
| 39.28 | 45.00       | -2 -2 19 19       | 0.0374               |
| 67.13 | -65.56      | -2 -1 19 9        | 0.0036               |
| 27.78 | 0.00        | -2 0 19 0         | 0.0028               |
| 67.13 | 65.56       | -2 1 19 -9        | 0.0036               |
| 39.28 | -45.00      | -2 2 19 -19       | 0.0374               |
| 69.61 | -151.39     | -1 3 9 28         | 0.0015               |
| 67.13 | 155.56      | -1 2 9 19         | 0.0036               |
| 67.13 | -155.56     | -1 2 9 19         | 0.0036               |
| 69.61 | 151.39      | -1 3 9 -28        | 0.0015               |
| 55.56 | 90.00       | 0 -4 0 38         | 0.0014               |
| 33.33 | -90.00      | 0 3 28 0         | 0.0014               |
| 27.78 | 90.00       | 0 -2 0 19         | 0.0028               |
| 0.00  | 0.00        | 0 0 0 0         | 1.0000               |
| 27.78 | -90.00      | 0 2 0 -19         | 0.0028               |
| 33.33 | 90.00       | 0 3 0 -28        | 0.0014               |
| 55.56 | -90.00      | 0 4 0 -38        | 0.0014               |
| 69.61 | -28.61      | 1 -3 -9 28       | 0.0015               |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>67.13</td>
<td>24.44</td>
<td>1</td>
<td>-2</td>
<td>-9</td>
<td>19</td>
<td>0.0036</td>
</tr>
<tr>
<td>67.13</td>
<td>-24.44</td>
<td>1</td>
<td>2</td>
<td>-9</td>
<td>-19</td>
<td>0.0036</td>
</tr>
<tr>
<td>69.61</td>
<td>28.61</td>
<td>1</td>
<td>3</td>
<td>-9</td>
<td>-28</td>
<td>0.0015</td>
</tr>
<tr>
<td>39.28</td>
<td>135.00</td>
<td>2</td>
<td>-2</td>
<td>-19</td>
<td>19</td>
<td>0.0374</td>
</tr>
<tr>
<td>67.13</td>
<td>-114.44</td>
<td>2</td>
<td>-1</td>
<td>-19</td>
<td>9</td>
<td>0.0036</td>
</tr>
<tr>
<td>27.78</td>
<td>180.00</td>
<td>2</td>
<td>0</td>
<td>-19</td>
<td>9</td>
<td>0.0028</td>
</tr>
<tr>
<td>67.13</td>
<td>-114.44</td>
<td>2</td>
<td>1</td>
<td>-19</td>
<td>-9</td>
<td>0.0036</td>
</tr>
<tr>
<td>39.28</td>
<td>-135.00</td>
<td>2</td>
<td>2</td>
<td>-19</td>
<td>-19</td>
<td>0.0374</td>
</tr>
<tr>
<td>64.79</td>
<td>59.04</td>
<td>3</td>
<td>-4</td>
<td>-28</td>
<td>38</td>
<td>0.0014</td>
</tr>
<tr>
<td>47.14</td>
<td>-45.00</td>
<td>3</td>
<td>-3</td>
<td>-28</td>
<td>28</td>
<td>0.0253</td>
</tr>
<tr>
<td>69.61</td>
<td>-61.39</td>
<td>3</td>
<td>-1</td>
<td>-28</td>
<td>9</td>
<td>0.0015</td>
</tr>
<tr>
<td>33.33</td>
<td>0.00</td>
<td>3</td>
<td>0</td>
<td>-28</td>
<td>0</td>
<td>0.0014</td>
</tr>
<tr>
<td>69.61</td>
<td>61.39</td>
<td>3</td>
<td>1</td>
<td>-28</td>
<td>-9</td>
<td>0.0015</td>
</tr>
<tr>
<td>47.14</td>
<td>45.00</td>
<td>3</td>
<td>3</td>
<td>-28</td>
<td>-28</td>
<td>0.0253</td>
</tr>
<tr>
<td>64.79</td>
<td>-59.04</td>
<td>3</td>
<td>4</td>
<td>-28</td>
<td>-38</td>
<td>0.0014</td>
</tr>
<tr>
<td>64.79</td>
<td>-149.04</td>
<td>4</td>
<td>-3</td>
<td>-38</td>
<td>28</td>
<td>0.0014</td>
</tr>
<tr>
<td>55.56</td>
<td>180.00</td>
<td>4</td>
<td>0</td>
<td>-38</td>
<td>0</td>
<td>0.0014</td>
</tr>
<tr>
<td>64.79</td>
<td>149.04</td>
<td>4</td>
<td>3</td>
<td>-38</td>
<td>-28</td>
<td>0.0014</td>
</tr>
<tr>
<td>31.43</td>
<td>135.00</td>
<td>7</td>
<td>-7</td>
<td>-66</td>
<td>66</td>
<td>-0.0108</td>
</tr>
<tr>
<td>31.43</td>
<td>-135.00</td>
<td>7</td>
<td>7</td>
<td>-66</td>
<td>-66</td>
<td>-0.0108</td>
</tr>
<tr>
<td>55.00</td>
<td>-45.00</td>
<td>8</td>
<td>-8</td>
<td>-75</td>
<td>75</td>
<td>-0.0094</td>
</tr>
<tr>
<td>55.00</td>
<td>45.00</td>
<td>8</td>
<td>8</td>
<td>-75</td>
<td>-75</td>
<td>0.0094</td>
</tr>
<tr>
<td>23.57</td>
<td>135.00</td>
<td>12</td>
<td>-12</td>
<td>-113</td>
<td>113</td>
<td>0.0063</td>
</tr>
<tr>
<td>23.57</td>
<td>-135.00</td>
<td>12</td>
<td>12</td>
<td>-113</td>
<td>-113</td>
<td>0.0063</td>
</tr>
<tr>
<td>62.85</td>
<td>-45.00</td>
<td>13</td>
<td>-13</td>
<td>-122</td>
<td>122</td>
<td>0.0057</td>
</tr>
<tr>
<td>62.85</td>
<td>45.00</td>
<td>13</td>
<td>13</td>
<td>-122</td>
<td>-122</td>
<td>0.0057</td>
</tr>
<tr>
<td>15.71</td>
<td>135.00</td>
<td>17</td>
<td>17</td>
<td>-160</td>
<td>-160</td>
<td>-0.0044</td>
</tr>
<tr>
<td>15.71</td>
<td>-135.00</td>
<td>17</td>
<td>17</td>
<td>-160</td>
<td>160</td>
<td>-0.0044</td>
</tr>
<tr>
<td>70.71</td>
<td>-45.00</td>
<td>18</td>
<td>-18</td>
<td>-169</td>
<td>169</td>
<td>-0.0041</td>
</tr>
<tr>
<td>70.71</td>
<td>45.00</td>
<td>18</td>
<td>18</td>
<td>-169</td>
<td>-169</td>
<td>-0.0041</td>
</tr>
<tr>
<td>7.86</td>
<td>135.00</td>
<td>22</td>
<td>-22</td>
<td>-207</td>
<td>207</td>
<td>0.0033</td>
</tr>
<tr>
<td>7.86</td>
<td>-135.00</td>
<td>22</td>
<td>22</td>
<td>-207</td>
<td>-207</td>
<td>0.0033</td>
</tr>
</tbody>
</table>
**PROGRAM ARGUMENTS:**

- Results File Name: `out.005`
- Corresponding Picture Name: `pic.005`
- Normalized Amplitude Threshold: 0.0010
- Screen Dot Shape: Round

**HALFTONE SPECIFICATIONS:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>%Area</th>
<th>Shape Code</th>
<th>Frequency Limit (c/in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writer</td>
<td>1411.1111</td>
<td>0.0000</td>
<td>1.000</td>
<td>2</td>
<td>50000.000</td>
</tr>
<tr>
<td>Screen</td>
<td>149.0000</td>
<td>0.0000</td>
<td>0.100</td>
<td>1</td>
<td>75.000</td>
</tr>
<tr>
<td>Visibi</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.000</td>
<td>5</td>
<td>75.000</td>
</tr>
</tbody>
</table>

**AMPLITUDE SPECTRUM:**

<table>
<thead>
<tr>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>Vector Description Writer</th>
<th>Screen</th>
<th>Normalized Amplitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>53.28</td>
<td>170.52</td>
<td>-5</td>
<td>-2</td>
<td>47</td>
</tr>
<tr>
<td>52.56</td>
<td>180.00</td>
<td>-5</td>
<td>0</td>
<td>47</td>
</tr>
<tr>
<td>53.28</td>
<td>-170.52</td>
<td>-5</td>
<td>2</td>
<td>47</td>
</tr>
<tr>
<td>63.80</td>
<td>-74.03</td>
<td>-4</td>
<td>-3</td>
<td>38</td>
</tr>
<tr>
<td>19.63</td>
<td>26.57</td>
<td>-4</td>
<td>-2</td>
<td>38</td>
</tr>
<tr>
<td>72.28</td>
<td>-75.94</td>
<td>-4</td>
<td>-1</td>
<td>38</td>
</tr>
<tr>
<td>17.56</td>
<td>0.00</td>
<td>-4</td>
<td>0</td>
<td>38</td>
</tr>
<tr>
<td>72.28</td>
<td>75.94</td>
<td>-4</td>
<td>1</td>
<td>38</td>
</tr>
<tr>
<td>19.63</td>
<td>-26.57</td>
<td>-4</td>
<td>2</td>
<td>38</td>
</tr>
<tr>
<td>63.80</td>
<td>74.03</td>
<td>-4</td>
<td>3</td>
<td>38</td>
</tr>
<tr>
<td>63.80</td>
<td>164.03</td>
<td>-3</td>
<td>-4</td>
<td>28</td>
</tr>
<tr>
<td>61.96</td>
<td>171.86</td>
<td>-3</td>
<td>-2</td>
<td>28</td>
</tr>
<tr>
<td>61.96</td>
<td>180.00</td>
<td>-3</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>61.96</td>
<td>-171.86</td>
<td>-3</td>
<td>2</td>
<td>28</td>
</tr>
<tr>
<td>63.80</td>
<td>-164.03</td>
<td>-3</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>53.28</td>
<td>-80.52</td>
<td>-2</td>
<td>-5</td>
<td>19</td>
</tr>
<tr>
<td>19.63</td>
<td>63.43</td>
<td>-2</td>
<td>-4</td>
<td>19</td>
</tr>
<tr>
<td>61.96</td>
<td>-81.86</td>
<td>-2</td>
<td>-3</td>
<td>19</td>
</tr>
<tr>
<td>12.41</td>
<td>45.00</td>
<td>-2</td>
<td>-2</td>
<td>19</td>
</tr>
<tr>
<td>70.66</td>
<td>-82.86</td>
<td>-2</td>
<td>-1</td>
<td>19</td>
</tr>
<tr>
<td>70.66</td>
<td>82.86</td>
<td>-2</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>12.41</td>
<td>-45.00</td>
<td>-2</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>61.96</td>
<td>81.86</td>
<td>-2</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>19.63</td>
<td>-63.43</td>
<td>-2</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>53.28</td>
<td>80.52</td>
<td>-2</td>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>72.28</td>
<td>165.94</td>
<td>-1</td>
<td>-4</td>
<td>9</td>
</tr>
<tr>
<td>70.66</td>
<td>172.86</td>
<td>-1</td>
<td>-2</td>
<td>9</td>
</tr>
<tr>
<td>70.66</td>
<td>180.00</td>
<td>-1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>70.66</td>
<td>-172.86</td>
<td>-1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>72.28</td>
<td>-165.94</td>
<td>-1</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>52.56</td>
<td>-90.00</td>
<td>0</td>
<td>-5</td>
<td>0</td>
</tr>
<tr>
<td>17.56</td>
<td>90.00</td>
<td>0</td>
<td>-4</td>
<td>0</td>
</tr>
<tr>
<td>61.33</td>
<td>-90.00</td>
<td>0</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>8.78</td>
<td>90.00</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>70.11</td>
<td>-90.00</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0.00</td>
<td>90.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70.11</td>
<td>90.00</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8.78</td>
<td>-90.00</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>61.33</td>
<td>90.00</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>17.56</td>
<td>-90.00</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>52.56</td>
<td>90.00</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>72.28</td>
<td>14.06</td>
<td>1</td>
<td>-4</td>
<td>-9</td>
</tr>
<tr>
<td>70.66</td>
<td>7.14</td>
<td>1</td>
<td>-2</td>
<td>-9</td>
</tr>
<tr>
<td>70.11</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>-9</td>
</tr>
<tr>
<td>70.66</td>
<td>-7.14</td>
<td>1</td>
<td>2</td>
<td>-9</td>
</tr>
<tr>
<td>72.28</td>
<td>-14.06</td>
<td>1</td>
<td>4</td>
<td>-9</td>
</tr>
<tr>
<td>53.28</td>
<td>-99.48</td>
<td>2</td>
<td>-5</td>
<td>-19</td>
</tr>
<tr>
<td>19.63</td>
<td>116.57</td>
<td>2</td>
<td>-4</td>
<td>-19</td>
</tr>
<tr>
<td>61.96</td>
<td>-98.14</td>
<td>2</td>
<td>-3</td>
<td>-19</td>
</tr>
<tr>
<td>12.41</td>
<td>135.00</td>
<td>2</td>
<td>-2</td>
<td>-19</td>
</tr>
<tr>
<td>70.66</td>
<td>-97.14</td>
<td>2</td>
<td>-1</td>
<td>-19</td>
</tr>
<tr>
<td>8.78</td>
<td>180.00</td>
<td>2</td>
<td>0</td>
<td>-19</td>
</tr>
<tr>
<td>70.66</td>
<td>97.14</td>
<td>2</td>
<td>1</td>
<td>-19</td>
</tr>
<tr>
<td>12.41</td>
<td>-135.00</td>
<td>2</td>
<td>2</td>
<td>-19</td>
</tr>
<tr>
<td>61.96</td>
<td>98.14</td>
<td>2</td>
<td>3</td>
<td>-19</td>
</tr>
<tr>
<td>19.63</td>
<td>-116.57</td>
<td>2</td>
<td>4</td>
<td>-19</td>
</tr>
<tr>
<td>53.28</td>
<td>99.48</td>
<td>2</td>
<td>5</td>
<td>-19</td>
</tr>
<tr>
<td>63.80</td>
<td>15.97</td>
<td>3</td>
<td>-4</td>
<td>-28</td>
</tr>
<tr>
<td>61.96</td>
<td>8.14</td>
<td>3</td>
<td>-2</td>
<td>-28</td>
</tr>
<tr>
<td>61.33</td>
<td>0.00</td>
<td>3</td>
<td>0</td>
<td>-28</td>
</tr>
<tr>
<td>61.96</td>
<td>-8.14</td>
<td>3</td>
<td>2</td>
<td>-28</td>
</tr>
<tr>
<td>63.80</td>
<td>-15.97</td>
<td>3</td>
<td>4</td>
<td>-28</td>
</tr>
<tr>
<td>63.80</td>
<td>-105.97</td>
<td>4</td>
<td>-3</td>
<td>-38</td>
</tr>
<tr>
<td>19.63</td>
<td>153.43</td>
<td>4</td>
<td>-2</td>
<td>-38</td>
</tr>
<tr>
<td>72.28</td>
<td>-104.06</td>
<td>4</td>
<td>-1</td>
<td>-38</td>
</tr>
<tr>
<td>17.56</td>
<td>180.00</td>
<td>4</td>
<td>0</td>
<td>-38</td>
</tr>
<tr>
<td>72.28</td>
<td>104.06</td>
<td>4</td>
<td>1</td>
<td>-38</td>
</tr>
<tr>
<td>19.63</td>
<td>-153.43</td>
<td>4</td>
<td>2</td>
<td>-38</td>
</tr>
<tr>
<td>63.80</td>
<td>105.97</td>
<td>4</td>
<td>3</td>
<td>-38</td>
</tr>
<tr>
<td>53.28</td>
<td>9.48</td>
<td>5</td>
<td>-2</td>
<td>-47</td>
</tr>
<tr>
<td>52.56</td>
<td>0.00</td>
<td>5</td>
<td>0</td>
<td>-47</td>
</tr>
<tr>
<td>53.28</td>
<td>-9.48</td>
<td>5</td>
<td>2</td>
<td>-47</td>
</tr>
</tbody>
</table>
**MOIRE PROGRAM RESULTS SUMMARY**

**PROGRAM ARGUMENTS:**

Results File Name = out.009  
Corresponding Picture Name = pic.009  
Normalized Amplitude Threshold = 0.0010  
Screen Dot Shape = Round

**HALFTONE SPECIFICATIONS:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>%Area</th>
<th>Shape Code</th>
<th>Frequency Limit (c/in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writer</td>
<td>1411.1111</td>
<td>0.0000</td>
<td>1.000</td>
<td>2</td>
<td>50000.000</td>
</tr>
<tr>
<td>Screen</td>
<td>150.0000</td>
<td>75.0000</td>
<td>0.100</td>
<td>3</td>
<td>75.000</td>
</tr>
<tr>
<td>Visibi</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.000</td>
<td>5</td>
<td>75.000</td>
</tr>
</tbody>
</table>

**AMPLITUDE SPECTRUM:**

<table>
<thead>
<tr>
<th>Frequency (c/in)</th>
<th>Angle (deg)</th>
<th>Vector Description</th>
<th>Normalized Amplitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.04</td>
<td>-121.63</td>
<td>-5 -3 39 -38</td>
<td>-0.0018</td>
</tr>
<tr>
<td>39.14</td>
<td>-104.88</td>
<td>-5 -1 21 -43</td>
<td>0.0021</td>
</tr>
<tr>
<td>70.18</td>
<td>-173.16</td>
<td>-5 0 12 -45</td>
<td>0.0039</td>
</tr>
<tr>
<td>23.59</td>
<td>-48.86</td>
<td>-5 1 3 -48</td>
<td>0.0011</td>
</tr>
<tr>
<td>45.59</td>
<td>165.07</td>
<td>-5 2 -6 -50</td>
<td>-0.0029</td>
</tr>
<tr>
<td>41.28</td>
<td>3.32</td>
<td>-5 3 -15 -53</td>
<td>-0.0031</td>
</tr>
<tr>
<td>60.20</td>
<td>-27.48</td>
<td>-4 -4 46 -27</td>
<td>0.0022</td>
</tr>
<tr>
<td>6.45</td>
<td>164.78</td>
<td>-4 -3 37 -29</td>
<td>0.0042</td>
</tr>
<tr>
<td>72.85</td>
<td>154.68</td>
<td>-4 -2 25 -31</td>
<td>0.0044</td>
</tr>
<tr>
<td>29.19</td>
<td>46.32</td>
<td>-4 -1 19 -34</td>
<td>-0.0049</td>
</tr>
<tr>
<td>65.16</td>
<td>128.11</td>
<td>-4 0 10 -36</td>
<td>-0.0015</td>
</tr>
<tr>
<td>61.53</td>
<td>42.93</td>
<td>-4 1 1 -39</td>
<td>-0.0047</td>
</tr>
<tr>
<td>72.85</td>
<td>101.55</td>
<td>-4 2 -8 -41</td>
<td>0.0044</td>
</tr>
<tr>
<td>60.20</td>
<td>-117.48</td>
<td>-4 4 -27 -46</td>
<td>0.0022</td>
</tr>
<tr>
<td>41.28</td>
<td>93.32</td>
<td>-3 5 -53 -15</td>
<td>-0.0031</td>
</tr>
<tr>
<td>65.58</td>
<td>69.24</td>
<td>-3 -3 35 -20</td>
<td>0.0022</td>
</tr>
<tr>
<td>64.25</td>
<td>-81.00</td>
<td>-3 -1 16 -25</td>
<td>0.0073</td>
</tr>
<tr>
<td>60.11</td>
<td>-145.56</td>
<td>-3 0 7 -27</td>
<td>-0.0073</td>
</tr>
<tr>
<td>56.15</td>
<td>-50.54</td>
<td>-3 1 -2 -30</td>
<td>0.0081</td>
</tr>
<tr>
<td>27.68</td>
<td>-149.88</td>
<td>-3 2 -11 -32</td>
<td>-0.0035</td>
</tr>
<tr>
<td>65.58</td>
<td>-20.76</td>
<td>-3 3 -20 -35</td>
<td>0.0022</td>
</tr>
<tr>
<td>6.45</td>
<td>74.79</td>
<td>-3 4 -29 -37</td>
<td>0.0042</td>
</tr>
<tr>
<td>68.04</td>
<td>148.37</td>
<td>-3 5 -38 -39</td>
<td>-0.0018</td>
</tr>
<tr>
<td>45.59</td>
<td>-104.93</td>
<td>-2 -5 50 -6</td>
<td>-0.0029</td>
</tr>
<tr>
<td>72.85</td>
<td>-168.45</td>
<td>-2 -4 41 -8</td>
<td>0.0044</td>
</tr>
<tr>
<td>27.68</td>
<td>-59.88</td>
<td>-2 -3 32 -11</td>
<td>-0.0035</td>
</tr>
<tr>
<td>46.07</td>
<td>173.11</td>
<td>-2 -2 23 -13</td>
<td>-0.0053</td>
</tr>
<tr>
<td>39.71</td>
<td>-5.54</td>
<td>-2 -1 14 -16</td>
<td>-0.0120</td>
</tr>
<tr>
<td>32.58</td>
<td>128.11</td>
<td>-2 0 5 -18</td>
<td>0.0161</td>
</tr>
<tr>
<td>67.15</td>
<td>14.03</td>
<td>-2 1 -4 -21</td>
<td>-0.0127</td>
</tr>
<tr>
<td>46.07</td>
<td>83.11</td>
<td>-2 2 -13 23</td>
<td>-0.0053</td>
</tr>
<tr>
<td>72.85</td>
<td>64.68</td>
<td>-2 4 -31 -28</td>
<td>0.0044</td>
</tr>
<tr>
<td>23.59</td>
<td>41.32</td>
<td>-1 -5 48 -3</td>
<td>0.0011</td>
</tr>
<tr>
<td>61.53</td>
<td>132.93</td>
<td>-1 -4 39 -1</td>
<td>-0.0047</td>
</tr>
<tr>
<td>56.15</td>
<td>39.46</td>
<td>-1 -3 30 -2</td>
<td>0.0081</td>
</tr>
<tr>
<td>67.15</td>
<td>104.03</td>
<td>-1 -2 21 -4</td>
<td>-0.0127</td>
</tr>
<tr>
<td>66.51</td>
<td>-116.30</td>
<td>-1 0 2 -9</td>
<td>-0.0077</td>
</tr>
<tr>
<td>39.71</td>
<td>-95.54</td>
<td>-1 2 -16 -14</td>
<td>-0.0120</td>
</tr>
<tr>
<td>64.25</td>
<td>-171.00</td>
<td>-1 3 -25 -16</td>
<td>0.0073</td>
</tr>
<tr>
<td>29.19</td>
<td>-41.68</td>
<td>-1 4 -34 -19</td>
<td>-0.0049</td>
</tr>
<tr>
<td>39.14</td>
<td>165.12</td>
<td>-1 5 -43 -21</td>
<td>0.0021</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>70.18</td>
<td>-83.16</td>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>65.16</td>
<td>-141.89</td>
<td>0</td>
<td>-4</td>
</tr>
<tr>
<td>60.11</td>
<td>-55.56</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>32.58</td>
<td>-141.89</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>66.51</td>
<td>-26.30</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>66.51</td>
<td>153.70</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>32.58</td>
<td>38.11</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>60.11</td>
<td>124.44</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>65.16</td>
<td>38.11</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>70.18</td>
<td>96.84</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>39.14</td>
<td>-14.88</td>
<td>1</td>
<td>-5</td>
</tr>
<tr>
<td>29.19</td>
<td>138.32</td>
<td>1</td>
<td>-4</td>
</tr>
<tr>
<td>64.25</td>
<td>9.00</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>39.71</td>
<td>84.46</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>66.51</td>
<td>63.70</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>67.15</td>
<td>-75.97</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>56.15</td>
<td>-140.54</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>61.53</td>
<td>-47.07</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>23.59</td>
<td>-138.68</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>72.85</td>
<td>-115.32</td>
<td>2</td>
<td>-4</td>
</tr>
<tr>
<td>46.07</td>
<td>-96.89</td>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>67.15</td>
<td>-165.97</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>32.58</td>
<td>-51.89</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>27.68</td>
<td>120.12</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>72.85</td>
<td>-6.89</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>27.68</td>
<td>11.55</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>45.59</td>
<td>75.07</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>68.04</td>
<td>-31.63</td>
<td>3</td>
<td>-5</td>
</tr>
<tr>
<td>6.45</td>
<td>-105.22</td>
<td>3</td>
<td>-4</td>
</tr>
<tr>
<td>65.58</td>
<td>159.24</td>
<td>3</td>
<td>-3</td>
</tr>
<tr>
<td>27.68</td>
<td>30.12</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>56.15</td>
<td>129.46</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>60.11</td>
<td>34.44</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>64.25</td>
<td>99.00</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>65.58</td>
<td>-110.76</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>41.28</td>
<td>-86.68</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>60.20</td>
<td>62.52</td>
<td>4</td>
<td>-4</td>
</tr>
<tr>
<td>72.85</td>
<td>-78.45</td>
<td>4</td>
<td>-2</td>
</tr>
<tr>
<td>61.53</td>
<td>-137.07</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>65.16</td>
<td>-51.89</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>29.19</td>
<td>-131.68</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>72.85</td>
<td>-25.32</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6.45</td>
<td>-15.22</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>60.20</td>
<td>152.52</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>41.28</td>
<td>-176.68</td>
<td>5</td>
<td>-3</td>
</tr>
<tr>
<td>45.59</td>
<td>-14.93</td>
<td>5</td>
<td>-2</td>
</tr>
<tr>
<td>23.59</td>
<td>131.32</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>70.18</td>
<td>6.84</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>39.14</td>
<td>75.12</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>68.04</td>
<td>58.37</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>