Histograms of gradients

Calculation of hog features

1. **Scale the image to 128x64 (may include rotation)**

2. **Find image gradient using Sobel edge operator**

filterx = [-1 0 1]

$$\text{filtery} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

I * filterx -> dx

I * filtery-> dy

dx

dy

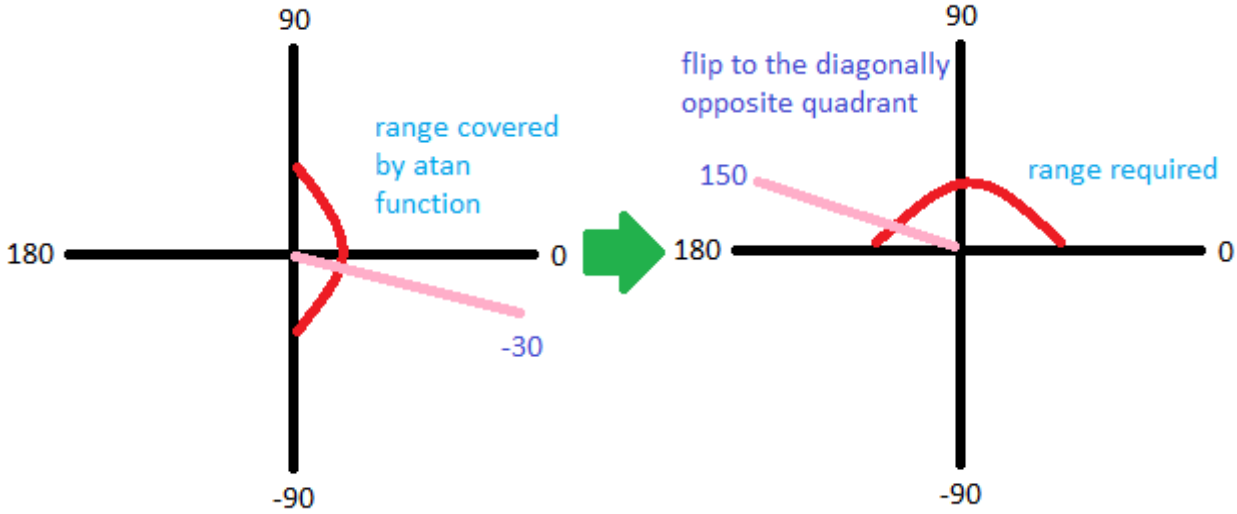
3. Find the image gradient magnitude and orientation

magnitude = $\sqrt{dx^2+dy^2}$

orientation = $\arctan(\frac{dy}{dx})$

magnitude

a. fix the orientation
tan inverse function returns -90 to 90, we need 0 to 180
so, if angle less than 90, add 180



b. fix the divide by zero error which might come while performing atan
find indexes where dx = 0, suppose locations
dx(locations) = 1
dy(locations) = 1000000
so atan(a/0) = atan(100000) = 90 degrees

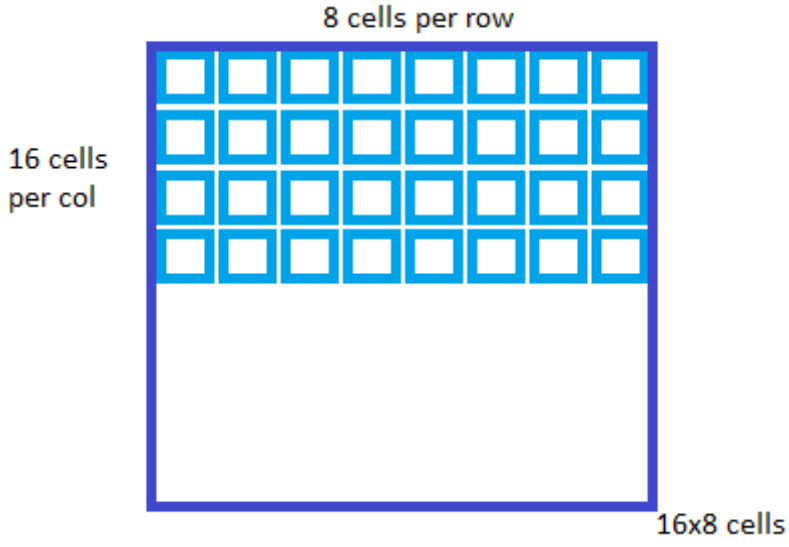
4. Create blank matrix histogram for image

5. Split the image into blocks and cells

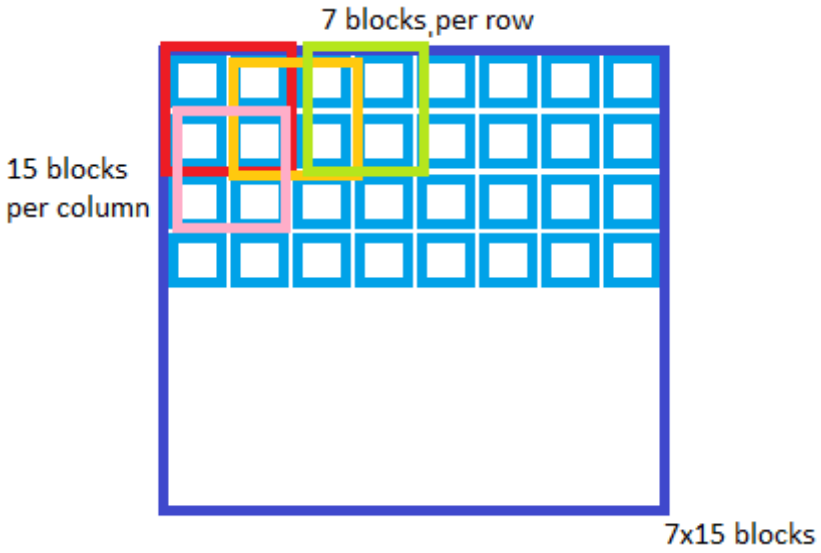
Here, the splitting is three level hierarchy:

image							
block				block			
cell	cell	cell	cell	cell	cell	cell	cell
pixel							
pixel							
pixel							
pixel							
pixel							
pixel							
pixel							
pixel							

Neighbouring pixels (8x8 pixels) are combined together to form cells. Cells are big enough to capture the local property at a big enough scale.



Now, if cell's histograms are combined then the connections between individual cells is lost. So to avoid that, 2x2 cells are combined into blocks and each block is taken so that there is 50 % overlap.



a. split the image into blocks of size 16x16 pixels and shift by 8 pixels

```
for i = 1 : 15
      for j = 1: 7
            % 1-16,8-24,16-32,24-40,32-48,40-56,48-64,...
            startxb = (i-1) * 8 + 1;
            startyb = (j-1) * 8 + 1;
            %get 16x16 pixels block
            blockm = magnitude(startxb : startxb + 15, startyb : startyb +15);
            blocko = orientation(startxb : startxb + 15, startyb : startyb +15);
```

for each block create a blank histogram

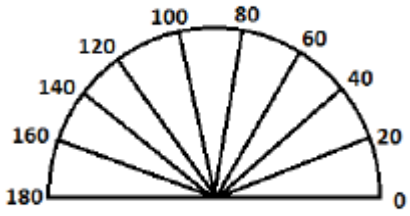
b. split the image into 2x2 cells each of 8x8 pixels

```
%divide each block into 2x2 cells
for a = 1 : 2
      for b = 1 : 2
            %1-8,9-16 (only two cells)
            startxc = 8 * (a - 1) + 1;
            startyc = 8 * (b-1) + 1;
            cellm = blockm(startxc:startxc+7, startyc: startyc+7);
            cello = blocko(startxc:startxc+7, startyc: startyc+7);
```

- note that startxb, startyb, startxc, startyc are physical address in their local region i.e. Startxb and startyb are defined in terms of image, startxc and startyc are defined in terms of blocks.

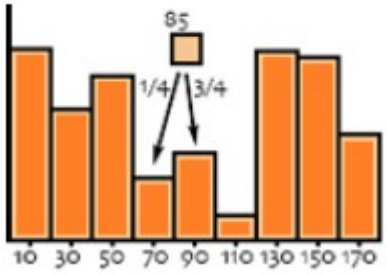
6. For each cell, find the histogram of orientations

orientation of each pixel is from 0 to 180
we will make into 9 bin (each of 20 degrees) weighted histogram



9 Bins

weight = 1. magnitude
= 2. linear voting



Bin centers

e.g. Angle = 85.
competiting parties = 70 and 90.
distance of 85 from 70 = 15
distance of 85 from 90 = 5
so weight of 70 = 5/20
and weight of 90 = 15/20
(more near -> more weight)

= 3. gaussian voting
same as sift. downweight the pixels near the edges of the block.

7. Append cell histogram into block histogram

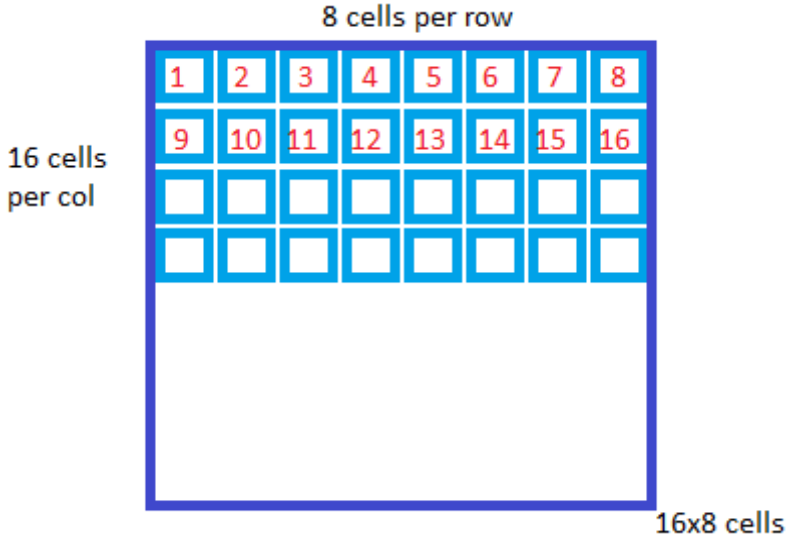
Each cell histogram has 9 values.
Total = 9 X 4

8. Append block histogram into image histogram

Each image has 105 blocks
total = 105 x 9 x 4 = 3780 values

Extra step:

- For visualizing pruposes, there is a need to store the logical address values for each of the cells.

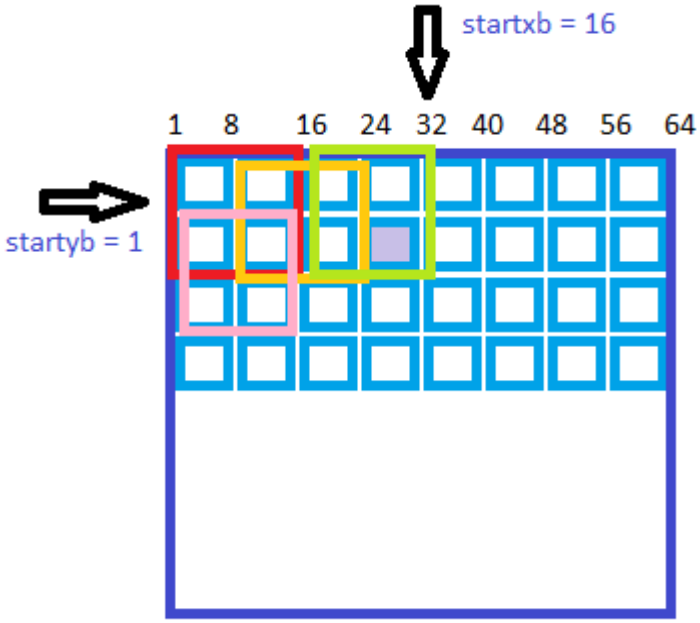


- Logical address of the cell = 1...128 which cell?
- For each cell histogram appended into block histogram, add the logical index into an array called logicalcellindex.
- Logical cell index can be obtained by following formula:

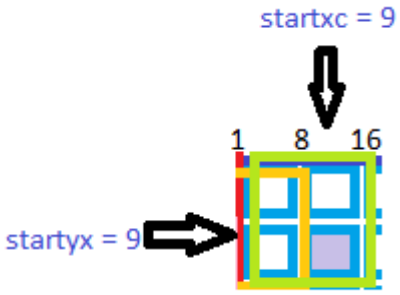
a. get the x,y of the starting pixel of the cell

```
x = startxb + startxc - 1;
y = startyb + startyc - 1;
```

suppose we are looking at the purple cell.



Startxb = 17 (green block is third block in column)
startyb = 1 (green block is in first row)



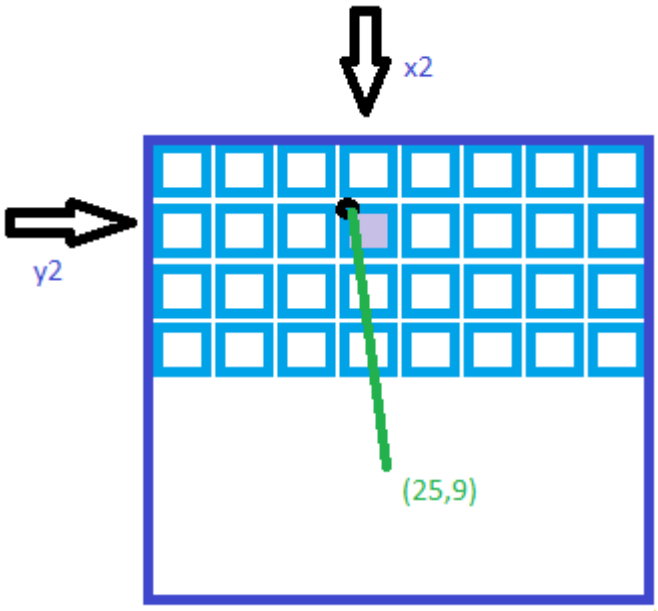
startxc = 9 (2nd cell from left)
startyb = 9 (2nd cell from top)

so find x,y of pixel by adding the two and subtracting one (because one based indexing)
x = 25
y = 9

b. find the index in terms of cells (8x16 format)
divide by 8 simple

```
x2 = (x-1) / 8 + 1;
y2 = (y-1) / 8 + 1;
```

x2 = 4
y2 = 2

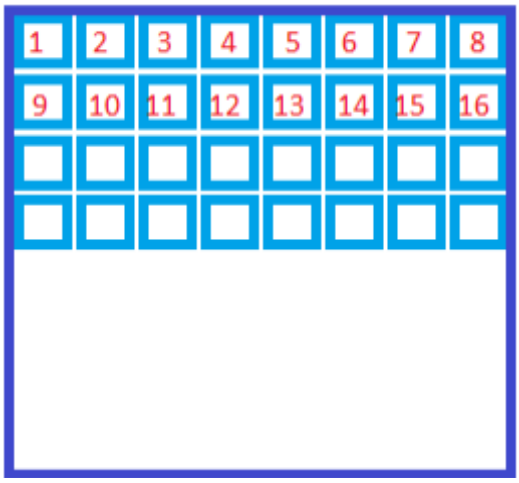


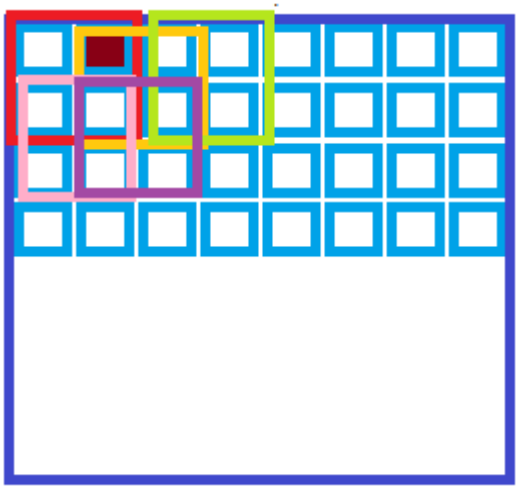
b. convert x2, y2 combination into one number
(1,1) -> 1, (2,1)->9,
mutlipy row by 8 and add column, simple.

```
logicalindex = (x2 - 1) * 8 + (y2);
```

Visualizing hog features

1. run a loop for cell index from 1 to 128 cells
2. find this cell index in the logical cell index array





logical cell index:

1	2	9	10	2	3	11	12		

The order is defined blockwise. Remember: each cell histogram is combined to block. So (1,2,9,10) -> red block, (2,3,10,11) -> yellow block.

For cell 2 (brown), locations = 2, 5

The brown cell is included in 2 blocks (red, yellow). Each cell can be include in max 4 blocks.

3. find corresponding histograms and find average

Each histogram is 9 values. So the index in hog features (3780 values), can be found by mutliplying by 9 and extracting 9 continuous values.

```
binsize = 9
hogindex = (location-1)*binsize + 1;
histogram = hogfeatures(1,hogindex: hogindex + binsize - 1);
```

So each cell has one histogram per block.
2 blocks -> 2 histograms.
So, add all the histograms and divide by total.

4. plot the average histogram

For every direction in average histogram, draw line with length = magnitude and direction = bin at the center of the cell.

a. find center of the cell

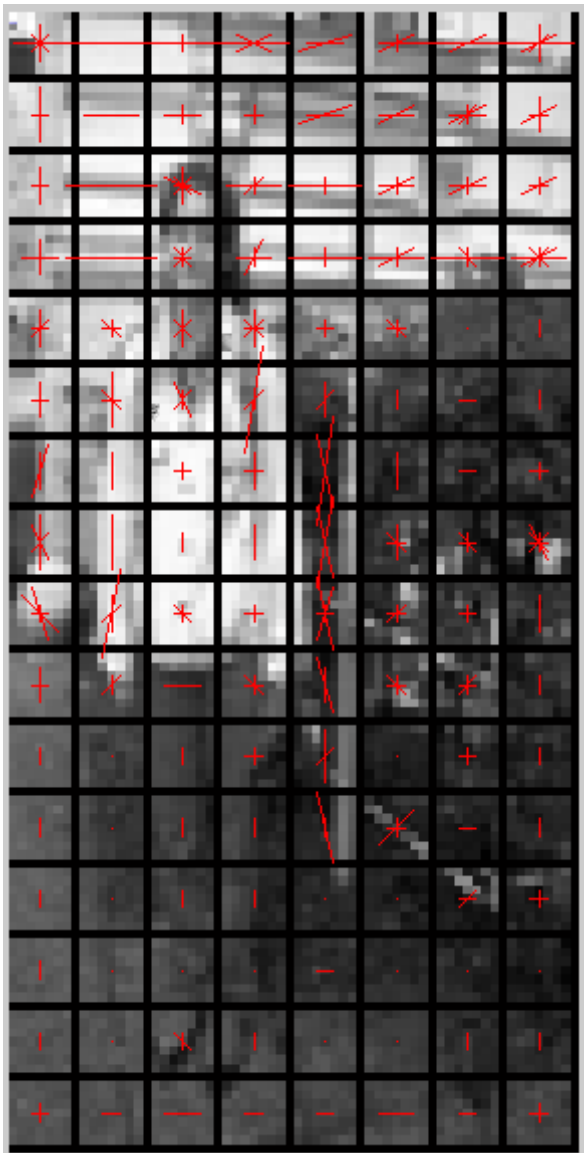
now, we need to do reverse operation of logical cell index.

```
%get x, y in terms of cell first
x = floor((cellindex - 1)/8) + 1;
y = (cellindex - (x-1) * 8);
%mutliply by 8
x = (x-1)*8 + 1;
y = (y-1)*8 + 1;
%add 3 to get the center
x = x + 3;
y = y + 3;
```

b. for all angles in the bin (1-9)

```
xcomponent = round(length/2 * cosd(angle * 20 - 10));
ycomponent = round(length/2 * sind(angle * 20 - 10));
%find x and y ranges
x1 = x-xcomponent;
y1 = y-ycomponent;
x2 = x+xcomponent;
y2 = y+ycomponent;
%draw line
plot([y1,y2],[x1,x2]
```

In magnified view:



As, you can see the lines show the overall direction of the edges for each cell in the grid.

*to display grid:

```
image(8:8:end,:,:) = 0;         %# Change every eighth row to black
image(:,8:8:end,:) = 0;
```

Default detector:

- 64x128 detection window
- rgb color space with no gamma correction
- [-1 0 1] gradient filter with no smoothing
- 16x16 pixels per blocks of four 8x8 pixel cells
- block spacing stride of 8 pixels (4 fold coverage of each cell)
- gaussian spatial window with sigma=8 on the block magnitude
- linear gradient voting into 9 orientation bins 0 – 180
- L2-Hys(Lowe-style clipped L2 norm) block normalization on block histograms
- linear svm

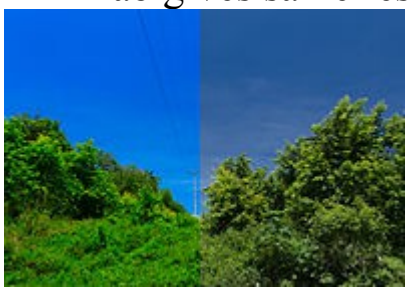
Extra steps:

1. **Color space**

- rgb preferred over grey scale

take each channel independently and calculate gradient in each channel. Take the one with largest norm as the pixel's gradient.

- lab gives same result as rgb



lab color space us designed to approximate human vision. Here, lab color space enhancement(right) is shown to given image (left)

- square root gamma compression gives low improvement

gamma encodes luminence

2. **Normalization of blocks**

-to reduce illumination and foreground-background constrast.
-similar to histogram stretching

a. L2-norm
v = v / sqrt(2-norm of v + epsilon)

b. L2-Hys
clipping maximum value to 0.2 and renormalizing

3. **window size**

16 pixels of boundary around the human

SVM

Type: gaussian kernel svm
soft C=0.01
SVMLight package http://svmlight.joachims.org/ - provides implementations in many languages