Non-Iterative Phase Retrieval from Magnitude of Complex-Valued Compact-Support Images

Andrew E. Yagle
Department of EECS, The University of Michigan, Ann Arbor, MI 48109-2122

Abstract— **The discrete phase retrieval problem is to reconstruct an unknown image having finite spatial extent from the magnitude of its discrete Fourier transform. Most methods for this problem take thousands of iterations to converge. We present a new non-iterative algorithm for M×M complex-valued images requiring: (1) rooting 5.2M polynomials of degree 2M; and (2) solution of a linear system of equations of size $5.2M^2$ whose solution elements are all either +1 or –1. A small example illustrate the new algorithm.**

Keywords— **Phase retrieval of complex images**
Phone: 734-763-9810. Fax: 734-763-1503.
Email: aey@eecs.umich.edu. EDICS: 2-REST.

I. INTRODUCTION

The problem of reconstructing an image known to have compact support from its Fourier transform magnitudes arises in several disciplines in optics. The image is reconstructed once the missing Fourier phase is recovered; hence the term "phase retrieval."

Since the image is assumed to have compact support, its Fourier transform may be sampled in wavenumber. Most images are approximately bandlimited to the extent that they may also be sampled spatially as well. This leads to the discrete version of this problem, in which a discrete-space image known to have finite spatial extent is to be reconstructed from the magnitude of its 2-D discrete Fourier transform (DFT). Note that this precludes methods based on oversampling of continuous images.

The algorithm most commonly used for phase retrieval is the hybrid input-output algorithm of Fienup. However, this algorithm usually requires thousands of iterations before it converges, and has some difficulty with complex-valued images unless the support is known exactly. Other methods require tracking zero curves or sheets of algebraic functions. We will make no attempt to survey the literature.

II. Problem Formulation

A. Problem Statement

We observe the autocorrelation of the image

$$y(i_1,i_2) = u(i_1,i_2) ** u(-i_1,-i_2)^* \qquad (1)$$

where $**$ denotes convolutions in both i_1 and i_2, and the 1-D convolution $*$ of causal signals is defined as

$$h(n)*u(n) = \sum_{i=0}^{n} h(n-i)u(i) = \sum_{i=0}^{n} h(i)u(n-i). \quad (2)$$

We omit commas between pairs of subscripts.

We make the following assumptions:

- *Support:* Image $u(i_1,i_2) = 0$ unless $0 \le i_1, i_2 \le$M;
- Autocorrelation $y(i_1,i_2)$=0 unless $0 \le |i_1|, |i_2| \le$M;
- $u(i_1,i_2)$ and $y(i_1,i_2)$ need **not** be real-valued.

The goal is to reconstruct the image $u(i_1,i_2)$ from its autocorrelation $y(i_1,i_2)$. Taking a 2-D discrete Fourier transform (DFT) of (1) shows that this is equivalent to computing the DFT $U(k_1,k_2)$ from its squared magnitude $Y(k_1,k_2)$=$|U(k_1,k_2)|^2$; hence the term "phase retrieval." No stochastic assumptions are made about $u(i_1,i_2)$, precluding methods using cumulants, ARMA or Poisson image models.

B. Problem Ambiguities

There are four ambiguities in 2-D phase retrieval:

1. *Scale:* If $u(i_1,i_2)$ is a solution, then $cu(i_1,i_2)$ is also a solution for any constant c such that $|c| = 1$;
2. *Translation:* If $u(i_1,i_2)$ is a solution, then the shifted $u(i_1+d_1, i_2+d_2)$ is also a solution for any integers (d_1,d_2). We eliminate this ambiguity by specifying the support of $u(i_1,i_2)$ to be $0 \le i_1, i_2 \le$M;
3. *Reversal:* Either $u(i_1,i_2)$ or its conjugate reversal $u(M-i_1, M-i_2)^*$ can regarded as the "true" image;
4. *Irreducibility:* The 2-D z-transform $U(x,y)$ of $u(i_1,i_2)$ is irreducible (i.e., cannot be factored). Here

$$U(x,y) = \sum_{i_1=0}^{M} \sum_{i_2=0}^{M} u(i_1,i_2) x^{-i_1} y^{-i_2} \qquad (3)$$

The 1-D phase retrieval problem has an additional ambiguity. There are almost surely 2^M solutions to an (M+1)–point 1-D phase retrieval problem if M is even. This can be seen by noting that the zeros of the 1-D z-transform of the autocorrelation occur in reciprocal pairs (if x_o is a zero, then $1/x_o^*$ is also). Either x_o or $1/x_o^*$ can be assigned to the 1-D signal; since there are M reciprocal pairs, this assignment can be made in 2^M ways. The choice disappears if x_0=$1/x_0^*$, that is, if $|x_0|$=1 (see the next section).

III. NEW ALGORITHM

A. Partition into 1-D Problems

Let $Y(x,y)$ and $U(x,y)$ be the 2-D z-transforms of $y(i_1,i_2)$ and $u(i_1,i_2)$, respectively. Then (1) becomes

$$Y(x,y) = U(x,y)U(1/x^*, 1/y^*)^*. \tag{4}$$

Setting $x = e^{j2\pi k_1/N}$ and $y = e^{j2\pi k_2/N}$ gives

$$Y_{k_1,k_2} = U_{k_1,k_2}U^*_{k_1,k_2} = |U_{k_1,k_2}|^2 \tag{5}$$

where the 2-D DFTs for any $N \geq 2M$ are

$$Y_{k_1,k_2} = Y(e^{j2\pi k_1/N}, e^{j2\pi k_2/N}) \tag{6}$$
$$U_{k_1,k_2} = U(e^{j2\pi k_1/N}, e^{j2\pi k_2/N}) \tag{7}$$

since $x = 1/x^*$ and $y = 1/y^*$ here.

Setting just $y = e^{j2\pi k_2/N}$ in (4) gives

$$Y_{k_2}(x) = U_{k_2}(x)U_{k_2}(1/x^*)^* \tag{8}$$

which is a 1-D phase retrieval problem. Note:

- Let $\{x_o\}$ be the M zeros of $U_{k_2}(x)$;
- Then the 2M zeros of $Y_{k_2}(x)$ are $\{x_o, 1/x_o^*\}$;
- The 1-D phase retrieval problem here is somehow to assign $\{x_o\}$ to $U_{k_2}(x)$ and $\{1/x_o^*\}$ to $U_{k_2}(1/x^*)^*$;
- This can be done in 2^M ways unless $x_o = 1/x_o^*$:

$$U_{k_2}(x) = e^{j\phi_{k_2}} \prod_{i=1}^{M}(x - x_i) \tag{9}$$

$$U_{k_2}(1/x^*)^* = e^{-j\phi_{k_2}} \prod_{i=1}^{M}(1/x - x_i^*) \tag{10}$$

$$U_{k_2}(1/x^*)^* = e^{-j\phi_{k_2}} \prod_{i=1}^{M}(x - 1/x_i^*) \left[\frac{x_i^*}{-x}\right] \tag{11}$$

This can be done for each $k_2 = 0 \ldots N$. Repeating the above with just $x = e^{j2\pi k_1/N}$ in (4) gives the 1-D phase retrieval problems, for each $k_1 = 0 \ldots N$,

$$Y_{k_1}(y) = U_{k_1}(y)U_{k_1}(1/y^*)^*. \tag{12}$$

where $U_{k_1}(y)$ has zeros $\{y_i\}$, so that

$$U_{k_1}(y) = e^{j\phi_{k_1}} \prod_{i=1}^{M}(y - y_i). \tag{13}$$

B. Linking 1-D Problems

These problems are not independent, since

$$U_{k_2}(x) = U_{k_1}(y) \text{ if } x = e^{j2\pi k_1/N}, y = e^{j2\pi k_2/N} \tag{14}$$

The magnitudes are automatically equal. The phases are equal if this congruence is true mod(2π):

$$\phi_{k_2} + \sum_{i=1}^{M} \arg[e^{j2\pi k_1} - x_i] \equiv \phi_{k_1} + \sum_{i=1}^{M} \arg[e^{j2\pi k_2} - y_i] \tag{15}$$

This is N^2 equations in 2MN independent choices:

- M choices: x_i vs. $1/x_i^*$, for each of N k_2;
- M choices: y_i vs. $1/y_i^*$, for each of N k_1;
- 2N additional unknowns $\{\phi_{k_1}, \phi_{k_2}\}$.

At first glance it seems that if $N > 2M$ we can solve this as a linear system of equations with unknowns ± 1. However, the mod(2π) issue is a problem. So:

- (1) Equate phase *differences* instead of phases:
- This eliminates the unknowns $\{\phi_{k_1}, \phi_{k_2}\}$ and
- Keeps the coefficients in (15) small; and also:
- (2) Eliminate equations in (15) for which the sum of the absolute values of the coefficients exceeds 2π.

This requires choosing $N > 2M$. We have found experimentally that choosing N=5.2M seems to give a sufficient number of equations in (15) to solve (15).

C. Problem Ambiguities

The ambiguities of 2-D phase retrieval noted above appear explicitly in this algorithm, as follows:

- The *reversal* ambiguity appears in the null vector of (15): Consisting of ± 1 for each choice of x_i vs. $1/x_i^*$, this can clearly be multiplied by –1, changing all the signs and replacing the image with its reversal;
- The *scale* ambiguity appears in the phase factors $\{e^{j\phi_k}\}$, which are computed using a rank-one factorization only to aan overall phase ambiguity;
- The *translational* ambiguity is eliminated as above by specifying the support constraint $0 \leq i_1, i_2 \leq M$.

IV. NUMERICAL EXAMPLE #1

A. Specifications and Numerical Results

We present a tiny numerical example to illustrate the algorithm. The complex image is 15×15, with real and imaginary parts left and right eyes. Hence:

- M=15 and N=(2.6)(15)=39 (the minimum is 29);

• Each 1-D problem has $\frac{1}{2}(2(15)–2)=14$ reciprocal pairs of roots, one of which must be chosen for each;
• The total number of choices is $2(39)(14)=1092$;
• The number of linear equations is $39^2=1521$;
• 1104 of 1521 had the sum of absolute values of its coefficients less than 2π, so they can be used in (15).

The major computational requirements were:

• Rooting 78 polynomials of degree 28 each;
• Computing the 1092 phases depicted in (15);
• Computing null vector, consisting entirely of ±1, of a 1104×1092 sparse matrix (see below).

This matrix had singular values

$$\sigma_1 = 4.2; \sigma_{1091} = 8 \times 10^{-5}; \sigma_{1092} = 2 \times 10^{-14} \quad (16)$$

so the null vector of the system (15) was well defined.

The structure of the sparse matrix (15) is shown.

B. Matlab Program

```
clear;load lena.mat;
X=xx(126:140,126:140);Z=xx(126:140,159:173);
X=X+j*Z;M=size(X,1);N=ceil(2.6*M);MN=(M-1)*N;
Y=conv2(X,fliplr(flipud(conj(X))));%Same as this:
%Y=fftshift(ifft2((abs(fft2(X,2*M-1,2*M-1))).^2))
E=exp(-j*2*pi*[0:N-1]/N)))+pi*[0:N-1]/N)';
FY1=fft(Y,N);FY2=fft(Y.',N);for I=1:N;
RY1(:,I)=sort(roots(FY1(I,:)));
RY2(:,I)=sort(roots(FY2(I,:)));
end;for I=1:N;for J=1:M-1;
FR1=unwrap(angle((ones(1,N)-RY1(J,I)*E;
A((I-1)*N+1:I*N,(J-1)*N+I)=FR1(1:N);
FR2=unwrap(angle((ones(1,N)-RY2(J,I)*E;
A(I:N:N*N,(J-1)*N+I+MN)=-FR2(1:N);end;end
for I=0:N-1;B([1:N-1]+(N-1)*I,:)=A([2:N]+N*I,:)...
-A([1:N-1]+N*I,:);end;
for I=0:N-2;C([1:N-1]+(N-1)*I,:)=B([1:N-1]+(N-1)*(I+1),:)...
-B([1:N-1]+(N-1)*I,:);end;
C=C';D=C(:,sum(abs(C))<6.28);D=D';
[U S V]=svd(D);X1=V(:,2*MN)/V(1,2*MN);
%Replace with: X1=D(:,1:2*MN-1)\D(:,2*MN);
P1=A(:,1:MN)*X1(1:MN);
Q1=reshape(P1,N,N)-pi*(M-1)*[0:N-1]'*ones(1,N)/N;
P2=A(:,MN+1:2*MN)*X1(MN+1:2*MN);
Q2=-reshape(P2,N,N)-pi*(M-1)*ones(N,1)*[0:N-1]/N;
Q3=exp(j*Q1)./exp(j*Q2);[U1 S1 V1]=svd(Q3);
P=reshape(P1,N,N)-pi*(M-1)*[0:N-1]'*ones(1,N)/N...
+ones(N,1)*(angle(V1(:,1)))';
EST=ifft2(abs(fft2(X,N,N)).*exp(j*P'));
EST=EST(1:M,1:M)/EST(1,1)*X(1,1);
subplot(221),imagesc(real(X)),colormap(gray),axis off
subplot(222),imagesc(real(EST)),colormap(gray),axis off
subplot(223),imagesc(imag(X)),colormap(gray),axis off
subplot(224),imagesc(imag(EST)),colormap(gray),axis off
```

V. NUMERICAL EXAMPLE #2

For entirely real-valued images, the following simplifications occur in the above algorithm:

• The roots of $U_{k_2}(x)$ are complex conjugates of the roots of $U_{N-k_2}(x)$, and similarly for $U_{k_1}(y)$. Hence only half as many polynomials need to be rooted;
• The *choices* of roots for $U_{k_2}(x)$ and $U_{N-k_2}(x)$ are also identical, so the number of unknowns ±1 of the linear system (15) is roughly halved;
• By conjugate symmetry of $U(k_1, k_2)$ and $Y(k_1, k_2)$, the number of equations is roughly halved.

A. Specifications and Numerical Results

This time the real-valued image is 25×25. So:

• M=25 and N=2.6(25)=65 (the minimum is 49);
• Each 1-D problem has $\frac{1}{2}(2(25)–2)=24$ choices;
• The total number of choices is $2(24)(65+1)/2=1584$;
• 1585(!) rows had the sum of absolute values of its coefficients less than 2π, so they can be used in (15).

The major computational requirements were:

• Rooting 130 polynomials of degree 48 each;
• Computing the 1584 phases depicted in (15);
• Computing null vector, consisting entirely of ±1, of a 1585×1584 sparse matrix (see below).

The structure of the sparse matrix (15) is shown.

B. Matlab Program

```
clear;load lena.mat;%Need M and N both odd.
X=xx(121:145,121:145);M=size(X,1);
N=ceil(2.6*M);N2=(N+1)/2;MN=(M-1)*N2;
Y=conv2(X,fliplr(flipud(X)));
E=exp(-j*2*pi*[0:N-1]/N)))+pi*[0:N-1]/N)';
FY1=fft(Y,N);FY2=fft(Y.',N);for I=1:N2;
RY1(:,I)=sort(roots(FY1(I,:)));
RY2(:,I)=sort(roots(FY2(I,:)));
end;for I=1:N2;for J=1:M-1;
FR1=unwrap(angle((ones(1,N)-RY1(J,I)*E;
A((I-1)*N+1:I*N,(J-1)*N2+I)=FR1(1:N);
FR2=unwrap(angle((ones(1,N)-RY2(J,I)*E;
A(I:N:N*N2,(J-1)*N2+I+MN)=-FR2(1:N2);if(I~=1);
FR3=unwrap(angle((ones(1,N)-conj(RY2(J,I))*E;
A(N2-I+N2+1:N:N*N2,(J-1)*N2+I+MN)=-FR3(1:N2);
end;end;end;for I=0:N2-1;
B([1:N-1]+(N-1)*I,:)=A([2:N]+N*I,:)-A([1:N-1]+N*I,:);end;
for I=0:N2-2;C([1:N-1]+(N-1)*I,:)=...
B([1:N-1]+(N-1)*(I+1),:)-B([1:N-1]+(N-1)*I,:);end;
C=C';D=C(:,sum(abs(C))<6.28);D=D';
X1=D(:,1:2*MN-1)\D(:,2*MN);X1=[X1;-1];
P1=A(:,1:MN)*X1(1:MN);
Q1=reshape(P1,N,N2)-pi*(M-1)*[0:N-1]'*ones(1,N2)/N;
P2=A(:,MN+1:2*MN)*X1(MN+1:2*MN);
Q2=-reshape(P2,N,N2)-pi*(M-1)*ones(N,1)*[0:N2-1]/N;
Q3=exp(j*Q1)./exp(j*Q2);[U1 S1 V1]=svd(Q3);%rank=1.
P=reshape(P1,N,N2)-pi*(M-1)*[0:N-1]'*...
ones(1,N2)/N+ones(N,1)*(angle(V1(:,1)))';
P=[P(:,1:N2)-[fliplr(P(1,2:N2));flipud(P(2:N,N2:-1:2))]];
EST=ifft2(abs(fft2(X,N,N)).*exp(j*P'));
EST=EST(1:M,1:M)/EST(1,1)*X(1,1);
subplot(221),imagesc(real(X)),colormap(gray),axis off
subplot(222),imagesc(real(EST)),colormap(gray),axis off
```

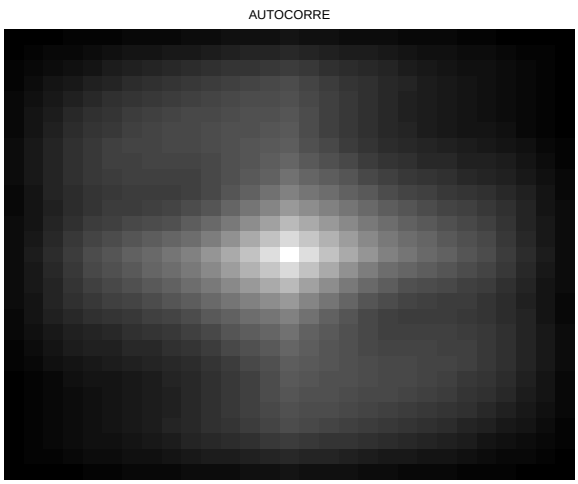

Fig. 1. Autocorrelation data for Example #1.

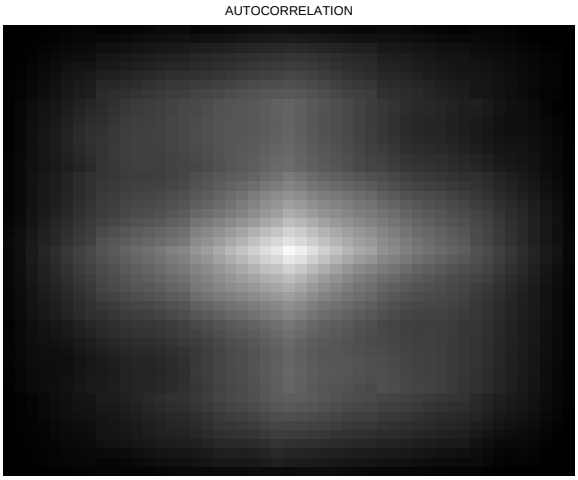

Fig. 4. Autocorrrelation data for Example #2.

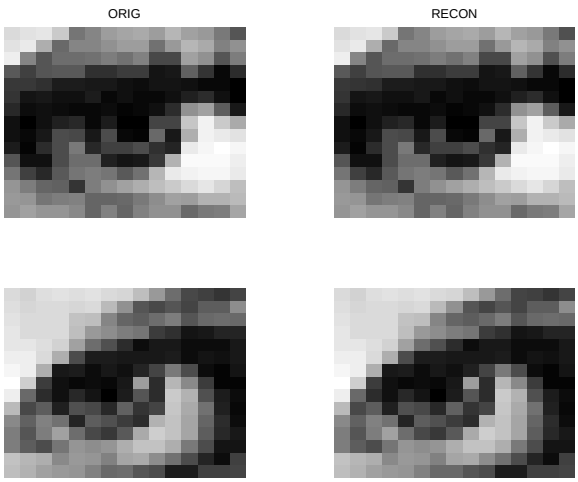

Fig. 2. Original and reconstructed images for Example #1.

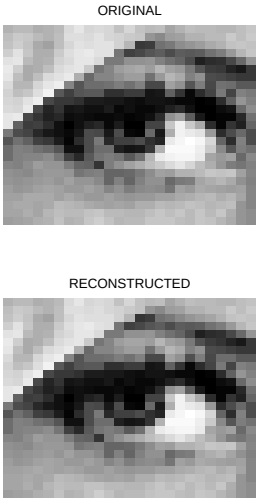

Fig. 5. Original and reconstructed images for Example #2.

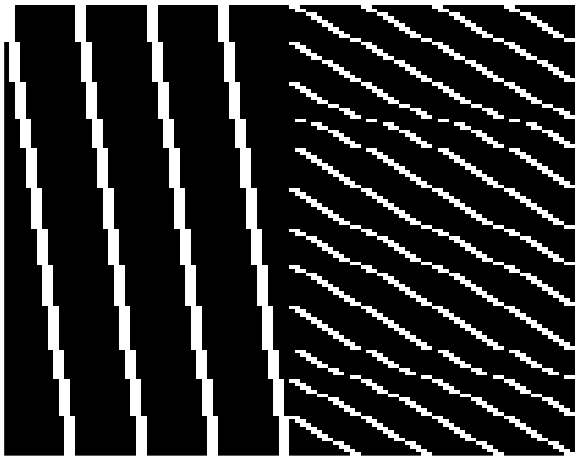

Fig. 3. Matrix sparsity structure for Example #1.

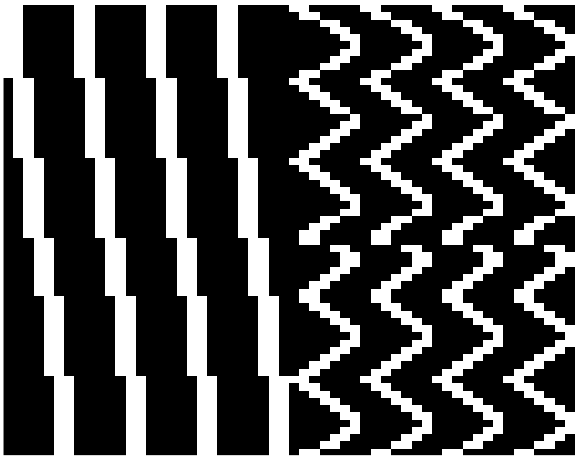

Fig. 6. Matrix sparsity structure for Example #2.