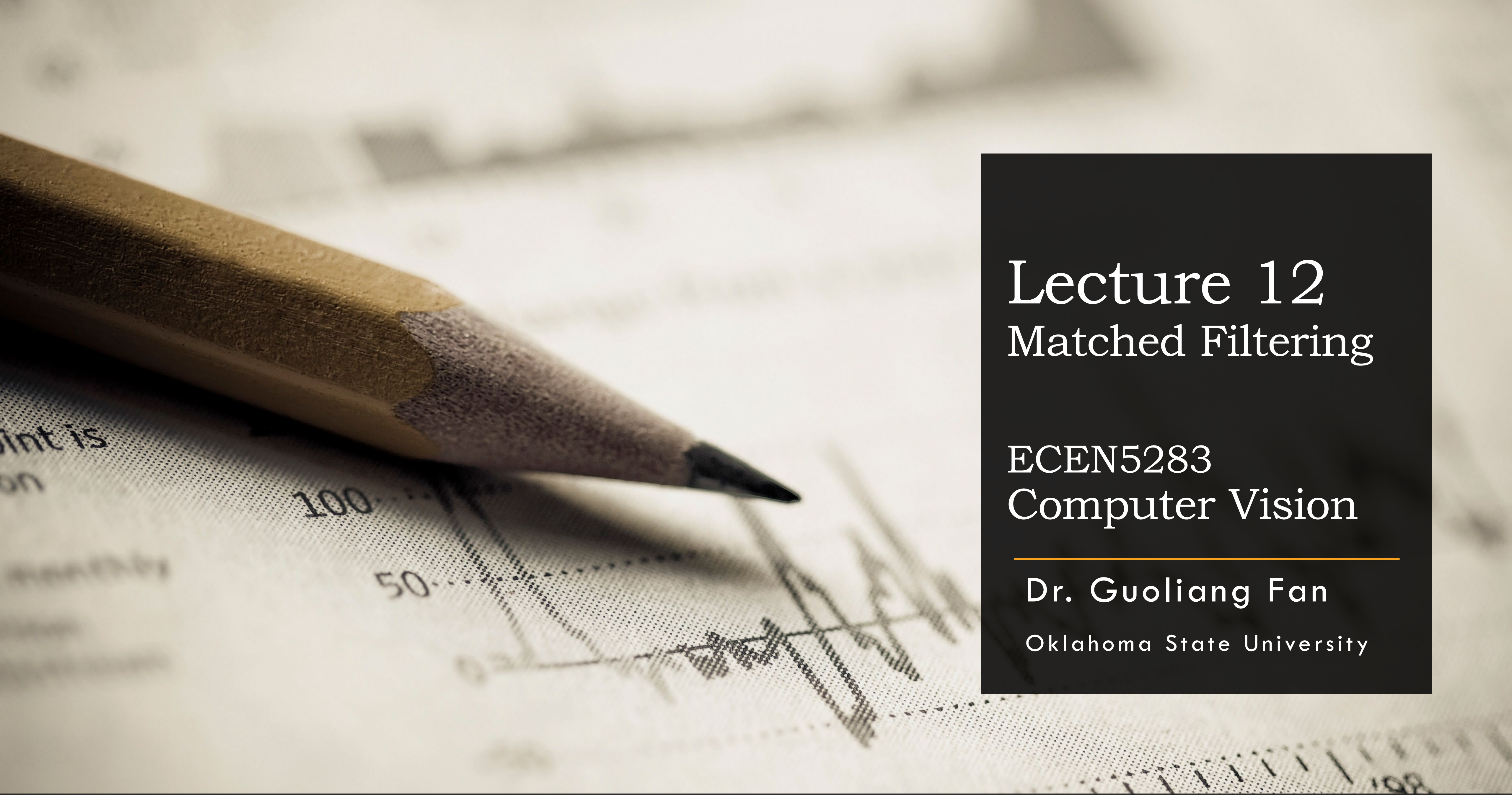


Lecture 12
Matched Filtering

ECEN5283
Computer Vision

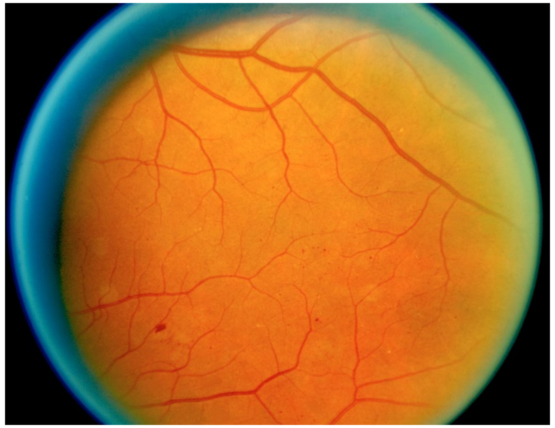
Dr. Guoliang Fan
Oklahoma State University

Goals

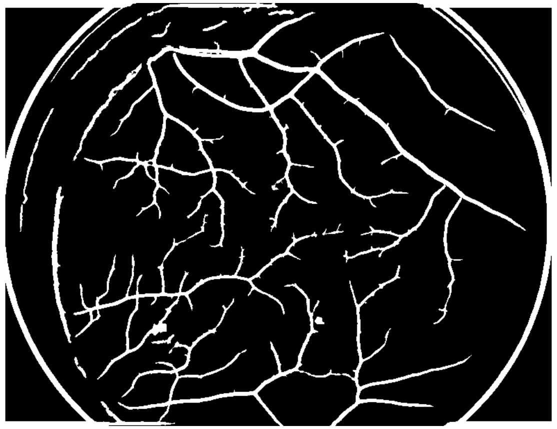
To review the Canny edge detector.

To implement the matched filter for blood vessel extraction in retinal images.

To develop a length filter to remove weak (isolated) edge points



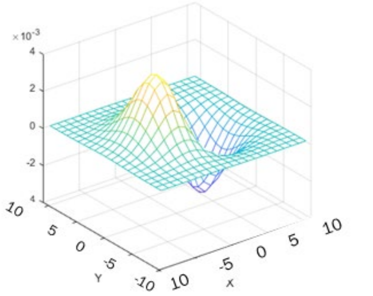
Retinal image

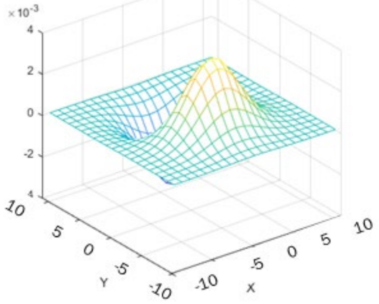


Matched filter

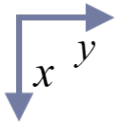
Canny Edge Detection

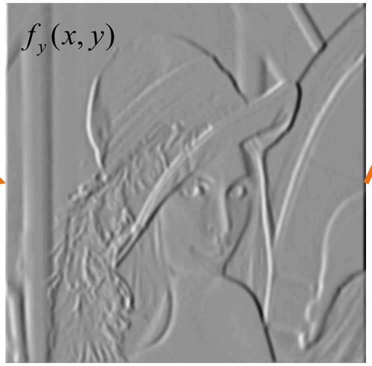
$$\nabla_x G_\sigma = \frac{-x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$





$$\nabla_y G_\sigma = \frac{-y}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$





$$|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2}$$

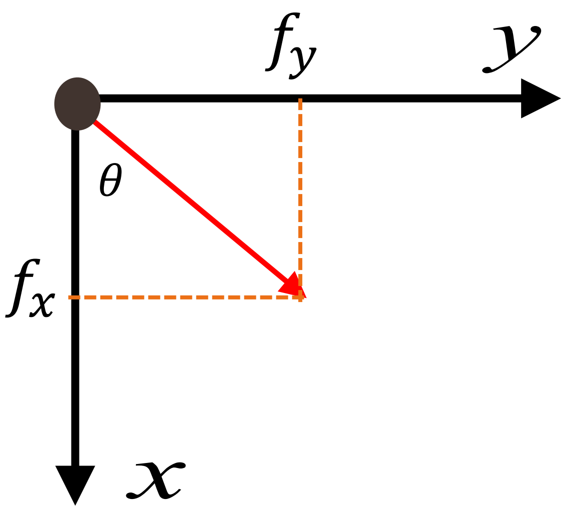
(gradient map)

Thresholding

Non-maximum suppression

Note: Non-maximum suppression is very time-consuming, It is a good idea to do it only for those pixels that have sufficiently large gradient magnitudes.

Special Cases of Gradient Orientation



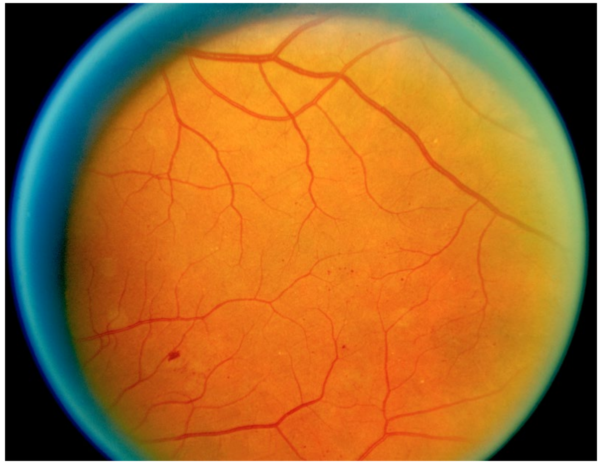
$|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2}$ Gradient magnitude

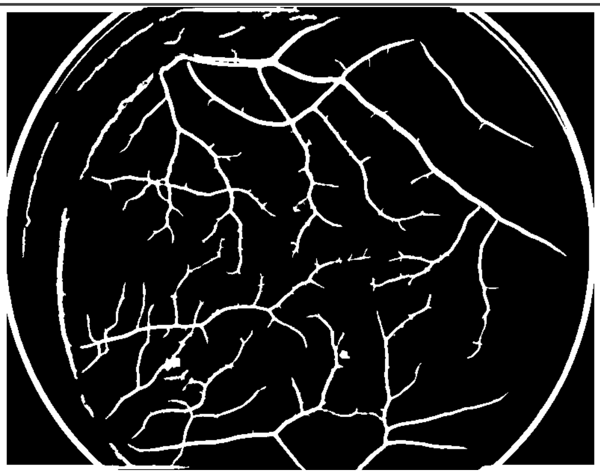
$\theta(x,y) = \tan^{-1}\left(\frac{f_y}{f_x}\right)$ $(\pm\pi)$ Gradient orientation

→	$\theta = \frac{\pi}{2}$	$f_x = 0, f_y > 0$
←	$\theta = -\frac{\pi}{2}$	$f_x = 0, f_y < 0$
↓	$\theta = 0$	$f_y = 0, f_x > 0$
↑	$\theta = \pi$	$f_y = 0, f_x < 0$
↘	$\theta = \frac{\pi}{4}$	$f_x = f_y > 0$
↖	$\theta = -\frac{3\pi}{4}$	$f_x = f_y < 0$
↙	$\theta = -\frac{\pi}{4}$	$f_x > 0, f_y < 0, \lvert f_x \rvert = \lvert f_y \rvert$
↗	$\theta = \frac{3\pi}{4}$	$f_x < 0, f_y > 0, \lvert f_x \rvert = \lvert f_y \rvert$

Why Matched Filter?

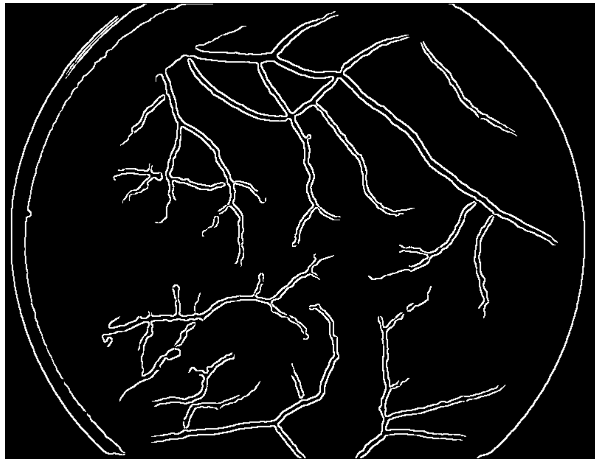
Retinal image





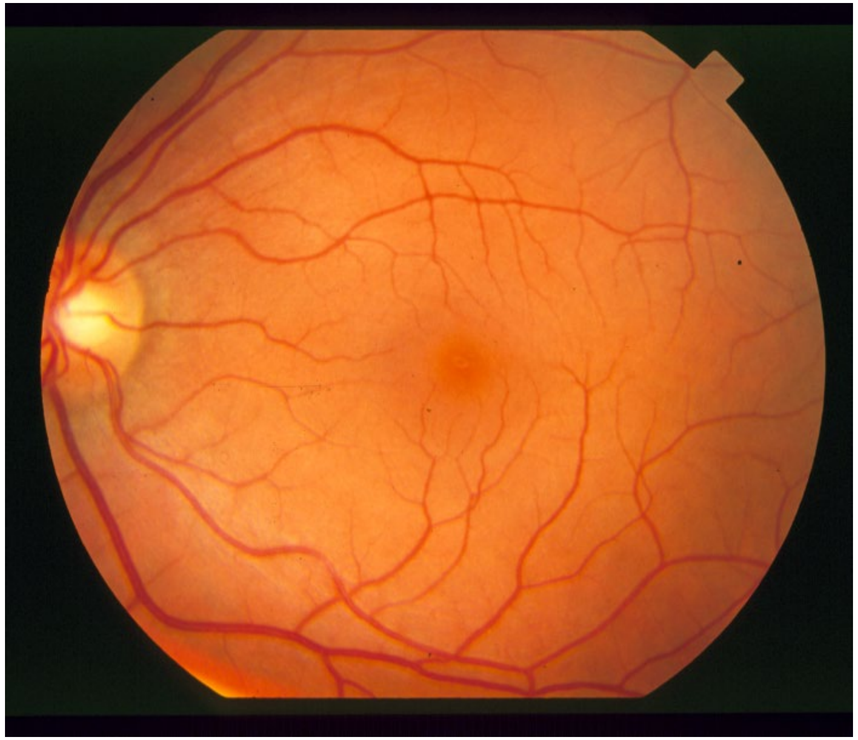
Matched filter

Canny detection

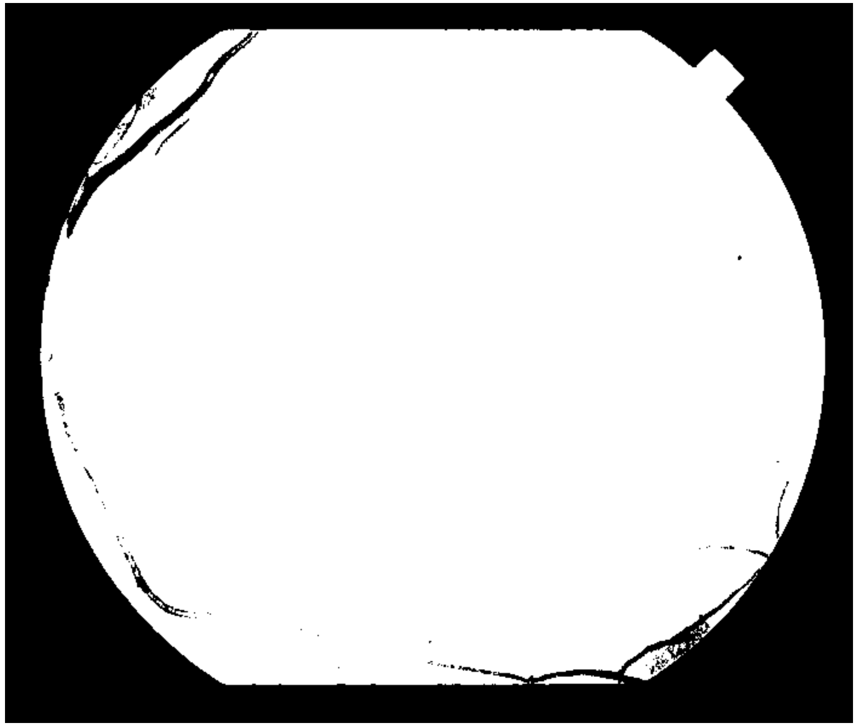


LoG detection

Why a simple thresholding method won't work?

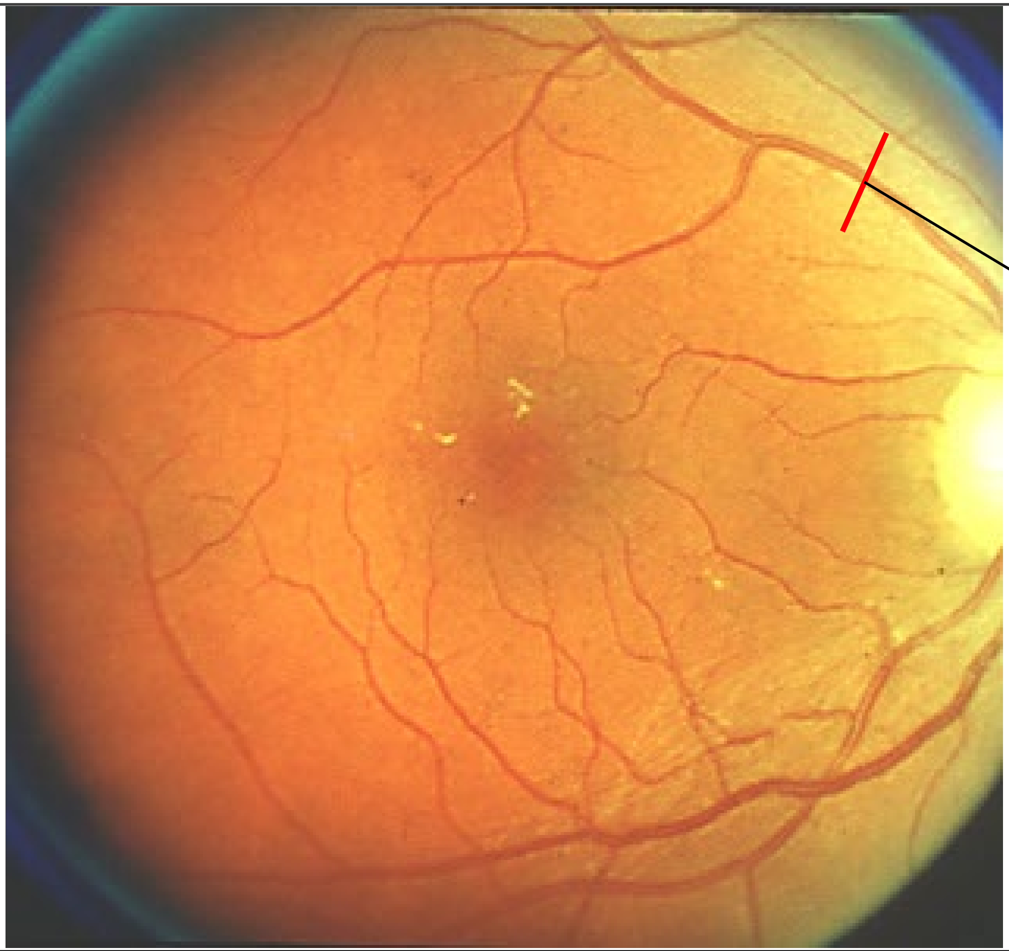


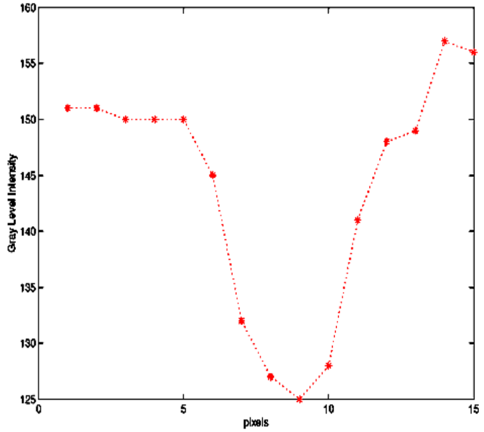
An original image



Binary images with different thresholds

Motivation of Matched Filtering

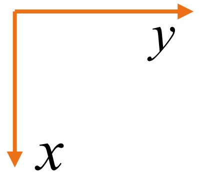




What is the Matched Filter?

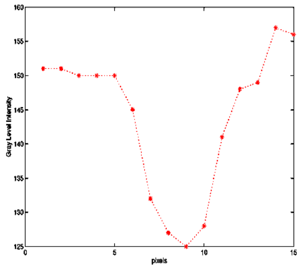
The matched filter is defined as a Gaussian function along one direction and constant along another direction

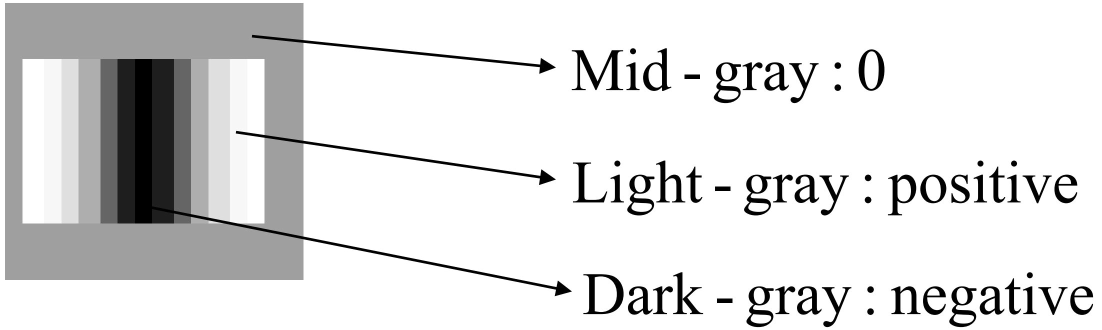
- Assumption: the cross-section of blood vessels is 1D Gaussian-like.



$$G(x,y) = -\frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{y^2}{2\sigma^2}\right)} - m_0 \ (x \in [-x_0, x_0])$$

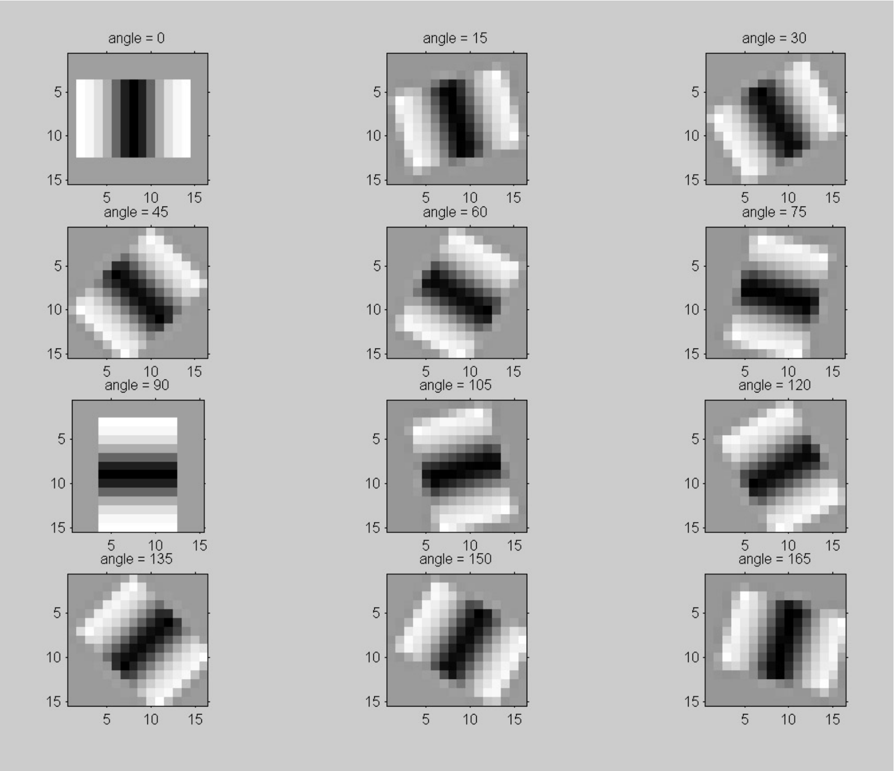
(m_0 is chosen to make kernel $G(x,y)$ have zero mean and m_0 determines the height of kernel)





Matched Filter Group (1)

$$\{G(x,y)_{\theta_1}, G(x,y)_{\theta 2}, \ldots, G(x,y)_{\theta_N}\}$$



Matched Filter Group (2)

In the Matlab, we can use the IMROTATE function to create a group of matched filters.

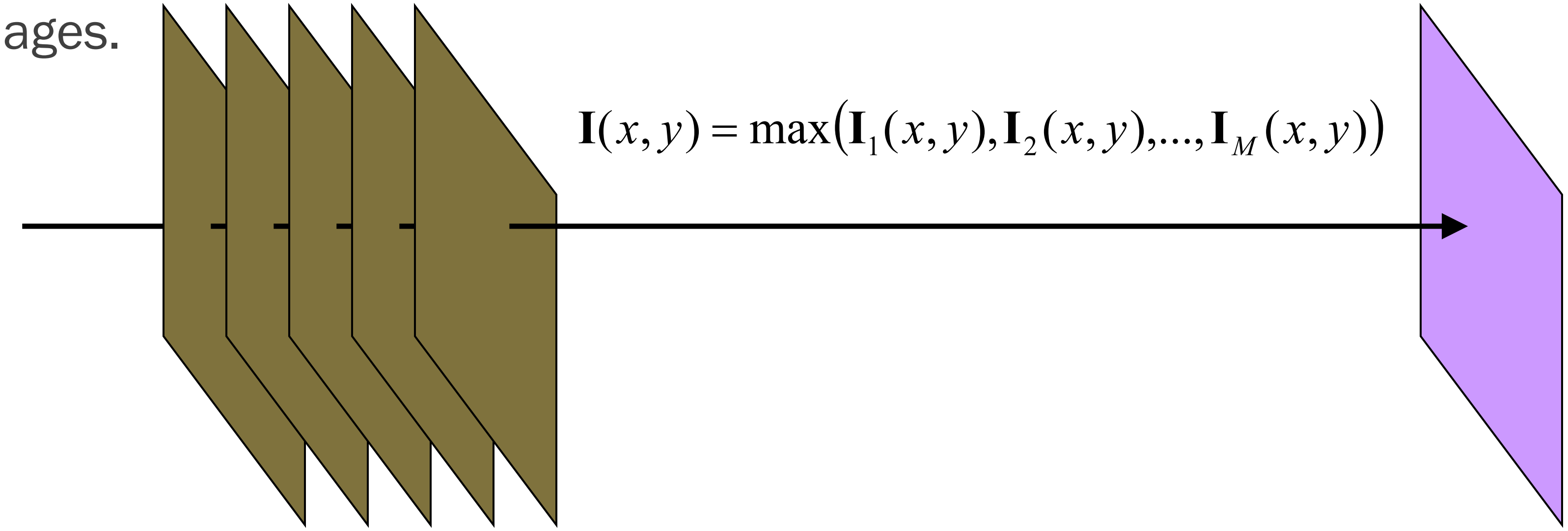
B = IMROTATE(A, ANGLE, 'bicubic', 'crop')

- Because kernel rotation will increase the effective size of the kernel. Therefore, the initial kernel is enlarged by adding more zeros around non-zero values.
- We would like to have a square-shaped odd-sized kernel for all matched filters that gives zero-phase filtering.

Matched Filtering for Blood Vessel Extraction

Step 1: After we create the matched filter group, we apply each kernel to the original image.

Step 2: Then we fuse all filtered images by assigning the pixel value to be the maximum one across all filtered images.

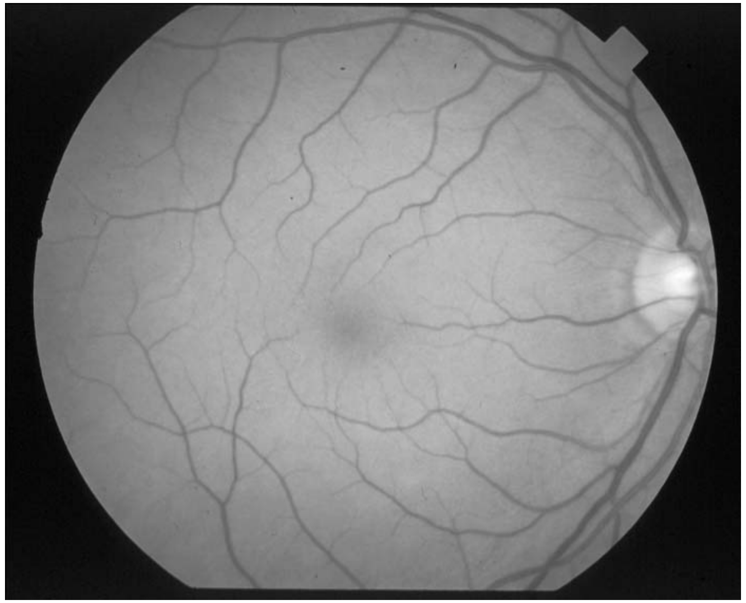


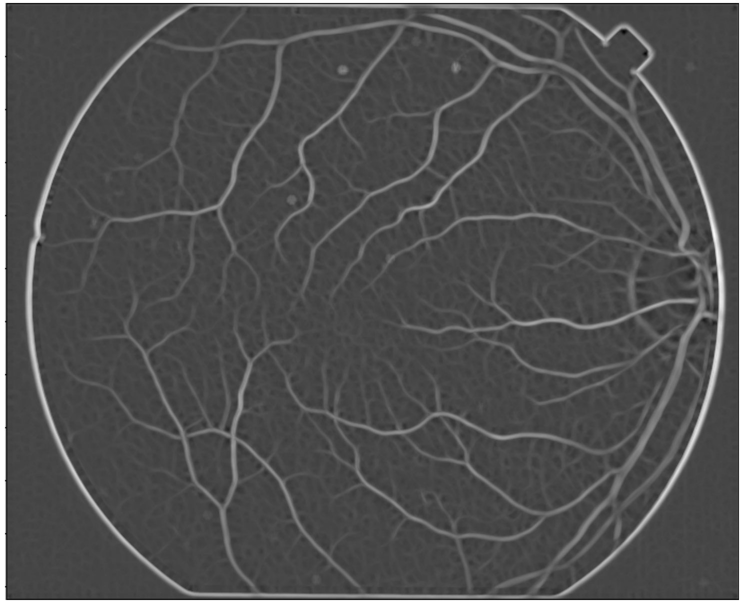
$$\mathbf{I}(x,y) = \max\left(\mathbf{I}_1(x,y), \mathbf{I}_2(x,y), \ldots, \mathbf{I}_M(x,y)\right)$$

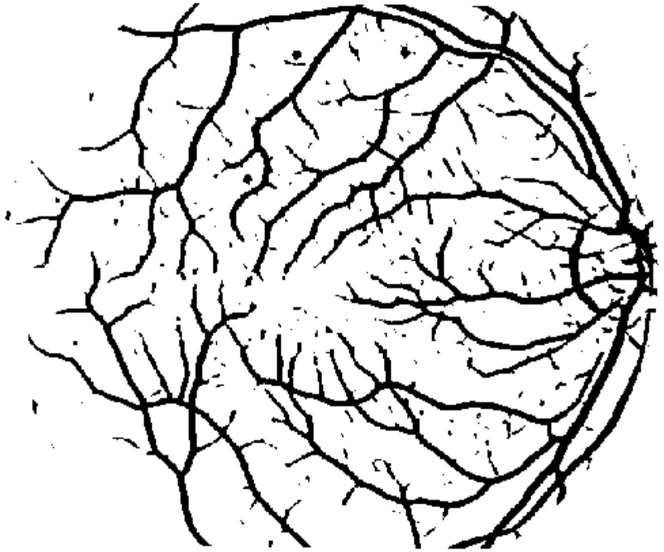
Step 3: Find an appropriate threshold (Matlab “GRAYTHRESH”).

Step 4: Use that threshold to binarize the image (Matlab “IM2BW”).

Fusion and Thresholding Results







Length Filtering using Matlab

Step 1: We use the Matlab function (BWLABEL) to find all connected component edge pixels in the image.

- L = BWLABEL(BW,N) returns a matrix L, of the same size as BW, containing labels for the N-connected (N=4 or 8) components in BW.

```
BW =

     1     1     1     0     0     0     0     0
     1     1     1     0     1     1     0     0
     1     1     1     0     1     1     0     0
     1     1     1     0     0     0     1     0
     1     1     1     0     0     0     1     0
     1     1     1     0     0     0     1     0
     1     1     1     0     0     1     1     0
     1     1     1     0     0     0     0     0
```

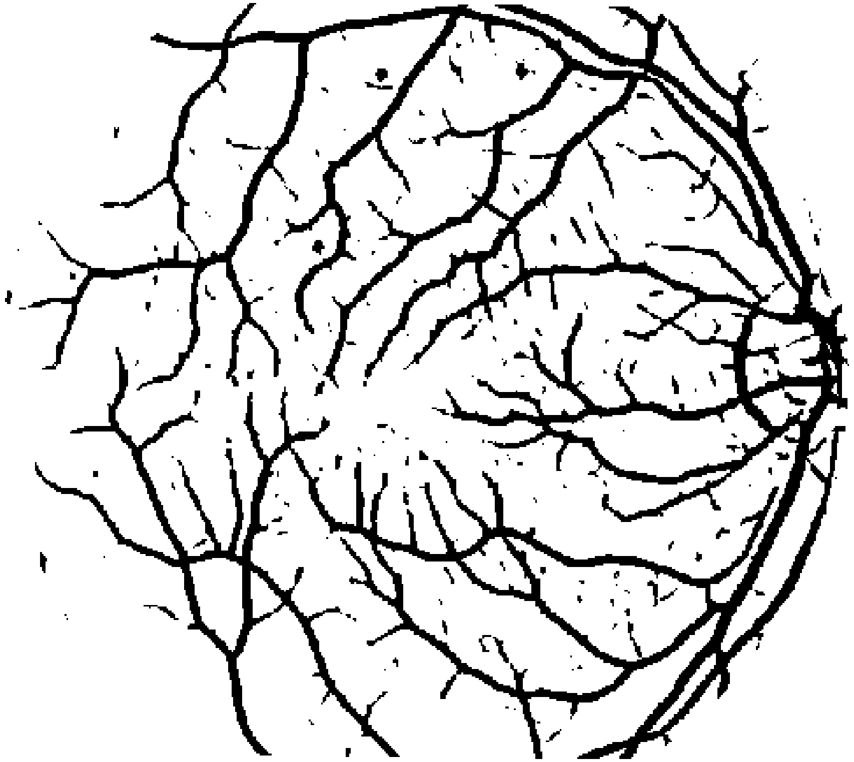
```
L =

     1     1     1     0     0     0     0     0
     1     1     1     0     2     2     0     0
     1     1     1     0     2     2     0     0
     1     1     1     0     0     0     3     0
     1     1     1     0     0     0     3     0
     1     1     1     0     0     0     3     0
     1     1     1     0     0     3     3     0
     1     1     1     0     0     0     0     0
```

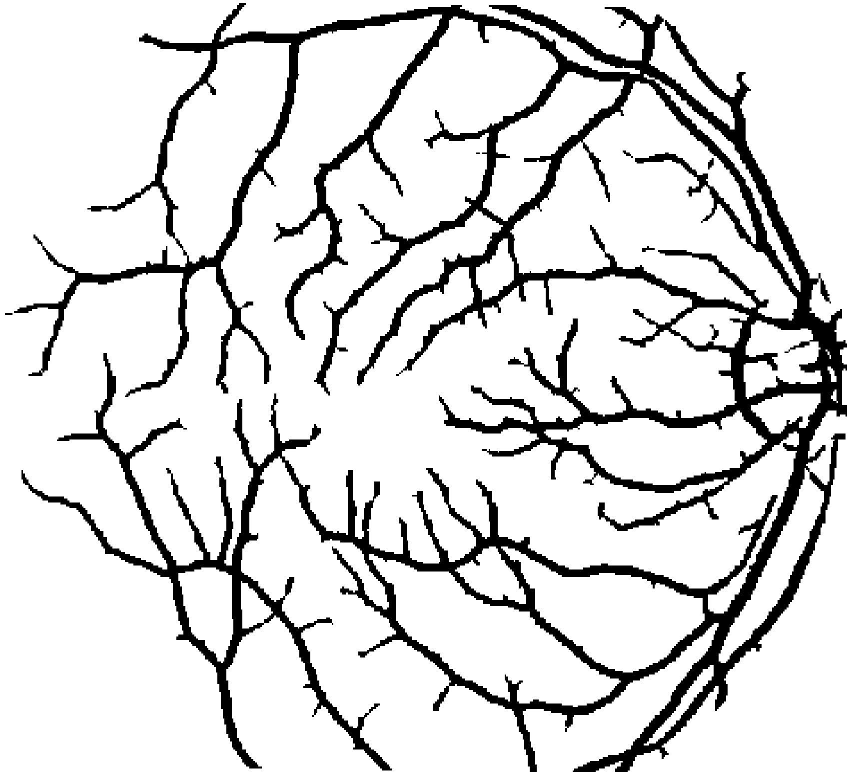
Step 2: We can use FIND to find the pixel coordinates and the number of pixel for certain class label

- [r, c]=find(L==2); ([r c] returns the x-y coordinate of all pixels of class 2. The dimension of r and c show the number of pixels in class 2.

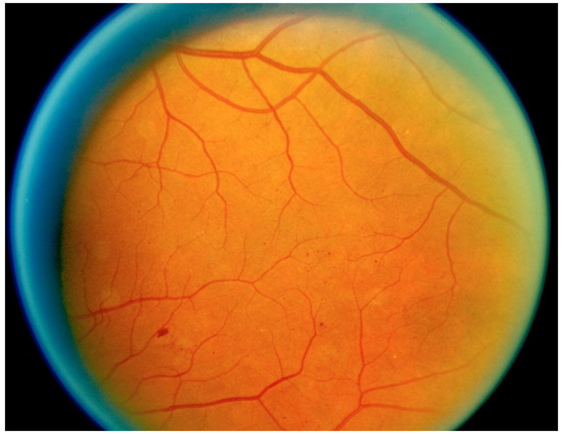
Step 3. We can delete the class labels with a small number of pixels.

Length Filtering Results

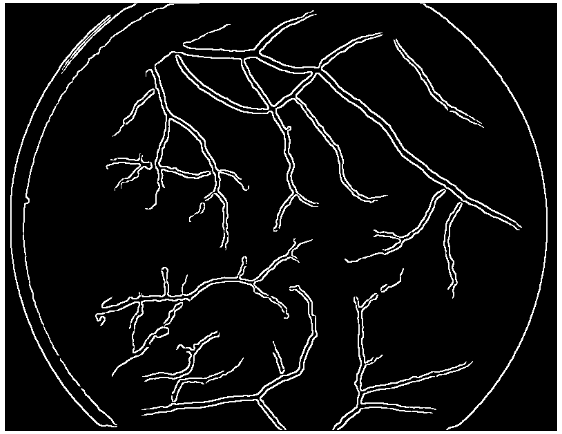




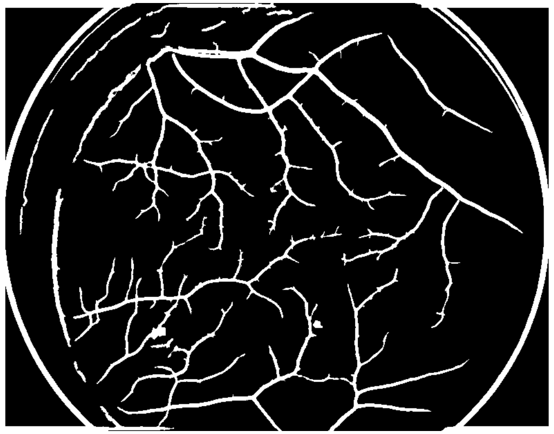
Project 2 (Due March 2, 2024)



Canny detection



LoG detection



Matched filter