COGS 119
MATLAB for Experimental Research

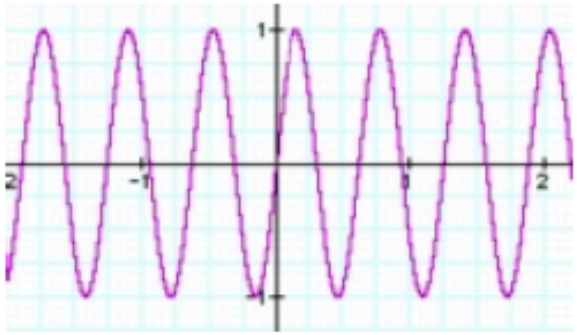
Fall 2014 – Week 8

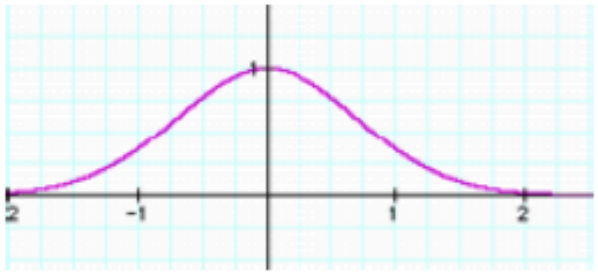
Basic Vision Science with Psychtoolbox

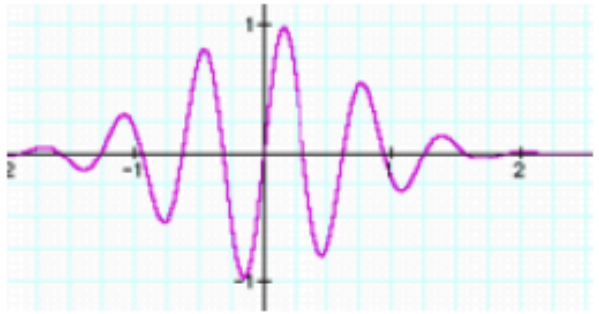
and Complete Experiment

Gabor

- Simple cells in the primary visual cortex (V1) of the primate brain can be modeled using Gabor filters, which is a product of sine wave and a Gaussian.

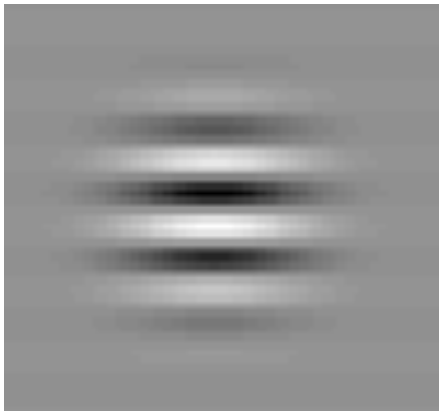






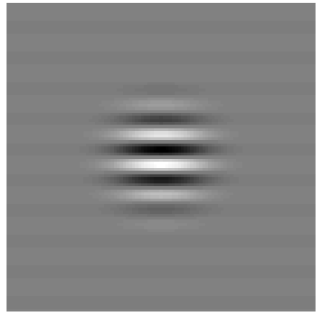
Gabor

- A gabor is characterized by several parameters:
  - Orientation (theta)
  - Spatial frequency (sf)
  - Width (sd)
  - Size



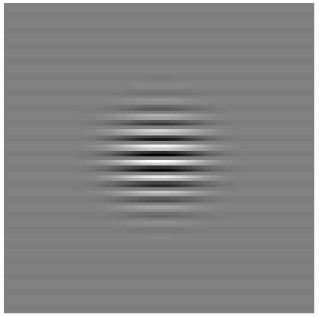
Example gabors

theta = 90
sf = 5
sd = 0.2
size = 100;

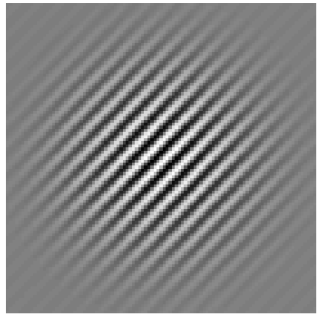


theta = 45
sf = 10
sd = 0.2
size = 100

theta = 90
sf = 10
sd = 0.2
size = 100;



theta = 45
sf = 10
sd = 0.4
size = 100



Creating a gabor in Matlab

Gabor = Sinewave * Gaussian

Step 1: Create Sinewave

Step 2: Create Gaussian

Step 3: Multiply

Creating a gabor in Matlab

```
1
2   function gabor = prep_gabor (sd, sf, theta, matsize)
3
4   % This function prepares a gabor of the input parameters to be displayed by
5   % psychtoolbox (disp_gabor.m). Input standard deviation (sd), spatial frequency (sf),
6   % orientation angle (theta) and matrix size
7
8 -  [x,y] = meshgrid(linspace(-1,1,matsize+1));
9 -  xc=0; yc=0;
10
11 -  ramp = cos(theta*pi/180)*(x-xc) + sin(theta*pi/180)*(y-yc);
12
13   % lets make a grating from this with the spatial frequency sf
14 -  grating = sin(2*pi*sf*ramp);
15
16   % make a gaussian centered at (xc,yc)
17 -  gauss = exp( -((x-xc).^2 + (y-yc).^2)/(2*sd^2));
18
19   % now we can combine them into a gabor
20 -  gabor = gauss.*grating;
21
22   % scale it from 0 to 255
23   % gabor = (gabor+1)*127.5;
24
25   % just for testing, comment out after checking gabor is OK
26 -  figure;
27 -  imagesc (gabor); colormap(gray); axis off; axis square%
28
29
```

prep_gabor.m

Creating a gabor in Matlab

```
1
2   function gabor = prep_gabor (sd, sf, theta, matsize)
3
4   % This function prepares a gabor of the input parameters to be displayed by
5   % psychtoolbox (disp_gabor.m). Input standard deviation (sd), spatial frequency (sf),
6   % orientation angle (theta) and matrix size
7
8   [x,y] = meshgrid(linspace(-1,1,matsize+1));
9   xc=0; yc=0;
10
11  ramp = cos(theta*pi/180)*(x-xc) + sin(theta*pi/180)*(y-yc);
12
13  % lets make a grating from this with the spatial frequency sf
14  grating = sin(2*pi*sf*ramp);
15
16  % make a gaussian centered at (xc,yc)
17  gauss = exp( -((x-xc).^2 + (y-yc).^2)/(2*sd^2));
18
19  % now we can combine them into a gabor
20  gabor = gauss.*grating;
21
22  % scale it from 0 to 255
23  % gabor = (gabor+1)*127.5;
24
25  % just for testing, comment out after checking gabor is OK
26  figure;
27  imagesc (gabor); colormap(gray); axis off; axis square%
28
29
```

inputs

sinewave

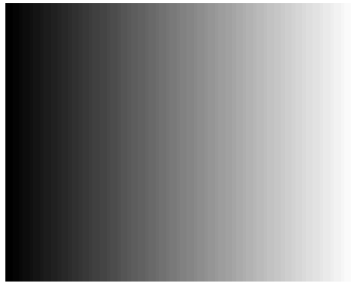
gaussian

multiply sinewave and gaussian

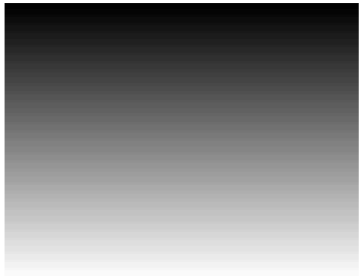
prep_gabor.m

Meshgrid

```
sd = 0.4   sf = 10   theta = 45   matsize = 100
>> [x,y] = meshgrid(linspace(-1,1,matsize+1));

>> imagesc(x); axis off; colormap('gray');
```

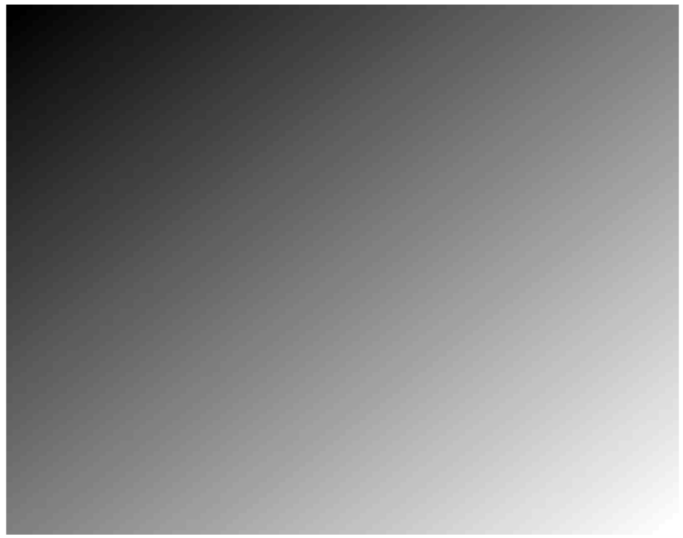


```
>> imagesc(y); axis off;
```



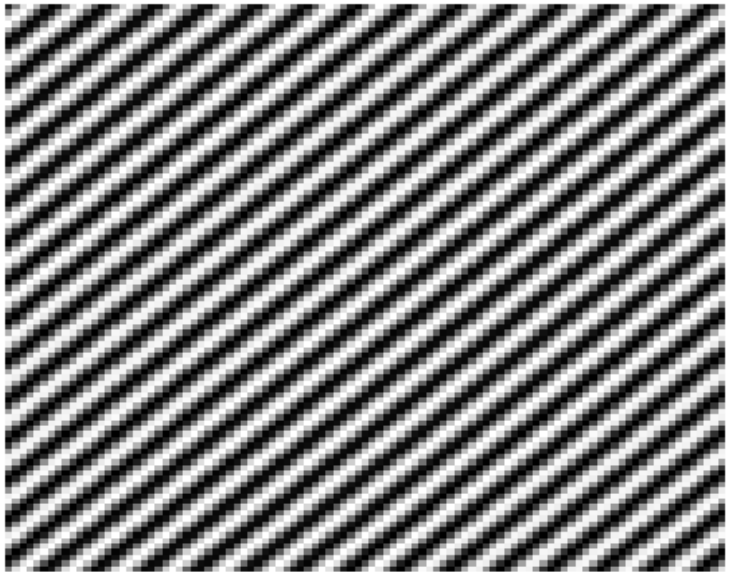
ramp

```
>> ramp = cos(theta*pi/180)*(x-xc) + sin(theta*pi/180)*(y-yc);

>> imagesc(ramp); axis off;
```



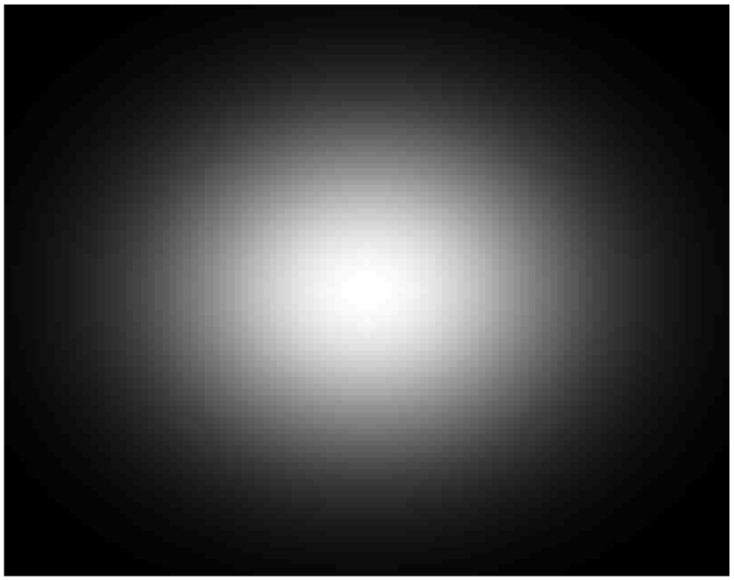
Sinewave (grating)

```
>> grating = sin(2*pi*sf*ramp);

>> imagesc(grating); axis off;
```



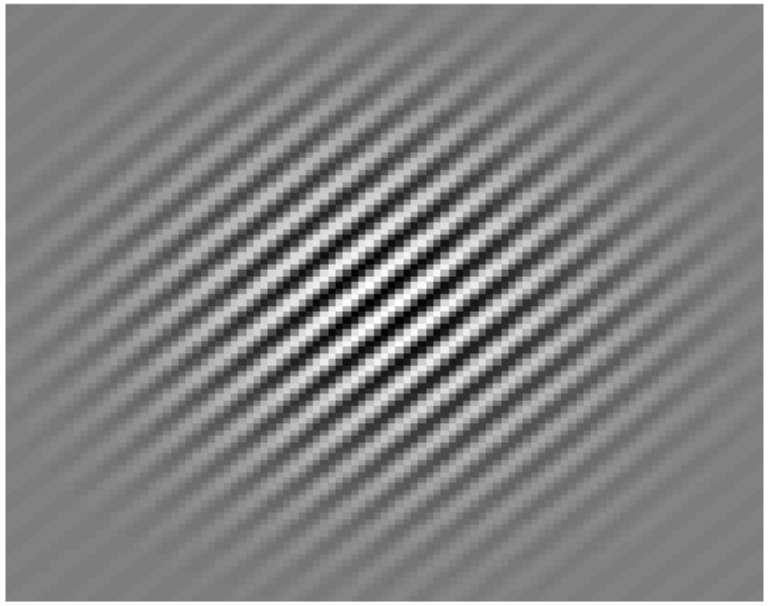
Gaussian

```
>> gauss = exp( -((x-xc).^2 + (y-yc).^2)/(2*sd^2));

>> imagesc(gauss); axis off;
```



Gabor

```
>> gabor = gauss.*grating;

>> imagesc(gabor); axis off;
```



Orienting gabor

- Download **gabor_sweep_orient.m** from the class website and run it.

- Now, let's look at the code closely to understand how the gabor changes orientation.

Displaying gabor in PTB

- Download **disp_gabor.m** from the class website and run it.

- **disp_gabor.m** prepares a gabor (calling **prep_gabor.m**) and displays it in Psychtoolbox.

- Inputs:
  - sd, sf, theta (gabor parameters)
  - offx, offy (offset from the center of the screen)

Displaying gabor in PTB

```
function disp_gabor (sd, sf, theta, offx, offy)
% This function displays a gabor of the input parameters to be displayed by
% psychtoolbox. Input standard deviation (sd), spatial frequency (sf),
% orientation angle (theta) and offset from center of screen in offx and
% offy
Screen('Preference', 'SkipSyncTests', 1);
screenNum=0;
res=[800 600];
matsize = 200; % size of gabor

[w,rect] = Screen('OpenWindow', screenNum, 0, [0 0 res(1) res(2)]);
[xc,yc] = RectCenter(rect);

% Retrieves color codes for black and white and gray.
black = BlackIndex(w);  % Retrieves the CLUT color code for black.
white = WhiteIndex(w);  % Retrieves the CLUT color code for white.
grey = (black + white) / 2;  % Computes the CLUT color code for gray.

% Taking the absolute value of the difference between white and gray will
% help keep the grating consistent regardless of whether the CLUT color
% code for white is less or greater than the CLUT color code for black.
inc = abs (white - grey);

% prepare gabor
gabor = prep_gabor (sd, sf, theta, matsize);
gabor = grey + inc * gabor + 1;
[gw, gh] = size (gabor); % width and height of image
gabortex = Screen('MakeTexture', w, gabor, [], [], []);
location = [xc - gw/2 + offx, yc - gh/2 + offy, xc + gw/2 + offx, yc+ gh/2+ offy];
% rightlocation = [3*xc/2 - imgwidth/2, yc - imgheight/2, 3*xc/2 + imgwidth/2, yc+imgheight/2];

% put it on screen
Screen(w, 'FillRect', grey);
Screen(w, 'Flip');
pause(1);
Screen(w, 'FillRect', grey);
Screen (w, 'DrawTexture', gabortex, [], location);
Screen(w, 'Flip');
pause(2);

Screen('CloseAll');
```

disp_gabor.m

Displaying gabor in PTB

inputs

```
2   function disp_gabor (sd, sf, theta, offx, offy)
3   % This function displays a gabor of the input parameters to be displayed by
4   % psychtoolbox. Input standard deviation (sd), spatial frequency (sf),
5   % orientation angle (theta) and offset from center of screen in offx and
6   % offy
7   Screen('Preference', 'SkipSyncTests', 1);
8   screenNum=0;
9   res=[800 600];
10  matsize = 200; % size of gabor
11
12  [w,rect] = Screen('OpenWindow', screenNum, 0, [0 0 res(1) res(2)]);
13  [xc,yc] = RectCenter(rect);
14
15  % Retrieves color codes for black and white and gray.
16  black = BlackIndex(w);  % Retrieves the CLUT color code for black.
17  white = WhiteIndex(w);  % Retrieves the CLUT color code for white.
18  grey = (black + white) / 2;  % Computes the CLUT color code for gray.
19
20  % Taking the absolute value of the difference between white and gray will
21  % help keep the grating consistent regardless of whether the CLUT color
22  % code for white is less or greater than the CLUT color code for black.
23  inc = abs (white - grey);
24
25  % prepare gabor
26  gabor = prep_gabor (sd, sf, theta, matsize);
27  gabor = grey + inc * gabor + 1;
28  [gw, gh] = size (gabor); % width and height of image
29  gabortex = Screen('MakeTexture', w, gabor, [], [], []);
30  location = [xc - gw/2 + offx, yc - gh/2 + offy, xc + gw/2 + offx, yc+ gh/2+ offy];
31  % rightlocation = [3*xc/2 - imgwidth/2, yc - imgheight/2, 3*xc/2 + imgwidth/2, yc+imgheight/2];
32
33  % put it on screen
34  Screen(w, 'FillRect', grey);
35  Screen(w, 'Flip');
36  pause(1);
37  Screen(w, 'FillRect', grey);
38  Screen (w, 'DrawTexture', gabortex, [], location);
39  Screen(w, 'Flip');
40  pause(2);
41
42  Screen('CloseAll');
```

Call prep_gabor.m

Make gabor texture

Draw gabor texture

disp_gabor.m

Experiment with gabors

- Let's display multiple gabors with various properties (sd, sf, theta) on various locations on the screen.
- Download **gabor_experiment.m** from the class website and run it.
- Let's look at the code closely to understand how it displays gabors.

gabor_experiment.m

Experiment has the following trial types:

- gabor with sd = 0.1, sf = 10, theta = 5;
- gabor with sd = 0.1, sf = 10, theta = 355;
- gabor with sd = 0.1, sf = 16, theta = 5;
- gabor with sd = 0.1, sf = 16, theta = 355;

gabor_experiment.m (trial set-up)

```
1   function data = gabor_experiment
2   % call prep_gabor.m to prepare the gabor and display it similar to what we
3   % did in display_gabor.m
4   KbName('UnifyKeyNames');
5   Screen('Preference', 'SkipSyncTests', 1);
6   screenNum=0;
7   res=[800 600]; % screen resolution
8   matsize = 200; % size of gabor
9   offx = 0; offy = 0;
10  Trialtime = 2;
11  isi = 0.5;  % inter stimulus interval
12  matsize = 200; % size of gabor
13
14  disp ['Welcome to the gabor experiment'];
15  data.Subnum = input (['Enter subject number: ']);
16  data.Date = date;
17  data.Data = [];
18
19  [fid message] = fopen('gabor_output.txt', 'w');
20  if fid == -1
21      fprintf('Couldn''t open output file.\n%s\n', message);
22  end
23  fprintf(fid, 'Subject no: %d', data.Subnum);
24  fprintf(fid, 'trial\tresponse\tRT\taccuracy\r\n');
25
26  % experiment has the following trial types:
27  % gabor with sd = 0.1, sf = 10, theta = 5;
28  % gabor with sd = 0.1, sf = 10, theta = 355;
29  % gabor with sd = 0.1, sf = 16, theta = 5;
30  % gabor with sd = 0.1, sf = 16, theta = 355;
31  rng('shuffle');
32  trials = [0.1 10 5; 0.1 10 355; 0.1 16 5; 0.1 16 355];
33  % trials has the order sd, sf, theta
34
35  offsets = [100 100; -120 100; 100 -50; -100 -100];
36  % offsets has the order x offset and y offset
37
38  % creating a new random order of the trials
39  trials = Shuffle(trials); offsets = Shuffle(offsets);
40  % Shuffle is a psychtoolbox function, read help
41  % trials and offsets are randomized separately - make sure this is what
```

Open a file to write

Set-up the trials – gabor parameters

```
>> help rng
```

rng

`rng('shuffle')`: seeds the random number generator based on the current time so that RAND, RANDI, and RANDN produce a different sequence of numbers after each time you call rng.

gabor_experiment.m (open window and set screen parameters)

```
43
44 -    while KbCheck; end
45
46
47 -    [w,rect] = Screen('OpenWindow', screenNum, 0, [0 0 res(1) res(2)]);
48     % define window w and open screen
49 -    [xc,yc] = RectCenter(rect);
50
51 -    black = BlackIndex(w);  % Retrieves the CLUT color code for black.
52 -    white = WhiteIndex(w);  % Retrieves the CLUT color code for white.
53 -    grey = (black + white) / 2;  % Computes the CLUT color code for gray.
54 -    inc = abs (white - grey);
55
```

gabor_experiment.m (prepare and display gabors)

```
for t = 1: size (trials,1)

    gabor = prep_gabor (trials (t,1), trials (t,2), trials(t,3), matsize);
    gabor = grey + inc * gabor + 1;
    [gw, gh] = size (gabor); % width and height of gabor
    gabortex = Screen('MakeTexture', w, gabor, [], [], []); % make gabor texture
    location = [xc - gw/2 + offsets(t,1), yc - gh/2 + offsets(t,2), xc + gw/2 + offsets(t,1), yc+ gh/2+ offsets(t,2)];

    % Get key press
    Starttime = GetSecs;
    while GetSecs < Starttime + Trialtime
        % put it on screen
        Screen(w, 'FillRect', grey);
        Screen (w, 'DrawTexture', gabortex, [], location); % draw gabor
        Screen(w, 'Flip'); % display gabor by flipping
        [keyIsDown, secs, keycode] = KbCheck;
        if keyIsDown
            response = KbName(keycode);
            resptime = secs - Starttime;
            break
        else
            response = 'none';
            resptime = 999;
        end

    end

    % record reaction time data into data
    data.Data = [data.Data resptime];

    % Write the trial information to the text file
    fprintf (fid, '%d\t%s\t%f\r\n', t, response, resptime);

    Screen(w, 'FillRect', grey);
    Screen(w, 'Flip');
    WaitSecs(isi);
end


Screen('CloseAll');
```

Call prep_gabor.m

Make a texture for the gabor and then draw it as in disp_gabor.m

Get user response and rt

Write into a file

Exercise

- Read Fine and Boynton chapter 5 and do the exercises there in Matlab.