INVESTIGATION INTO THE GERCHBERG-SAXTON AND ERROR-REDUCTION ALGORITHMS

Submitted by

JAMES JOSEPH HOSKER

In partial fulfillment of the requirements

for the degree of

Master of Science

in

Electrical Engineering

TUFTS UNIVERSITY

February 1990

ABSTRACT

This document provides a detailed analysis of the creation and implementation of the Gerchberg-Saxton, error-reduction and hybrid input-output algorithms. The Gerchberg-Saxton algorithm is an image enhancement algorithm which is used to retrieve distorted images. In particular, this document investigates the phase retrieval of an image in which only information on the magnitude of the image in the Fourier domain is known. The document is broken into four sections and one appendix. The first section provides background and historical information on the algorithms. The second section provides an introduction to and explanation of the Gerchberg-Saxton, error-reduction and input-output algorithms. The third section describes the software implementation of the algorithms, including the problems discovered during the programming of the algorithms. The fourth section provides conclusions and recommendations in the use of these algorithms. Finally, an appendix is added to show images which were retrieved using these algorithms.

ACKNOWLEDGEMENTS

This document was written to fulfill the requirements for a Masters Degree in Electrical Engineering at Tufts University. This document was made possible with the assistance of two people: first, my advisor and the director of the Tufts University Electro-Optics Technology Center (EOTC), Professor Robert A. Gonsalves; and second, the lead scientist of Department D-53 in the MITRE Corporation, Dr. Marvin D. Drake. Finally, this document was also made possible through the MITRE Graduate Degree Program in Science and Engineering.

TABLE OF CONTENTS

SECTION		PAGES
1	Background	1 - 3
2	Introduction	4 - 15
2.1	Gerchberg-Saxton Algorithm	4 - 6
2.2	Error-Reduction Algorithm	6 - 11
2.3	Input-Output Algorithm	12 - 15
3	Software Implementation of the Algorithm	16 - 44
3.1	Problems in the Applications of the Algorithms	16 - 22
3.2	Outline of the Final Algorithm	22 - 27
3.3	Details on the Software Implementation of the Algorithm	28 - 44
3.3.1	Memory	29 - 33
3.3.2	Fast Fourier Transform (FFT)	33 - 35
3.3.3	Operations	35 - 36
3.3.4	Mean Squared Error (MSE) Analysis	36 - 43
3.3.5	Plot	43 - 44
4	Recommendations and Conclusions	45 - 47
Appendix		48 - 58
List of References		59 - 61

LIST OF FIGURES

FIGURE PAGE

1 The error-reduction or generalized Gerchberg-Saxton algorithm 7

2 The input-output algorithm 12

3 RMS error of E_{Ok} vs. β for the hybrid input-output algorithm (Log Graph) ... 15

4 The final algorithm 24

5 The object plane of an input matrix 29

6 The memory requirements of a 16 by 16 input matrix 32

7 The memory requirements of a 64 by 64 input matrix 32

8 The memory requirements of a 128 by 128 input matrix 33

9 The FFT of a two-dimensional matrix 34

10 Transposition of a two-dimensional matrix 34

11 Error vs. number of iterations (Log Graph) 39

12 RMS error of E_{Ok} vs. the number of iterations (Log Graph) 42

SECTION 1

BACKGROUND

R. W. Gerchberg and W. O. Saxton first presented their findings in 1972 in the journal *Optik* which was published in West Germany. They developed single and dual intensity algorithms for the rapid solution of the phase of a complete wave function whose intensity in the diffraction and imaging planes of an imaging system were known. They further presented evidence that the error between successive iterations of the algorithm must decrease and that a unique solution can always be found.[1-2] Robert A. Gonsalves separately discovered a similar algorithm in the United States in 1976. He describes the phase retrieval as an extraction of the phase from the modulus of a complex signal. He gives examples where the modulus of the Fourier transform of a signal is used to retrieve a corrupted or noisy version of the signal through a computer simulation.[3] The mathematical and theoretical premise for the work of Gerchberg, Saxton and Gonsalves in using the modulus of the Fourier transform to retrieve the phase has existed since 1963.[4-6]

It was not until James R. Fienup investigated the topic of phase retrieval using modulus information of the signal that research and information on this phase retrieval technique again progressed. In many articles from 1978 to the present, Fienup presented many different existing techniques for phase retrieval including the Gerchberg-Saxton algorithm, the steepest-descent method, and gradient search methods. In his experimental comparison of phase-retrieval algorithms, he found a combination of two algorithms that produced optimal results. In particular, Fienup presented a generalized version of the Gerchberg-Saxton algorithm which he called the error-reduction algorithm and presented hybrid algorithm of the error-reduction algorithm which he called the input-output algorithm.[6-13] The error-reduction and input-output algorithms are the focus of this document because the combination of these two techniques provide the optimal unique solution in the fewest number of iterations.

There are many application of the error-reduction and input-output algorithms. The number of problems that can be solved by this iterative algorithm (the error-reduction algorithm) is only limited by the ingenuity in defining different combinations of information that might be available in both domains. The best way to classify each of the problems is

by the type of information available. The following examples of the application of the algorithms are outlined in detail by Feinup in reference 14.

The first problem this algorithm can solve are those problems in which the modulus or magnitude of a complex function and the modulus of its Fourier transform are known and one wants to know the phase of the Fourier transform pair. This application is useful for phase retrieval in electron microscopy; for phase retrieval in wavefront sensing; for design optimization of radar signals and antenna arrays having desirable properties; and for phase coding and spectrum shaping problems found in computer generated holograms as well as other applications. Electron microscopy was one of the earliest applications of the error-reduction algorithm and perhaps has been the most investigated application. Wavefront sensing is similar to electon microscopy, but spectrum shaping is a complex synthesis problem.

The second problem is one where the function is known to be real and nonnegative and the modulus of its Fourier transform is known or measured. This application is useful for the phase problems associated with x-ray crystallography, Fourier transform spectroscopy, pupil function retrieval, and imaging through atmospheric turbulence using interferometer data. This problem of phase retrieval arises in spectroscopy as a one-dimensional problem, in astronomy as a two-dimensional problem, and in x-ray crystallography as a three-dimensional problem. For the one-dimensional problem, the use of the iterative algorithm is of limited interest since the solution is not generally unique. For the problem of reconstructing a two-dimensional nonnegative function from the modulus of it Fourier transform, the use of the iterative algorithm is of great importance in astronomical reconstruction and pupil synthesis. This application of the algorithm is the focus of this document.

The third problem is one where a low-pass filtered function is measured and the function is known to have a finite extent. By saying the low-pass filtered function is measured, it means that the complex Fourier transform of the function is measured only over a certain interval and by saying it has a finite extent, it means that the function is zero outside of some known region. This application is useful for the spectral extrapolation, for the superresolution problem of band-limited time signals, or for the imaging of objects of finite extent.

The fourth problem is one where the function is known to be nonnegative and of finite extent. Its complex Fourier transform is measured only over a partially filled aperture. This application is useful for the interpolation of the complex visibility function

over long baseline radio interferometry and for the missing-cone problem in x-ray tomography.

The fifth problem is one where the modulus of a complex function is given and one wants to find an associated phase function that results in a Fourier transform whose complex values are in part of a prescribed set of quantized complex values. This application is useful for the reduction of quantized noise in computer-generated holograms and in coded signal transmission.

The sixth problem is one where the modulus of complex function is reconstructed from the phase of the function given that the Fourier transform of the function has finite support. Fienup uses the relaxation-parameter algorithm to solve this application.[14] Essentially, he adds an extra step to the error-reduction algorithm. This document will not deal with relaxation-parameter algorithm.

The application that has been most intriguing is the application described in the second problem. For this document, the Gerchberg-Saxton algorithm is used to retrieve a distorted image using the modulus of the Fourier transform of the original undistorted image, which is real and nonnegative. The algorithms are able to retrieve the original image from a distorted image using an exact estimate of the Fourier modulus of the orginal undistorted image. This document will outline an algorithm capable of retrieving this image and will provide an in-depth analysis on how the error is reduced after each iteration.

SECTION 2

INTRODUCTION

This section is broken into three subsections. The first subsection describes the Gerchberg-Saxton algorithm. The second subsection describes the error-reduction algorithm. Finally, the third subsection describes the hybrid input-output algorithm.

2.1 GERCHBERG-SAXTON ALGORITHM

In the many applications described in the section 1 of this document, one usually wants to retrieve an object f(x) which is related to its Fourier transform F(u). The functions f(x) and F(u) are a unique Fourier transform pair.

$$\begin{aligned} F(u) &= |F(u)| \exp[i\psi(u)] = \mathcal{FT}[f(x)] \\ &= \int_{-\infty}^{\infty} f(x) \exp(-i2\pi u \cdot x/N)\, dx \end{aligned} \tag{1}$$

$$\begin{aligned} f(x) &= |f(x)| \exp[i\alpha(x)] = \mathcal{FT}^{-1}[F(u)] \\ &= \int_{-\infty}^{\infty} F(u) \exp(i2\pi u \cdot x/N)\, du \end{aligned} \tag{2}$$

where $\mathcal{FT}$ is an abbreviation for the Fourier transform and $\mathcal{FT}^{-1}$, for the inverse Fourier transform. In this case, x is a multidimensional spacial coordinate and u is a multidimensional spacial frequency coordinate. For the purpose of this document, the analysis is limited to two-dimensions. If the data is stored in a computer, then the algorithm will be used with the discrete Fourier transform, where for the case of two-dimensions u has coordinates u_x and u_y and x has coordinates x_x and x_y where u_x, u_y, x_x , and x_y = 0,1,2,3, ... (N-1).

$$F(u) = \sum_{x=0}^{N-1} f(x) \exp(-i2\pi u \cdot x/N) \tag{3}$$

$$f(x) = N^{-2} \sum_{u=0}^{N-1} F(u) \exp(i2\pi u \cdot x/N) \quad (4)$$

Obviously, the computer simulation will use the Fast Fourier Transform (FFT) to perform the analysis.[19]

For the original analysis, Gerchberg and Saxton were interested in retrieving the phase from a dual intensity measurement as in electron microscopy.

$$f(x) = |f(x)| \exp[i\alpha(x)] \quad (5)$$

$$F(u) = |F(u)| \exp[i\psi(u)] \quad (6)$$

They wanted to recover $\alpha(x)$ from $|f(x)|$ and $\psi(u)$ from $|F(u)|$. For a single intensity measurement, they were interested in recovering $\psi(u)$ or $f(x)$ given $|F(u)|$ and constraints that $f(x)$ is real and nonnegative.

$$f(x) \geq 0 \quad (7)$$

The Gerchberg-Saxton algorithm is successful in solving these two problems. The error-reduction algorithm and input-output algorithms expanded on the types of problems that can be solved as described in section 1 of this document. These algorithms involve an iterative process of transforming back and forth between the object-domain and the Fourier-domain after the application of measured or known constaints in each of the two domains. The term object-domain can also be referred to as the function-domain, the spacial-domain, or the X-domain. The term Fourier-domain can also be referred to as the spacial frequency-domain or the U-domain. In this document, I will try to remain consistent and refer to the object-domain and Fourier-domain as the X-domain and the U-domain repectively.

The Gerchberg-Saxton algorithm was initially created to recontruct phase from dual intensity measurements. The algorithm has four basic steps:

1. Fourier transform an estimate of the original object.

2. Replace the modulus of the computed Fourier transform with the measured Fourier modulus to form a estimate of the Fourier transform.

3. Inverse Fourier transform the estimate of the Fourier transform.

4. Replace the modulus of the resulting computed image with the measured object modulus to form a new estimate of the image.

In the form of equations, the Gerchberg-Saxton algorithm can be reduced to the following:

$$G_k(u) = |G_k(u)| \exp[i\varphi_k(u)] = \mathcal{FT}[g_k(u)], \quad (8)$$

$$G'_k(u) = |F(u)| \exp[i\varphi_k(u)], \quad (9)$$

$$g'_k(x) = |g'_k(x)| \exp[i\theta'_k(x)] = \mathcal{FT}^{-1}[G'_k(u)], \quad (10)$$

$$g_{k+1}(x) = |f(x)| \exp[i\theta_{k+1}(x)] = |f(x)| \exp[i\theta'_k(x)], \quad (11)$$

where g_k, θ_k, G'_k, and φ_k are all estimates of f, α, F and ψ respectively and where k represents the k^{th} iteration.

2.2 ERROR-REDUCTION ALGORITHM

As already stated in the section 1 of this document, the Gerchberg-Saxton algorithm can be used for any problem where partial constraints are known in each of the two domains, usually the X-domain and the U-domain. Transforming back and forth between the two domains while satisfying the contraints in each domain before returning to the other domain is a generalized version of the Gerchberg-Saxton algorithm. The generalized Gerchberg-Saxton algorithm can be used for any problem in which contraints are known in each of the two domains - the X-domain and U-domain. The contraints are usually from measured data or information which is obtained *a priori*.[7] Fienup refers to this generalized algorithm as the error-reduction algorithm.

The error-reduction algorithm is reduced to four basic steps:

1. Fourier transform $g_k(x)$, which is an estimate of f(x).

2. Make changes in $G_k(u)$, the computed Fourier transform; then satisfy the Fourier-domain or U-domain contraints to form $G'_k(u)$, which is an estimate of F(u).

3. Inverse Fourier transform $G'_k(u)$.

4. Make changes in $g'_k(x)$, the computed image; then satisfy the object-domain or X-domain constraints to form $g_{k+1}(x)$, which is a new estimate of the image.

The first three steps for a single intensity measurement are the same as for the Gerchberg-Saxton algorithm, but the fourth step is given by the following:

$$g_{k+1}(x) = \begin{cases} g'_k(x), & \text{for } x \notin \gamma \\ 0, & \text{for } x \in \gamma \end{cases} \qquad (12)$$

In this case, γ represents the set of points at which $g'_k(x)$ violates the X-domain constraints. For the purpose of this document, the violation occurs when $g'_k(x)$ is negative. However, other violations can occur when $g'_k(x)$ exceeds the known diameter of the object. Figure 1 depicts the error-reduction or generalized Gerchberg-Saxton

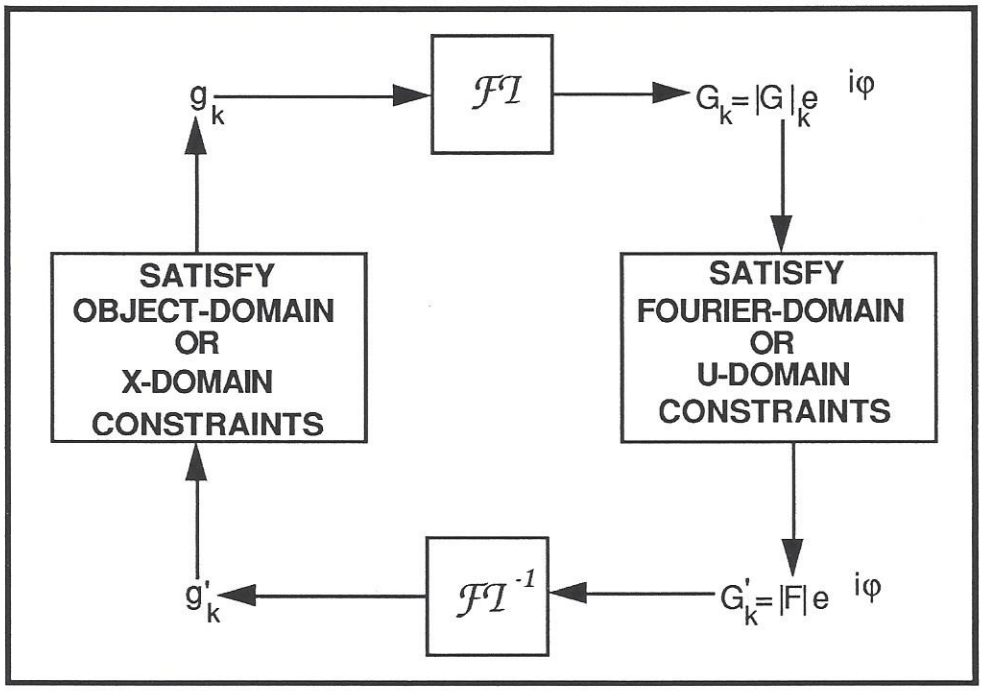

Figure 1 - The error-reduction or generalized Gerchberg-Saxton algorithm.

algorithm. In figure 1, |F| is the saved Fourier modulus or magnitude of the original object or estimate of the object.

The iterations of the algorithm continue until all the U-domain and X-domain constraints are satisfied. When this occurs, a solution has been found by the algorithm. The algorithm converges when the mean-squared error goes to zero. In the U-domain, the squared error is the sum of the squares of the difference between $G_k(u)$, the computed Fourier transform, and $G'_k(u)$, the estimate of F(u) after the U-domain constraints have been applied.[7]

$$e^2_{Fk} = N^{-2} \sum_u |G_k(u) - G'_k(u)|^2 , \quad \text{for generalized measurements} \quad (13)$$

$$= N^{-2} \sum_u [|G_k(u)| - |F(u)|]^2 , \quad \text{for dual intensity measurements} \quad (14)$$

In this case e^2_{Fk} is the mean-squared error in the U-domain or Fourier-domain and e^2_{Ok} is the mean squared error in the X-domain or object-domain.[7,14]

$$e^2_{Ok} = \sum_x |g_{k+1}(x) - g'_k(x)|^2 , \quad \text{for generalized measurements} \quad (15)$$

$$= \sum_x [|f(x)| - |g'_k(x)|]^2 , \quad \text{for dual intensity measurements} \quad (16)$$

$$= \sum_{x \varepsilon \gamma} [g'_k(x)]^2 , \quad \text{for single intensity measurements} \quad (17)$$

For the error-reduction algorithm, the mean-squared error in the U-domain can be converted to the X-domain using Parseval's theorem.[15] The energy computed in the X-domain must equal the energy computed in the U-domain, which is given by the following derivation:

$$e_{Fk}^2 = N^{-2} \sum_u |G_k(u) - G'_k(u)|^2 \qquad (18)$$

If $H_1(u) = |G_k(u) - G'_k(u)|$, then

$$e_{Fk}^2 = N^{-2} \sum_u [H_1(u)]^2$$

$$= N^{-2} \sum_u H_1(u) \cdot H_1(-u)$$

$$= N^{-2} \cdot N^2 \sum_x [h_1(x)]^2$$

where $h_1(x) = |g_k(x) - g'_k(x)|$,

$$e_{Fk}^2 = \sum_x |g_k(x) - g'_k(x)|^2 \qquad (19)$$

Both $g_k(x)$ and $g_{k+1}(x)$ always satisfy the X-domain constraints and $g_{k+1}(x)$ is always the nearest value of $g'_k(x)$ that satisfies the X-domain constraints.[7,14] Therefore, the error in the $k^{th}+1$ iteration is always less than or equalled to the error in the k^{th} iteration:

$$|g_{k+1}(x) - g'_k(x)| \leq |g_k(x) - g'_k(x)| \qquad (20)$$

or equivalently,

$$e_{Ok}^2 \leq e_{Fk}^2 \qquad (21)$$

Similarly, if Parseval's theorem is applied to the X-domain,

$$e_{Ok}^2 = \sum_x |g_{k+1}(x) - g'_k(x)|^2 \qquad (22)$$

If $|g_{k+1}(x) - g'_k(x)| = h_2(u)$, then

$$e^2_{Ok} = \sum_x [h_2(x)]^2$$

$$= N^{-2} \sum_u H_2(u) \cdot H_2(-u)$$

$$= N^{-2} \sum_u [H_2(u)]^2$$

where $H_2(x) = |G_{k+1}(u) - G'_k(u)|$,

$$e^2_{Ok} = N^{-2} \sum_u |G_{k+1}(u) - G'_k(u)|^2 \qquad (23)$$

Both $G'_k(u)$ and $G'_{k+1}(u)$ always satisfy the U-domain constraints and $G'_{k+1}(u)$ is always the nearest value of $G_{k+1}(u)$ that satisfies the U-domain constraints.[7,14] Therefore, the error in the $k^{th}+1$ iteration is always less than or equalled to the error in the k^{th} iteration:

$$|G_{k+1}(u) - G'_{k+1}(u)| \leq |G_{k+1}(u) - G'_k(u)| \qquad (24)$$

or equivalently,

$$e^2_{F\,k+1} \leq e^2_{Ok} \qquad (25)$$

Combining equations 21 and 25 for the X-domain and U-domain analysis[7,14]:

$$e^2_{F\,k+1} \leq e^2_{Ok} \leq e^2_{Fk} \qquad (26)$$

Therefore, the error can only decrease with each successive iteration.

For the case of the use of the error-reduction algorithm in this document, the normalized mean-squared error is monitored. The normalized mean-squared error is defined in the U-domain as:

$$E_{Fk}^2 = \frac{\int_{-\infty}^{\infty} |G_k(u) - G'_k(u)|^2 \, du}{\int_{-\infty}^{\infty} |G'_k(u)|^2 \, du} \tag{27}$$

where k is for the k^{th} iteration.[14] The normalized mean-squared error is defined in the X-domain as:

$$E_{Ok}^2 = \frac{\int_{-\infty}^{\infty} |g_{k+1}(x) - g'_k(x)|^2 \, dx}{\int_{-\infty}^{\infty} |g'_k(x)|^2 \, dx} \tag{28}$$

where k is for the k^{th} iteration.[14] In the U-domain, the integrand in the numerator is the squared modulus of the amount by which the computed function violates the constraints of the U-domain. A similar explanation is true for the X-domain.

As will be discussed in section 3 of this document, the error-reduction or generalized Gerchberg-Saxton algorithm gave very disappointing results. The error initially decreases rapidly but after the first few iterations begins to stagnate. The speed of the convergence depends on the constraints used in the application of the algorithm. Convergence for dual intensity measurements and one-dimenional single intensity applications is excellent, but convergence is slow for two-dimensional single intensity applications. Fienup performs error analysis in which he discovers that in some cases more than 10^5 iterations are necessary to reduce the root mean squared (RMS) error by 10^{-2} to 10^{-3}.[7] However, as Feinup, myself and others have discovered, there are plateaus where the convergence is extremely slow. After these plateaus, the error again decreases rapidly within a few iterations and then stagnates on another plateau.[1,2,7,16] Other algorithms had to be developed to work in conjunction with the error-reduction algorithm.

2.3 INPUT-OUTPUT ALGORITHM

The input-output algorithm is similar to the error-reduction algorithm except for the X-domain operations. The first three steps of the input-output algorithm are the same as the error-reduction algorithm. The input-output algorithm, shown in figure 2, always has an output that satisfies the U-domain constraints; therefore, if $g'_k(x)$ also satisfies the X-domain constraints, then it is the solution.[7,14] The input $g_k(x)$ is no longer the current best estimate of the object unlike the error-reduction algorithm; rather it is the driving function for the next output of $g'_k(x)$. Therefore, the input $g_k(x)$ does not necessarily satisfy the X-domain constraints. This allows ingenuity in devising X-domain constraints that select the next input and for devising new algorithms to accelerate the convergence process.

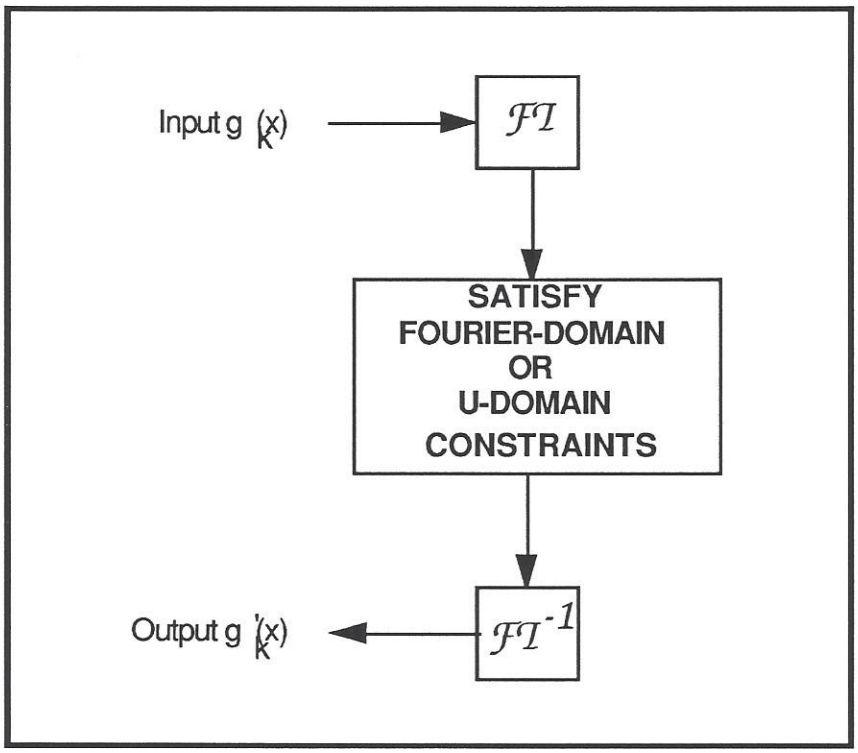

Figure 2 - The input-output algorithm.

A small change in the input results in a small change in the output in the same direction. For a small change in the input, the expected value for the change in the ouput is a constant α multiplied by the change in the input.[7,14] By changing the input, the output can be steered in the direction of the correct output. If a change of $\Delta g(x)$ is needed

in the output, then the change made to the input would be $\beta\Delta g(x)$, where β is a constant equalled to:

$$\beta = \frac{1}{\alpha} \tag{29}$$

There are many classes of input-output algorithms such as the basic input-output, the output-output, and hybrid input-output algorithms which Fienup describes in detail in references 7 and 14. This document will only briefly describe the basic input-output algorithm and in particular, the hybrid input-output algorithm.

For the problem of phase retrieval in a single intensity measurement, the change in the output should be the following:

$$\Delta g_k(x) = \begin{cases} 0, & x \notin \gamma \\ -g'_k(x), & x \in \gamma \end{cases} \tag{30}$$

where γ is the set of points that violates the X-domain constraints. When the constraints are satisfied, there is no change required in the output; when the constraints are not satisfied, the change in the output needed to satisfy the X-domain constraints drives it to zero. The process that drives it to zero occurs when the change in the input is the negative of the output at the points that violate the X-domain constraints.[7] The next input is then the following:

$$\begin{aligned} g_{k+1}(x) &= g_k(x) + \beta\Delta g_k(x) \\ &= \begin{cases} g_k(x), & x \notin \gamma \\ g_k(x) - \beta g'_k(x), & x \in \gamma \end{cases} \end{aligned} \tag{31}$$

This is called the basic input-output algorithm.[7]

In the hybrid input-output algorithm the next input $g_{k+1}(x)$ is created in the following way:

$$g_{k+1}(x) = \begin{cases} g'_k(x), & x \notin \gamma \\ g_k(x) - \beta g'_k(x), & x \in \gamma \end{cases} \qquad (32)$$

where γ is the set of points that violates the X-domain contraints. The hybrid input-output algorithm avoids the stagnation that occurs in the output-output algorithm and the error-reduction algorithm. If for a given value of x the output remains negative for more than one iteration, the corresponding point in the output grows more positive until eventually it becomes nonnegative. Notice if β is equalled to one, then the hybrid input-output algorithm reduces to the error-reduction algorithm.[7]

These input-output algorithms can be used on their own. However, for the input-output algorithms, the error E_{Fk} is usually meaningless, since the input $g_k(x)$ is no longer an estimate of the object. Therefore, the mean-squared error in the X-domain usually increases or becomes negative. Figure 3 shows a graph of the root mean squared (RMS) error vs. the β of the input-output algorithm for a specific 16 by 16 input matrix after 30 iteration of the algorithm (see section 3.3.4 for more information on how E_{Ok} is calculated). With each execution of the hybrid input-output algorithm, the algorithm started with the same distorted image but each had a different value of β. Each individual β for the input test matrix or test object produced different results - some were good results others were bad results. If β was too small the RMS of E_{Ok} did not reduce enough or increased; where as if too large a β was used, the RMS of E_{Ok} increased too much. However, as β increases, the RMS of E_{Ok} may increase but this may not be a bad result because in some cases, especially in the combination of the error-reduction and hybrid input-output algorithms, it is good for E_{Ok} to increase. It increases the energy in certain areas of the distorted image to improve and accelerate the convergence of the algorithm to the correct image. Reducing the RMS of E_{Ok} in the hybrid input-output algorithm does not necessarily mean an improvement in the quality of the image just as an increase in the RMS of E_{Ok} in the hybrid input-output algorithm does not mean an increase the amount of distortion within the image. However, when the hybrid input-output algorithm is used in conjunction with the error-reduction algorithm, it reduces the stagnation problem of the error-reduction algorithm and accelerates the convergence process of the error-reduction algorithm in significantly fewer iterations (see figure 12 in section 3.3.4). Fienup proves that this is a successful strategy for single intensity applications of the algorithm.[7,14]

For this document, section 3 will investigate the application of the error-reduction and hybrid input-ouput algorithms to problems were only information on the magnitude of an object is known in the U-domain. This is a single intensity application of the algorithm. In the X-domain, all that is known is that the object is real and nonnegative. Therefore, the stagnation encountered in single intensity applications can be overcome by alternating between the error-reduction algorithm and the hybrid input-output algorithm. The application of the error-reduction algorithm will prevent the error in hybrid input-output algorithm to increase if it begins to diverge from the object. In section 3 of this document, it is found that 10 to 30 iterations of the input-output algorithm followed by 5 to 10 iterations of the error-reduction algorithm is the best strategy. This combination of the error-reduction and hybrid input-output algorithms significantly reduces the iterations needed to retrieve the phase of the object from, in some cases, more than 10^5 to a few hundred iterations.

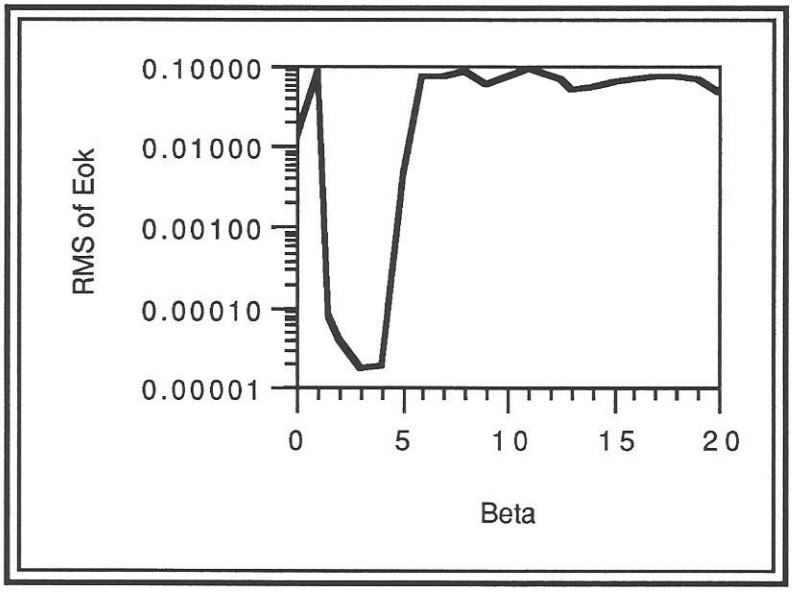

Figure 3 - RMS error of E_{Ok} vs. β for hybrid input-ouput algorithm (Log Graph).

SECTION 3

SOFTWARE IMPLEMENTATION OF THE ALGORITHM

The software implementation of the algorithm was performed on an IBM* PS/2 - 80 under DOS* version 3.32 using version 5.0 of Turbo Pascal*. To use the program, a computer must be PC compatible and must have a color graphics card. The simulation was designed for a specific application of the error-reduction algorithm. The application of this algorithm is similar to the second application discussed in section 1 of this document. In the test case, the error-reduction algorithm is used to retrieve the phase of a two-dimensional single intensity measurement which is real and nonnegative. This application is perhaps the most difficult because in most cases the algorithm is applied to dual intensity measurement or one-dimensional single intensity measurements. By single intensity, there is information on only the real components of an object in one of the domains (for this case the X-domain) and information on the real and the imaginary components in the other domain (for this case the U-domain); by dual intensity, there is information on the real and the imaginary components in both the X-domain and the U-domain. An algorithm for the recontruction of a multidimensional sequence from the modulus or magnitude of its Fourier transform has been the objective of many research efforts as described in sections 1 and 2 of this document.

This section is broken down into three subsections. Subsection 1 will outline the initial task of the simulation and discuss the problems that were encountered in the software simulation. Subsection 2 will outline the final algorithm that was used to solve the initial problem. Subsection 3 will detail the software implementation of the algorithms with a technical overview of each of the modules that were used to create the final algorithm.

3.1 PROBLEMS IN THE APPLICATIONS OF THE ALGORITHMS

Initially, I wanted to implement this application of the algorithms and create a

---

* IBM and DOS are registered trademarks of the International Business Machine Corporation. Turbo Pascal is a registered trademark of the Borland International Corporation.

database of estimated Fourier object moduli that the algorithms could compare against a corrupted input object. Unfortunately, do to problems in the programming of the algorithm, this was not possible because there was just not enough time. However, at MITRE, the development is continuing to complete the initial application of the algorithm, but as this section will outline, there are some problems with this kind of application. There were many problems to overcome just to get a working algorithm.

First, a simple algorithm was programmed for a one-dimensional single intensity measurement. The input was a waveform which was corrupted by a random phase and then retrieve using the error-reduction algorithm. The test was a complete success. In fact, the algorithm retrieved every waveform successfully and within 10 to 200 iterations. I thought that this application was going to work just as successfully in two-dimensions but I was wrong.

As with the one-dimensional application, the two-dimensional application for an object had the following constraints in the X-domain and the U-domain:

X-domain: All components must be real and nonnegative, which means that all the imaginary terms were set to zero and that all negative real terms were set to zero.

U-domain: The new best estimate of $G_k'(u)$ was equalled to the saved Fourier modulus or magnitude of the initial object F(u) multiplied by the $G_k(u)$ and divided by the modulus or magnitude of $G_k(u)$:

$$G_k'(u) = |F(u)| \, [G_k(u)/|G_k(u)|] \tag{33}$$

Including time to research articles and information on the algorithms, it took approximately four to five weeks to program a version of the error-reduction algorithm with 36 grey scale levels for the graphical display of a 8 by 8, 32 by 32, or 64 by 64 object. Unfortunately, the results were terrible due to the problem of stagnation. At first, it looked as if the algorithm was working for symmetrical objects (i.e. squares and rectangles); however, a closer look at the numerical calculation found that it was not working at all. Futhermore, for non-symmetrical or strange shapes (i.e an airplane) the algorithm almost never converged. Thousands of iteration were made on many 32 by 32 objects, and little to no change was made to the corrupted image. It was at this point that alternative algorithms had to be investigated to overcome the problem of stagnation and

the hybrid input-output algorithm, developed by Fienup, was chosen. This algorithm was the best choice for two-dimensional single intensity measurement applications.

The results of the stand alone application of the hybrid input-output algorithm were dismal and so was the combined application of the error-reduction and hybrid input-ouput algorithms to the retrieval problem. Not only was stagnation a problem but with the application of the hybrid input-ouput algorithm, excessive divergence became a problem. I took a closer look at my code from my FFT implementation which uses transposition to the implementation of the error-reduction and hybrid input-output algorithms. However, no significant problems were found. I then solicited the help of Professor Gonsalves of Tufts University and Dr. Marvin Drake of the MITRE Corporation. Dr. Drake went through the algorithm and offered the use of an alternate FFT program from MatLab. The use of this FFT algorithm yielded no difference numerically or otherwise in any of the results. Professor Gonsalves also applied different FFT algorithms with the same disappointing results.

Professor Gonsalves and myself, then broke the algorithm down into a 16 by 16 form to be applied to a 8 by 8 object. We both went through the code line by line. The numerical analysis of the error-reduction algorithm performed exactly as Fienup describe in references 7 through 14. It stagnated after the first few iterations. However, the addition of the hybrid input-output algorithm did not accelerate the process at all as claimed by Fienup. Professor Gonsalves and myself then tried several different modifications to the original error-reduction algorithm including a unique random number generation process that creates the random phase which corrupts the original object (see figure 5 in section 3.2). The random phase gerneration algorithm uses the same random phase in the object plane it does in the rest of the three planes (see section 3.2 for more information). Again the application of this random phase modification made very little improvement.

Finally, a different application of the algorithm was tested. The application that was implemented was the first application described in section 1 of this document. This dual intensity application had information on the real and imaginary components in the X-domain and the U-domain of the modulus of the original object, specifically $|f(x)|$ and $|F(u)|$. These two moduli were used to retrieve corrupted objects successfully. It was at this point that is was evident that my code was working correctly and instead something was wrong with the claims of Fienup. Research was the last option available in uncovering more information on the algorithms; otherwise, this application would have to be abandoned.

With more research, information was found on problems that others had with this specific single intensity application of the algorthim. Professor Monson H. Hayes of the Georgia Institute of Technology described similar problems in the magnitude-only reconstruction with information on the modulus of the object in only the U-domain when he used the error-reduction and hybrid input-output algorithms of Fienup. His results described the same problems that I was having in the coding of the error-reduction algorithm. Hayes claims that there are two problems preventing the development of such an algorithm. First, there is the non-linear relationship between coefficients of a multidimensional sequence and the modulus or magnitude of its Fourier transform. Second, without any phase information, it is not generally possible to form an accurate estimate of the unknown sequence from just the modulus or magnitude of its Fourier transform. Unlike the phase-only iteration technique described by Hayes, he observes that the magnitude-only iteration will not generally converge to the correct solution or to any solution even if the desired object or sequence satisfies the constraints and the uniqueness criteria.[17]

The breakthrough came when a letter from Fienup responding to the claims of Hayes was found. In this letter, Fienup comments on the article by Hayes in which Fienup refutes the claim of Hayes that the magnitude of the Fourier transform does not uniquely specify a sequence or object. Fienup outlines two important constraints which must occur. First the bounds must be known in the X-domain and second the sequence or object must be real and nonnegative. However, it may still not be possible to reconstruct the sequence or object if the region of finite support is not known. *Fienup then states that when an upper bound in the X-domain is placed on the region of support, the algorithm converges must faster in 100 or 200 iterations than when using only the nonnegativity constraint in which case several thousand iterations are necessary to retrieve the sequence or object.*[18] This discovery led to the needed breakthrough in my coding of the algorithms. Apparently, the error-reduction algorithm will converge after several thousand iterations but Fienup accelerated the process by placing a maximum bound in the X-domain based on the maximum intensity his instruments were capable of measuring. In this way, Fienup reduced the number of iteration in the algorithm to a few hundred, but he never stated that he added this maximum bound constraint in the X-domain in any of his journal articles in order to achieve his results. Fienup goes on to state that for some objects or sequences, this technique of image retrieval may not perform well even with the added constraint in the X-domain.

To get a working algorithm, it took four to five months of endless modifications to and research on the algorithm. At this stage, there was little time left for the development of some added features. With the addition of the maximum bound in the X-domain, the error-reduction and hybrid input-ouput algorithms worked successfully. The number of iterations droped from thousands to a few hundred. However, there were still more problems to address.

The second problem was discovered in the implementation of the hybrid input-output algorithm. As described in section 3.2 of this document, the error-reduction algorithm and hybrid input-output algorithm were combined into one algorithm that alternated between the two algorithms after a few iterations of each one. Feinup describes in references 7 to 14, that the best results were achieved with a β of one to three. However, for my own analysis the number of iterations was still in the thousands to retrieve some objects. Therefore, the final algorithm in section 3.2 was modified to first allow the error-reduction algorithm to iterate until it stagnates. When the error-reduction algorithm stagnated, the MSE would go to zero. The final algorithm would then save the image and then the hybrid input-output algorithm would start to iterate with a β of 2. After 20 to 30 iteration of the hybrid input-output algorithm, the hybrid input-output algorithm would stop and the error-reduction alogorithm would begin until it again stagnated. At this point the image would be checked. If there was no improvement from the saved image, then the hybrid input-ouput algorithm would start over with the saved image but this time with a β equalled to the former β plus 2. Therefore, the next set of iterations of the hybrid input-output algorithm would have a β of 4. This process would continue until there was improvement in the new image as compared to the saved image and as compared to the save modulus of the original estimated object that the algorithm desires to retrieve. However, the results of this modification were not good. Therefore, the final algorithm merely stopped when stagnation occurred in the error-reduction algorithm and then prompted the operator for a β and for the number of iterations to use this β in the hybrid input-output algorithm (see section 3.2).

The hybrid input-output algorithm in general was very sensitive to the β and to the number of iterations. As stated in section 2.3, the MSE analysis is useless during the application of the hybrid input-output algorithm because the error may increase before it begins to decrease or it may decrease and then increase. If the β was to high, the MSE increased too much and the image quality deteriorated. If there were too many iteration with a specific β, the results went from good to terrible as the image quality improved and then deteriorated. If the β was too small, the image did not change at all. Therefore, the best way to implement the final algorithm (described in section 3.2) was to allow the

operator to decide what β to use and for how many iterations of the algorithm to use it before returning to the error-reduction algorithm. The best results were achieve when a value of β (from 3 to 45) was used in the beginning of the algorithm but returned to a low value of β (0 to 3) when the operator saw the image quality improving. As noted with the hybrid input-output algorithm, sometimes the error or distortion had to increase significantly before the hybrid input-output and error-reduction algorithms decrease the error and improved the quality of the image.

The third problem was the shifting of the image, which occurred during the execution of the error-reduction and hybrid input-output algorithms. The image would sometimes move around within the 8 by 8, 32 by 32, or 64 by 64 area of pixels. To prevent this from occurring, two diagonal corner pixels were chosen in the object. These two pixels were then set to the maximum intensity possible. For the case of 36 grey scale level, they were set to 36. The two pixels became part of the retrieved image and prevented the retrieved image from moving around. The addition of these two pixels merely added a constant DC term to the original object. Alternatively, a high intensity border was also placed around the object that was to be retrieved which also prevented movement.

The fourth problem was retrieving the mirrored image of the original object. This is due to the fact that the error-reduction and the hybrid input-output algorithm have an inability to tell the difference between g(x) and g(-x). Therefore, depending on the random phase that is generated, the algorithms can retrieve g(-x) from g(x). This was only a problem when the retreived object was compared against the the stored estimate of the object.

Finally, both algorithms exhibited extreme sensitivity to the stored Fourier modulus of the object. This stored modulus must be an accurate estimate of the object that is to be retrieved. This modulus, $|F(u)|$, will be used by the hybrid input-output algorithm to steer the convergence of the object in the right direction. However, in testing the algorithms, some interesting results were discovered. If the stored modulus of the estimate is very different from the original input or object, the algorithm has a tendency to retrieve an object that resembles the stored modulus because the hybrid input-output algorithm steers the retrieval process in the direction of the stored modulus. However, both algorithms have difficulty in retrieving a complete object. If the stored modulus of the estimate is close to the original input or object, the algorithms have a tendency to retrieve an object that resembles the stored modulus because the hybrid input-output algorithm again steers the retrieval process in the direction of the stored

modulus. The recognition of bounds of distorted objects and the sensitivity to the Fourier modulus will be discussed in further detail in section 4.

3.2 OUTLINE OF THE FINAL ALGORITHM

This subsection will detail the final algorithm that was implemented but specific parts of the algorithm are described in full detailed in section 3.3. The final algorithm was a combination of the error-reduction and hybrid input-output algorithms. The algorithm itself is designed to retrieve a two-dimensional object from a distorted image that is real and nonnegative using the Fourier modulus of the original object. The following are the X-domain and U-domain constraints used in the implementation of the final algorithm which is a combination of the error-reduction and hybrid input-output algorithms described in section 2:

Error-Reduction Algorithm

X-domain: All terms must be real and nonnegative, which means all the imaginary components and all the negative real components were set to zero during each iteration of the algorithm. If any real components were greater than the maximum bound (in this case 36 for 36 grey scale levels) or exceeded the maximum intensity, then that real component was reset to the value of the maximum bound (in this case 36).

U-domain: The new best estimate of $G'_k(u)$ was equalled to the saved Fourier modulus or magnitude of the initial object F(u) multiplied by the $G_k(u)$ and divided by the modulus or magnitude of $G_k(u)$:

$$G'_k(u) = |F(u)| \, [G_k(u)/|G_k(u)|] \tag{33}$$

and

$$G_{k+1}(u) = G'_k(u) \tag{34}$$

Hybrid Input-Output Algorithm

X-domain: All components must be real and nonnegative, which means all imaginary components were set to zero and all negative real components are set to the following as described in section 2.3:

$$g_k(x) = g_s(x) - \beta g'_k(x) \tag{35}$$

where $g_k(x)$ is the new object based on $g_s(x)$, which is the saved $g_k(x)$ from the previous iteration, and based on $g'_k(x)$, which is the estimate of the object after the X-domain constraints are applied. All other components are retained:

$$g_k(x) = g'_k(x) \tag{36}$$

This description of the hybrid input-ouput algorithm is similar to the description found in section 2.3. If any real components were greater than the maximum bound (in this case 36 for 36 grey scale levels) or exceeded the maximum intensity, then that real component was reset to the value of the maximum bound (in this case 36).

U-domain: The new best estimate of $G'_k(u)$ was equalled to the saved Fourier modulus or magnitude of the initial object F(u) multiplied by the $G_k(u)$ and divided by the modulus or magnitude of $G_k(u)$:

$$G'_k(u) = |F(u)| \, [G_k(u)/|G_k(u)|] \tag{33}$$

and then

$$G_{k+1}(u) = G'_k(u) \tag{34}$$

To allow the final algorithm to converge more rapidly and clearly, any real component is set to zero if it is less than 0.0001 instead of zero in both the error-reduction and hybrid input-output algorithms.

The final algorithm is depicted in figure 4. First, a two-dimensional input or object is sampled. This input or object is called f(x), which must be real and nonnegative. The

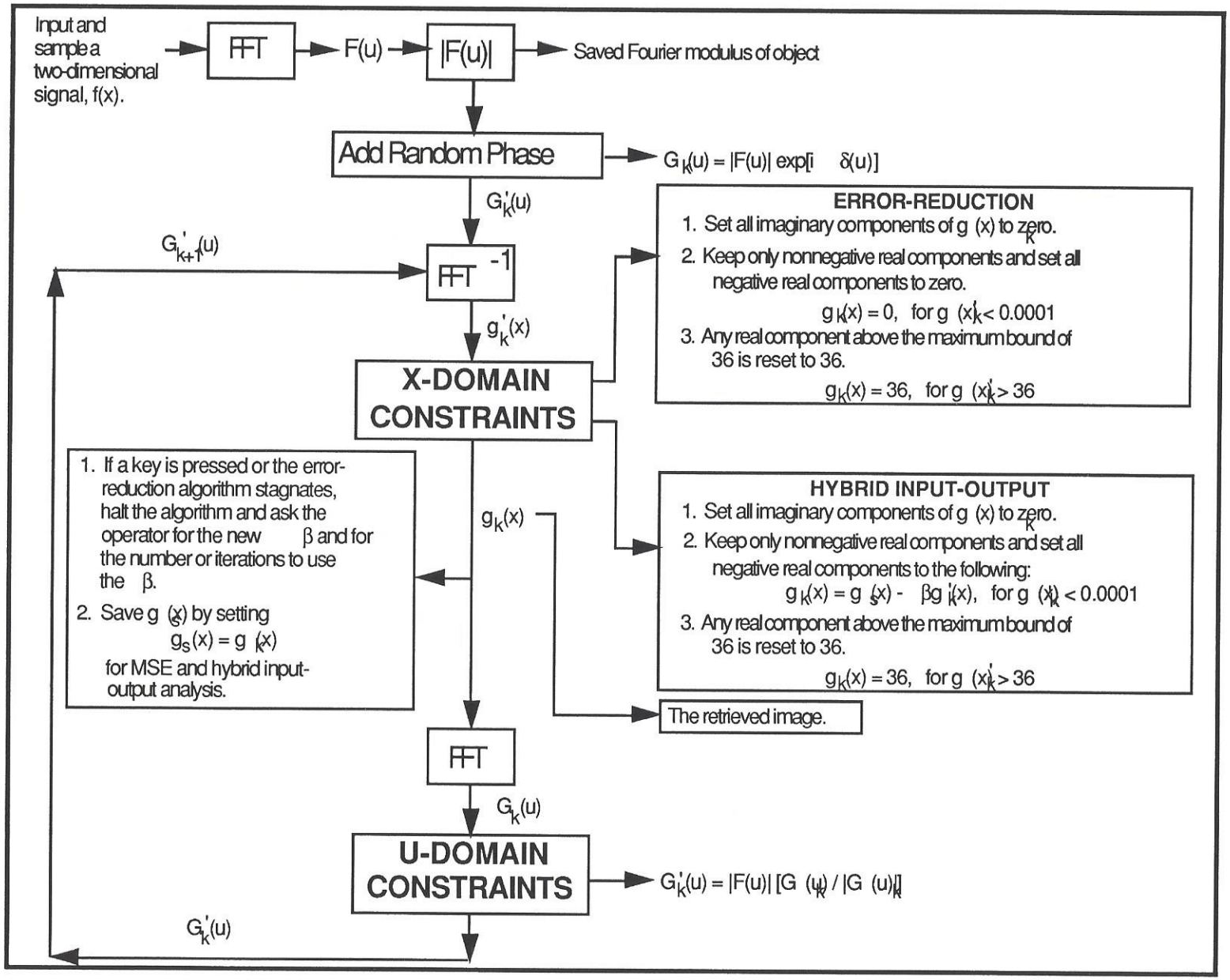

Figure 4 - The final algorithm.

object f(x) is displayed on the screen of the computer for the operator to view. The operator can then continue by pressing return or quit the program by pressing 'q' on the keyboard. Then f(x) is Fourier transformed using the Fast Fourier Transform (FFT) to yield F(u). The modulus or magnitude of F(u) is calculated and saved. Therefore, $|F(u)|$ becomes the saved Fourier estimate of the object or input, which will be used in the implementation of the U-domain constraints. The magnitude of F(u) is displayed on the screen of the computer for the operator to view. The operator can then continue by pressing return or quit the program by pressing 'q' on the keyboard. The Fourier modulus is then multiplied by a random phase $\exp[i\delta(u)]$ to create $G'_k(u)$. The random phase distorts and corrupts the image with noise. $G'_k(u)$ is then used as the input into the final algorithm. The magnitude of $G'_k(u)$ is displayed on the screen of the computer for the operator to view and is called $R(u) = |G'_k(u)|$. The operator can then continue by pressing return or quit the program by pressing 'q' on the keyboard.

$G'_k(u)$ is inversed Fourier transformed using the inverse FFT to create $g'_k(x)$. The X-domain constraints are then imposed on $g'_k(x)$. There are three X-domain contraints for both the error-reduction and hybrid input-output algorithms. The first X-domain constraint of the error-reduction algorithm sets all the imaginary components to zero; the second X-domain constraint keeps only nonnegative real components and sets any real negative components to zero; and the third X-domain constraint sets any real component above the maximum bound value of 36 to the maximum value of 36 for 36 grey scale levels. The first X-domain constraint of the hybrid input-output algorithm sets all the imaginary components to zero; the second X-domain constraint keeps only nonnegative real components and set all the negative components to the following:

$$g_k(x) = \begin{cases} g'_k(x), & \text{if real components are nonnegative} \\ g_s(x) - \beta g'_k(x), & \text{if real components are negative} \end{cases} \qquad (37)$$

where β is determined by the operator, $g_s(x)$ is the object $g_k(x)$ from the previous iteration of the final algorithm and $g_k(x)$ is the new value of $g'_k(x)$ after the X-domain constraints have been applied; and the third X-domain constraint sets any real component above the maximum bound value of 36 to the maximum value of 36 for 36 grey scale levels. The application of the X-domain constraints for the error-reduction and hybrid input-ouput algorithm yields $g_k(x)$.

The final algorithm will then evaluate $g_k(x)$ and stop if the error-reduction algorithm stagnates or the operator presses a key to halt the execution of the final

algorithm. When the algorithm stops, the operator can press return to enter new data, 's' to save that iteration to the hard disk and then enter new data, or 'q' to quit the program. When the operator presses return or the 's' key, the operator will be prompted for a new β and for the number of iterations to use the β. If a β other than 1 is selected, then the final algorithm will implement the hybrid input-output algorithm with that β and for the number of iterations indicated by the operator. Upon completing the iterations of the hybrid input-output algorithm, the final algorithm will implement the error-reduction algorithm until it stagnates again or until the operator again presses a key to halt the execution of the final algorithm. If the operator selects a β of 1, then the final algorithm will implement the error-reduction algorithm for the number of iteration indicated by the operator and then will continue executing the error-reduction algorithm until it stagnates. When stagnation occurs in the error-reduction algorithm, the final algorithm will again stop and prompt the operator for the same information again. The operator is always able to stop the algorithm during any iteration by merely pressing any key on the keyboard to interrupt the final algorithm and either modify the selection of the β or force a return to the error-reduction algorithm. If the operator press a key, then the algorithm will stop after the next iteration is complete. The algorithm determines if the error-reduction algorithm stagnates using the mean squared error (MSE) analysis which is described in detail in section 3.3.

Finally, before continuing the algorithm stores $g_k(x)$ as $g_s(x)$ for the next iteration of the hybrid input-output algorithm for the MSE analysis. At this point, $g_k(x)$ is Fourier transformed by the FFT to create $G_k(u)$. The U-domain constraints are imposed on $G_k(u)$ to yield $G'_k(u)$, which is created using the modulus or magnitude of the saved Fourier transform of the original object or estimate, F(u):

$$G'_k(u) = |F(u)| \, [G_k(u)/|G_k(u)|] \tag{33}$$

Here, $|F(u)|$ is multiplied by the phase information of $G_k(u)$ divided by the modulus or magnitude of $G_k(u)$. After creating $G'_k(u)$, $G'_{k+1}(u)$ is set equal to $G'_k(u)$ and then the final algorithm begins its next iteration with $G'_{k+1}(u)$ as the next input to be inversed Fourier transformed by the FFT.

The operator has total control over the retrieval process. Besides prompting the operator for an new β and for the number of iterations to use the β in the hybrid input-output algorithm, the operator has other options. As the final algorithm starts its iterations, the operator will have a picture of the original object in the left corner of the screen of the computer with the heading 'GS=0' below the image indicating zero iterations.

Along with the original object, the operator will be able to see some of the iterations of the final algorithms during the retrieval process. Every 5 iterations, an image is added to the screen of the computer. When the screen is full of images, there is an upgrade of one of the images on the screen of the computer excluding the image of the original object. In addition, there is an upgrade of one of the images on the screen of the computer when the operator interrupts the execution of the algorithm by pressing any key on the keyboard. The image displayed is the last iteration of the algorithm before it was interrupted. Below each image on the computer screen is displayed the type of iteration along with the iteration number. For an iteration of the error-reduction algorithm, 'GS=' appears below the image along with the iteration number; and for an iteration of the hybrid input-output algorithm, 'IO=' appears below the image along with the iteration number. Between each iteration, the RMS of E_{Ok} is displayed at the bottom of the screen (see section 3.3.4 for more information on the RMS of E_{Ok}). The final algorithm saves iterations 5, 10, 20, 40, 60, 80, 100, 250, 500, 1000, 1500, 2000, 3000, 4000, 5000, 7500, and 10000 in data files on the hard disk called 'd5.dat' for the 5^{th} iteration, 'd10.dat' for the 10^{th} iteration and similar names for the other iterations. The final algorithm also saves the original input matrix in a data file called 'ORIGINAL.DAT' which also can be displayed. The maximum number of iterations is 10,000 after which the program completes its execution. Furthermore, after the final algorithm halts, the operator can save an iteration by pressing the key 's', which represents the word 'save'. When the algorithm halts and the key 's' is pressed, the operator will be prompted for the normal information (a new β and the number of iterations to use this β). The final algorithm will automatically save that iteration on the hard disk in a data file with a name similar to the data files stored automatically. Finally, after the final algorithm halts, the operator has the capability to end the execution of the final algorithm at any time by press the key 'q', which repesents the word 'quit'. The program project will end execution but will automatically save the last iteration to a data file called 'LASTIT.DAT', which like other data files can be displayed.

In addition a separate program was developed to plot the saved data files on the screen of the computer. It prompts the operator for the name of the file the operator wants to display and gives an error message if the operator specifies a file that does not exist or a file that is not a Gerchberg-Saxton file. If it is a Gerchberg-Saxton file, then it plots that file on the screen of the computer for the operator to view. This program is helpful upon the completion of the final algorithm.

3.3 DETAILS ON THE SOFTWARE IMPLEMENTATION OF THE ALGORITHM

This subsection details the software coding of the final algorithm. Each subsection highlights the major components that make up the final algorithm. The subsections in this section will document the memory, FFT, MSE and ploting implementations which need to be discussed in more detail than described in the source code. The final algorithm has four modules that create program project. The first module is called 'PROJECT.PAS' and is the main control module in the execution of the final algorithm. The second module is the FFT program, called 'FFT_LIB.PAS' and contains all the FFT procedures and functions. The third module is the operations library, called 'OP_LIB.PAS' and contains all the different procedures for the operations performed by the final algorithm including the implementation of the X-domain and U-domain constraints. The final module is the plot library, called 'PLOT_LIB.PAS' and contains all the procedures that plot images on the screen for the operator to view. Each procedure and function is described in the source code.

Before execution program project (PROJECT.EXE), the operator should first modify the file 'INPUT.DAT'. The file 'INPUT.DAT' should contain the original object that is to be retrieved. Upon the completing the execution of program project, the operator can display any of the saved images by execution program nplot (NPLOT.EXE), which was described at the end of section 3.2. Program nplot will not be described in any futher detail in this document but the source code can be found in the main module of program nplot, 'NPLOT.PAS'.

There are three sizes of objects used to test the final algorithm: 16 by 16, 64 by 64 and 128 by 128. In this document, X stands for the number of columns and Y stands for the number of rows in an X by Y matrix. Twice the bandwidth in the U-domain must be allowed than in the X-domain in order to retrieve the entire image. A bed of zeros twice the size of the image is necessary to retrieve the image in full detail. Using a bed of zeros less than twice the size of the image, may not completly retrieve the image. The image is embeded in either the center of a matrix of zeros or in the left hand corner of a matrix of zeros. In all the examples in the appendix, the image is placed in a bed of zeros that it twice the size of the image. For example, a 16 by 16 matrix only uses an 8 by 8 plane for the object. The other three 8 by 8 planes are nothing more than zeros. The zeros are used as in the one-dimensional signal application to give the object an area to be retrieved from. On each successive iteration of the algorithm, these three zero planes in two-dimensions are reset to zero as part of the X-domain constraints. This allows the

retrieval process to be influence by the effects of the data in the U-domain because the bandwidth in the U-domain is usually larger than the bandwidth in the X-domain. Therefore, the other three planes are used to allow the object to expand in the U-domain. Figure 5 shows how this setup works for a 16 by 16 matrix where the object plane is in

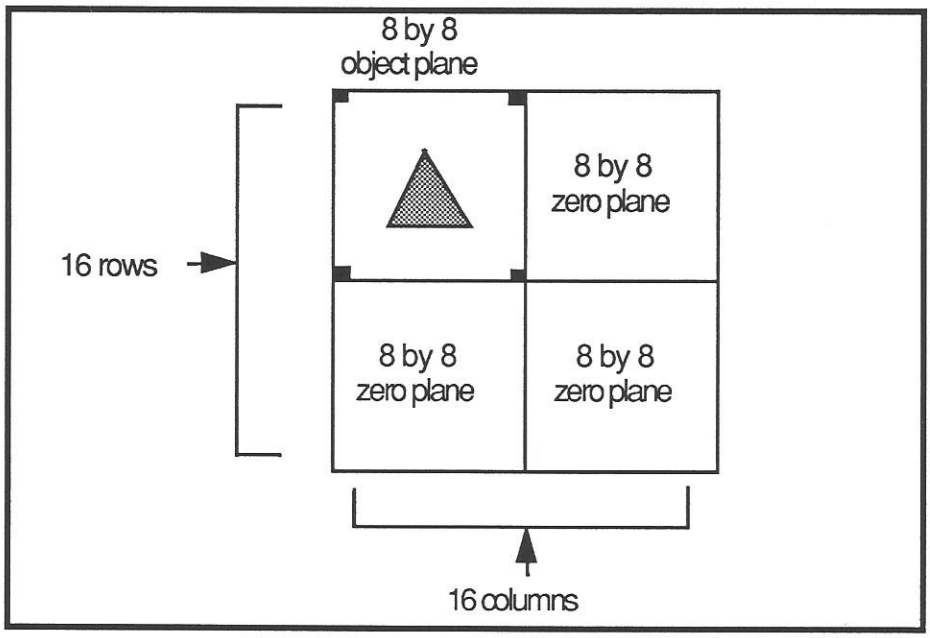

Figure 5 - The object plane of an input matrix.

the upper left corner. An alternate technique is to place the object plane in the center of the 16 by 16 matrix and surround the object plane with zeros. There are four dark pixels in diagonol corners used to prevent the image from moving around the screen during the retrieval process as described in section 3.1. There are similar setups for a 64 by 64 and 128 by 128 matrix. Therefore, the actual size of the object is contained in a 8 by 8 matrix for a 16 by 16 input, a 32 by 32 matrix for a 64 by 64 input, and 64 by 64 matrix for a 128 by 128 input. The 16 by 16, 64 by 64, or 128 by 128 input should be contained in the file 'INPUT.DAT'.

3.3.1 MEMORY

The memory of a PC is a constraint because for DOS the pascal source code was restricted to 640 KB. Therefore, the operator must determine three parameters before recompiling the code using version 5.0 of Turbo Pascal. First, the object must be within a square amount of pixels such as 8 by 8, 32 by 32, or 64 by 64. The operator then must

decide the size of the blocks in memory and then set the following constants in the module: max_samples, sample_block and blocks.

1. max_samples: is a set to the square size of the entire input matrix. For example for a 16 by 16 matrix, it is set to 16; for 64 by 64, it is set to 64; and for a 128 by 128, it is set to 128.

2. sample_blocks: is the size of the sample block. The sample block is the square size of the blocks which make up the size of the entire input matrix. For example for a 16 by 16 matrix, the object is broken into four planes each 8 by 8.

3. blocks: is the number of planes that make up the object. For a 16 by 16 matirx, there are four blocks each containing an 8 by 8 matrix that makes up the input matrix.

4. sqr_sample_block: is automatically calculated as the square of the size of the sample block. If the sample block is 8, its square is 64, which is the total number of entries in the sample block.

5. sqr_max_samples: is automatically calculated as the square of the size of the maximum number of samples. If the maximum number of input samples for the object is 16, then its square is 256, which represents the total number of entries in the input matrix.

6. rows: is automatically calculated as the maximum samples divided by the number of blocks:

$$(\text{max\_samples} / \text{blocks}) \qquad (38)$$

It represents the number of rows in a sample block. The value of sample_blocks represents the number of columns in a sample block. For a 16 by 16 matrix, the following is the calculation for the number of rows in a sample block:

$$\text{rows} = (16 / 4) = 4$$

Therefore, from the above calculations the memory in extended heap is parsed using an array of size 'blocks' that holds a pointer to extended heap. This pointer in extended heap points to an array which is the size of 'max_samples' by 'rows' (column by row). The main data structure in extended heap is called 'data'. It is an array of size 'blocks' that contains the pointers to arrays in extended heap. Each entry in the arrays in extended heap has two fields. The first field is called 'real_pt' for the real component of that pixel or sample and the second field is called 'image_pt' for the imaginary component of that pixel or sample.

For a 16 by 16 matrix, 'max_samples' is 16, 'sample_blocks' is 8 and 'blocks' is 4. Figure 6 depicts the memory requirements of a 16 by 16 matrix. In this case there are four blocks of memory in extended heap, which have pointers from 'data'. The reason for using an array in extended heap is that the access to the data through an array is much faster than access through a linked list. Moreover, Turbo Pascal as well as Microsoft Pascal are unable to compile a data stucture such as a matrix in the form of an array that is larger than 64 KB. Therefore, pointers to extended heap to matrices less than 64 KB were needed to accelerate the algorithms. Even though it is not necessary to implement this kind of memory structure for a 16 by 16 matrix, it is necessary for a 64 by 64 and 128 by 128 matrix. For a 64 by 64 matrix, 'max_samples' is 64, 'sample_blocks' is 16 and 'blocks' is 4. Figure 7 depicts the memory requirements for a 64 by 64 matrix. Finally, for a 128 by 128 matrix, 'max_samples' is 128, 'sample_blocks' is 16 and 'blocks' is 8. Figure 8 depicts the memory requirements for a 128 by 128 matrix. In order to execute this algorithm with a 128 by 128 input matrix, changes must be made to the source code. Five data structures of type 'fftp' and of size 128 by 128 have two real field each of size 8 bytes. These five data structures are needed to perform the added MSE and hybrid input-output analysis. This would require 1.35 MB of memory which is not possible with only 640 KB of memory for a conventional computer. If the two real fields in the data structures of type 'fftp' are changed to short integers (4 bytes each) and the MSE analysis is removed from the source code, only four data structures are needed to execute the source code. This would only require 525 KB of memory. Theoretically, a even more streamlined version of the source code can be created to run a 256 by 256 input matrix; however, it is probably wiser to tranfer the code to a workstation for an input matrix of this size. Other memory combinations are possible by setting 'max_samples', 'sample_blocks' and 'blocks' to different values to meet the requirements. However, the original matrix must be square. When executing the program project with a 64 by 64, or larger matrix application, the operator should compile and executable version of the code to the hard disk and leave Turbo Pascal; otherwise, the computer will run out of memory with Turbo Pascal in the background.

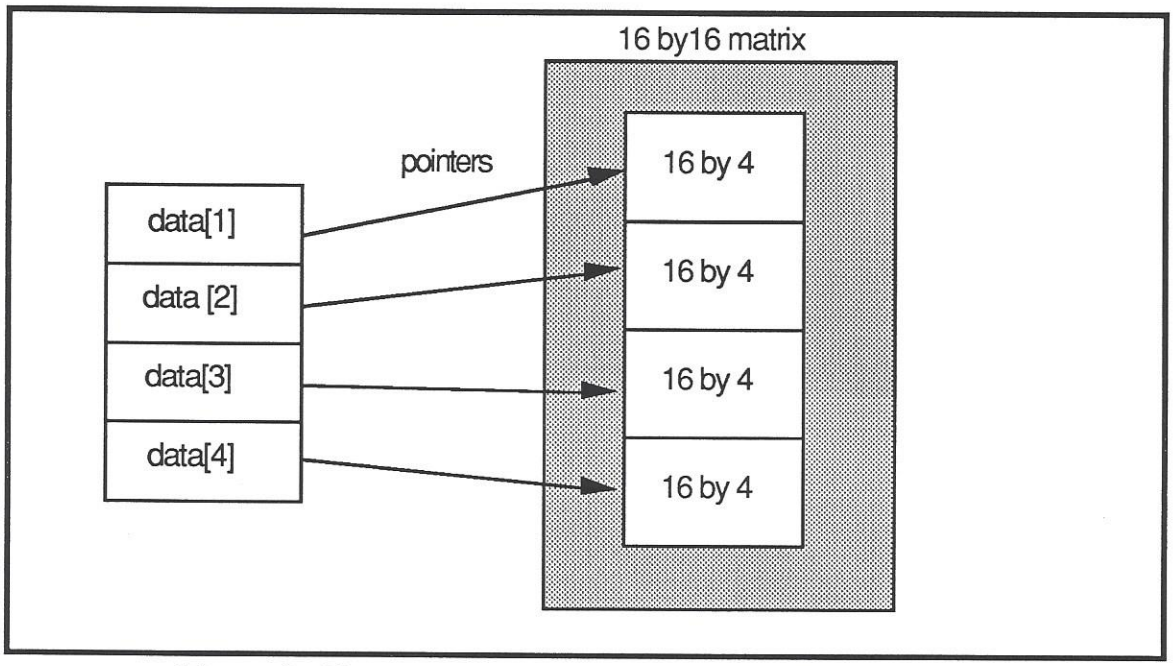

Figure 6 - The memory requirements of a 16 by 16 input matrix.

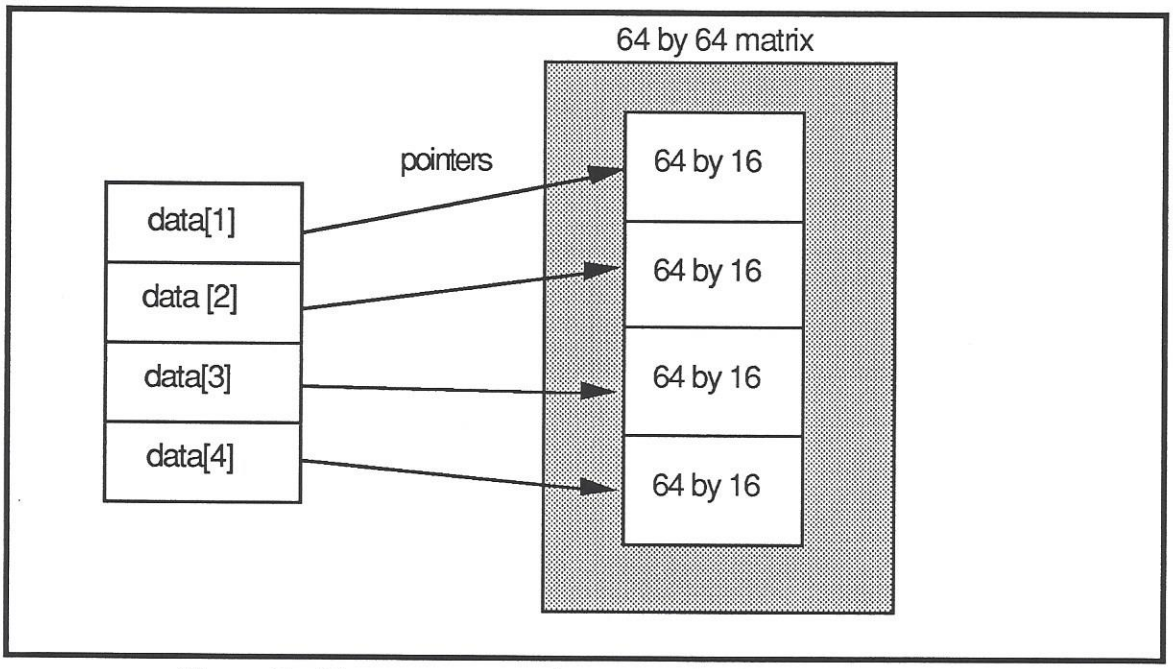

Figure 7 - The memory requirements of a 64 by 64 input matrix.

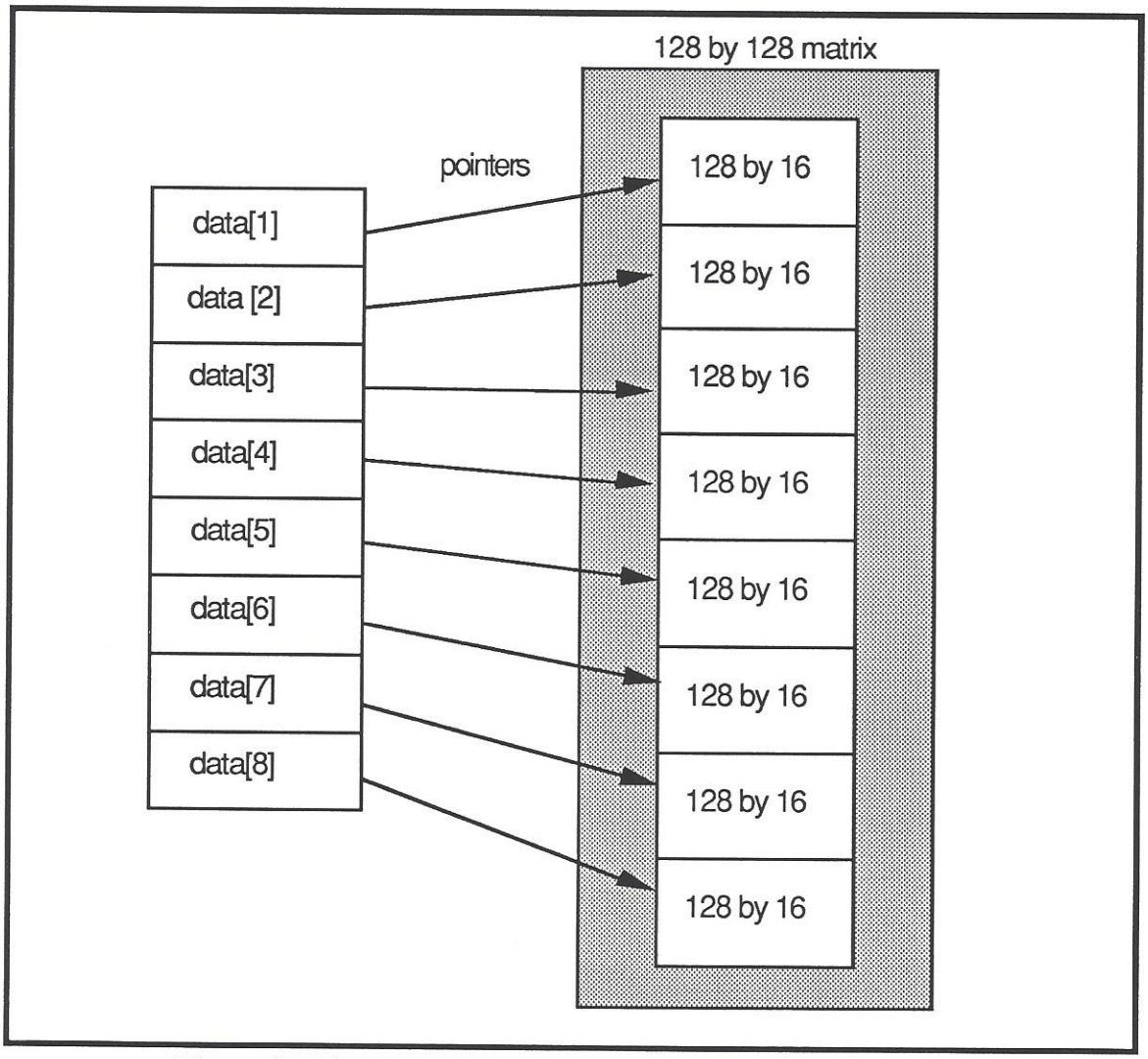

Figure 8 - The memory requirements of a 128 by 128 input matrix.

3.3.2 FAST FOURIER TRANSFORM (FFT)

The discrete Fourier transform analysis used in the final algorithm was implemented using the Fast Fourier Transform (FFT).[19] In this application, the data of the two-dimensional object was stored in a two-dimensional matrix, 16 by 16. To take the FFT of a two-dimensional matrix, first Fourier transform all of the rows in the matrix; second, transpose the matrix; third, Fourier transform all the rows again; and fourth, transpose the matrix again. This essentially reduces to taking the Fourier transform of

the rows and then the Fourier transform of the columns. Transposition merely turns the rows into columns or the columns into rows. Figure 9 depicts the Fourier transform process using the FFT. Figure 10 shows how a 16 by 16 matrix is transposed.

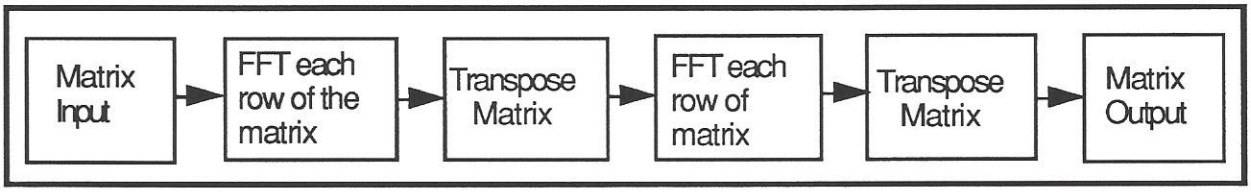

Figure 9 - The FFT of a two-dimensional matrix.

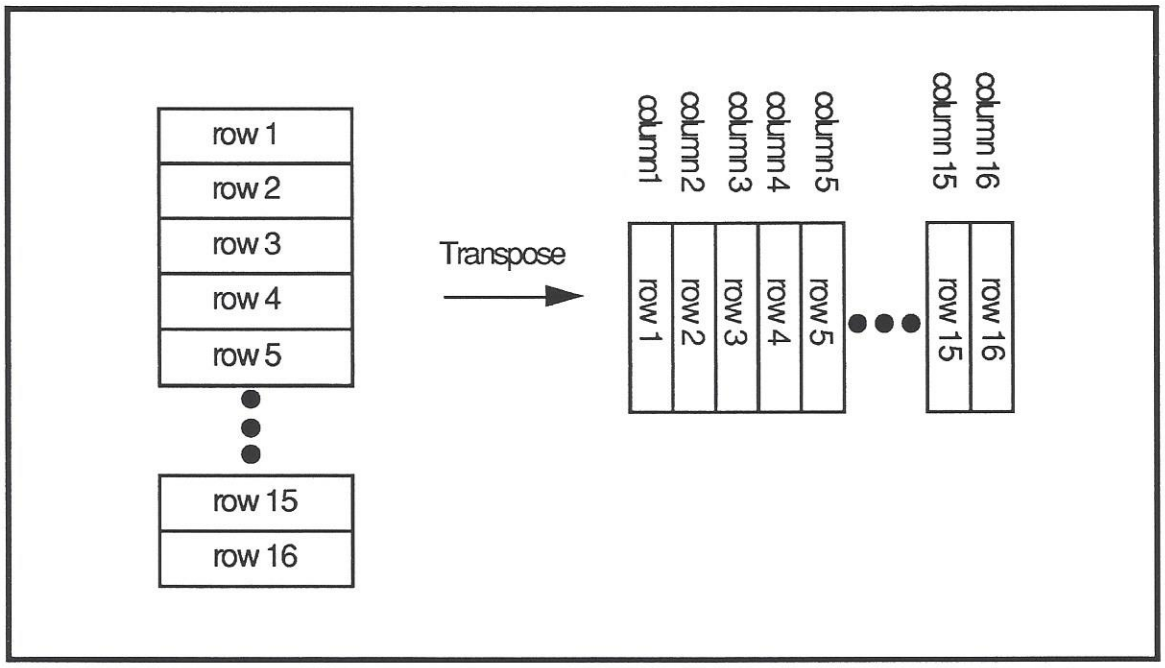

Figure 10 - Transposition of a two-dimensional matrix.

Therefore, the (column,row) entry of (1,1) in the matrix stays in the same position but now the (row,column) entry is (1,1). However, the (column,row) entry of (2,1) changes to the (row,column) entry of (2,1). This effectively turns the rows into columns or the columns into rows. Transposition is complicated even further in the source code because of the way data is stored in extended heap (see section 3.3.1).

The FFT analysis is located in the FFT module called 'FFT_LIB.PAS' and contains all the functions and procedures that perform an discrete FFT on an input matrix. Function power_down determines the power of a base number that created the number

sent to the function power_down. It is useful in determining the power of 2. It is used to determine the number of stages in the FFT algorithm. Procedure find_trans_entry is used to find the transpose entry within the memory technique implemented in program project (see section 3.3.1). Procedure transpose tranposes the two-dimensional matrix for the FFT algorithm. Procedure nfft takes the FFT of one row in a matrix. Procedure two_d_fft breaks the matrix into individual rows to be sent to procedure nfft and also calls procedure transpose.

3.3.3 OPERATIONS

Many of the operations performed in the final algorithm are located in the operations library, which is called module 'OP_LIB.PAS'. Procedure dispose_pointers disposes of all the pointers that point to matrices in extended heap. Procedure save_info saves information by writing it to the hard disk in files previously described in section 3.2. Procedure get_input reads the 'INPUT.DAT' file, which contains the original object. Procedure make_sine_cos_tbl makes the sine and cosine tables for the analysis used to retrieve the object. This allows the FFT to run more rapidly because the FFT does not have to calculate the sine and cosine function. Instead, the FFT merely uses a look up array or table to access the sine and cosine values. Procedure multiply_2_d multiplies the two-dimensional arrays together that have both real and imaginary components. Multiplying one matrix of real and complex components with another is accomplished using the following process:

$$(x + jy) \cdot (w + jz) = (xw - yz) + j(yw + xz) \tag{39}$$

$$\text{real component} = (xw - yz) \tag{40}$$

$$\text{complex component} = (yw + xz) \tag{41}$$

Procedure random_phase distorts the original object by calculating random phase matrix of real and imaginary components and then by multiplying the random phase matrix with the Fourier transform of the orginal object matrix. Random phase is created in the following manner:

$$\cos(2 \cdot \pi \cdot r) + j \cdot [\sin(2 \cdot \pi \cdot r)] \tag{42}$$

where 'r' is a randomly generated number created by using the random number generation program in Turbo Pascal. An alternate technique is a random phase generation program

created by Professor Gonsalves which program project and the final algorithm uses (see section 3.1). Either form of random phase generation is valid. The random phase generation program of Professor Gonsalves merely accelerates the convergence process, but even with an established random phase generation process, the final aglorithm is capable of retrieving any object. Procedure theta_mod applies the U-domain constraints to a matrix for both the error-reduction and hybrid input-output algorithms. Procedure save_prev_data saves the data of a matrix for the next iteration to be used in the MSE analysis or the hybrid input-ouput algorithm. Procedure save_error_data saves the data of a matrix for the MSE analysis. Procedure x_domain_contraints applies the X-domain constraints of the error-reduction or the hybrid input-output algorithm depending on which algorithm is executing during an iteration. In addition, this procedure performs preliminary calculations for the MSE analysis of the hybrid input-output algorithm. Procedure calculate_error is explained in section 3.3.4. Procedure save_mod_file saves a particular iteration on the hard disk.

3.3.4 MEAN SQUARED ERROR (MSE) ANALYSIS

The mean squared error (MSE) analysis is performed in the operations library in module 'OP_LIB.PAS' in procedure calculate_error. The mean squared error analysis is only used with the error-reduction algorithm to determine if it converged or stagnated. From section 2.2, the normalized MSE is expressed in the following equations:

$$E_{Ok}^2 = \frac{\int_{-\infty}^{\infty} |g_{k+1}(x) - g'_k(x)|^2 \, dx}{\int_{-\infty}^{\infty} |g'_k(x)|^2 \, dx} \tag{27}$$

$$= \frac{e_{Ok}^2}{\int_{-\infty}^{\infty} |g'_k(x)|^2 \, dx} \tag{43}$$

and

$$E^2_{Fk} = \frac{\int_{-\infty}^{\infty} |G_k(u) - G'_k(u)|^2 \, du}{\int_{-\infty}^{\infty} |G'_k(u)|^2 \, du} \tag{28}$$

$$= \frac{e^2_{Fk}}{\int_{-\infty}^{\infty} |G'_k(u)|^2 \, du} \tag{44}$$

and therefore using equations 19 and 20:

$$e^2_{Ok} \leq e^2_{Fk} \tag{21}$$

The unnormalized MSE causes the algorithm to stop when the following is true:

$$e^2_{Ok} = e^2_{Fk} \tag{46}$$

Applying this analysis to a discrete implementation of the MSE analysis, the following must be true, given that $g_{k+1}(x)$ is $g_k(x)$ and $g_k(x)$ is $g_s(x)$ in Figure 4 of section 3.2.

$$e^2_{Fk} = \sum_x |g_k(x) - g'_k(x)|^2 = \sum_x |g_s(x) - g'_k(x)|^2 \tag{47}$$

$$e^2_{Ok} = \sum_x |g_{k+1}(x) - g'_k(x)|^2 = \sum_x |g_k(x) - g'_k(x)|^2 \tag{48}$$

$$e^2_{Fk} = \int_{-\infty}^{\infty} |g_k(x) - g'_k(x)|^2 \, dx = \sum_x |g_s(x) - g'_k(x)|^2$$

$$= \sum_x \{ [\mathrm{Re}(g_s(x)) - \mathrm{Re}(g'_k(x))]^2 + [\mathrm{Im}(g_s(x)) - \mathrm{Im}(g'_k(x))]^2 \} \tag{49}$$

$$e_{Ok}^2 = \int_{-\infty}^{\infty} |g_{k+1}(x) - g'_k(x)|^2 \, dx = \sum_x |g_k(x) - g'_k(x)|^2$$

$$= \sum_x \{ [Re(g_k(x)) - Re(g'_k(x))]^2 + [Im(g_k(x)) - Im(g'_k(x))]^2 \} \quad (50)$$

Finally, to normalize the error, it is also necessary to calculate the following:

$$n_k = \int_{-\infty}^{\infty} |g'_k(x)|^2 \, dx = \sum_x |g'_k(x)|^2 \quad (51)$$

$$= \sum_x \{[Re(g'_k(x))]^2 + [Im(g'_k(x))]^2\} \quad (52)$$

Therefore, the error reduces to the following:

$$\text{Error} = \frac{e_{Fk}^2 - e_{Ok}^2}{n_k} \quad (53)$$

And

$$0 \leq \text{Error} \leq \text{Maximum Value} \quad (54)$$

When this occurs, the error-reduction algorithm halts and prompts the operator for new information. The maximum value is typically from 0.00001 to 0.0001. Figure 11 shows the error analysis over 40 iterations on a 16 by 16 input test matrix using only the error-reduction algorithm with the maximum value set to 0.00005. However, the results were poor because as you can see the MSE actually increased during some of the iterations. Again, this analysis seemed to disprove the claims of Fienup until it was realized that using e_{Fk} has no meaning when applied to a single intensity measurement for the error-reduction algorithm. This anlaysis was design for dual intensity measurments,which explains the increase in the normalized MSE at about 25 iterations. Therefore, using this error analysis on the error-reduction algorithm with only a single intensity input only shows that the MSE gradually decreases as the number of iterations continues. There is a rapid descrease in the MSE in the first few iterations. However, as Fienup, Hayes and

myself experienced, this rapid decrease is followed by a longer period (thousands of iterations) of stagnation that will last long past 40 iterations. Moreover, the decrease initially in the normalized MSE does not necessarily mean that the image has been retrieved only that it has converged. Using this MSE analysis on the hybrid input-ouput algorithm is useless because e_{Fk} has no meaning and would cause the MSE anlaysis to produce extremely negative and positive values. The error e_{Fk} is meaningless since

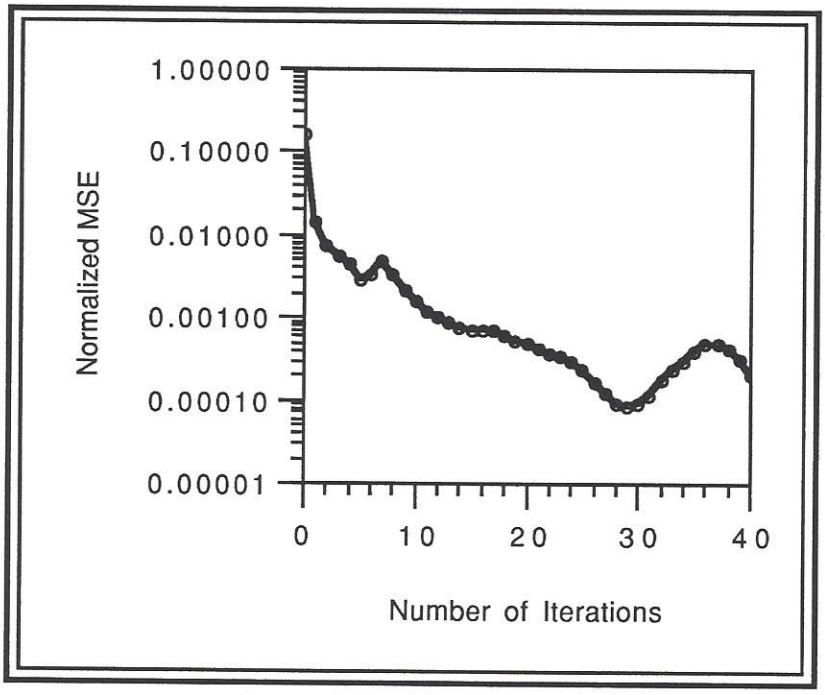

Figure 11 - Error vs. number of iterations (Log Graph).

$g_k(x)$ is no longer a estimate of the object as discussed in section 2.3. This explanation also is true for this particular application of the error-reduction algorithm because it does not use the modulus of the original object, $|f(x)|$, in the X-domain constraints to retrieve the image. Instead, the X-domain constraints is that the object is real and nonnegative. Therefore, for a single intensity measurement, the following evaluation of the e_{Ok} was derived for the error-reduction algorithm to determine if the image converged or stagnated using equation 48:

$$e^2_{Ok} = \sum_x |g_k(x) - g'_k(x)|^2 \qquad (48)$$

E_{Ok} is the root mean square (RMS) value of e^2_{Ok} using equation 51.

$$n_k = \sum_x |g'_k(x)|^2 \tag{51}$$

where from equation 52,

$$\sum_x |g'_k(x)|^2 = \sum_x \{[Re(g'_k(x))]^2 + [Im(g'_k(x))]^2\} \tag{52}$$

Using equations 48 and 50 for e^2_{Ok} and equations 51 and 52, E_{Ok} is:

$$E_{Ok} = \sqrt[2]{\frac{e^2_{Ok}}{n_k}} \quad \text{,error reduction} \tag{55}$$

and

$$0 < E_{Ok} < \text{Maximum Value} \tag{56}$$

If E_{Ok} is less than the maximum value, then the algorithm halts and prompts the operator for input. This RMS application works successfully for the error-reduction algorithm. Again modification of the original MSE anlaysis in section 2.2 is necessary for the single intensity application of this algorithm. The maximum value is set by the variable max_error in the main module 'PROJECT.PAS' and can be changed if the code is recompiled. Max_error was set to 0.18. During the iterations of the error-reduction algorithm, the final algorithm will halt three iterations after the RMS error of E_{Ok} becomes less than 0.18. This was done to allow the error to decrease during the iterations of the error-reduction algorithm. As will be shown in figure 12, the RMS error is typically much less than 0.18 when the error-reduction and hybrid input-output algorithms are combined. Therefore, at least three iterations of the error-reduction algorithm are performed after the iterations of the hybrid input-output algorithm have completed.

The following is the RMS analysis for the hybrid input-ouput algorithm using equation 17:

$$e^2_{Ok} = \sum_{x \varepsilon \gamma} |g'_k(x)|^2 \qquad (17)$$

where,

$$\sum_{x \varepsilon \gamma} |g'_k(x)|^2 = \sum_{x \varepsilon \gamma} \{[Re(g'_k(x))]^2 + [Im(g'_k(x))]^2\} \qquad (57)$$

When x is a member of γ, then it violates the X-domain constraints and must be re-evaluated using the hybrid input-output algorithm. The normalized equation for the RMS of E_{Ok} is the following using equations 17 and 57 for e^2_{Ok} and equations 51 and 52 for n_k:

$$E_{Ok} = \sqrt[2]{\frac{e^2_{Ok}}{n_k}} \quad \text{, hybrid input-output} \qquad (58)$$

Figure 12 shows a comparison of the RMS of E_{Ok} vs. the number of iteration for the error-reduction algorithm and for the combination of the error-reduction and hybrid input-output algorithms. The original object was a 16 by 16 input test matrix for the 40 iterations of the algorithm. Line 1 represents only the error-reduction algorithm. Line 2 represents an alternating strategy of 5 iterations of the error-reduction algorithm followed by 15 iterations of the hybrid input-output algorithm until 40 iterations are reached. Line 3 represents an alternating strategy of 2 iterations of the error-reduction algorithm followed by 18 iterations of the hybrid input-output algorithm until 40 iterations are reached. A β of 5 was chosen as the optimum β after some analysis. Line 1 is represented by circles; line 2, squares; and line 3 diamonds. Each test was given the same starting distorted object before the iterations began. For the error-reduction algorithm the error level offs and begins to stagnate after only 2 iterations and it will take many more before the image will converge to the original object. In this case after 40 iterations, the image not only stagnates but is a long way from converging to the image or even from improving the quality of the image. For line 2, after the five iterations of the error-reduction algorithm, initially the error begins to decrease, stagnates at the tenth iteration, increases after the tenth iteration and then decresease sharply. It converges to the original image at iteration number 29. For line 2, the error sharply decreases at a faster rate than line 3 and then levels off. Line 3 converged to the original image in 35 iterations. For different objects, there are perhaps many optimum strategies that can be

devised to retrieve the image in the fewest number of iterations. Line 2 and line 3 of figure 12 represent only two of thousands of possibilities.

However, it is a good recommendation that the combining of the error-reduction and hybrid input-ouput algorithm is the best way to retrieve an image because the error-reduction algorithm prevents the hybrid input-output algorithm from increasing the error significantly in the original image. For the case of the hybrid input-output algorithm, the visual quality improves but E_{Ok} does not. However, if a few iterations of the

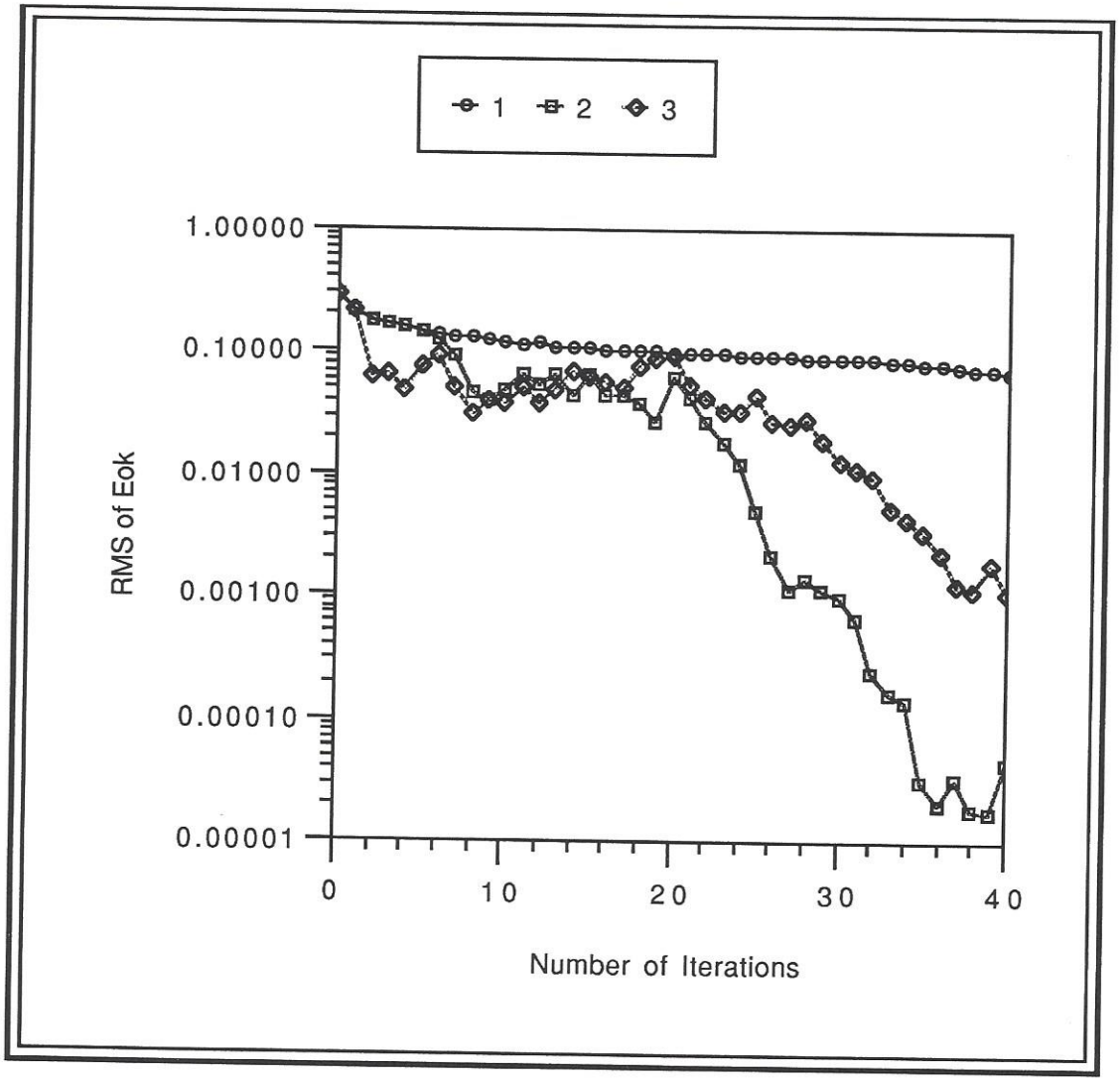

Figure 12 - RMS of E_{Ok} vs. the number of iterations (Log Graph); (a) line 1 is the error reduction algorithm; (b) line 2 is 5 iteration of the error-reduction algorithm followed by 15 iterations of the hybrid input-output algorithm; (c) line 3 is 2 iteration of the error-reduction algorithm followed by 18 iterations of the hybrid input-output algorithm.

error-reduction algorithm are combined with the hybrid input-output algorithm, the visual quality during the iterations of the error-reduction algorithm improves the quality of the distorted image very little but causes E_{Ok} to decrease rapidly until it is consistent with the quality of the image. E_{Ok} in line 2 decreases sharply during the iterations 20 to 25 of the error-reduction algorithm even though in the previous 15 iterations the error increases and decreases during the iterations of the hybrid input-output algorithm. The same is true for line 3 at iterations 20 and 21, although it is less noticeable. Again alternating between the two algorithms prevents the error in the hybrid input-output algorithm from eventually increasing due to the value of β even after the distorted image has converged to the orginal object. As line 2 shows at iterations 36 to 40, the error begins to increase even after the image has converged to the original object. If the algorithm was allowed to continue the next five iterations of the error-reduction algorithm would sharply decrease the error in the image and would prevent the introduction of any added distortion. Clearly, the strategy in line 2 or 3 is far superior than only using the error-reduction algorithm. This analysis reinforces the claims of Fienup and dispells those of Hayes; therefore, an algorithm exists that reduces the error and distortion in a image when only the magnitude of the Fourier tranform of the object is known and when the object is real and nonnegative.

3.3.5 PLOT

The plot library is contained in module 'PLOT_LIB.PAS'. It creates the patterns and displays the images on the screen of a PC computer with color graphics. The object or image is displayed on the screen of the computer in five colors which created the 36 grey scale levels. Light blue is for the value zero in the bit pattern; white is for the value 1; light grey is for the value 3; and black is for the value 4. Only the object plane of the matrix is displayed; therefore, the zero planes are not displayed because they are of no importance to the operator. Procedure display_pix plots one pixel on the screen of the computer. Procedure create_patterns creates all 36 grey scale patterns. Procedure display_image scales each data point of the matrix that is to be displayed. The possible values procedure display_image can scale to are 0 to 36. Procedure display_image calls procedure display_pix to map it to the screen of the computer. Procedure pause stops the program, gives instructions to the operator and determines what keys have been pressed.

Procedure beep_op causes the computer to beep at the operator. Procedure dip_dis locates and sets the x-y axis for the image to be displayed as well as determining how many images to display on the screen.

The 36 grey scales can be modified to create 64 greyscale levels. However, it can be used with only the 16 by 16 and 64 by 64 matrices that have object planes of 8 by 8 or 32 by 32 respectively; otherwise, for a 128 by 128 input matrix, a 64 by 64 object plane with 64 grey scale levels will exceed the number of pixels on a conventional PC monitor. The implementation of 64 grey scale levels can be done for a 16 by 16 and 64 by 64 input matrix, if the operator enters module 'PROJECT.PAS' and changes the bit_size constant from 3 to 4. Then the operator must enter module 'OP_LIB.PAS' and make changes to procedure x_domain_constraints. In procedure x_domain_contraints, the maximum bound should be changes from 36 to 64. Finally, the appropriate changes should be made to the module 'PLOT_LIB.PAS'.

SECTION 4

RECOMMENDATIONS AND CONCLUSIONS

There are many conclusions and recommendations to make concerning the final algorithms. First, there is an algorithm for the reconstruction of a multidimensional sequence or image from the estimated modulus or magnitude of its Fourier transform given that the original object is real and nonnegative. Alternating between the error-reduction and hybrid input-output algorithms provides the optimal phase retrieval technique for this application. Second, the use of these algorithms for two-dimensional image retrieval is of great importance in astronomy and pupil synthesis. As described in section 1, there have been many other successful applications of both algorithms to other problems but perhaps the greatest potential for these algorithms is in optical applications. Both the error-redcution and hybrid input-output algorithms are very healthly and robust algorithms which can retrieve the worst distorted images using information on the Fourier modulus. The error-reduction algorithm reduces the error in the distorted image with each iteration but only changes the quality of the distorted image slightly. The hybrid input-output algorithm improves the quality of the distorted image but can increase the error in the distorted image if the iterations continue for too long. Therefore, these two algorithms complement each other perfectly when implemented together. The appendix provides examples of the ability of the final algorithm to retrieve distorted images.

However, there are still many problems to be addressed. First, if an estimate of the object is known, then it can be applied to retrieve a corrupted image. Monitoring the RMS in the application of the error-reduction algorithm can not only determine whether or not the image has converged or stagnated but can also provide information as to whether or not the estimate of the object represents the corrupted image. In addition, there should be a further investigation into the effects that a Fourier modulus has on the retrieval process if it is only an estimate of the original modulus. The sensitivity of the final algorithm is caused by the error-reduction and hybrid input-output algorithm acting as the driving force behind the retrieval process using the Fourier modulus. However, more work is need to investigate the sensitivity of the error-redcution and hybrid input-output algorithms to the estimate of the Fourier modulus of the original object as well as the senstivity of the hybrid input-output algorithm to the β input. Second, Hayes points to some of these problems in his discussion of his magnitude-only reconstruction algorithms using information on the Fourier modulus.[17] He discovers it is not always possible to

know the boundaries of the distorted image from just the Fourier modulus. However, Hayes discusses the application of the algorithms to the phase-retrieval problem for a discrete multidimensional sequence. He develops a similar iterative procedure as the error-reduction algorithm for the reconstruction of a signal from the modulus of its Fourier transform. The information necessary to use his recursion algorithm is the boundary values of the distorted image to help determine an estimate of the Fourier modulus. However, Hayes is not always able to determine these boundary values from the Fourier modulus alone. Information in the X-domain as to the boundary values and in the U-domain as to an estimated Fourier modulus are needed. If the distored image has a region of support with a certain geometry, then it is possible to determine the boundary values.[20] However, this area needs more research. Third, Paxman and Fienup analyzed an optical implementation of an algorithm similar to the error-reduction algorithm based on the phase-diversity method of Professor Gonsalves.[21] The technique was used on a mirrored telescope that suffered from phase errors due to unaligned segments which had to be within a fraction of a wavelength. Paxman and Fienup analyzed this algorithms with the effects of added noise. These results are important because in the final algorithm the estimate of the Fourier modulus was noise free and therefore an exact estimate of the object. However, if there is added noise to the estimate, then the noise will effect the convergence of the image because the noisy estimate of the Fourier modulus generally the driving force behind the algorithms. They encountered many problems in this optical implementation. It suffices to say that although the computer simulated implementation of these algorithms works sucessfully, more work is needed for an optical implementation.

There are many recommendations to make. First, persistence and patience in the presuit of a working algorithm is necessary. Second, the anlaysis in section 3 was hindered by the memory contraints of 640 KB on a PC. A Sun or Apollo workstation with a larger memory and faster speed is definitely needed especially for the FFT analysis. The pascal code should be transferred to and re-programmed on a workstation in C++ to increase the speed and program ease. Third, a β that automatically changes itself in the algorithm is needed rather than a β that is manually entered. Although this was investigated, the work in this area should be addressed again. More error analysis and investigation is needed to develop an automatically adjusting β. Fourth, work needs to be done on the establishment of boundary conditions on distorted images as well as the optical implementation of the algorithms. Finally, an optical implementation is perhaps the only way of improving the processing speed of the algorithm and achieve real time analysis due to the intensive FFT calculations; otherwise, for anything above a 64 by 64 image a Pixar or workstation is needed to perform the analysis to avoid days of calculations on an IBM PS/2 - 80 with a 25 MHz clock speed. I hope to continue this work

at the MITRE Corporation on a workstation and finally to use the final algorithm on an optical computer being developed at MITRE to retrieve and identify badly distorted images.

APPENDIX

This appendix provides three concrete examples of images that are retrieved and discusses some of the results in the application of this algorithm to single intensity phase retrieval using the Fourier modulus. Here a 32 by 32 pixel image which is real and nonnegative is placed in a 64 by 46 array in a similar fashion as described in section 3.3 and as shown in figure 5. Again in this appendix, the 32 by 32 pixel image is placed in a bed of zeros that is twice the size of the image. Each iteration of the aglorithm for the 64 by 64 array takes about 51 seconds to process on an IBM PS/2 - 80 with a 16 MHz clock of which the foward and inverse Fourier transforms each take 15 seconds. Therefore, the two Fourier transforms take 30 seconds and the other error and algorithm analysis takes about 21 seconds.

There are two known applications of this aglorithm in two dimensions. First, in pupil synthesis, it is possible to retrieve the pupil function of a diffraction limited optical system given a point spread function at a particular point in the image plane. The point spread function is the square of the Fourier modulus of the pupil function and the optical transfer function is the autocorrelation of the pupil function. Alternatively, an optical system could be designed to synthesize a pupil function that would produce a desired point spread function. Second, in astronomy, atmospheric turbulence causes a distortion in the resolution of the image but it is possible to measure the Fourier modulus of the image out to the diffraction limits of the telescope using interferometer data. Using the data on the modulus of the Fourier transform of the image, F(u) can be retrieved as well as the object. Furthermore, the autocorrelation function can be computed from F(u) allowing the diameter of the object to be determined.

The first image is a simple optical aperture consisting of a white rectangle inside a black rectangle which is in turn inside a gray rectangle. The image is totally distorted by a random phase. Figures 1 to 15 shows the image before and after entering the final algorithm. As in Figure 4 in section 3.2, the original image is Fourier transformed, a random phase is added in the Fourier-domain and the image is then inverse Fourier transformed to get an initial esitmate. The final aglorithm continues until a solution is reached in the final iteration. In this case, the distortion prior to entering the iterative portion of algorithm was total. Notice that it took many extra iterations to retrieve the last few pixels of the image and that the original image was retrieved and not the mirrored image.

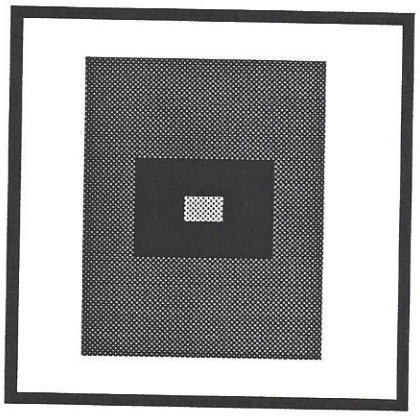

Figure 1 - The original image, f(x).

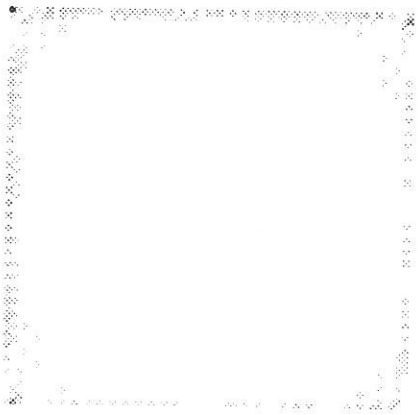

Figure 2 - The Fourier transform of the orginal image, F(u).

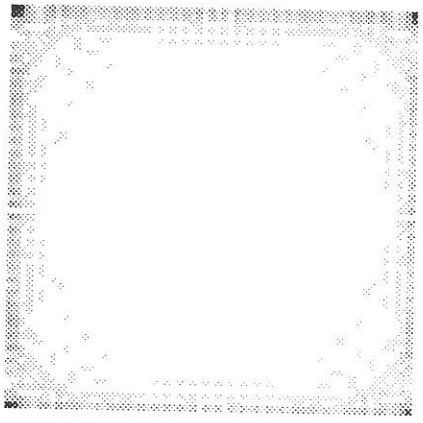

Figure 3 - The modulus of the Fourier transform of the original image, |F(u)| .

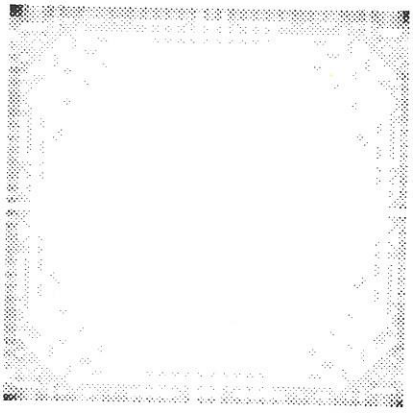

Figure 4 - The modulus of the image after the random phase is added to |F(u)|.

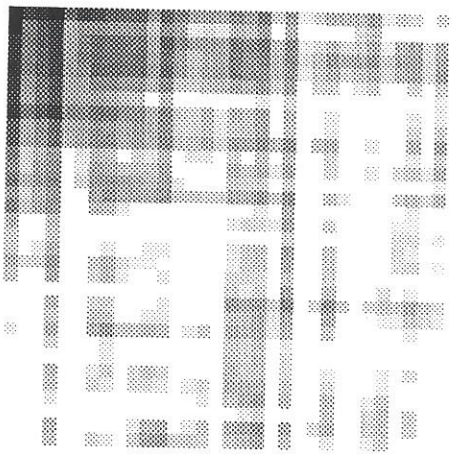

Figure 5 - The initial estimate of the original image.

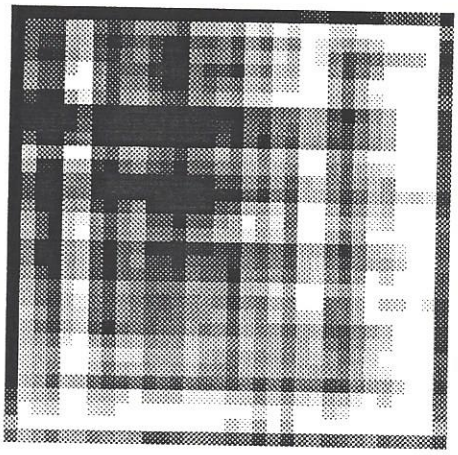

Figure 6 - The retrieved image after 5 iterations.

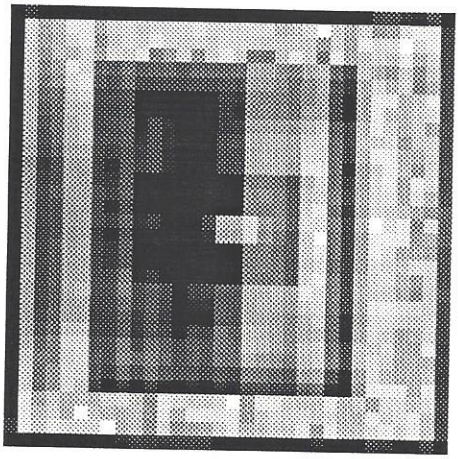

Figure 7 - The retrieved image after 20 iterations.

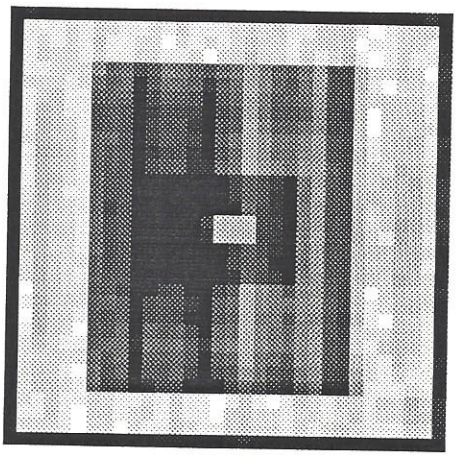

Figure 8 - The retrieved image after 40 iterations.

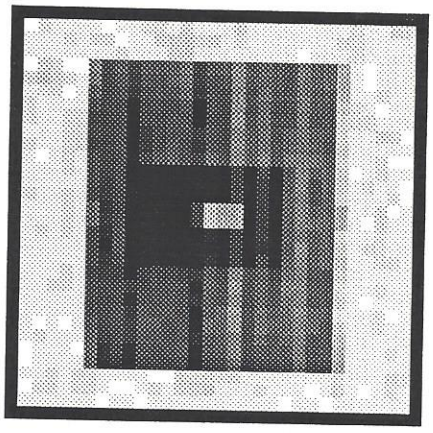

Figure 9 - The retrieved image after 100 iterations.

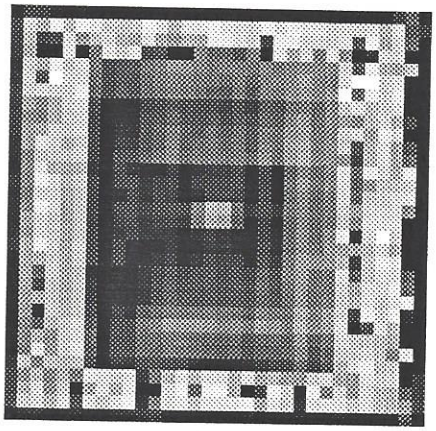

Figure 10 - The retrieved image after 226 iterations.

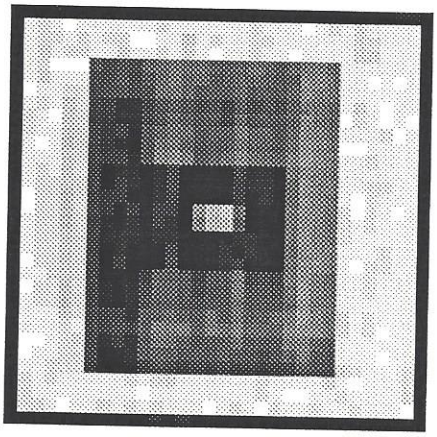

Figure 11 - The retrieved image after 250 iterations.

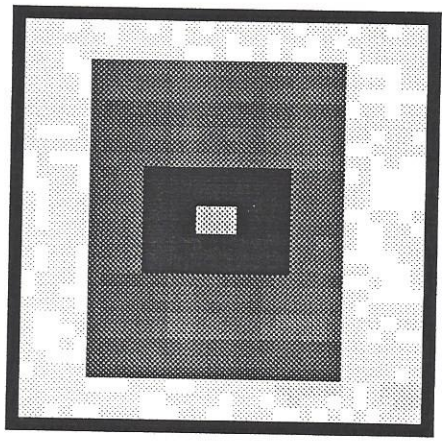

Figure 12 - The retrieved image after 500 iterations.

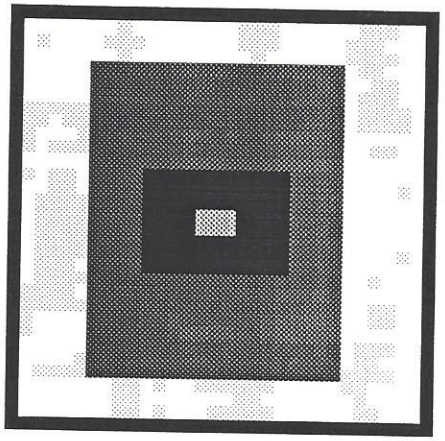

Figure 13 - The retrieved image after 689 iterations.

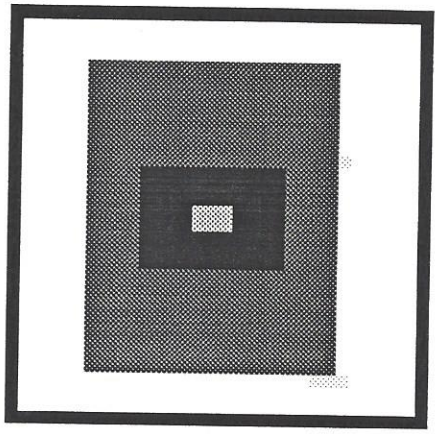

Figure 14 - The retrieved image after 773 iterations.

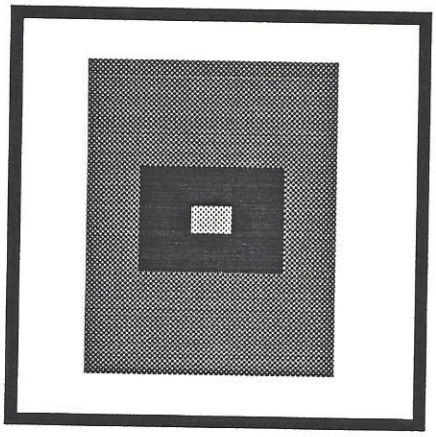

Figure 15 - The retrieved image after 895 iterations (the final iteration).

The second image is the same optical aperture as the first image (shown in figure 1 of this appendix) but this time the initial estimate is closer to the original image. The same Fourier transform and modulus of the Fourier transform from the first image are used as shown in figures 2 and 3 of this appendix respectively. Figures 16 to 22 show the image before and after entering the final algorithm. In this case, the final algorithm takes fewer iterations to retrieve the image because the initial estimate has much less distortion and was close to the original image. Moreover, it only took five iterations of the algorithm for a fairly good quality image to appear as opposed to the 250 iterations of the algorithm it took for the first image to achieve about the same quality (shown in figure 11 of this appendix). However, in the second image it still took many extra iterations to retrieve the last few pixels of the image.

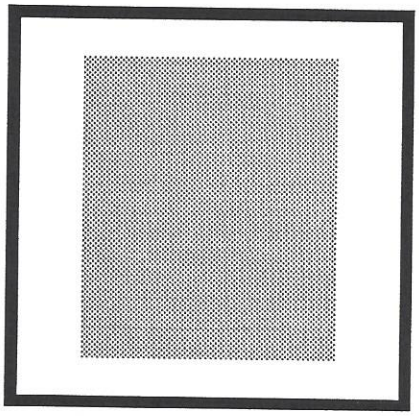

Figure 16 - The initial estimate of the original image.

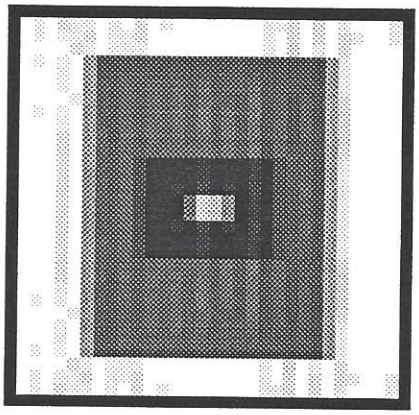

Figure 17 - The retrieved image after 5 iterations.

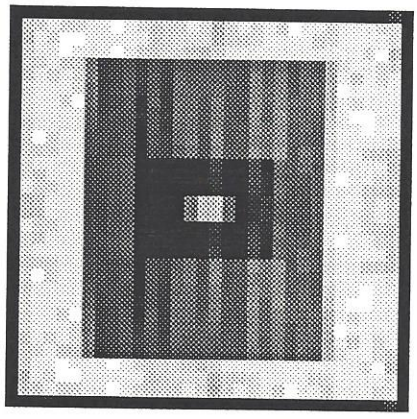

Figure 18 - The retrieved image after 20 iterations.

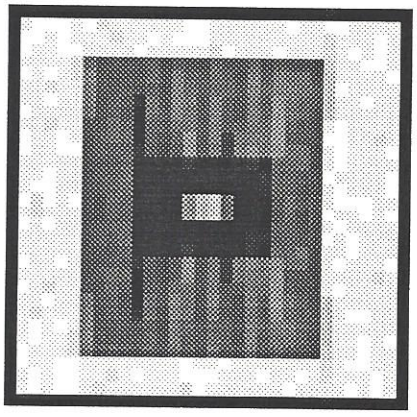

Figure 19 - The retrieved image after 100 iterations.

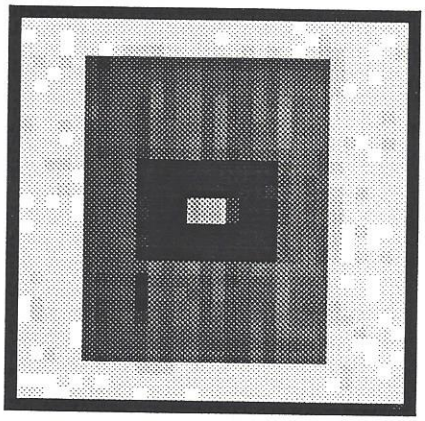

Figure 20 - The retrieved image after 250 iterations.

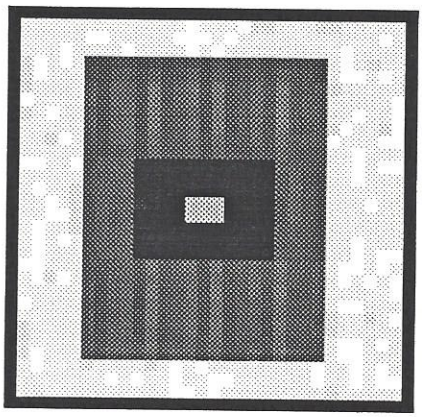

Figure 21 - The retrieved image after 358 iterations.

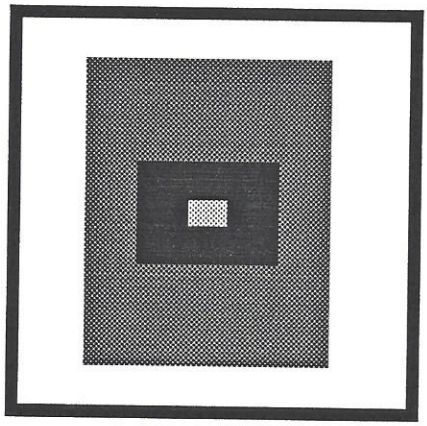

Figure 22 - The retreived image after 523 iterations (the final iteration).

The third image is a more complicated image, a fighter. Figures 23 to 35 show the image before and after entering the final algorithm. It took many more iterations to

retrieve the fighter. In this case, the distortion prior to entering the iterative portion of the algorithm was total as in the first image. Figure 26 is added to show that when the inverse Fourier transform of |F(u)| is taken, one does not get the original image. Notice that the mirrored image of the original image was retrieved.

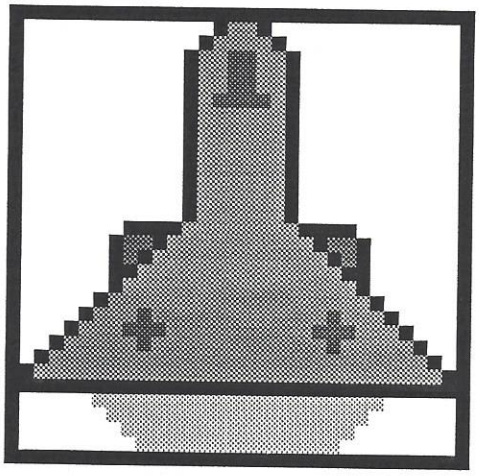

Figure 23 - The original image, f(x).

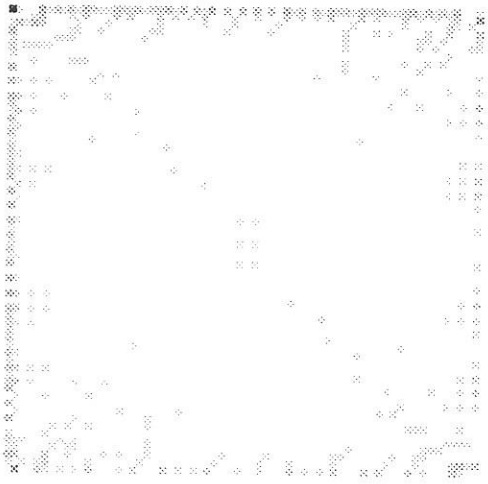

Figure 24 - The Fourier transform of the original image, F(u).

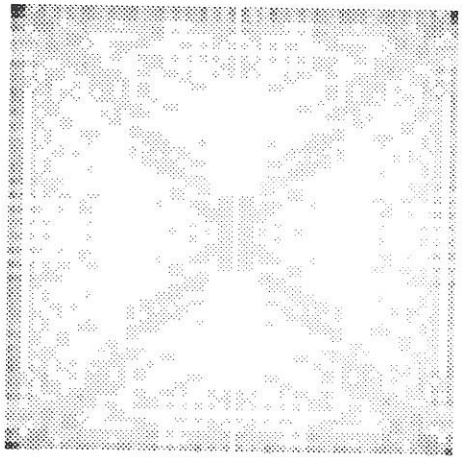

Figure 25 - The modulus of the Fourier transform of the original image, |F(u)|.

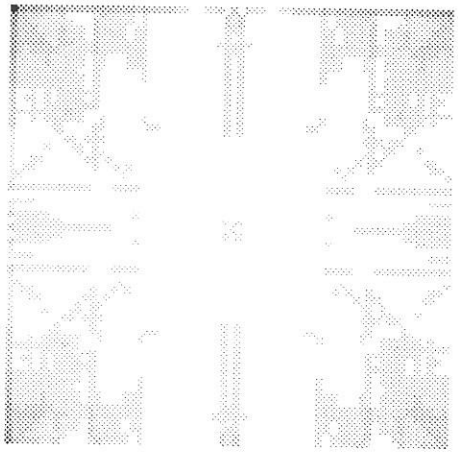

Figure 26 - The inverse Fourier transform of |F(u)|.

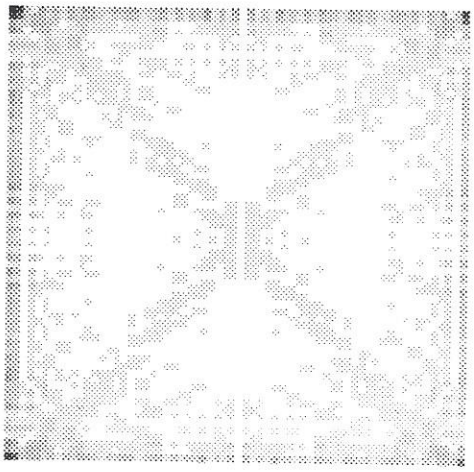

Figure 27 - The modulus of the image after the random phase is added to |F(u)|.

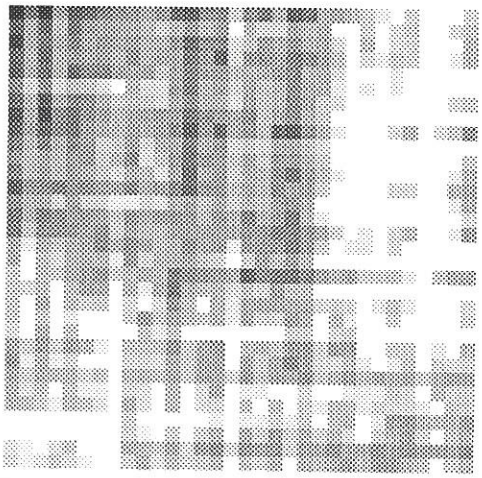

Figure 28 - The initial estimate of the original image.

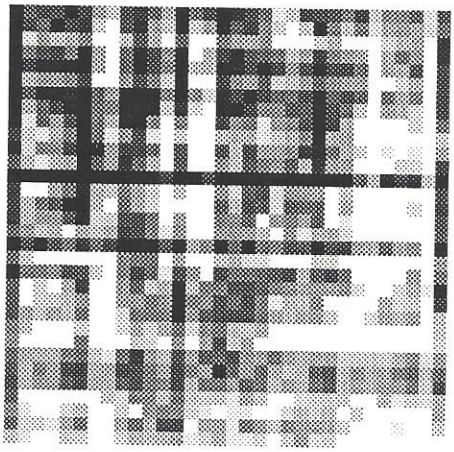

Figure 29 - The retrieved image after 5 iterations.

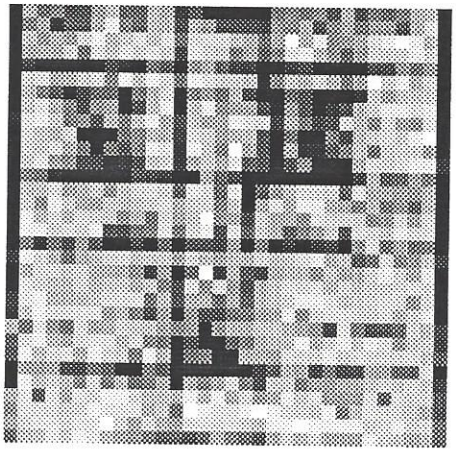

Figure 30 - The retrieved image after 40 iteratrions.

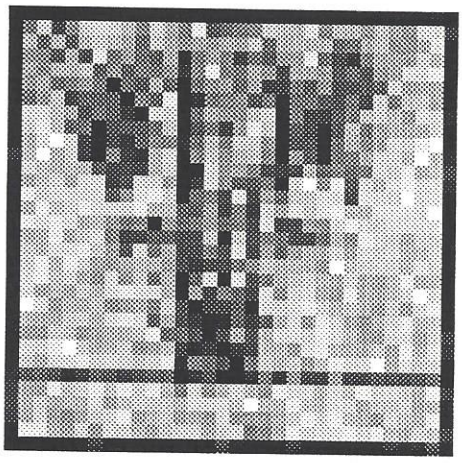

Figure 31 - The retrieved image after 100 iterations.

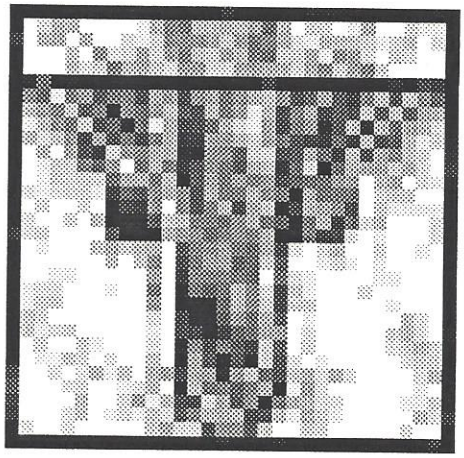

Figure 32 - The retrieved image after 288 iterations.

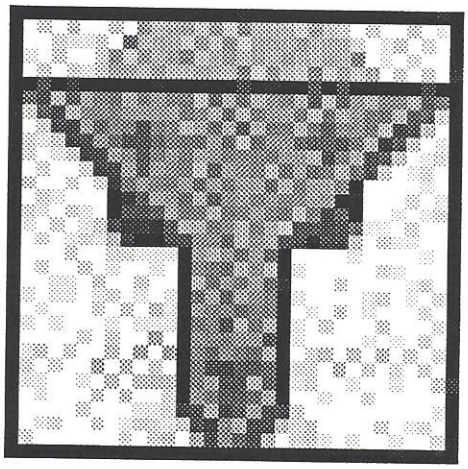

Figure 33 - The retrieved image after 656 iterations.

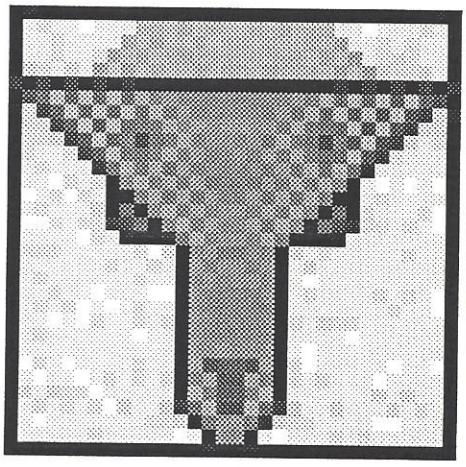

Figure 34 - The retreived image after 1000 iterations.

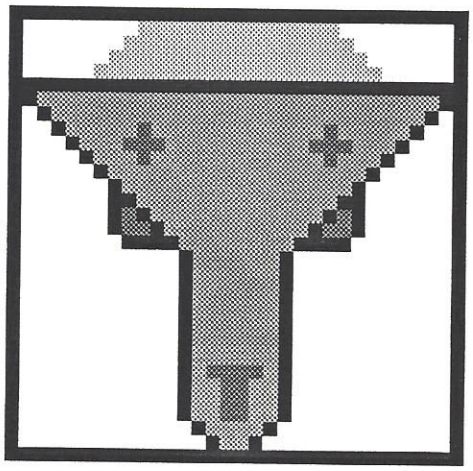

Figure 35 - The retrieved image after 1342 iterations (the final iteration).

It is clear that if the intial estimate is close to the original image, it will take fewer iterations to retrieve the image. If there is less phase distortion, then the image will converge faster. However, it seems to take many extra iterations to completely retrieve the image in full detail even after the retrieved image is close to the original image. Finally, choosing the correct β for the hybrid input-output algorithm can clearly accelerate or delay the retrieval process. Other attempts at retrieving the same two images can add or subtract from the number of iterations given the same distortion or the same initial estimate if a different β is used during different iterations.

LIST OF REFERENCES

1. R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik*, Vol. 35, No. 2, 1972, p.237-246.

2. W. O. Saxton, *Computer Techniques for Image Processing in Electron Microscopy*, NewYork: Academic, 1978.

3. R. A. Gonsalves, "Phase Retrieval from modulus data," *Journal of the Optical Society of America* , Vol. 66, No. 9, September 1976, p.961-964.

4. Adrain Walther, "The question of phase retrieval in optics," *Optic Acta*, Volume 10, 1963, p.41-49.

5. D. Kohler and L. Mandel, "Source reconstruction from the modulus of the correlation function: a practical approach to the phase problem of optical coherence theory," *Journal of the Optical Society of America*, Volume 63, No. 2, February 1973, p.126-134.

6. J. W. Goodman and A. M. Silvestri, "Some effects of Fourier-domain phase quantization," *IBM Journal of Research and Development*, Volume 14, No. 5, September 1970, p.478-484.

7. J. R. Fienup, "Phase retrieval algorithms: a comparison," *Applied Optics*, Volume 21, No. 15, August 1982, p.2758-2769.

8. J. R. Fienup, "Iterative method applied to image reconstruction and to computer-generated holograms," *Optical Engineering*, Volume 19, No. 3, May/June 1980, p.297-305.

9. J. R. Fienup, T. R. Crimmins, and W. Holsztynski, "Reconstruction of the support of an object from the support of its autocorrelation," *Journal of the Optical Society of America*, Volume 72, No. 5, May 1982, p.610-624.

10. J. R. Fienup, "Reconstruction of an object from the modulus of its Fourier transform," *Optics Letters*, Volume 3, No. 1, July 1978, p.27-29.

11. J. R. Feinup, "Space object imaging through the trubulent atmosphere," *Optical Engineering*, Volume 18, No.5, September-October 1979, p.529-534.

12. G. B. Feldkamp and J. R. Feinup, "Noise properties of images reconstructed from Fourier modulus," in *Proceedings of 1980 International Optical Computing Conference*, SPIE, Volume 231, 1980, p. 84.

13. J. R. Feinup and et. al., see *Advanced Institute on Transformations in Optical Signal Processing* (1981: Seattle, Washington), "Proceedings of the SPIE Advanced Institute on Transformations in Optical Signal Processing," Bellingham Washington, SPIE (International Society for), v.373.

14. J. R. Feinup, "Reconstruction and synthesis applications of an iterative algorithm," in *Transformations in Optical Signal Processing*, W. T. Rhodes, J. R. Fienup and B. E. A. Saleh, Eds., v.373, Society of Photo-Optical Instrumentation Engineers: Bellingham, Washington, 1983, p.147-160.

15. E. O. Brigham, *The Fast Fourier Transform*, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1974, p.64.

16. N. C. Gallagher and B. Lui, "Convergence of a spectrum shaping algorithm," *Applied Optics*, Volume 13, No. 11, November 1974, p.2470-2471.

17. Monson H. Hayes, "The reconstruction of a multidimensional sequence from the phase or Magnitude of its Fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume ASSP-30, No. 2, April 1982, p.140-154.

18. J. R. Feinup, Comments on "The reconstruction of a multidimensional sequence from the phase or Magnitude of its Fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume ASSP-31, No. 3, June 1983, p.738-739.

19. James W. Cooley and John W. Tukey, "An algorithm for the machine calculation of a complex Fourier series," *Mathematics of Computations*, Volume 19, No. 90, April 1965, p.297-301.

20. Monson H. Hayes and Thomas F. Quatieri, "Recusive phase retrieval using boundary conditions," *Journal of the Optical Society of America*, Volume 73, No. 11, November 1983, p.1427-1433.

21. R. G. Paxman and J. R. Fienup, "Optical misalignment sensing and image reconstruction using phase diversity," *Journal of the Optical Society of America*, Volume 5, No. 6, June 1988, p.914-923.