Spatial Light Modulator (SLM) Pattern Generator

Harsh Jain, PhD
National Center for Biological Sciences, TIFR, Bengaluru

March 26, 2025

Contents

1 Abstract

This Technology/Solution focuses on wavefront shaping using Fourier optics to achieve tailored laser beam patterns in a microscope. It involves generating specific phase distributions for Spatial Light Modulators (SLMs) to manipulate laser wavefronts effectively. MATLAB and Python scripts are presented that compute the phase distributions based on computational technique called the G S algorithm, enabling precise beam shaping. Additionally, manufacturer-provided correction patterns can be incorporated to enhance accuracy. These codes are designed for phase modulation using spatial light modulators such as the Hamamatsu LCOS - SLM. To get a target pattern such as an optical tweezers trap, the projection pattern for the SLM needs to be back calculated using algorithms such as GS Algorithm. Here, we present MATLAB and PYTHON codes for calculating and projecting such patterns for an SLM that is recognized as an extended display by the Operating System. The technology is applicable in Biology, Physics and Chemistry experiments.

2 MATLAB Script: Binary SLM Phase Generator

The MATLAB script below implements an algorithm to generate phase patterns for binary SLMs using the Gerchberg-Saxton algorithm. It initializes an image, generates a target pattern, computes a 256-bit phase distribution, and then converts it into a binary pattern for the SLM. For the code below, a binary 8-bit image can be input by the user, or a matrix can be calculated. Examples for the case of a circle and a spot are included in the code.

```
%% Algorithm to generate Pattern for projection onto Spatial Light Modulators

% Resolution of the SLM
ResX=1380
ResY=2048

%------------ We start with all black image and generate a target field intensity

% Generating all black image
E_target=zeros(ResX,ResY);

% E_target(300-100,396+60)=255;
% E_target(300-100,396+40)=255;


%% Generate the image of the required pattern
% We wish to generate a circle, with center (a,b) and radius r

a=ResX/2;
b=ResY/2;

r=1000.0;
delta=500;

% to generate a circle:
% for i=1:1:4*a
%     for j=1:1:4*b
%         if(  ((a-i)^2 +(b-j)^2>r) &&  ( (a-i)^2 +(b-j)^2<(r+delta) )   )
%             E_target(uint16(i),uint16(j))=255;
%             sprintf('%d %d',i,j)
%
%         end
```

```matlab
%       end
% end

% To generate a single spot
E_target(a+100,b)=255;

imshow(E_target)
title('target Pattern')
%  target pattern


%%
%--------------- We generate a 256 bit phase distribution
E_target=fftshift(E_target);

Phase=rand(ResX,ResY)*(2*pi);
Phase_image=exp(-1i*Phase);
for j=1:1:10

    holo=ifftshift(ifft2(E_target.*Phase_image));

    Phase_holo=exp(-1i*angle(holo));

    E_focus=fft2(Phase_holo);
    Phase_image=exp(-1i*angle(E_focus));
end

Phase=angle(holo);

% For a general 256 bit SLM
Phase=uint8(mod(Phase+2*pi,2*pi)/(2*pi)*255);
imshow(Phase)
title('256 bit phase distribution pattern')

%%
% ----------- We convert the generated phase distribution
    to binary
%for the forthdd SLM which is binary
PhaseMat= angle(holo)>0;
Phase= uint8(PhaseMat*255);
% addSave=''
imshow(ifftshift(Phase))
title('Generated 2 bit pattern')
imwrite(ifftshift(Phase),'GS_generated.bmp')
%This is the final generated image
```

3 Python Code: Laguerre-Gaussian Beam Pattern Generation

The Python script generates a Laguerre-Gaussian Beam Pattern, also referred hereto as an axicon pattern. It defines a function to map phase values into Hamamatsu SLM's 8-bit range and applies a mathematical function to produce an axicon-like phase distribution.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 18 22:16:52 2017

@author: Admin
"""
#import slmpy
import time
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

M_x=792; # x-resolution of SLM
M_y=600; # y-resolution of SLM
A=np.zeros([M_x,M_y])
pi=np.pi;

Cx=400; # x-Center of axicon pattern
Cy=210;

def phase2hamamatsu(Phase_xy):  # Hamamatsu SLM phase bit
    varies from 0 to 224
    for p in range(M_x):
        for q in range(M_y):
            Phase_xy[p][q]=np.mod(Phase_xy[p][q],2*pi)
                *224/(2*pi)
    return Phase_xy


# Generating image for the pattern around center (Cx,Cy)
for i in range(792):
    for j in range(600):
        n=20
```

```
        r_o=1e4;
        r=(i-Cx)**2+(j-Cy)**2
        theta=np.arctan((j-Cy)/(i-Cx+1e-6))
        A[i,j]=np.mod(n*theta-2*pi*r/r_o,2*pi)

im=A; # 8-bit Image from 0 to 255 (uint8)
im=phase2hamamatsu(im)
im=np.transpose(im)
im=im.astype(np.uint8)
image=Image.fromarray(im)
image.save('Axicon.bmp')



#%%
import slmpy
import time
import numpy as np
from PIL import Image
slm = slmpy.SLMdisplay(isImageLock = True)
resX, resY = slm.getSize()
# We use images twice smaller than the resolution of the
   slm
i=0

testIMG = np.asarray(Image.open('Axicon.bmp'))
slm.updateArray(testIMG)
time.sleep(500)
slm.close()
```

4 Further Work

In this section, we are going to show how this code is useful to calculate a pattern when the alignment is unknown. To align incoming beam with the axicon pattern of our SLM, we use the above code with various values of Cx and Cy. Figure 1 top image shows that we achieved an aligned circle with values of Cx and Cy that were obtained via a 2d search. Other three images show the misalignment. It must be noted, what we're trying to achieve here is to orient the center of the spiral pattern on the SLM with the incoming Laser beam.

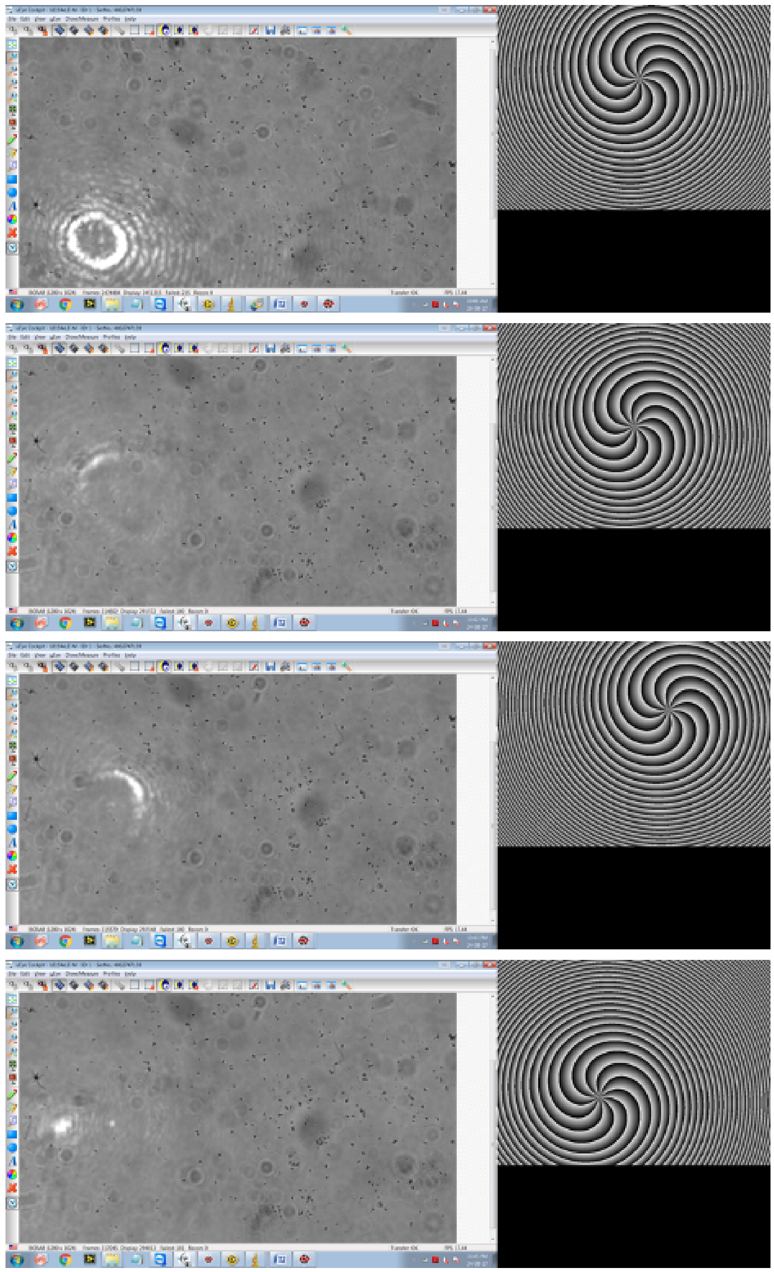

Figure 1: Figure presents four screenshots corresponding to iterations for different positions chosen for the center of the axicon pattern. Left hand side is the microscope image obtained with a camera, and the right hand side shows the projected pattern. The topmost image shows the correct position of center which was found using trial and error.