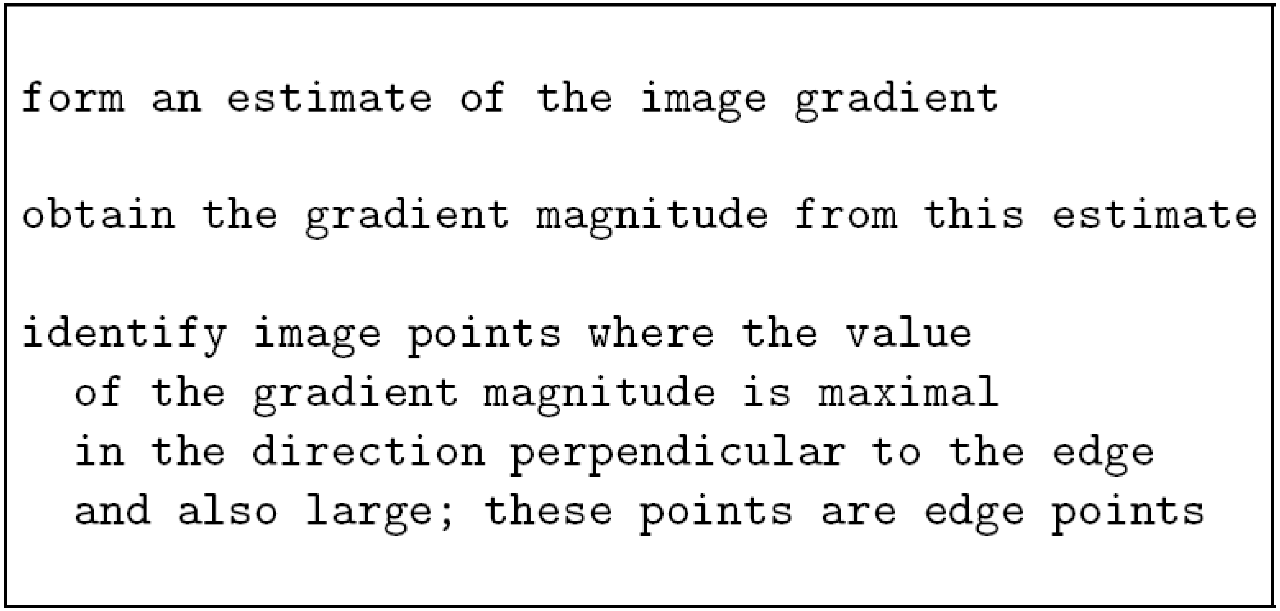
Lecture 14
Matched Filter for Edge Detection
ECEN 5283 Computer Vision

Dr. Guoliang Fan
School of Electrical and Computer Engineering
Oklahoma State University

Goals

- To review gradient-based Canny edge detection.
- To implement the matched filter for edge detection in retinal images, i.e., blood vessel extraction.
- To develop a length filter to remove weak (isolated) edge points.

Canny Edge Detection Algorithm

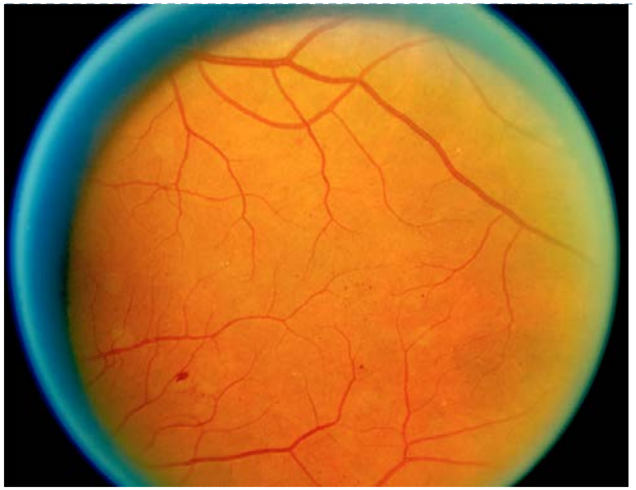


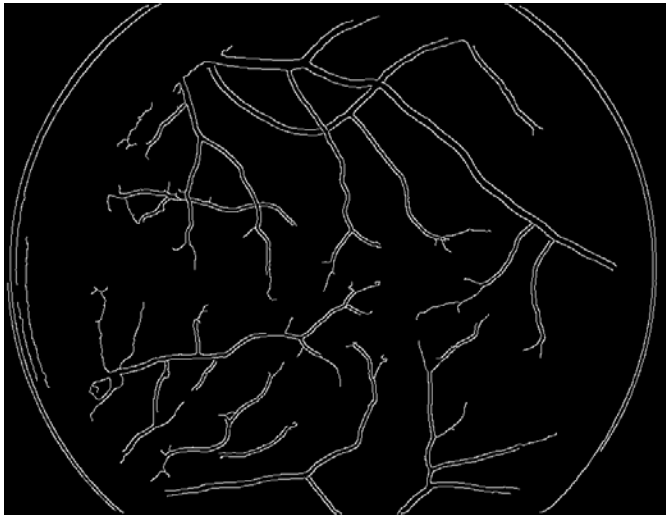
```
form an estimate of the image gradient

obtain the gradient magnitude from this estimate

identify image points where the value
  of the gradient magnitude is maximal
  in the direction perpendicular to the edge
  and also large; these points are edge points
```

Algorithm 9.1: *Gradient based edge detection.*

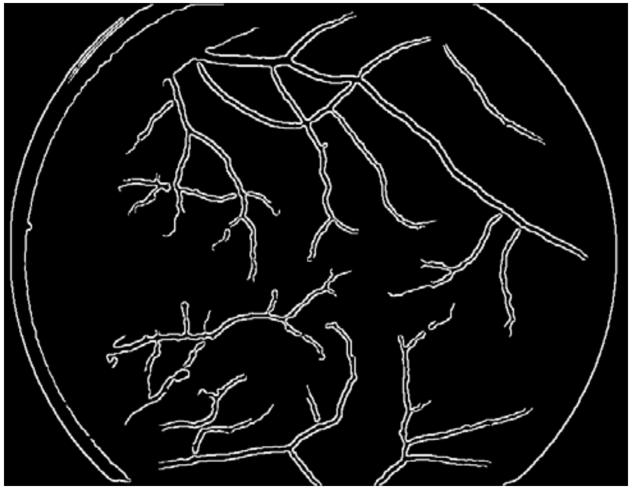
Note: Non-maximum suppression is very time-consuming, It is a good idea to do it only for those pixels that have sufficiently large gradient magnitudes.

Project 2 (Due Feb. 25, 2015)



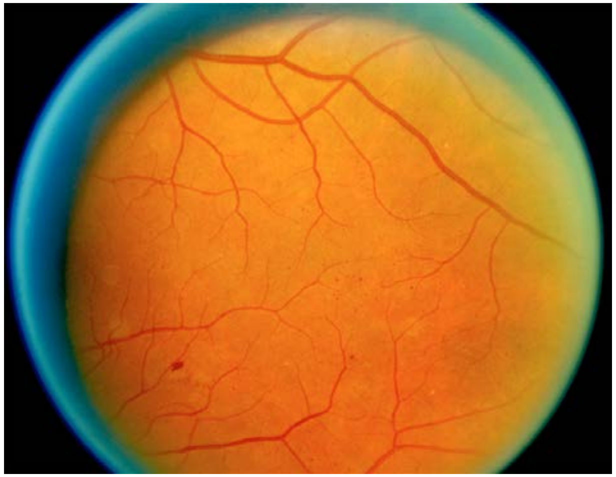


Canny detection

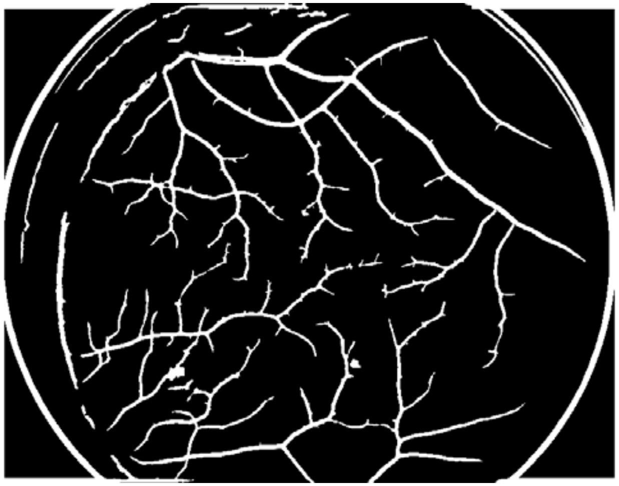


LoG detection

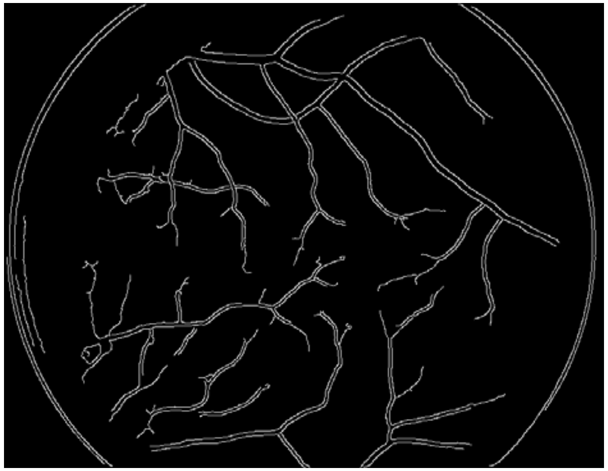
Why Matched Filter?



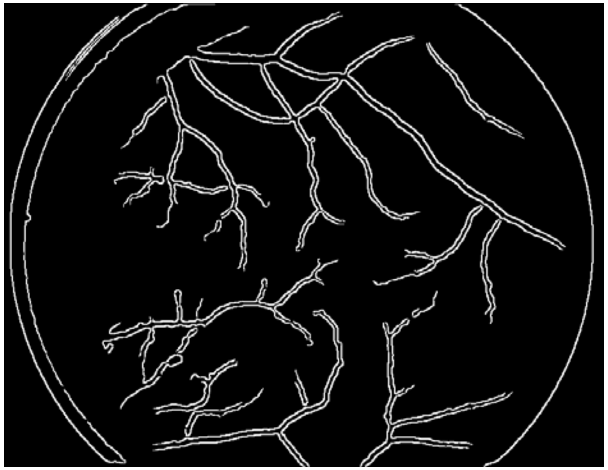
Retinal image



Matched filter

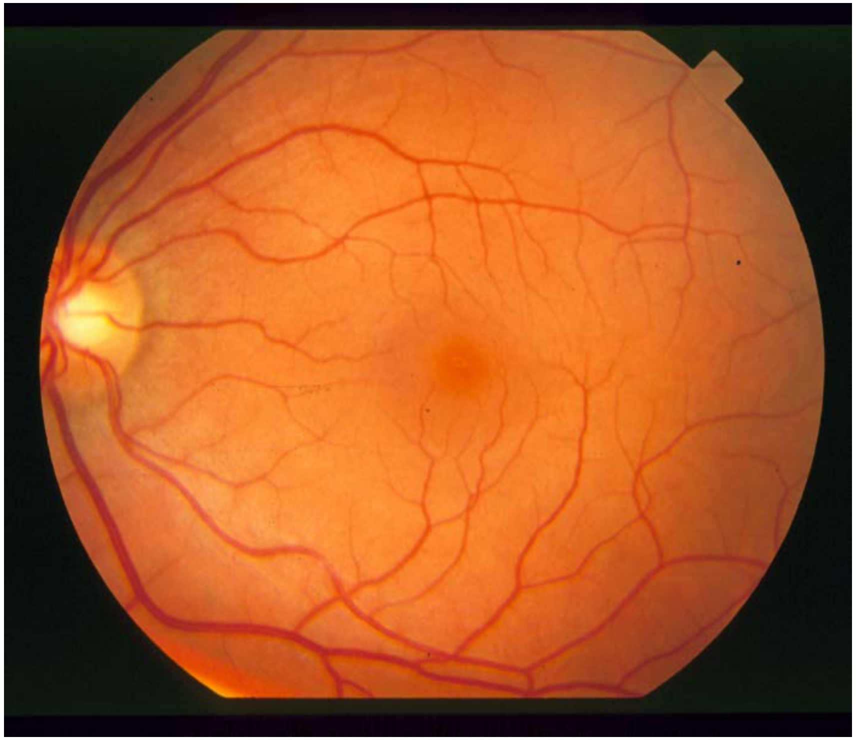


Canny detection

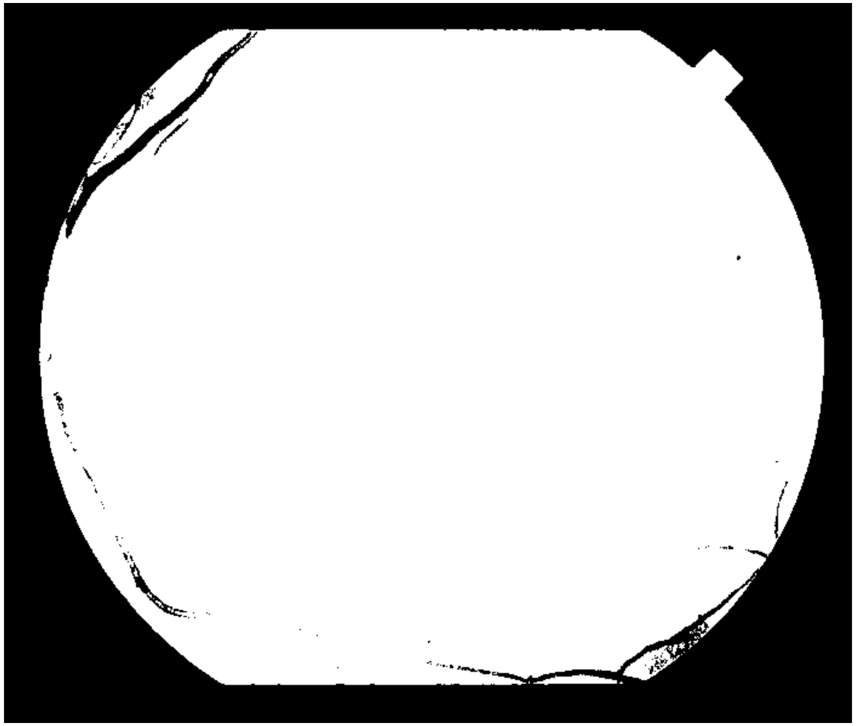


LoG detection

Why a simple thresholding method won't work?

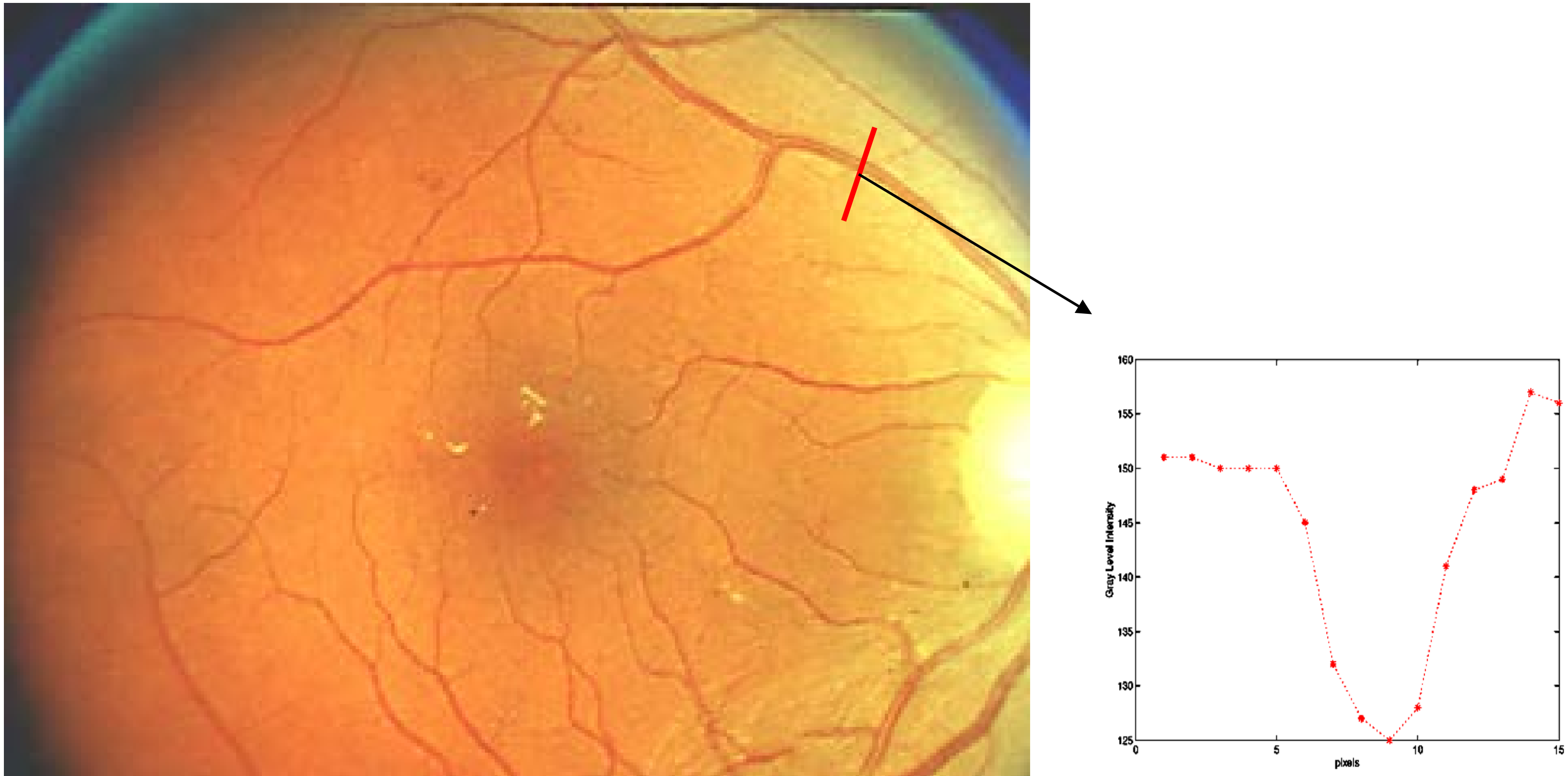


An original image



Binary images with different thresholds

Matched Filter Motivation

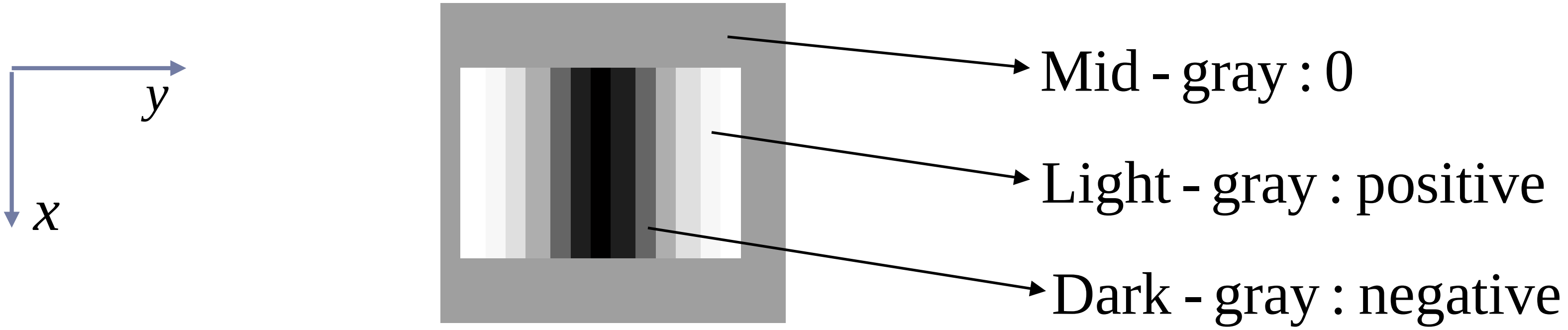


What is the Matched Filter?

- Assumption: the cross-section of blood vessels is Gaussian-like.
- The matched filter is defined as a Gaussian function along one direction and constant along another direction

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

(m_0 is chosen to make kernel $G(x,y)$ have zero mean)



Mid - gray : 0

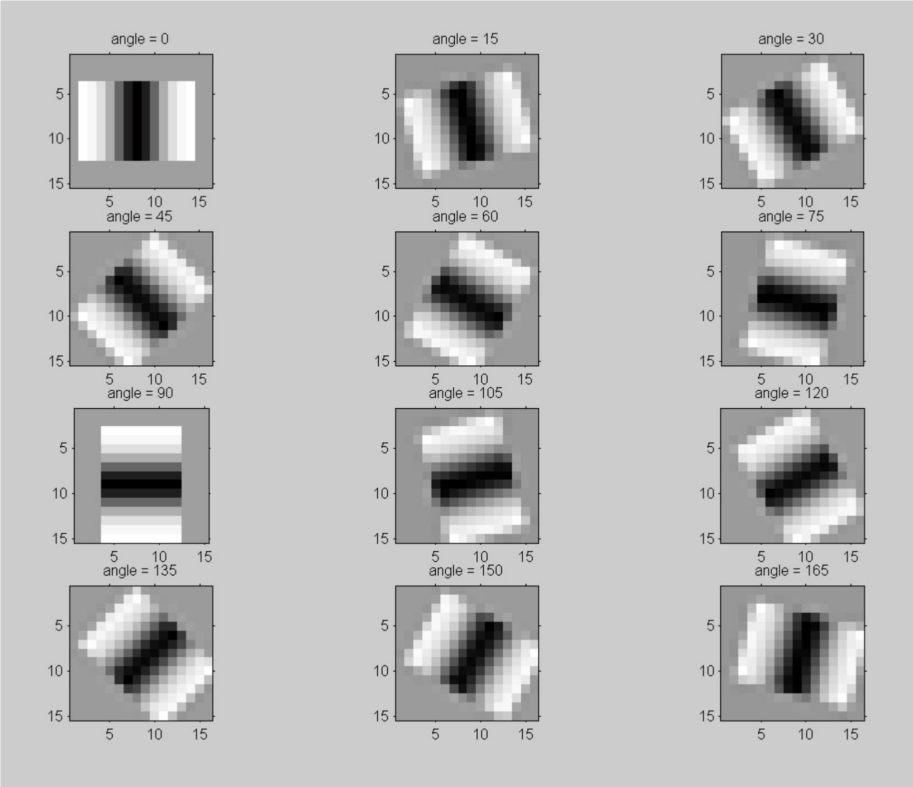
Light - gray : positive

Dark - gray : negative

Matched Filter Group (1)

- A group of rotated matched filters

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

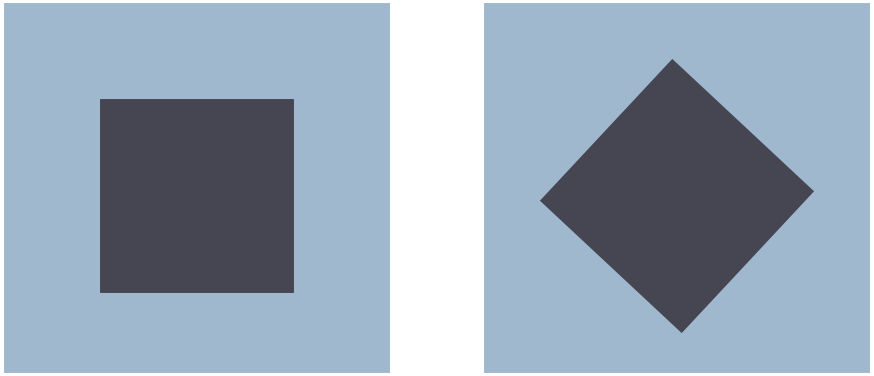


Matched Filter Group (2)

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

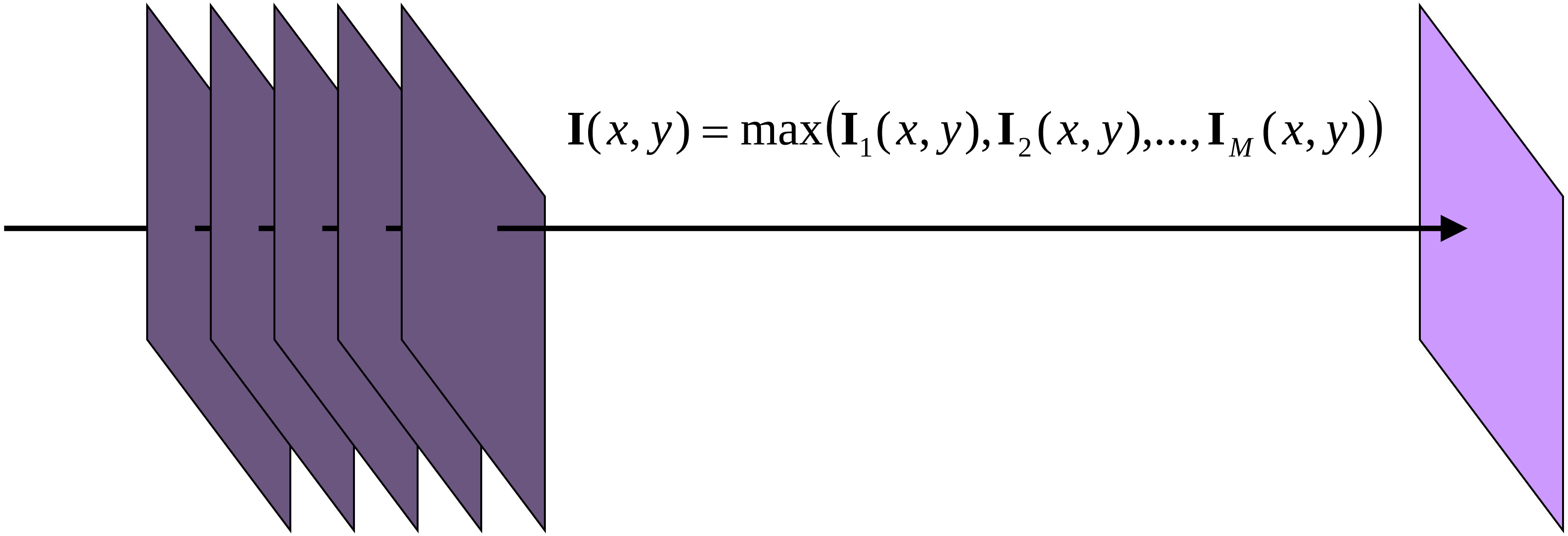
  B = IMROTATE(A, ANGLE, ’bicubic’, ’crop’)

  - 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.
  - Also, we would like to have a square-shaped odd-sized kernel for all matched filters that gives zero-phase filtering.



Matched Filtering for Edge Detection

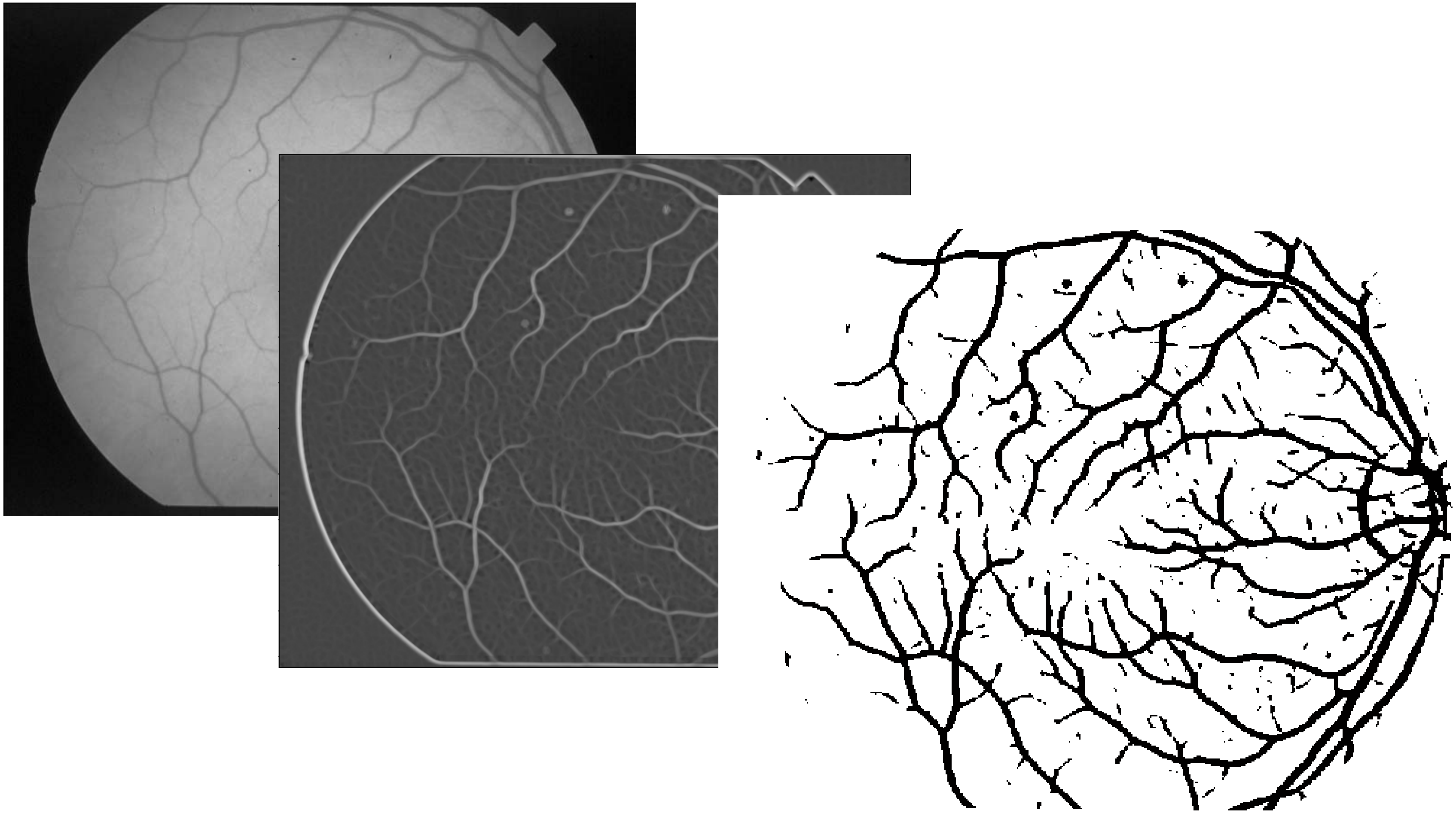
- Step 1: After we create the matched filter group, we apply each kernel to the original image.
- Step 2: Then we fuse all filtered images by assigning the pixel value to be the maximum one across all filtered images.



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

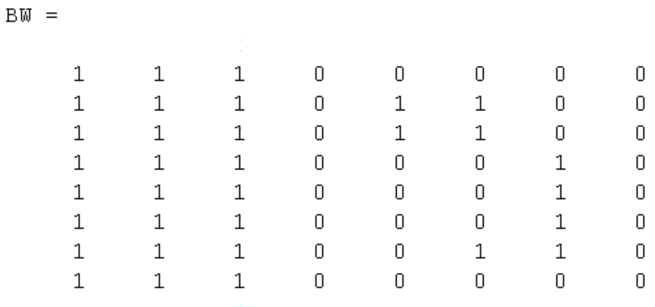
- Step 3: Find an appropriate threshold (Matlab "GRAYTHRESH").
- Step 4: Use that threshold to binarize the image (Matlab "IM2BW").

Matched Filtering Results

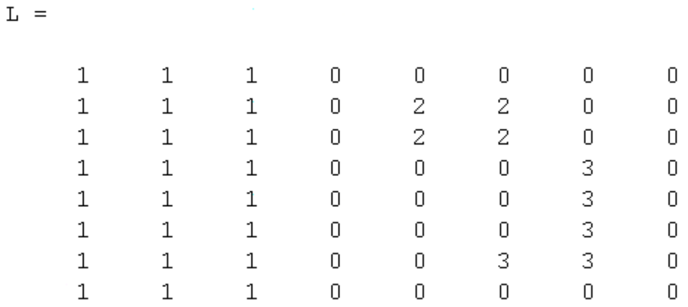


Length Filtering using Matlab

- Step 1: We can 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