Computer generated holography (diffractive optics) Phys 423

In this lab, we will investigate the design of computer-generated holograms which could be implemented on a binary phase-only spatial light modulator (SLM).

The task we are going to set ourselves is to design a hologram that will take a beam of light and turn it into a two-dimensional array of spots of light in the back focal plane of a lens. Such spot arrays have been used in optical tweezers applications and could be a critical part of the construction of a neutral-atom quantum computer.

We are going to do several tasks:

1. A two-dimensional direct search design of a spot array.
2. A search by simulated annealing. Theoretically, you can find the optimal hologram but it might take a very long time.
3. Direct search for a more complicated output than a spot array.

References

- An example of the type of device that could implement the holograms is the spatial light modulator seen here.
  https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=10378
  Actually, this device has a range of phase modulation levels, not just the two that we will consider here.
- A well-known paper on the use of a spatial light modulator for optical tweezers is Curtis, J.E., Koss, B.A. and Grier, D.G., 2002. Dynamic holographic optical tweezers. *Optics communications*, *207*(1-6), pp.169-175 which is included with the lab manual. See Figure 1 in the paper.
- A review of simulated annealing can be found at https://en.wikipedia.org/wiki/Simulated_annealing
- A few pages of the book, Numerical Recipes, are included with the lab materials which introduces the simulated annealing method.
- A nice summary of computer-generated holography is at https://en.wikipedia.org/wiki/Computer-generated_holography.

Background

We can think of a hologram as some sort of a structure that modulates the phase and/or amplitude of an input light field to generate a new light field with a desired amplitude or intensity distribution. When we think about display holography this means that we want to create a hologram that turns a beam of light into a light distribution identical to that which would occur if the original object were in place.

Display holography has not yet penetrated to mass-consumer products, and perhaps it never will, but the use of diffractive structures for light field shaping has found widespread application in several areas of science and engineering including optical tweezers, atom trapping and laser beam shaping.

The simplest “hologram” we can think of is perhaps the diffraction grating. This device takes a single beam of light and distributes it among several output beams. In physics 132 you observed the behavior of these gratings and in physics 323 you will have analyzed it in considerable detail, finding the relative intensity of the output beams under monochromatic illumination and how the beams behave as the wavelength of the light changes.

A diffraction grating modulates the amplitude of the incident light but we are going to examine diffraction from a device that modulates the phase of the light. However, we will only have access to two distinct levels – 0 and π. The type of hologram we are going to focus on is where the desired light distribution occurs at the focus of a lens – sometimes called a Fourier plane hologram. So how do we decide the distribution of phases needed for a given task? There is no general direct algorithm but one thing we can do is search through many different arrangements of pixels and see which one gives an output closest to the desired output.

The spatial light modulator (SLM)

We are going to design our hologram with a view to implementing it on a binary, phase-only spatial light modulator (SLM). This type of spatial light modulator is often a liquid crystal on silicon (LCOS) ferroelectric liquid crystal (FLC) device. Basically, the SLM is a display. It is comprised of a large number of pixels that are individually addressable. Display sizes typically run into millions of pixels with each pixel on the order of 10 microns on a side. The desired pattern is computed and loaded onto the display much as you would display an image on a cell phone screen.

How does the SLM affect the light? In this type of SLM the modulating liquid crystal acts as a half-waveplate, one where the fast axis can be switched between two orientations.

An optical arrangement that utilizes a spatial light modulator is shown in figure 1, and you can see a practical use of this type of arrangement in figure 1 of the paper by Curtis et al.

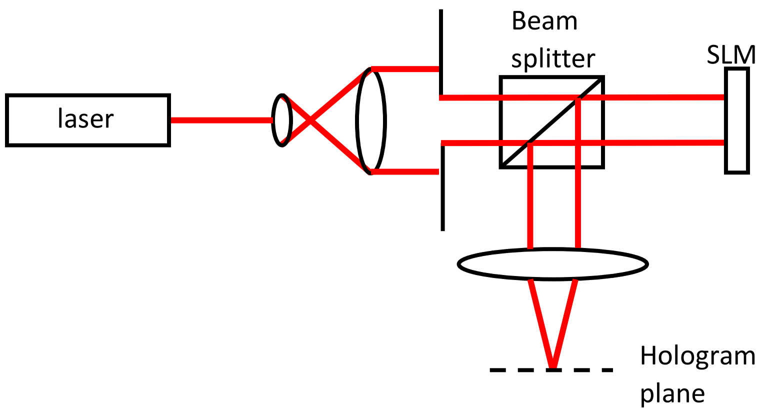

Figure 1. An experimental optical arrangement for displaying a computer-generated hologram. Light from the laser is expanded and collimated. The beam passes through a (non-polarizing) beam splitter

and is incident on the spatial light modulator (SLM). After reflection a lens, the hologram is formed one focal length from the lens.

Designing the hologram

By direct search, where we design the hologram by a trial-and-error approach.

Known as “binary direct search” what we do is this: start with a random distribution of pixels (0 and π phase shift). Calculate the output irradiance that this gives by computing the Fourier transform of the distribution and then multiplying this transform by its own complex conjugate. Now, randomly choose one of the pixels and flip its state – if it was 0 make it π, if it was π make it 0. Recalculate the irradiance due to this distribution. Is it closer to the irradiance you want than the irradiance due to the previous distribution of pixels? If it is then keep that pixel change, if not set the pixel back to its original state. Keep repeating this until you get an irradiance output close to your desired field. A reasonable “error function” to keep track of the difference between the target pattern and the pattern the hologram generates is the difference of squares given by

$$E = \sum_i (t_i - o_i)^2$$

where i indexes the pixels in the target image (t) and the output (o). Here is a short program to get started. If you paste it directly into MATLAB it should run and give a set of plots.

```
%direct search for binary hologram
clear; clf;
rng('shuffle');

sze=64; %the size of the SLM
cent=sze/2+1; %this is the center of the target

% the next bit makes a target which the hologram will create
target=zeros(sze,sze);
target(cent,cent+16)=1;
target(cent,cent-16)=1;
target(cent+12,cent)=1;
target(cent-12,cent)=1;
target(cent+12,cent+16)=1;
target(cent+12,cent-16)=1;
target(cent-12,cent+16)=1;
target(cent-12,cent-16)=1;
target(cent,cent)=1;

s=sqrt(sum(sum(target.^2)));
target=target/s; %making the average "energy" a number we can compare with
            %the eventual hologram output
```

```
hol=rand(sze,sze); %a set of numbers between 0 and 1

%now set the pixels plus or minus one.
for i=1:sze;
   for j=1:sze;
      if hol(i,j)<0.5;
         hol(i,j)=-1;
      else
         hol(i,j)=1;
      end
   end
end

%do the first pass through the hologram
holf=fft2(hol);
holf=fftshift(holf);
holf2=holf.*conj(holf);
holf2=holf2./(sqrt(sum(sum(holf2.^2))));


err1=sum(sum((target-holf2).^2)); % the first difference between the target
                                  % and the hologram output.

iter=20000;
err_arr=zeros(1,iter); %an array so we can plot the error
for i=1:iter
   rpix_x=round(1+rand*(sze-1)); %pick a random x
   rpix_y=round(1+rand*(sze-1)); %pick a random y

   hol(rpix_x, rpix_y)=-1*hol(rpix_x, rpix_y); %flip that pixel

   holf=fft2(hol);
   holf=fftshift(holf);
   holf2=holf.*conj(holf);

   holf2=holf2/(sqrt(sum(sum(holf2.^2))));
   err2=sum(sum((target-holf2).^2))

   if err2>err1
      hol(rpix_x, rpix_y)=-1*hol(rpix_x, rpix_y); %error has increased
                                                  % so set the pixel back
   else
      err1=err2;
   end
   err_arr(i)=err1; %record the error
end
subplot(2,2,1)
imagesc(target)
```

```
title('target pattern')
subplot(2,2,2)
imagesc(holf2)
title('hologram output')
subplot(2,2,3)
imagesc(hol);
title('hologram')
subplot(2,2,4)
plot(err_arr);
title('error')
```

The above program is crude. For example, it just runs for a preset number of iterations. It would be better to set some limit on the error signal and terminate the program when the error is small enough.

By simulated annealing

If you run the direct search several times you will see that sometimes it gets stuck and you do not get a good output. This is because any pixel flip leads to a larger error than the previous distribution of pixel phases. Although you have not found a very good hologram, the algorithm has no way of getting away from the current distribution. If you think of the hologram design process as one where you want to minimize the error function, then rather than finding the *global minimum* you are stuck in a *local minimum*. Many years ago, it was realized that it was not a good practice to reject *all* the changes that led to an increase in error. Rather, some of these unfavorable changes should be kept and thus there is a chance of escaping the local minimum. The process starts with a fairly high probability of keeping an unfavorable change and over time as we (hopefully) approach an optimum solution then this probability is decreased. Because this process has some analogy with the slow cooling of metals (annealing) to create a low-energy state of the metal structure, it is called simulated annealing. There is a compact and readable review of the simulated annealing method copied from the book "Numerical Recipes", and included with the lab materials.

The algorithm is the following

- Start with a random distribution of pixels and compute the difference between the hologram output and the target.
- Flip a pixel in the hologram, compute the output, and compare it against the target.
- Did the error go down? If yes, accept the pixel change.
- Did the error go up? If it did, then "throw a die" to see if you will keep this change or reject it.
- Flip another pixel and repeat the process, but reduce the probability of accepting an unfavorable pixel flip.

So how do we decide whether or not to accept a pixel flip? We will use the Metropolis algorithm, which is mentioned in the handout from Numerical Recipes. First, calculate the change in energy going from one state to another. Call it ΔE, and if it is less than zero we accept the flip. If it is more than zero then calculate $\exp(-{}^{\Delta E}/_{T})$ where T is "temperature", and is a positive number. Pick a number from the uniform distribution (obtained with rand) and if this is less than $\exp(-{}^{\Delta E}/_{T})$ we accept the change. Otherwise, do not accept the change and put the pixel back to its previous state. On proceeding to the next step, we decrease the temperature slightly. Note that at early times, when the temperature is large, the exponential term is closer to one so that unfavorable flips are often accepted and when the system “cools” the exponential term is closer to zero. Hopefully by that time we are closer to a good solution. You can cool the system by including a factor such as 0.999 which you multiply the temperature by at the end of each iteration.

Now, there are several questions that must be addressed such as “what is a reasonable range for temperature” and “what is a good cooling schedule”. These are questions that you will need to address – by trial and error.

Lab report

When getting your programs sorted out and debugged, I strongly recommend you start with small arrays (e.g. 64x64 or 32x32) before trying to work with the larger arrays requested for the lab report.

Please submit:

- Code and results for the binary direct search generation of a 4x4 spot array on a 256x256 binary phase SLM. The spots should be separated by 16 pixels. Please include an image of the desired spot array, an image of the final hologram, an image (or surf plot) of the hologram output, and a graph of the error as a function of iteration number.
- Code and results for the simulated annealing generation of a 4x4 spot array on a 256x256 binary phase SLM. Please include an image of the desired spot array, an image of the final hologram, an image (or surf plot) of the hologram output, and a graph of the error as a function of iteration number.
- Code and results for the binary direct search generation of a rectangular array of 16x16 pixels on a 256x256 binary phase SLM. Please include an image of the desired output, an image of the final hologram, an image (or surf plot) of the hologram output, and a graph of the error as a function of iteration number.