# Main Algorithms

```
In[1]:= dir = NotebookDirectory[];
img = ColorConvert[
   Import[dir <> "DRIVE/training/images/21_training.tif"], "Grayscale"];
imgData = ImageData[img];
```

## Helper function

```
In[200]:=
PNGtoGray[filename_]:=Reverse[Map[Map[First,#]&,Import[filename,"Data"]]];
GIFtoGray[filename_]:=255-Reverse[Map[Map[First,#]&,Import[filename,"Data"]]];
showGrayImage[img_]:=Graphics[Raster[img/Max[img]],Frame→False];
showBinaryImage[img_]:=Graphics[Raster[img],Frame→False];
showBinaryOverlay[img_,mask_]:=Graphics[Raster[MapThread[MapThread[If[#2==1,{1,0,0},{#1,#1
 {img,mask}]],Frame→False];
showGrayOverlay[img_,mask_]:=Graphics[{Raster[img/Max[img]],Raster[Map[Map[{#,0,0,If[#==1,
```

## $S_{op}$

We remove noise while preserving most of the capillaries using geodesic reconstruction of the opened images into the original image $S_0$

Each structuring element $L_i$ (every 15) is a 15-pixel long 1-pixel wide. Its size is approximately the range of the diameter of the biggest vessels for 512 × 512 × 8 images of retinal angiography.

In the image $S_{op}$, every isolated round and bright zone whose diameter is less than 15 pixels has been removed. Being a supremum of openings by reconstruction this operation is an opening, called linear opening by reconstruction of size 15.

$$S_{op} = \gamma_{S_0}^{rec}\left(\mathrm{Max}_{i=1\ldots12}\{\gamma_{L_i}(S_0)\}\right)$$

My question:
Is this equation means
1. apply erosion with structure elements $L_1$ ... $L_{12}$ to $S_0$
2. Find the maximum for every pixel
3. apply geodesic opening with marker image being original image

$$\gamma_{S_0}^{rec} = \text{geodesic opening}$$
$$\gamma_{S_0}^{rec}(S) = \sup_{d\in\mathbb{N}}\left(\Delta_{S_0}^{d}(S)\right) \quad // \; S_0 = \text{marker image}$$

## Step 1: Construct $L_i$

In[4]:=
```
angles = Table[{Cos[t], Sin[t]}, {t, 0, 11 π / 12, π / 12}]
```

Out[4]=

$$\left\{\{1, 0\}, \left\{\frac{1+\sqrt{3}}{2\sqrt{2}}, \frac{-1+\sqrt{3}}{2\sqrt{2}}\right\}, \left\{\frac{\sqrt{3}}{2}, \frac{1}{2}\right\}, \left\{\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right\}, \left\{\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}, \left\{\frac{-1+\sqrt{3}}{2\sqrt{2}}, \frac{1+\sqrt{3}}{2\sqrt{2}}\right\}, \{0, 1\}, \left\{-\frac{-1+\sqrt{3}}{2\sqrt{2}}, \frac{1+\sqrt{3}}{2\sqrt{2}}\right\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}, \left\{-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right\}, \left\{-\frac{\sqrt{3}}{2}, \frac{1}{2}\right\}, \left\{-\frac{1+\sqrt{3}}{2\sqrt{2}}, \frac{-1+\sqrt{3}}{2\sqrt{2}}\right\}\right\}$$

In[69]:=
```
Graphics[{ Point[angles]}]
```

Out[69]=

In[10]:=
```
linearStrut =
  Table[Round[{Cos[t], Sin[t]} * #] &/@ (Range[15] - 8), {t, 0, 11 π / 12, π / 12}];
```

In[49]:=
```
kernelFromStructure[structure_] := Block[
   {size = 2 Max @ Abs @ MinMax @ structure + 1},
   Transpose@ReplacePart[Table[0, size, size],
    Thread[Append[structure, {0, 0}] + (size + 1)/2 → 1]]
]
```

In[78]:=
```
GraphicsRow[showBinaryImage[kernelFromStructure[#]] &/@ linearStrut]
```

Out[78]=

In[21]:=
```
kernels = kernelFromStructure[#] &/@ linearStrut;
```

## Step 2: Applying Opening with $L_i$

In[22]:=
```
AbsoluteTiming[erosionByLinearStrucImageList = Erosion[img, #] &/@ kernels]
```

In[28]:=
```
AbsoluteTiming[erosionByLinearStrucImageDataList =
  ImageData[#] &/@ erosionByLinearStrucImageList]
```

In[80]:=
```
maxErosion = MapThread[Max, erosionByLinearStrucImageDataList, 2];
```

In[81]:=
```
maskImage = Image[maxErosion]
```

But the built-in GeodesicOpening doesn't have imageMarker.
Therefore, I implemented GeodesicOpening myself.

$\gamma_{S_0}^{rec}$ = geodesic opening

$\gamma_{S_0}^{rec}(S) = \sup\limits_{d \in \mathbb{N}}\left(\Delta_{S_0}^{d}(S)\right)$ // $S_0$ = marker image

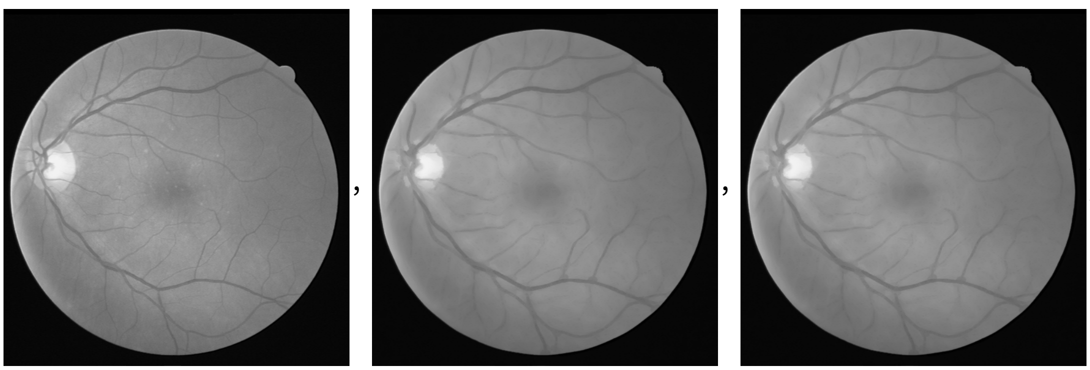
Pick the maximum until the Dilation doesn't change

```
geodesicOpening[marker_, mask_]:= FixedPointList[GeodesicDilation[#, mask]&, marker]
```

In[89]:=
```
geodesicOpening[marker_, mask_]:=
    MapThread[Max, ImageData/@ FixedPointList[GeodesicDilation[#, mask]&, marker], 2]
```

In[82]:=
```
markerImage = img;
```

In[83]:=
```
geodesicDilationList = geodesicOpening[markerImage, maskImage]
```

Out[83]= {  }

In[84]:=
```
geodesicDilationImageDataList = ImageData /@ geodesicDilationList
```

Out[84]=
```
{{ … 1 … }, { … 1 … }, { … 1 … }}
```
large output | show less | show more | show all | set size limit...

In[95]:=
```
AbsoluteTiming[SopData = geodesicOpening[markerImage, maskImage]]
```

Out[95]=
```
{0.271236, { … 1 … }}
```
large output | show less | show more | show all | set size limit...

In[96]:=
```
Sop = Image[SopData]
```

In[99]:=
```
Dimensions /@ {erosionByLinearStrucImageDataList}
```

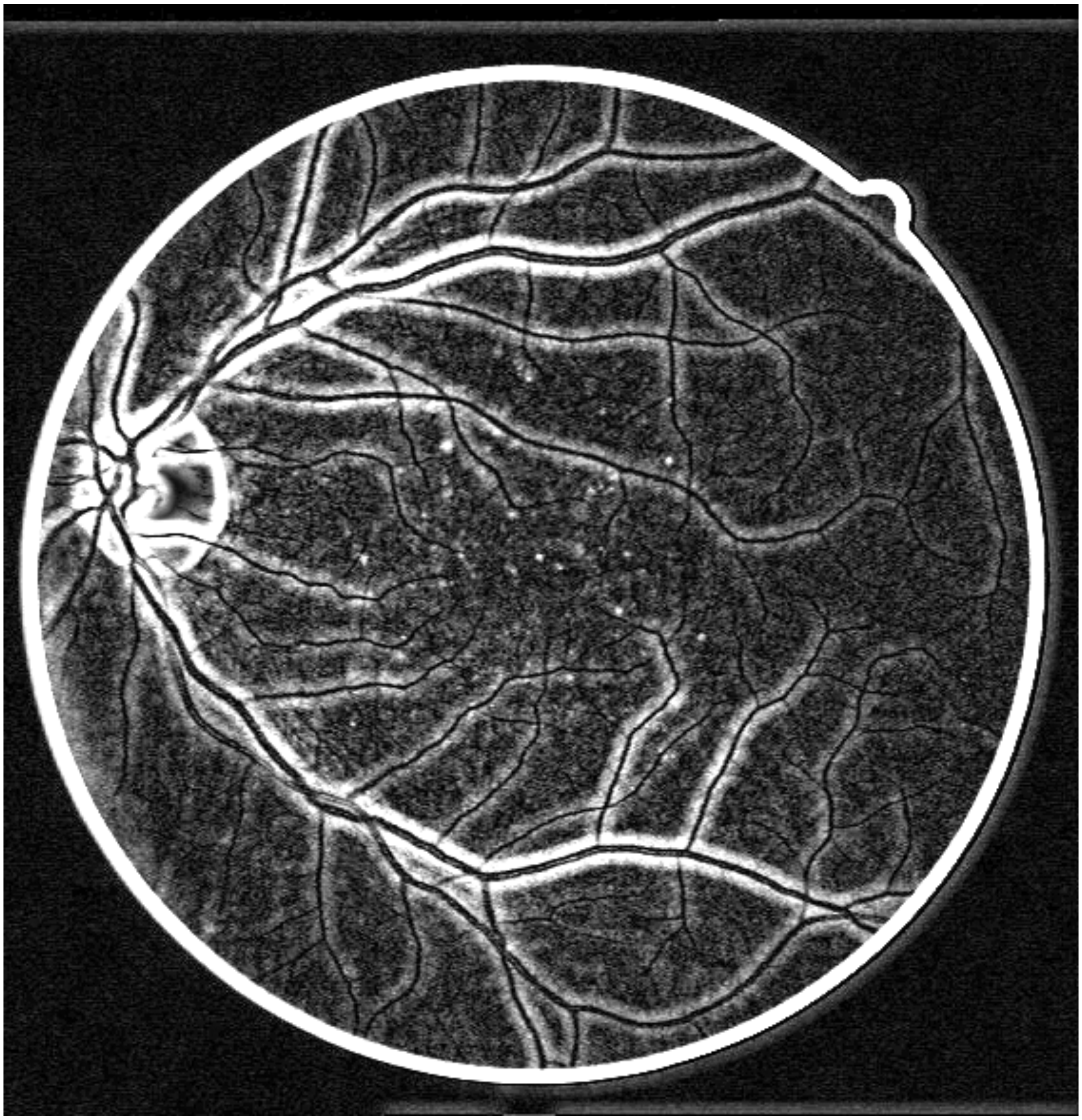
Out[99]=
```
{{12, 584, 565}}
```

```
In[119]:= AbsoluteTiming[topHatList = (SopData - #) & /@ erosionByLinearStrucImageDataList]

In[126]:= AbsoluteTiming[SsumData = Total[topHatList]]

In[127]:= Ssum = Image[SsumData]
```

Out[127]=



This transformation (a sum of top hats) reduces small bright noise and improves the contrast of all linear parts. Vessels could be manually segmented with a simple threshold on $S_{sum}$

## Alternating Filter

In[123]:=
```
? GaussianFilter
```

GaussianFilter[*image*, *r*] filters *image* by convolving with a Gaussian kernel of pixel radius *r*.

GaussianFilter[*image*, *r*, {$n_1$, $n_2$}] convolves *image* with a kernel formed from the $n_i^{\text{th}}$ derivatives of the discrete Gaussian.

GaussianFilter[*image*, {*r*, $\sigma$}, …] uses a Gaussian kernel with radius *r* and standard deviation $\sigma$.

GaussianFilter[*image*, {{$r_1$, $r_2$}, …}] uses radii $r_i$ etc. in vertical and horizontal directions.

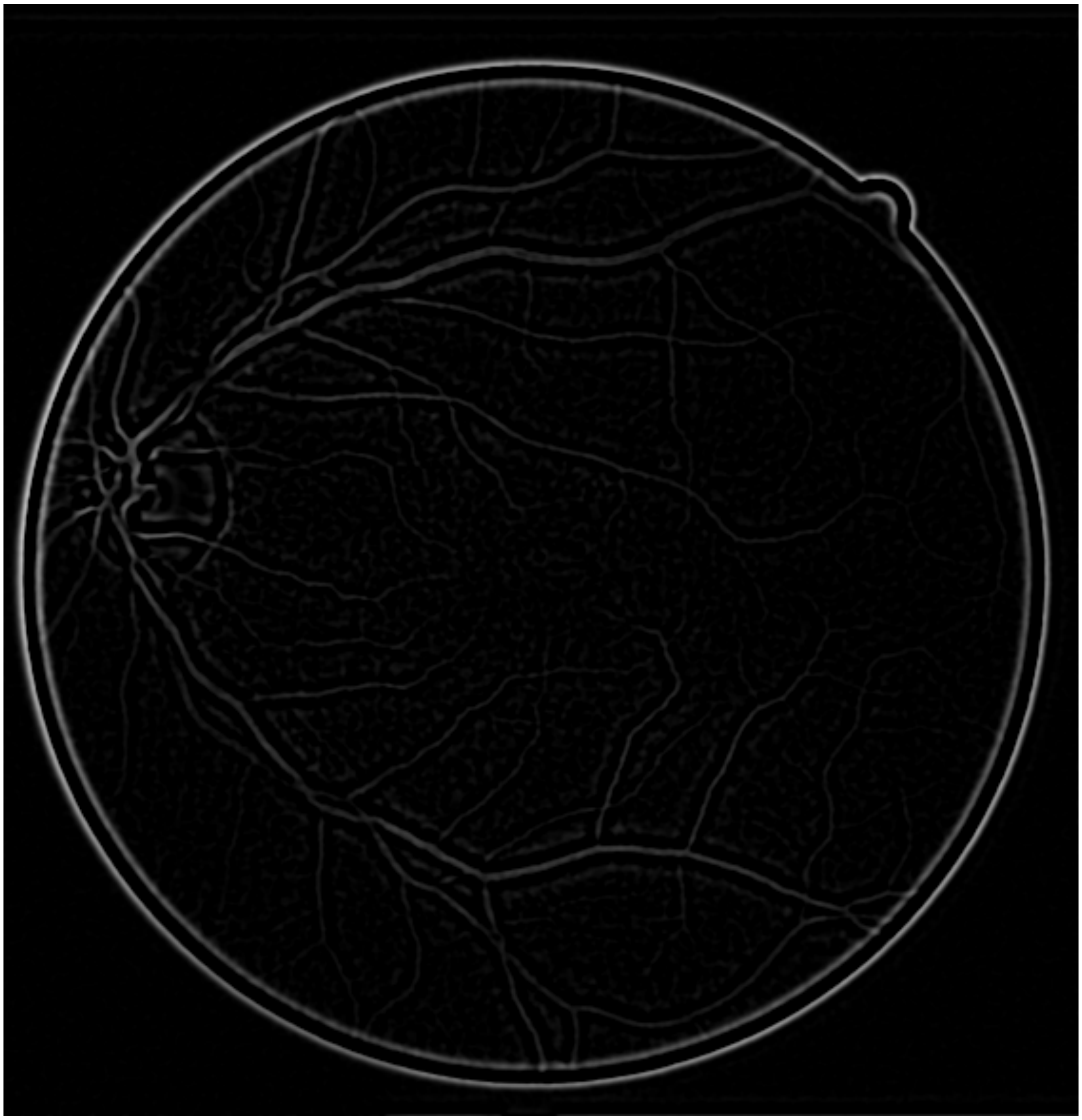
GaussianFilter[*data*, …] applies Gaussian filtering to an array of *data*. >>

In[128]:=
```
GaussianFilter[Ssum, {7, 7 / 4}]
```

Out[128]=

In[140]:=
```
Slap = LaplacianFilter[GaussianFilter[Ssum, {7, 7 / 4}], 1]
```

Out[140]=



## Final Result

### SI

1. Apply Erosion with Linear Structure Element

In[142]:=
```
SlapErosionByLinearStrucImageList = Erosion[Slap, #] & /@ kernels;
```

2. Pick the Maximum pixel-wise

In[143]:=
```
AbsoluteTiming[SlapErosionByLinearStrucImageDataList =
  ImageData[#] & /@ SlapErosionByLinearStrucImageList]
```

Out[143]=
```
{0.00004, {… 1 …}}
```

large output | show less | show more | show all | set size limit...

In[154]:=
```
slapMaxData = MapThread[Max, SlapErosionByLinearStrucImageDataList, 2];
```

3. Apply GeodesicOpening

In[163]:=
```
slapMax = Image[slapMaxData]
```

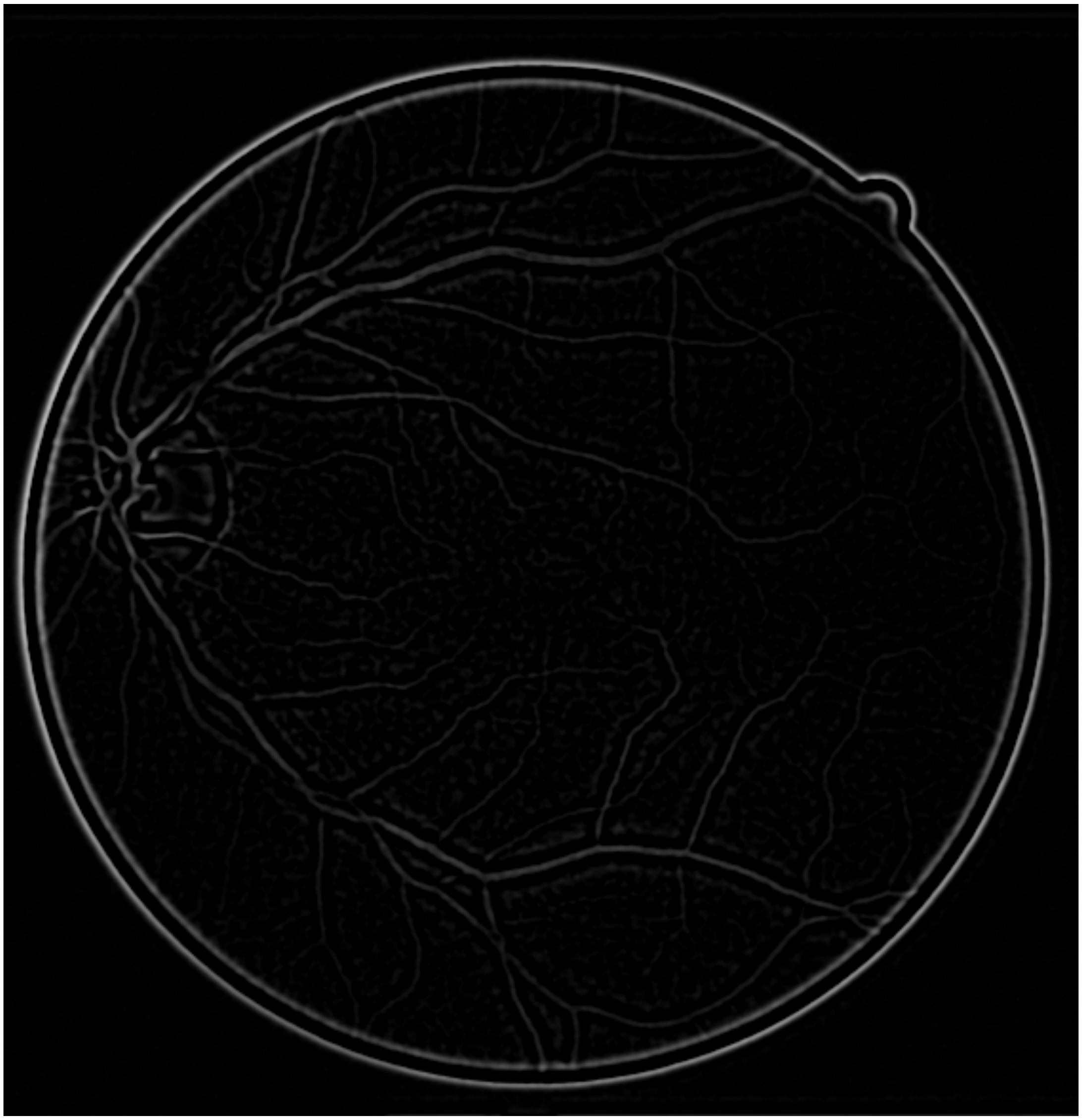
In[164]:=
```
S1Data = geodesicOpening[Slap, slapMax]
```

Out[164]=
```
{… 1 …}
```

large output | show less | show more | show all | set size limit...

In[165]:=
```
S1 = Image[S1Data]
```

Out[165]=



## S2

1. Apply Dilation with Linear Structure Element

In[159]:=
```
SlapDilationByLinearStrucImageList = Dilation[S1, #] & /@ kernels;
```

2. Pick the Maximum pixel-wise

In[160]:=
```
AbsoluteTiming[SlapDilationByLinearStrucImageDataList =
  ImageData[#] & /@ SlapDilationByLinearStrucImageList]
```

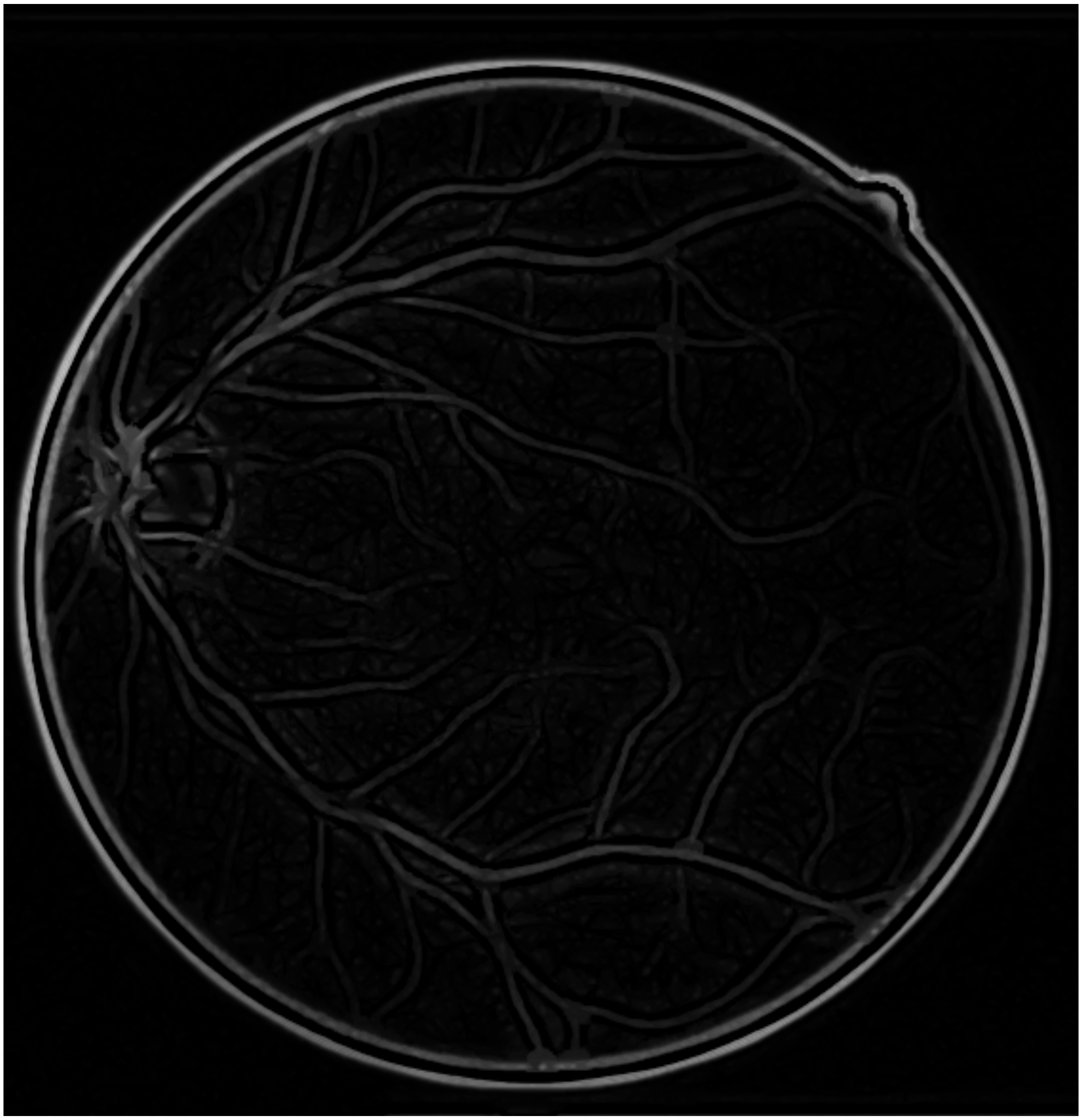
Out[160]=
```
{0.00003, {… 1 …}}
```

large output | show less | show more | show all | set size limit...

In[166]:=
```
slapMinData = MapThread[Min, SlapDilationByLinearStrucImageDataList, 2];
```

In[167]:=
```
slapMin = Image[slapMinData]
```

Out[167]=



3. Apply GeodesicOpening with marker image = S1

In[169]:=
```
S2Data = geodesicOpening[S1, slapMin]
```

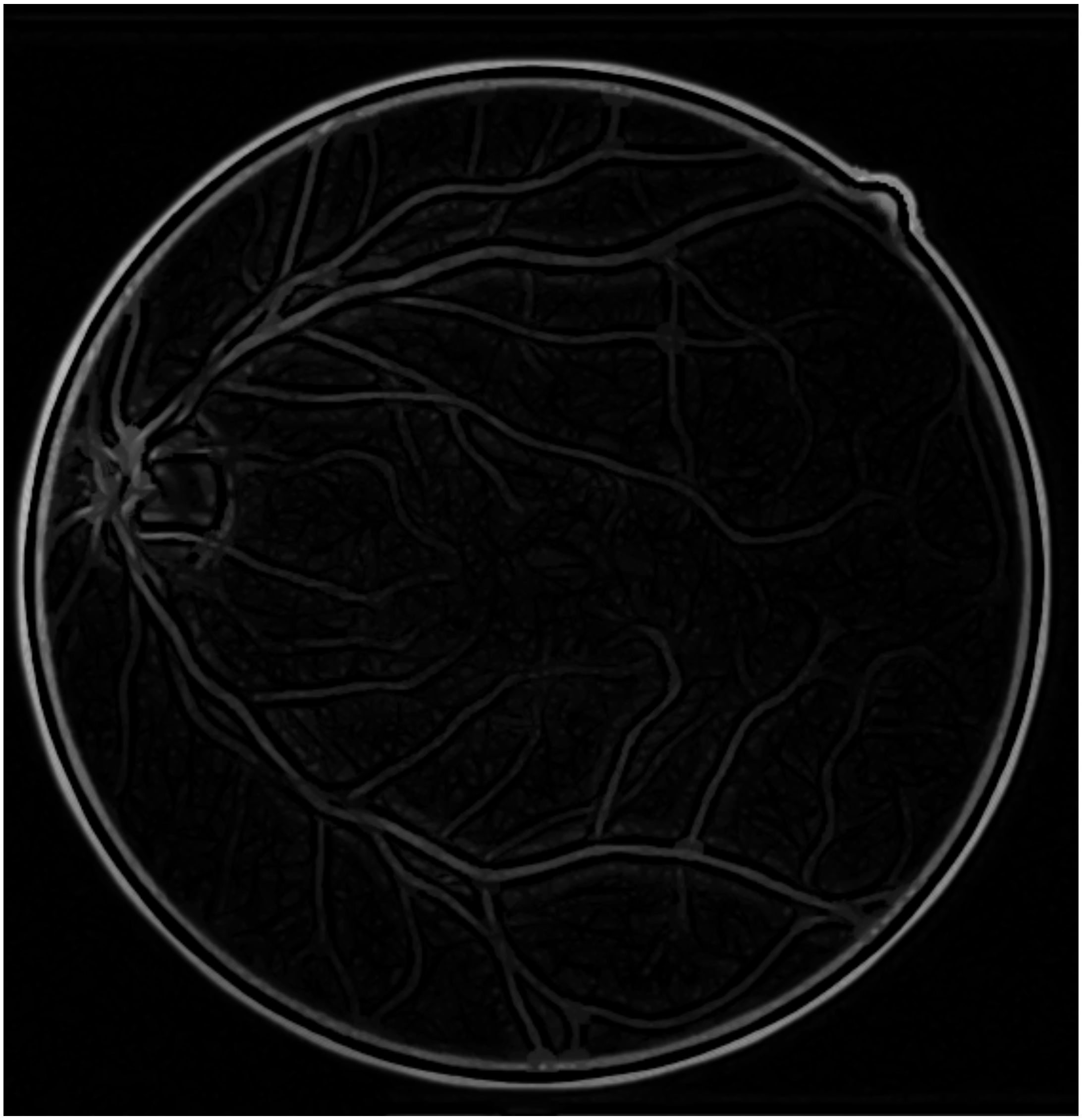
Out[169]=

{ ··· 1 ··· }

large output | show less | show more | show all | set size limit...

In[172]:=
```
S2 = Image[S2Data]
```

Out[172]=



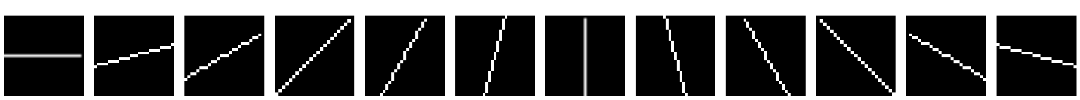
## $S_{res}$

### new linear struct e = 2

In[222]:=
```
linearStrut30 =
  Table[Round[{Cos[t], Sin[t]} * #] &/@ (Range[30] - 16), {t, 0, 11 π/12, π/12}];
```

In[223]:=
```
GraphicsRow[showBinaryImage[kernelFromStructure[#]] &/@ linearStrut30]
```

Out[223]=



In[224]:=
```
kernels30 = kernelFromStructure[#] &/@ linearStrut30;
```

## 1. Erosion with $L_i$ e = 2

In[225]:=
```
erosionOnSres = Erosion[S2, #] & /@ kernels30
```

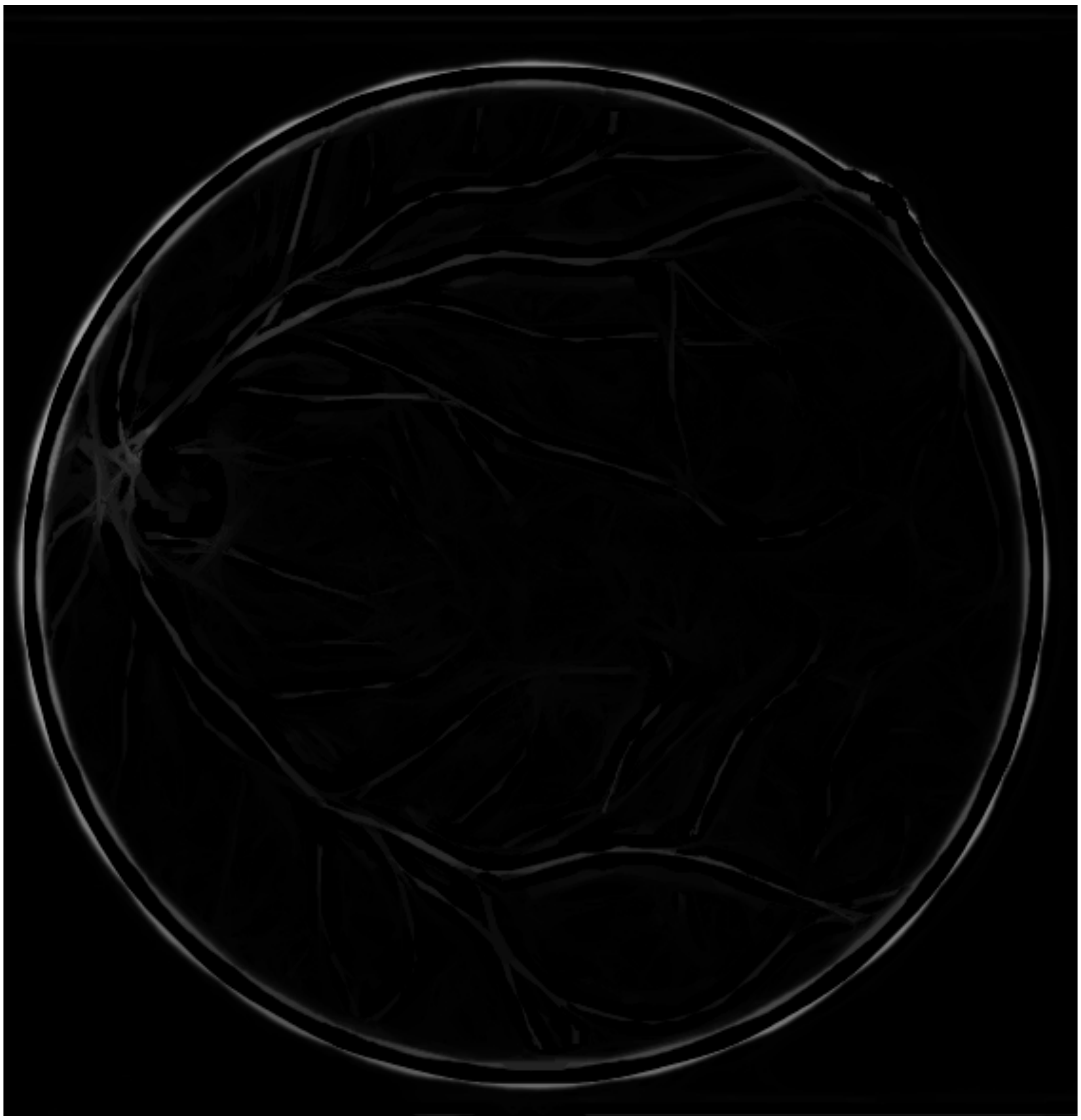
In[226]:=
```
erosionOnSresData = ImageData /@ erosionOnSres;
```

## 2. Pick maximum

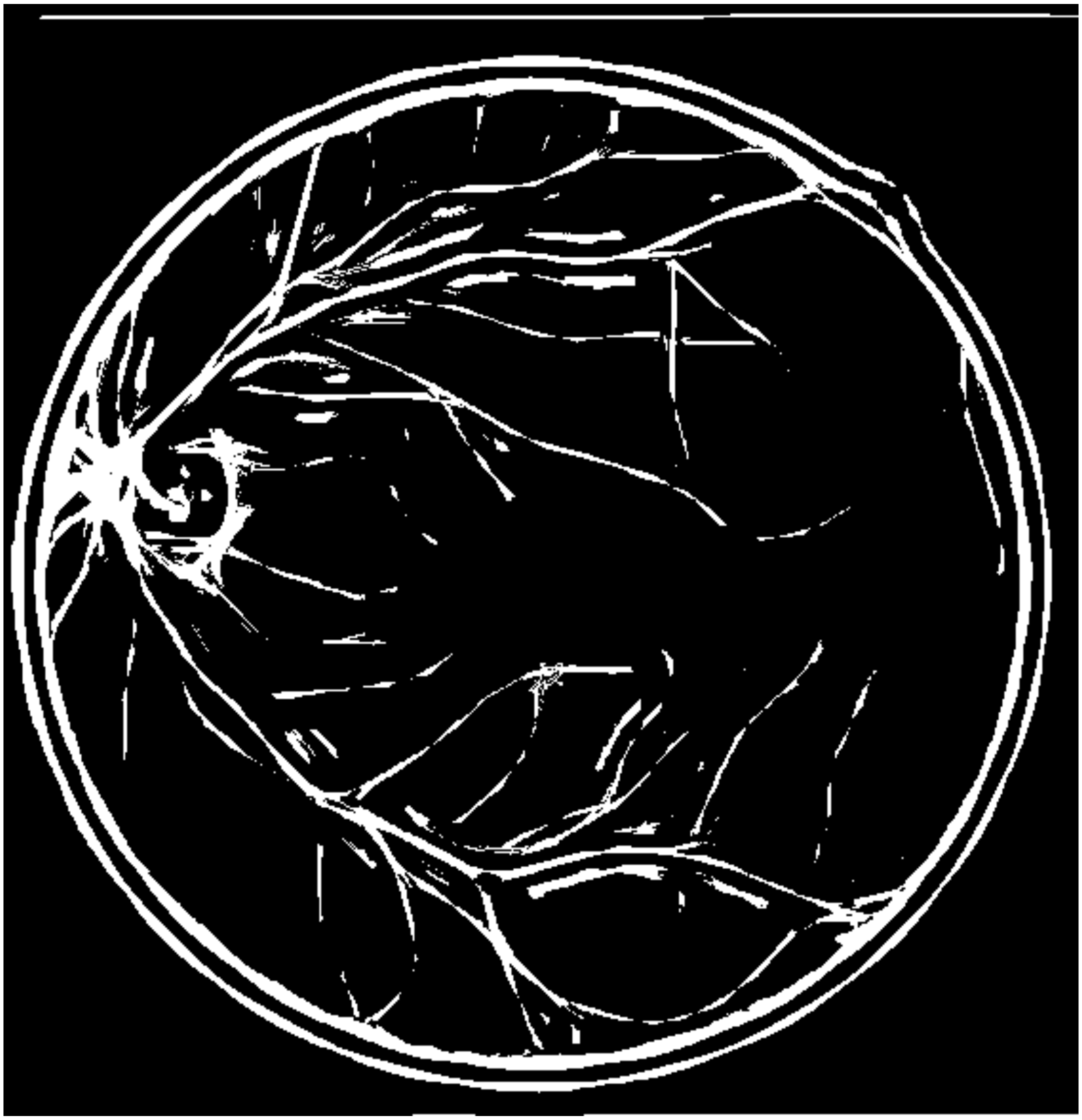
In[227]:=
```
erosionOnSresMaxData = MapThread[Max, erosionOnSresData, 2];
```

In[228]:=
```
Sres = Image[erosionOnSresMaxData]
```

Out[228]=



In[237]:= 
```
bin = Binarize[Sres, 0.025]
```

Out[237]= 

In[238]:= 
```
binData = ImageData[bin];
```

In[239]:=
```
showGrayOverlay[imgData, binData]
```

Out[239]=