Table of Contents

Description

```
%title: GS Algorithm
%desc: an IFTA to determine phase distribution in CGH for beam shaping
%author: Muhammad Syahman Samhan
%email: mssamhan@students.itb.ac.id
%last update: November 27, 2019
%adapted from Gerchberg and Saxton (1971)
%developed from
    %1. Musa Aydin (Sultan Mehmet Vakif University) and
    %2. Dae Gwang Choi (KAIST, 2019)
```

General Variables

```
clc
clear all
close all
pixel = 2048;               %pixel size
sizex = 0.432;              % for specific SLM (HoloEye)
sizey = 0.432;              %
x = linspace(-sizex, sizex, pixel);
y = linspace(-sizey, sizey, pixel);
[X,Y] = meshgrid(x,y);  %meshgrid for Input Beam and SLM
x0=0; y0=0;             %center of CGH and Input Beam
i_num = 30;             %number of iteration
error = [];             %error array is empty for first
```

Generate Gaussian Input Beam

```
sigma = 0.74;                                       %beam waist size
 (depends on experiment laser)
A = 1;                                              %input amplitude
theta = ((X-x0).^2 + (Y-y0).^2)./(2*sigma^2);   %phase of Input beam
input = A*exp(-theta);                              %input amplitude for a
 given position
```

Target Image

```
Target_Ori = rgb2gray(imread('square.bmp'));  %specify input
Target = im2double(Target_Ori);              %changing the target into
 matrix of doubles with precision
A = fftshift(ifft2(fftshift(Target)));       %performing GS
 initialization step, IFFT
% Notes: 2 times fftshift is used to shift the matrix (q1<->q3, q2<-
>q4)
SR_Ori = rgb2gray(imread('SR.bmp'));         %signal region input
threshold = 0.5;
SR = (double(SR_Ori) >= threshold);          %change signal region into
 matrix of 1 and 0
NR = 1-SR;                                   %noise region = outside of
 signal region
MR = SR;                                     %define measure region as
 the same with signal region
N_m = nnz(MR);
Target_m = MR.*Target;
```

Perform Gerchberg - Saxton (GS) Algorithm (check wikipedia for pseudo code)

```
for i=1:i_num
    B = abs(input).*exp(1i*angle(A));
    C = fftshift(fft2(fftshift(B)));
    D = abs(Target).*exp(1i*angle(C));
    A = fftshift(ifft2(fftshift(D)));
    %Error Calculation: Root Mean Squre Error of Measure Region
        C_m = MR.*C;
        error_cur = sqrt((1/N_m)*sum(sum(((abs(C_m)./
max(max(abs(C_m)))) - abs(Target_m)./max(max(abs(Target_m)))).^2 ) ));
        error = [error; error_cur];
end
```

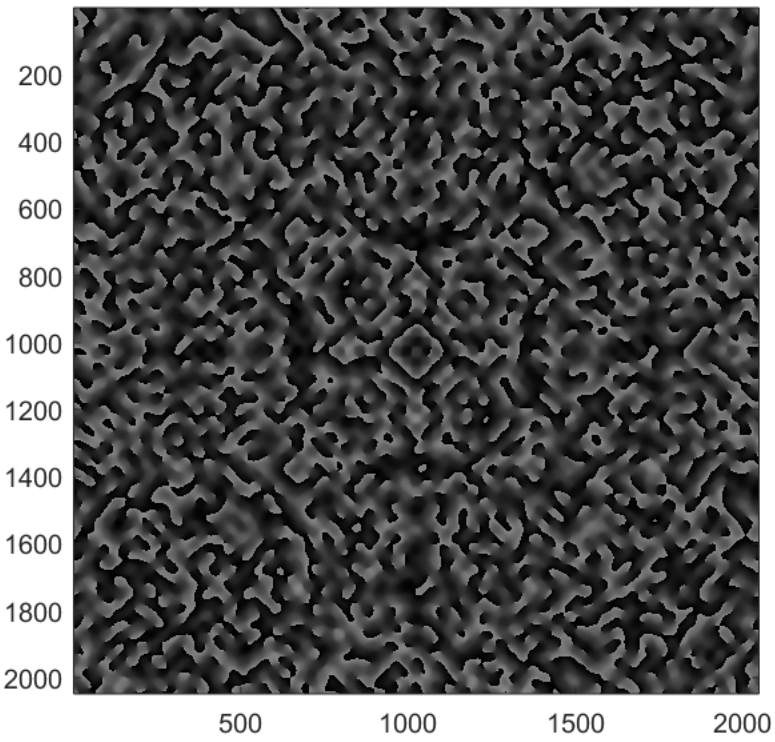
Calibration Section by Dae Gwang

```
Crop=angle(A)+pi;   % change range, from -pi to pi into 0 to 2pi
aaaa=136;           % maximum gray level for 2pi phase shift (depends
 on SLM)
Crop=floor(aaaa.*Crop./(2.0*pi)); % converting the phase shift into
 gray level

% This line changes the black into the white
for j=1:pixel
    for k=1:pixel
        if Crop(j,k) == 0;
            Crop(j,k)=aaaa;
        end
    end
```

```
end

% This line is the main calibration section, which we got after
 experiment
for qi=1:pixel
    for qj=1:pixel
        k= Crop(qi,qj);

 Crop(qi,qj)=round(0.863*k-0.00596*(k)^2+0.00008035*k^3-0.000000238*k^4+0.00000000
    end
end
```

Figure of Calibrated CGH

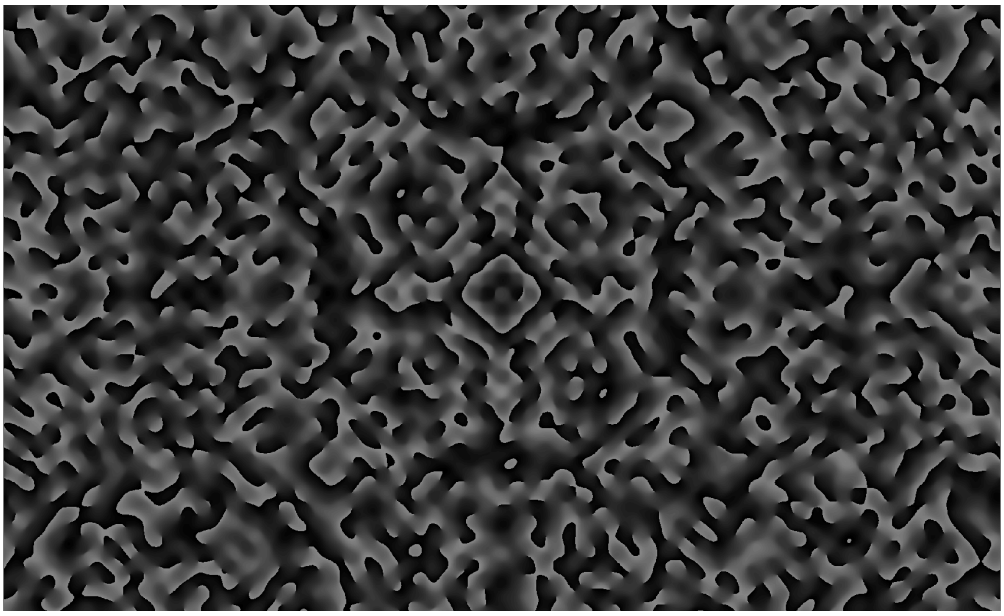
```
figure()
Crop=Crop';
imagesc(Crop), axis image ,colormap('gray'), caxis([0,255]);
```



Setting the position, magnification, etc. of the CGH

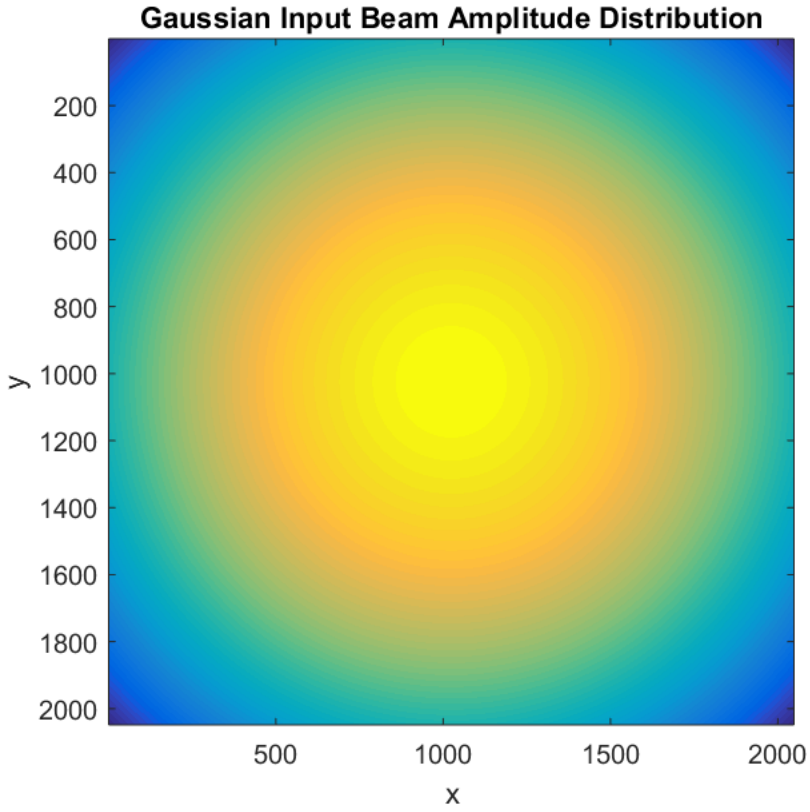
```
magni=1.65;

set(gcf, 'Units', 'Normalized', 'OuterPosition', [1 0.00 1 1]);
```

```
set(gca, 'Units', 'normalized','Position',[0. -0.3 1 magni]);
set(gcf, 'Toolbar', 'none', 'Menu', 'none','menubar','none','NumberTitle','off');
set(gca,'Yticklabel',[],'Xticklabel',[],'ytick',[],'xtick',[])
```

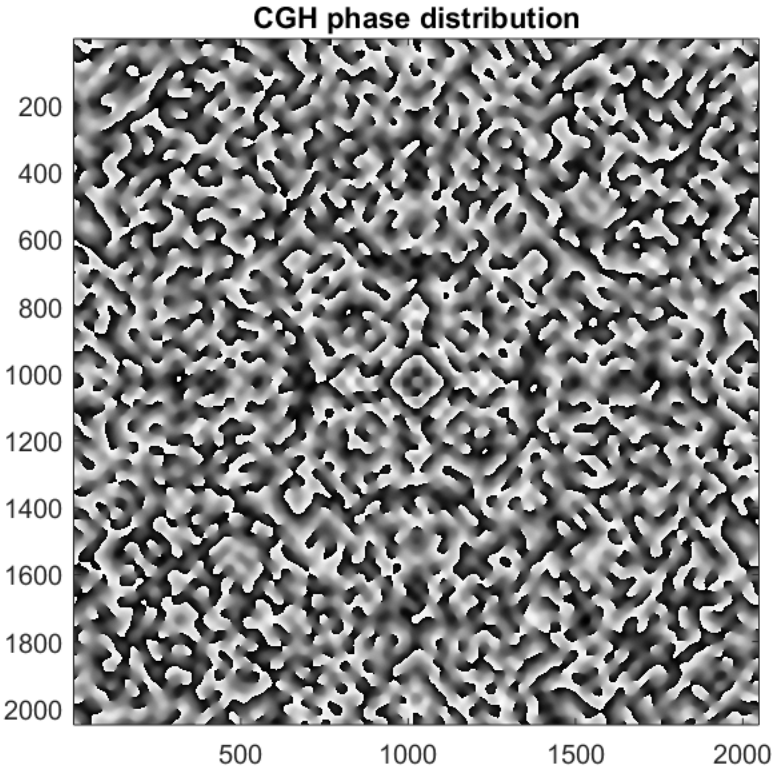


Show Result

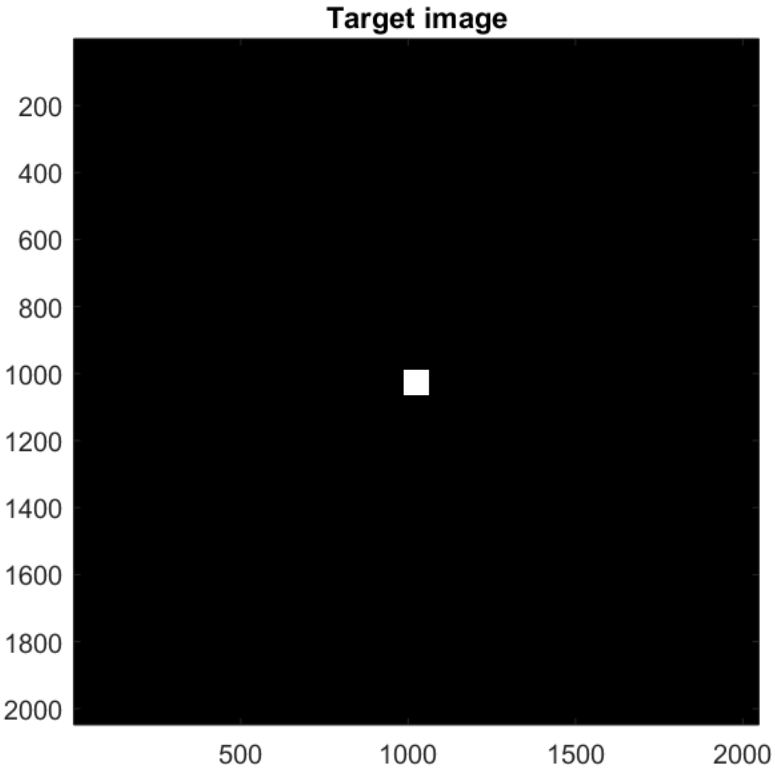
```
figure %Input Beam Distribution
    imagesc(input), axis image;
    title('Gaussian Input Beam Amplitude Distribution')
    xlabel('x')
    ylabel('y')
```

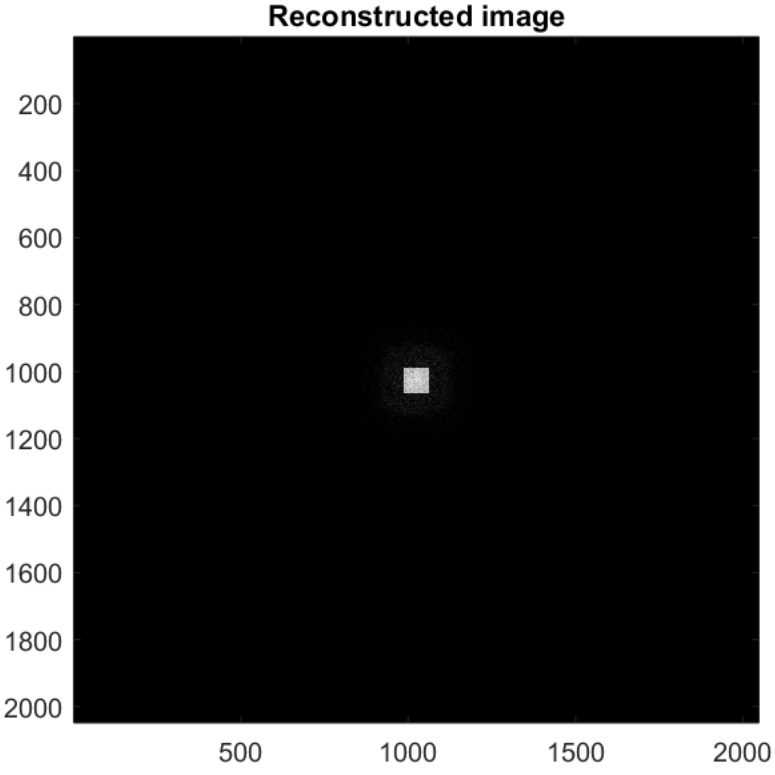



```
figure %CGH Phase Distribution Result
    imagesc(angle(A)), axis image, colormap('gray');
    title('CGH phase distribution');
```

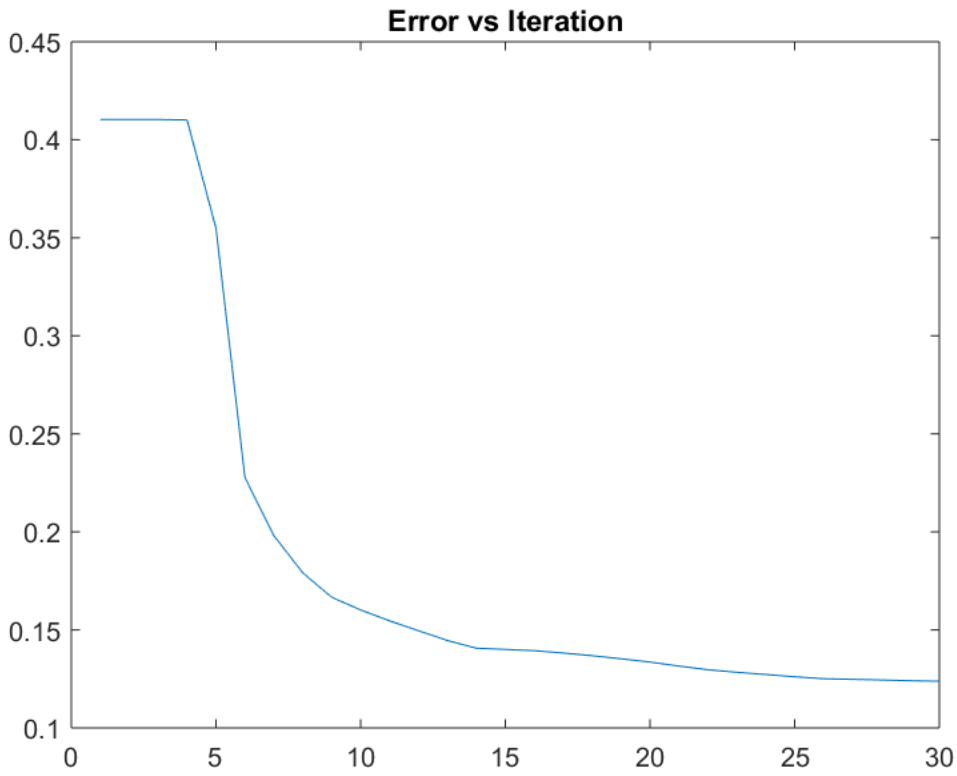



```
figure %Original image
    imagesc(Target./max(max(Target))),axis image, colormap('gray');
    title('Target image')

figure %Reconstructed image
    imagesc(abs(C)./max(max(abs(C)))), axis image, colormap('gray');
    title('Reconstructed image');
```


Target image
200
400
600
800
1000
1200
1400
1600
1800
2000
500
1000
1500
2000


Reconstructed image
200
400
600
800
1000
1200
1400
1600
1800
2000
500
1000
1500
2000

```
figure %Error vs iteration
    i = 1:1:i;
    plot(i,(error'));
    title('Error vs Iteration');
```



Published with MATLAB® R2016b