Digital Video
Spring 2016
HW 3

Ambika Verma
Av28944

Q 1)

Frequency for sinusoids was set to pi/2, resulting in Lambda value of 4.

Equations used –

%for orientation

```
x_theta=x*cos(theta)+y*sin(theta);
y_theta=-x*sin(theta)+y*cos(theta);
```

%filter design

```
gb= exp(-0.5*((x_theta.^2/sigma_x^2)+(y_theta.^2/sigma_y^2))).*exp(i*((2*pi/lambda*x_theta)+psi));
```

[1]

and

```
gb= exp(-0.5*((x_theta.^2/sigma_x^2)+(y_theta.^2/sigma_y^2))).*exp(-i*((2*pi/lambda*x_theta)+psi));
```

[2]

for the replica of the filter in the specified orientation

Parameters used –

Filter 1(using equation 1 above) and 3(using equation 2 above)-

```
Lambda=4
Theta=45
gamma=1;
sigma=1;
psi=0;
```

Filter 2(using equation 1 above) and 4(using equation 2 above)-

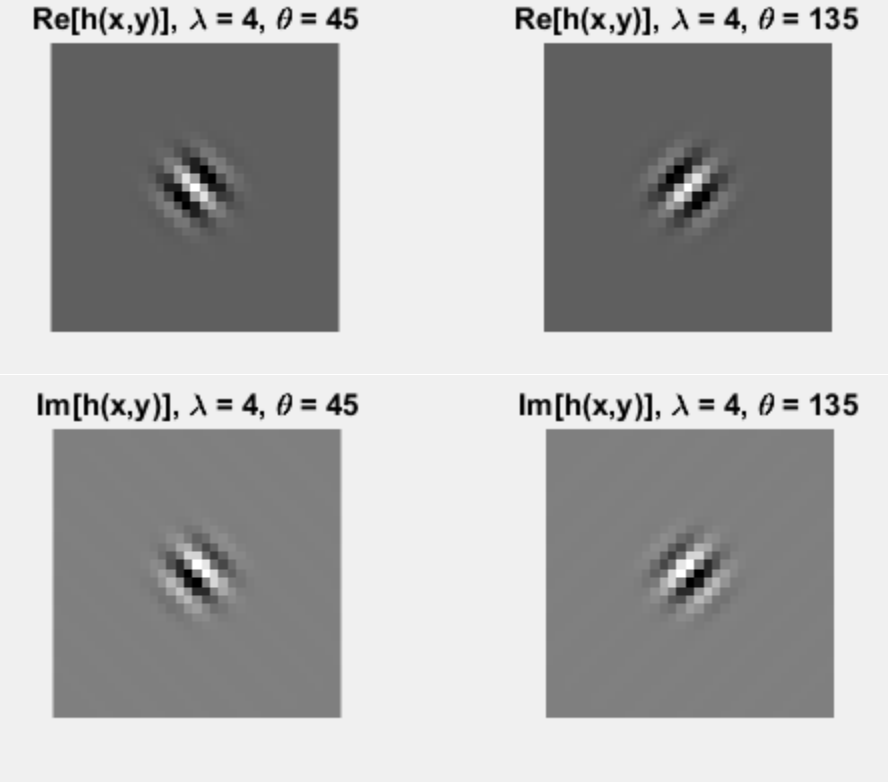
```
Lambda=4
Theta=135
gamma=1;
sigma=1;
psi=0;
```

Following code was used to generate gabor filters –

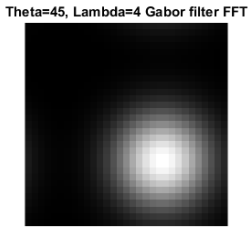
```
clear all
close all
clc
theta=pi/4;
theta2=3*pi/4;
lambda=4;
gamma=1;
sigma=1;
```

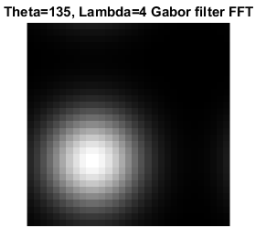
```
psi=0;

sigma_x = sigma;
sigma_y = sigma/gamma;

nstds = 20;
xmax = max(abs(nstds*sigma_x*cos(theta)),abs(nstds*sigma_y*sin(theta)));
xmax = ceil(max(1,xmax));
ymax = max(abs(nstds*sigma_x*sin(theta)),abs(nstds*sigma_y*cos(theta)));
ymax = ceil(max(1,ymax));
xmin = -xmax; ymin = -ymax;
[x,y] = meshgrid(xmin:xmax,ymin:ymax);

x_theta=x*cos(theta)+y*sin(theta);
y_theta=-x*sin(theta)+y*cos(theta);

gb= exp(-.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*exp(i*(2*pi/lambda*x_theta+psi));
gbf=fft2(gb);

gb3= exp(-.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*exp(-i*(2*pi/lambda*x_theta+psi));
gbf3=fft2(gb3);

xmax = max(abs(nstds*sigma_x*cos(theta2)),abs(nstds*sigma_y*sin(theta2)));
xmax = ceil(max(1,xmax));
ymax = max(abs(nstds*sigma_x*sin(theta2)),abs(nstds*sigma_y*cos(theta2)));
ymax = ceil(max(1,ymax));
xmin = -xmax; ymin = -ymax;
[x,y] = meshgrid(xmin:xmax,ymin:ymax);

x_theta=x*cos(theta2)+y*sin(theta2);
y_theta=-x*sin(theta2)+y*cos(theta2);

gb2= exp(-.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*exp(i*(2*pi/lambda*x_theta+psi));
gbf2=fft2(gb2);

gb4= exp(-.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*exp(-i*(2*pi/lambda*x_theta+psi));
gbf4=fft2(gb4);

figure
subplot(2,2,1)
imshow(abs(fftshift(gbf)),[]);
subplot(2,2,2)
imshow(abs(fftshift(gbf2)),[]);
subplot(2,2,3)
imshow(abs(fftshift(gbf3)),[]);
subplot(2,2,4)
imshow(abs(fftshift(gbf4)),[]);
figure
```

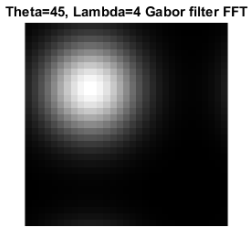
```
subplot(2,2,1)
surf(abs(fftshift(gbf)));
subplot(2,2,2)
surf(abs(fftshift(gbf2)));
subplot(2,2,3)
surf(abs(fftshift(gbf3)));
subplot(2,2,4)
surf(abs(fftshift(gbf4)));
```

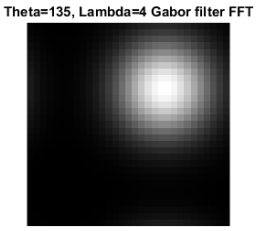


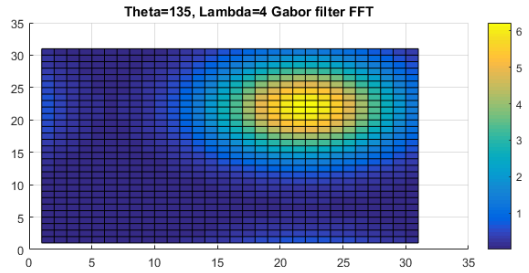
Taking FFT of the above designed filters resulted in following figures –

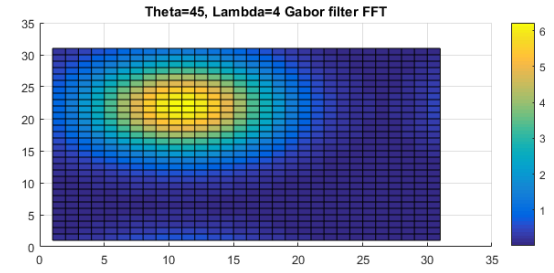


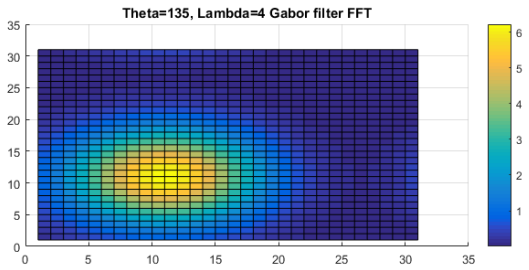


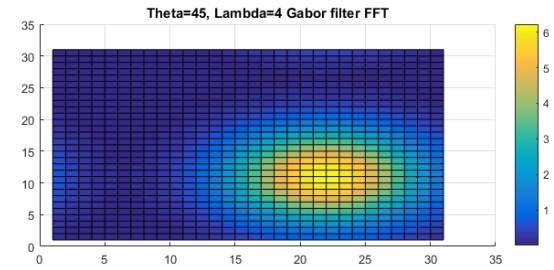












From the graph above peak value is approximately 6 resulting in half peak at approx 3 (green band). It is clear from the above plot that the filters are intersecting at half peaks.

Gabor filter was then applied to video as follows –

```
clear all
close all
clc
wavelength = 4;
orientation = [45 135];
g = gabor(wavelength,orientation,'SpatialFrequencyBandwidth',1,'SpatialAspectRatio',1);
v = VideoReader('C:\Users\user1\Downloads\basketball.mp4');
i=0;
framecell=[];
outputVideo1 = VideoWriter('C:\Users\user1\Downloads\basketball45');
```

```
outputVideo2 = VideoWriter('C:\Users\user1\Downloads\basketball135');
outputVideo3 = VideoWriter('C:\Users\user1\Downloads\basketball45_2');
outputVideo4 = VideoWriter('C:\Users\user1\Downloads\basketball135_2');
outputVideo = VideoWriter('C:\Users\user1\Downloads\basketball1final2');
outputVideo1.FrameRate = v.FrameRate;
outputVideo2.FrameRate = v.FrameRate;
outputVideo3.FrameRate = v.FrameRate;
outputVideo4.FrameRate = v.FrameRate;
outputVideo.FrameRate = v.FrameRate;
open(outputVideo1)
open(outputVideo2)
open(outputVideo3)
open(outputVideo4)
open(outputVideo)
while hasFrame(v)
frame = readFrame(v);
   frame_gray=rgb2gray(frame);

out1 = imgaborfilt(frame_gray,g(1));
out2 = imgaborfilt(frame_gray,g(2));
out3 = imgaborfilt(frame_gray,g(1));
out4 = imgaborfilt(frame_gray,g(2));
out1d=downsample(out1,2);
out2d=downsample(out2,2);
out3d=downsample(out3,2);
out4d=downsample(out4,2);
out1d=downsample(out1d',2);
out2d=downsample(out2d',2);
out3d=downsample(out3d',2);
out4d=downsample(out4d',2);

out=cat(1,out2d',out1d');
outn=cat(1,out3d',out4d');
outf=cat(2,out,outn);
writeVideo(outputVideo1,uint8(out1))
writeVideo(outputVideo2,uint8(out2))
writeVideo(outputVideo3,uint8(out3))
writeVideo(outputVideo4,uint8(out4))
writeVideo(outputVideo,uint8(outf))
end
close(outputVideo1)
close(outputVideo2)
close(outputVideo3)
close(outputVideo4)
close(outputVideo)
```

*Output video is available as basketballfinal2.avi

Each filter responds to structures in the image oriented(45 or 135 degrees) in the direction of the filter.
Top left and bottom right filter – 135 degrees
Top right and bottom left filter – 45 degrees

Q4)

```
clear all
close all
clc
v=VideoReader('C:\Users\Dell\Downloads\flag.mp4');
outputVideoA=VideoWriter('C:\Users\Dell\Downloads\a');
outputVideoB=VideoWriter('C:\Users\Dell\Downloads\b');
outputVideoA.FrameRate=v.FrameRate;
outputVideoB.FrameRate=v.FrameRate;
open(outputVideoA)
open(outputVideoB)
numFrames = v.NumberOfFrames;
for k=2: numFrames
   k
frame = read(v,k);
prev_frame=read(v,k-1);
frame_gray=cv.cvtColor(frame,'RGB2GRAY');
prev_frame_gray=cv.cvtColor(prev_frame,'RGB2GRAY');
flow = cv.calcOpticalFlowFarneback(prev_frame_gray, frame_gray, 'WinSize',8,'Iterations',100);
a=flow(:,:,1);
a=uint8(255*(a/max(max(a))));
b=flow(:,:,2);
b=uint8(255*(b/max(max(b))));
writeVideo(outputVideoA,a)
writeVideo(outputVideoB,b)
end
close(outputVideoA)
close(outputVideoB)
```

Flow outputs for window size of 8 were more robust and less sensitive to noise as compared to those with window size 2, but were more blurred.

The function uses the following algorithm –

prevImg(y,x) ~ next(y + flow(y,x,2),x + flow(y,x,1))

a(m,k)= flow(:,:,1); %vertical motion
b(m,k)= flow(:,:,2); %horizontal motion

*8 output videos are available in folder as –
a_2_10.avi (for window size 2 and 10 iterations) and so on