Metehan Kaya
21401258
CS 484

HOMEWORK 3

Part 1 – Parameters used for superpixel segmentation

$kSuperpixels = 750$, the number of superpixels I want to create. However, NumLabels that is the actual number of superpixels that were computed is different, in general. $kClusters = 6$ and $denominator = 5$ that is used for threshold on superpixels' sizes that are covered by circles.

Part 2 – Segmentation results using superpixel segmentation

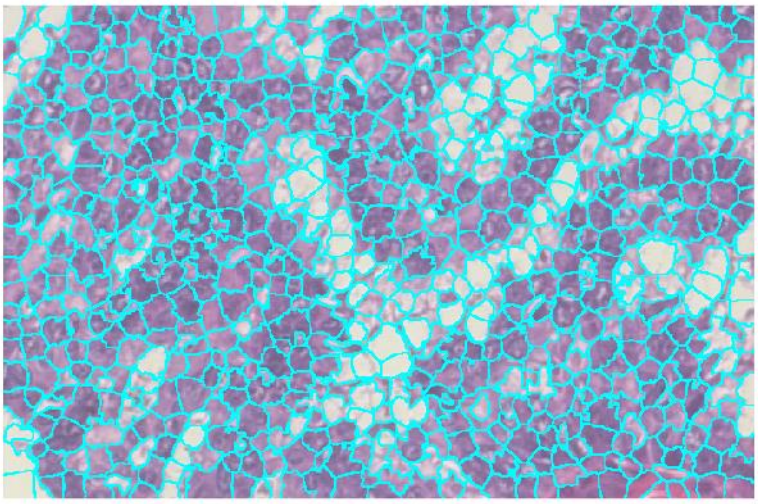

Figure 1. Segmentation result using superpixel segmentation of image 1

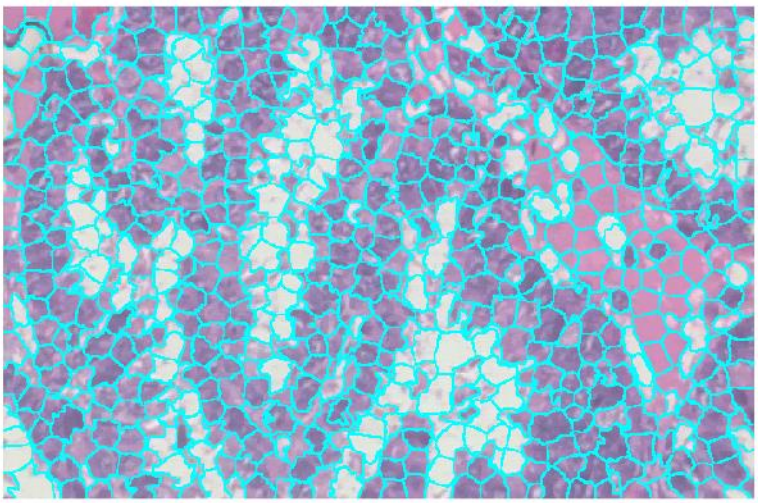

Figure 2. Segmentation result using superpixel segmentation of image 2

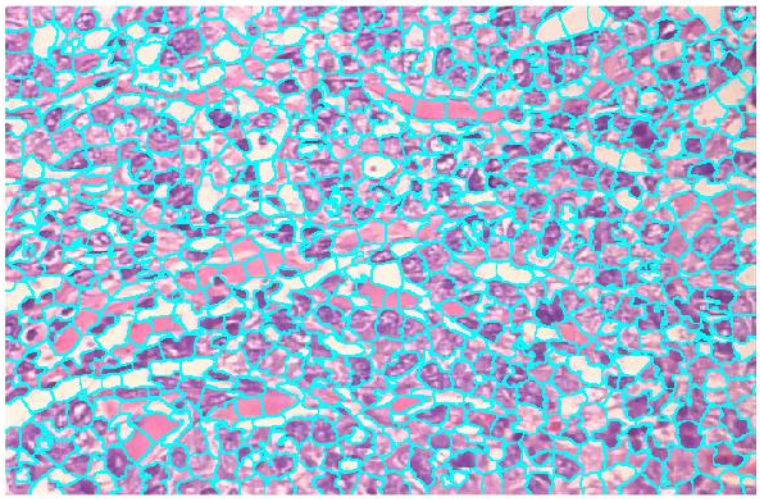

Figure 3. Segmentation result using superpixel segmentation of image 3

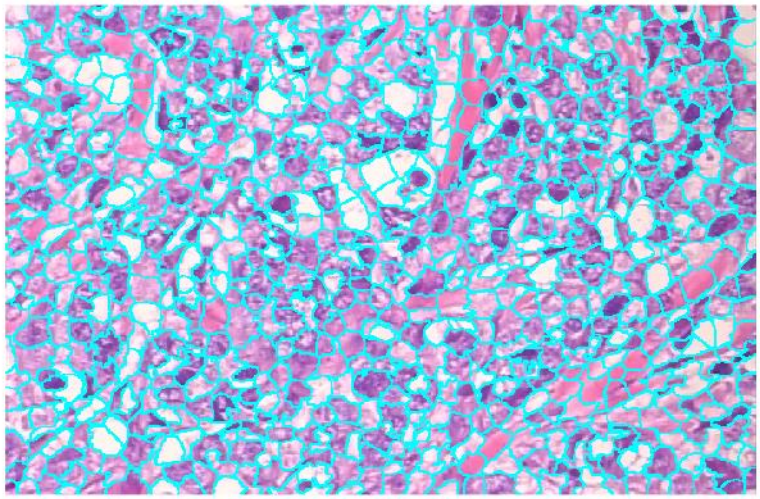

Figure 4. Segmentation result using superpixel segmentation of image 4

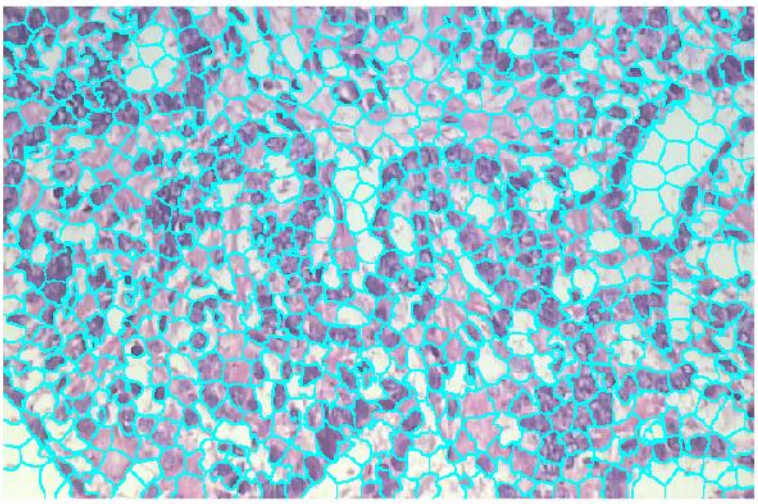

Figure 5. Segmentation result using superpixel segmentation of image 5

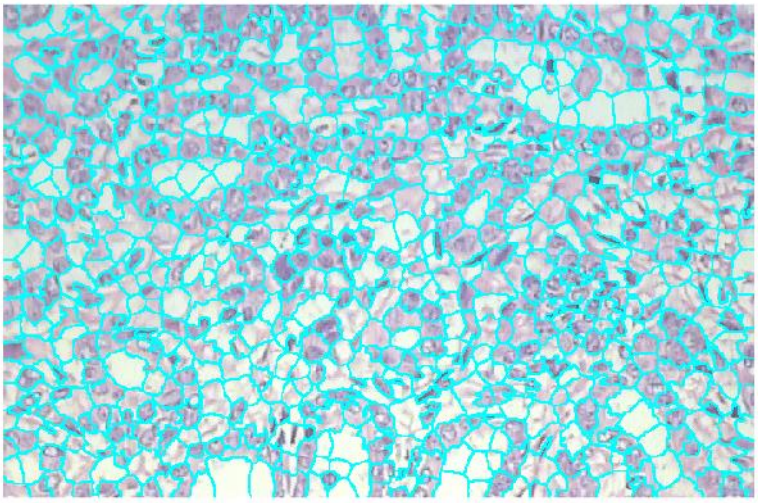

Figure 6. Segmentation result using superpixel segmentation of image 6

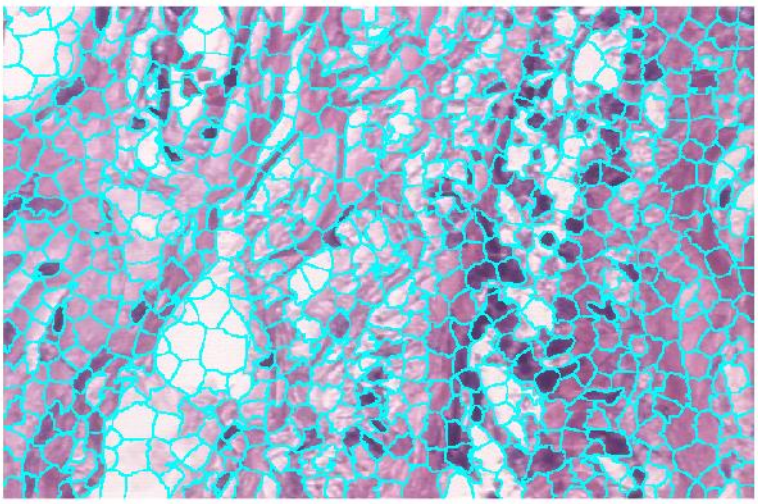

Figure 7. Segmentation result using superpixel segmentation of image 7

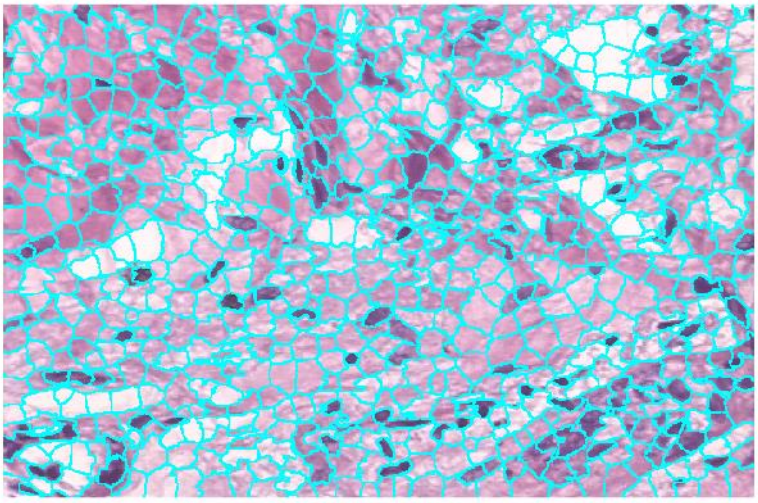

Figure 8. Segmentation result using superpixel segmentation of image 8

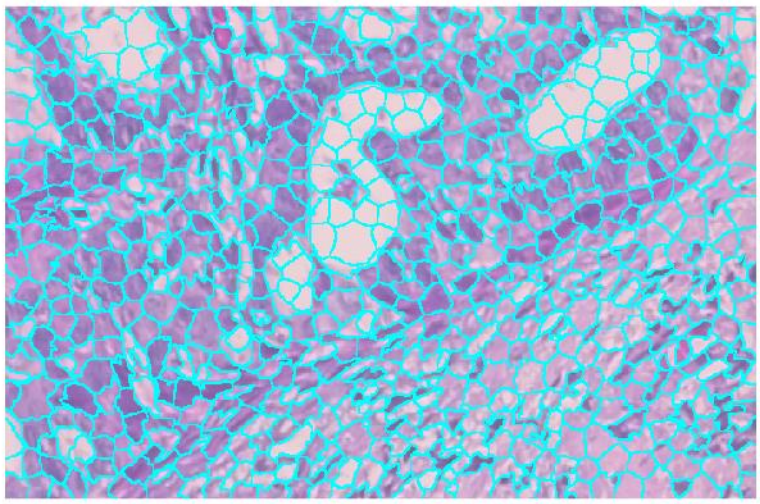

Figure 9. Segmentation result using superpixel segmentation of image 9

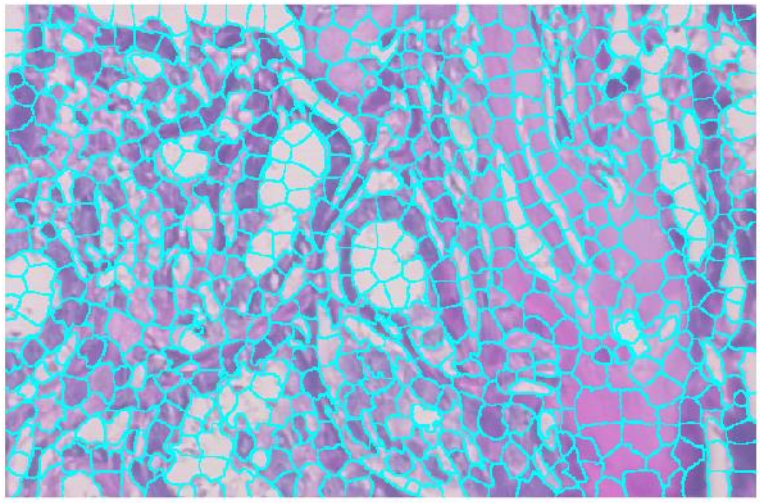

Figure 10. Segmentation result using superpixel segmentation of image 10

Part 3 – Parameters for Gabor texture feature extraction

Scale = [2 , 4 , 5 , 10]. I used 2 and 5 in the report.

Orientation = [30 , 45 , 60 , 90]. I used 30 and 60 in the report.

Part 4 – Gabor texture feature examples

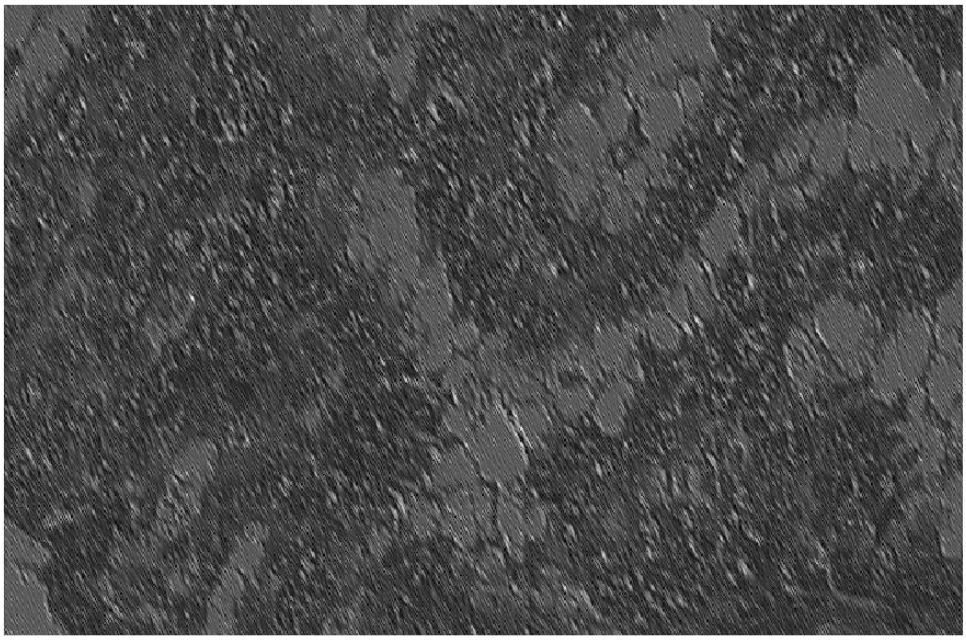

Figure 11. Gabor texture features examples of image 1, scale = 2, orientation = 30

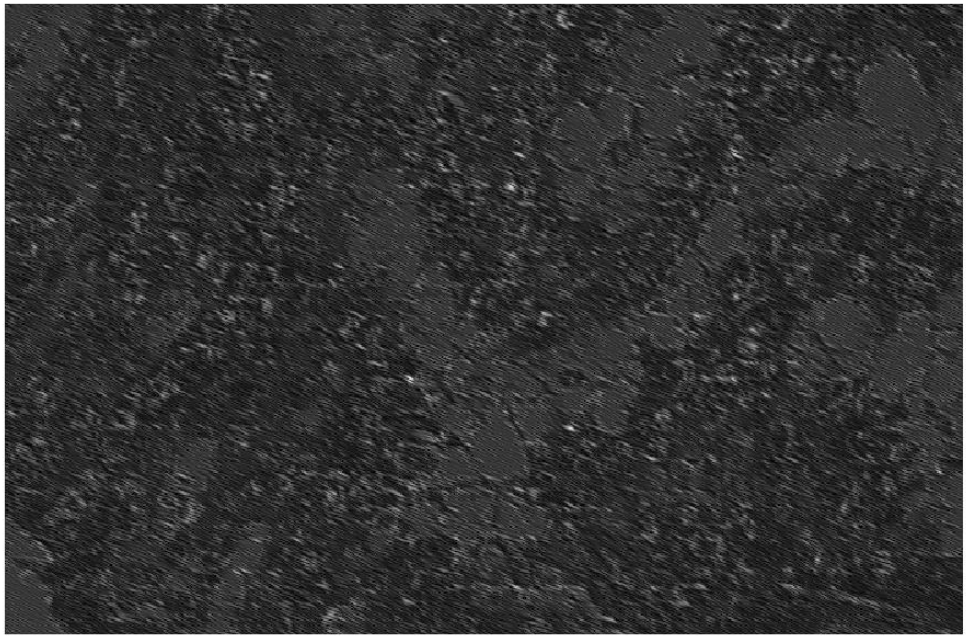

Figure 12. Gabor texture features examples of image 1, scale = 2, orientation = 60

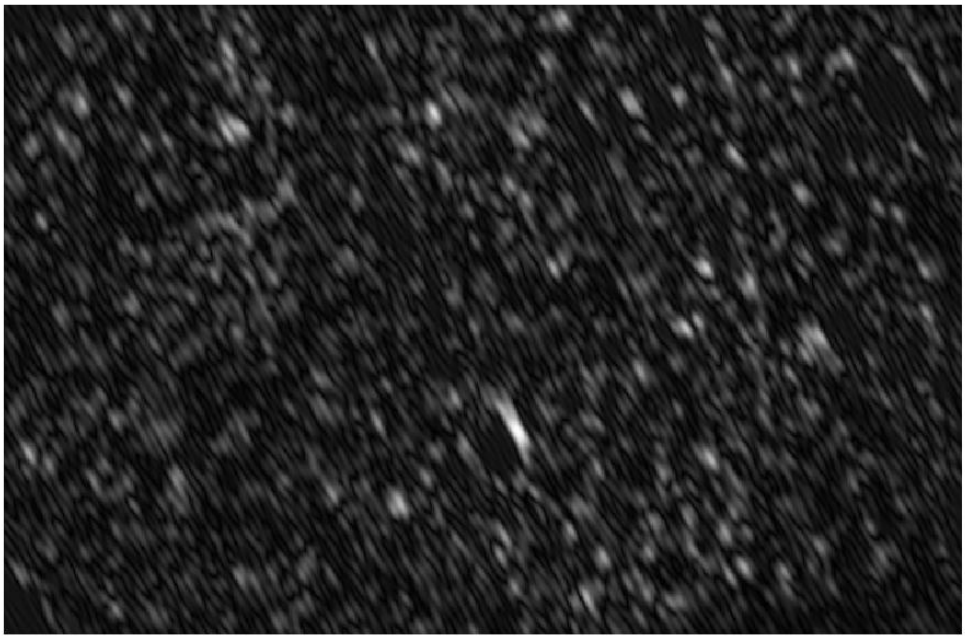

Figure 13. Gabor texture features examples of image 1, scale = 5, orientation = 30

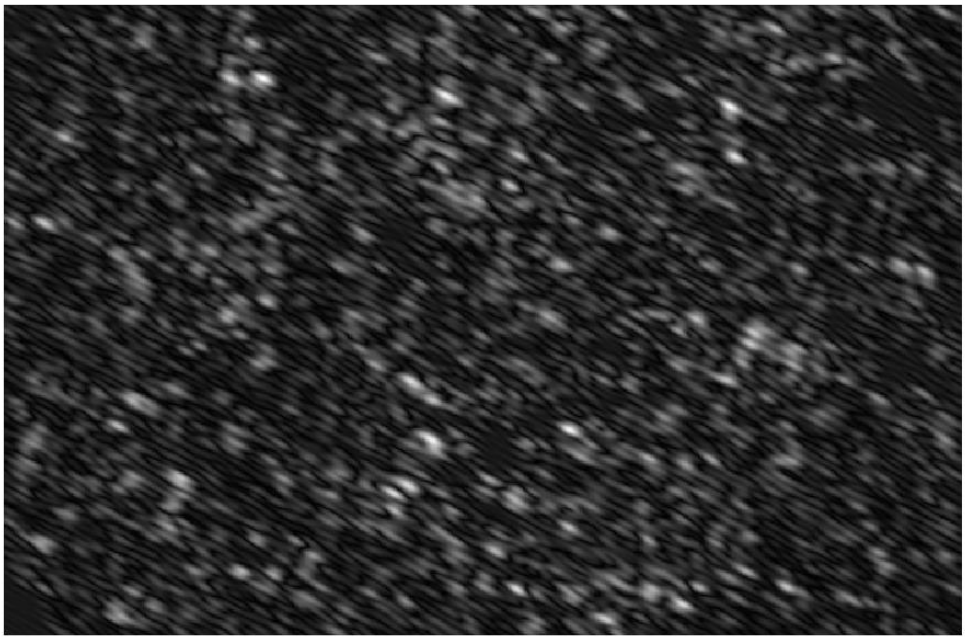

Figure 14. Gabor texture features examples of image 1, scale = 5, orientation = 60

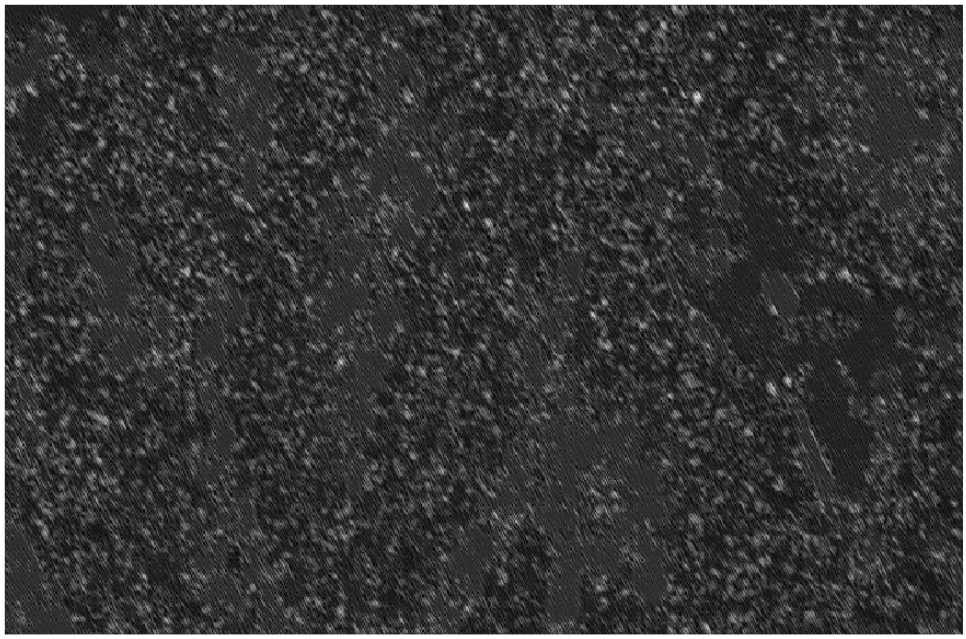

Figure 15. Gabor texture features examples of image 2, scale = 2, orientation = 30

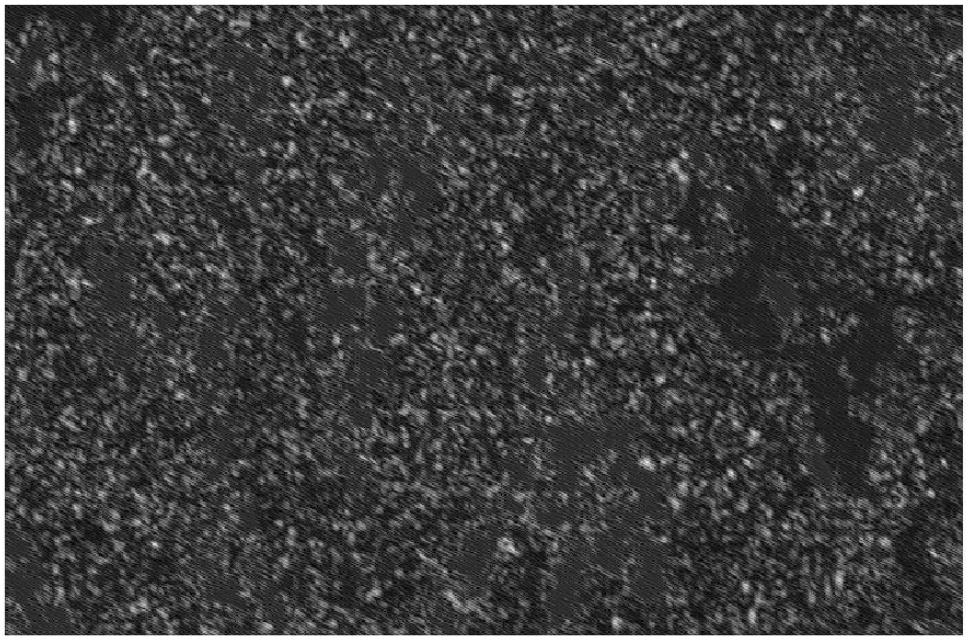

Figure 16. Gabor texture features examples of image 2, scale = 2, orientation = 60

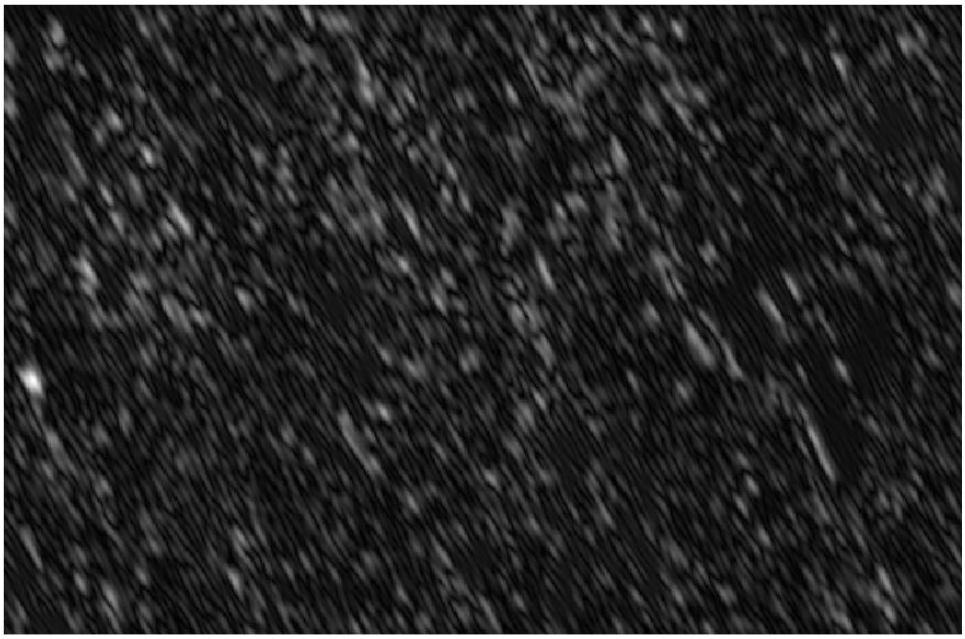

Figure 17. Gabor texture features examples of image 2, scale = 5, orientation = 30

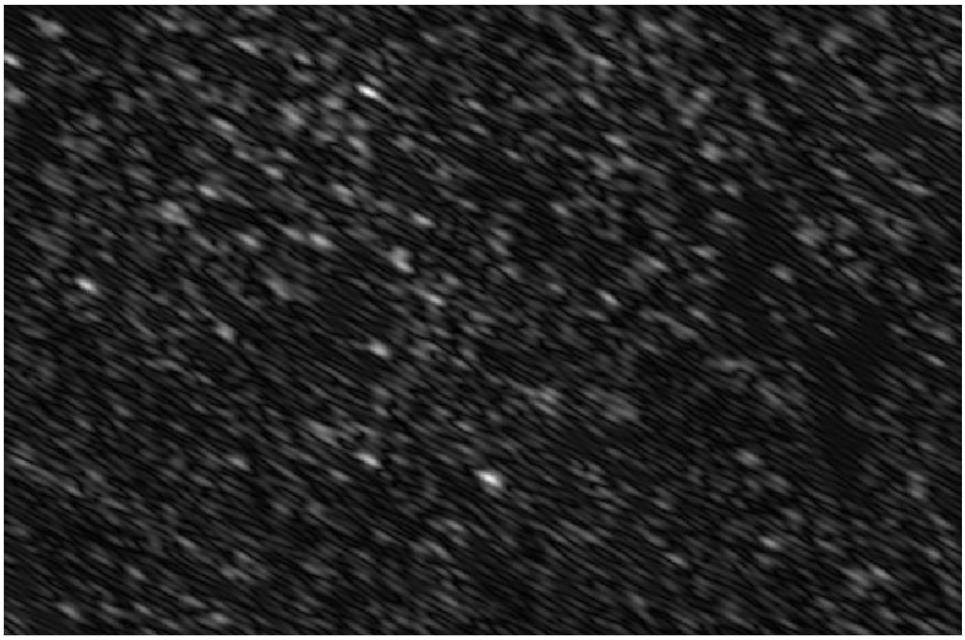

Figure 18. Gabor texture features examples of image 2, scale = 5, orientation = 60

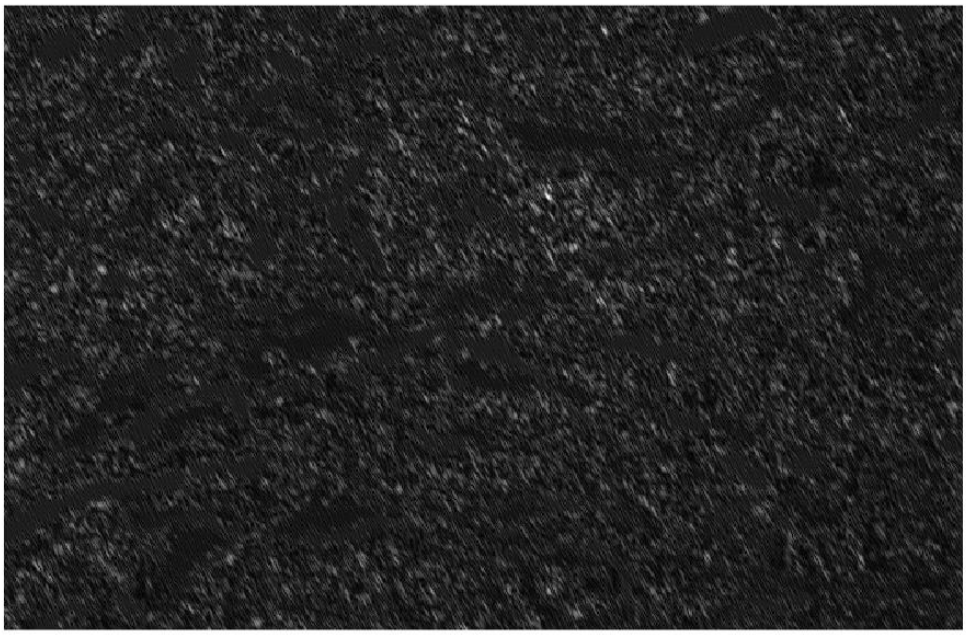

Figure 19. Gabor texture features examples of image 3, scale = 2, orientation = 30

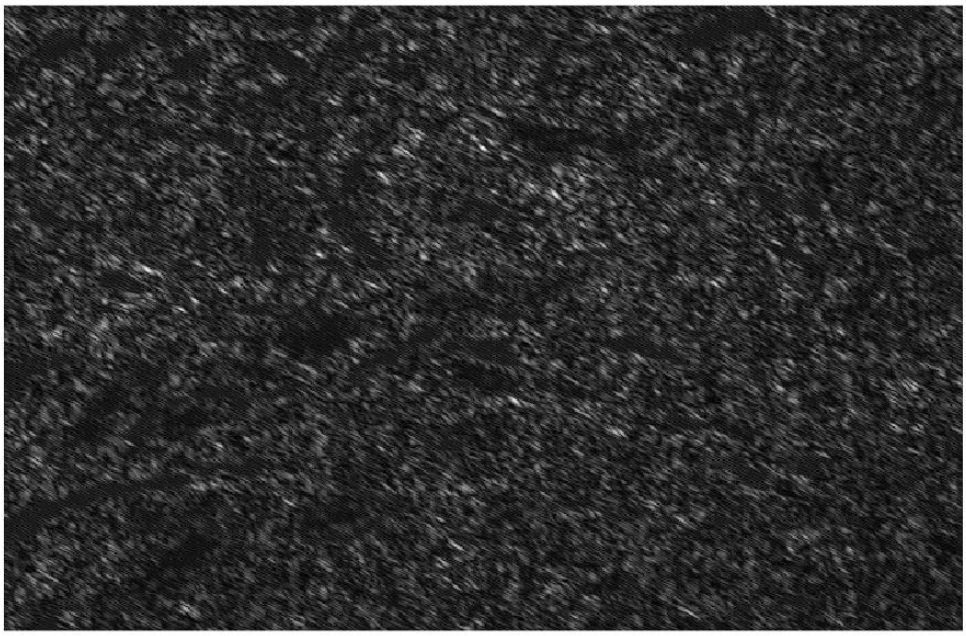

Figure 20. Gabor texture features examples of image 3, scale = 2, orientation = 60

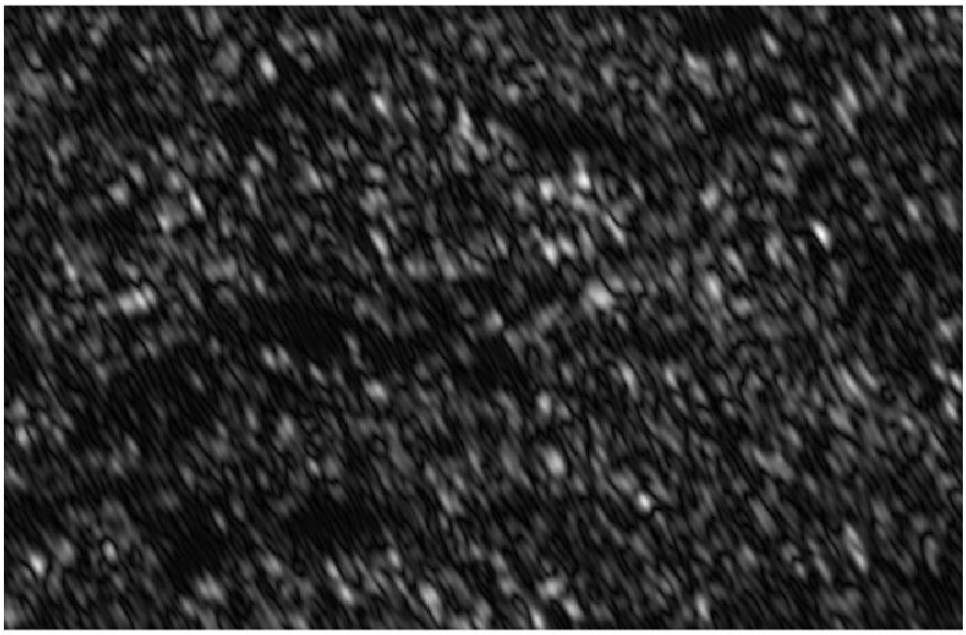

Figure 21. Gabor texture features examples of image 3, scale = 5, orientation = 30

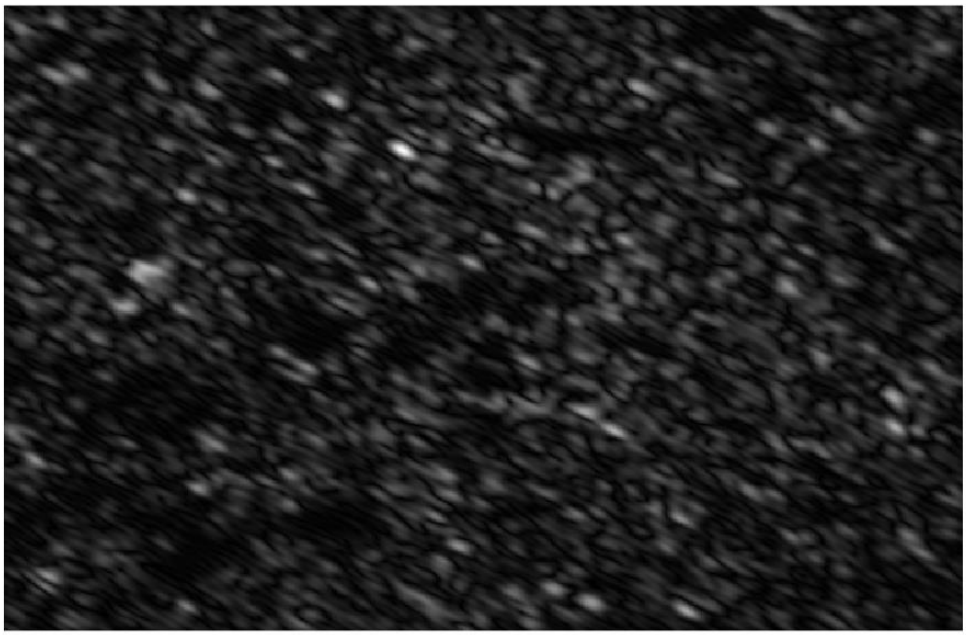

Figure 22. Gabor texture features examples of image 3, scale = 5, orientation = 60

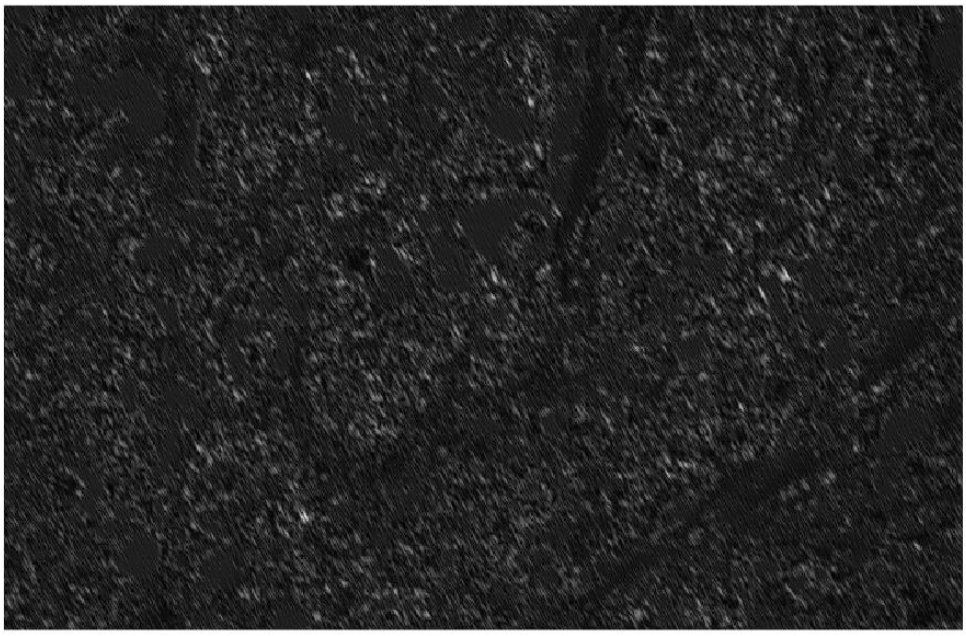

Figure 23. Gabor texture features examples of image 4, scale = 2, orientation = 30

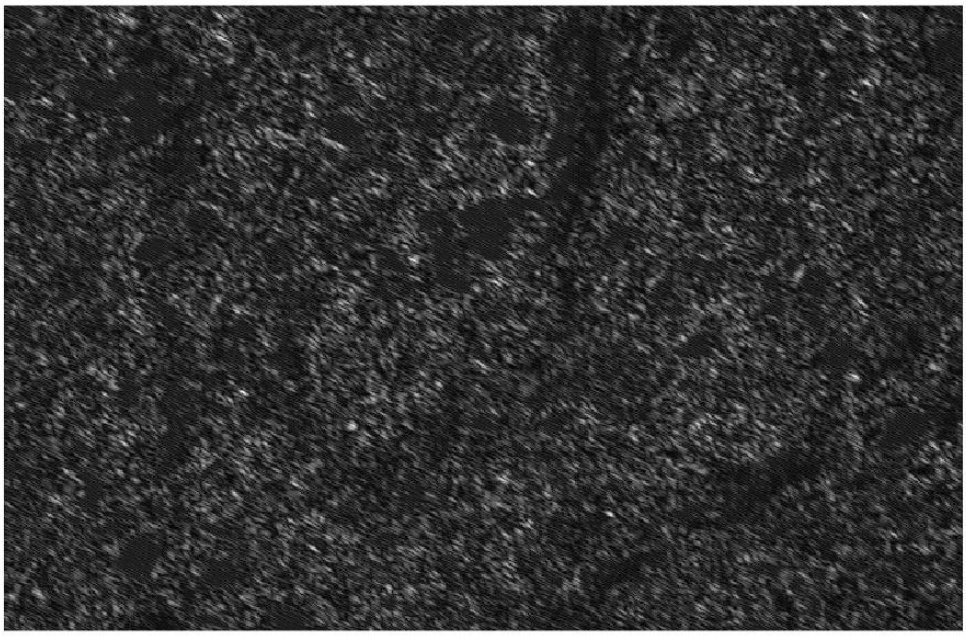

Figure 24. Gabor texture features examples of image 4, scale = 2, orientation = 60

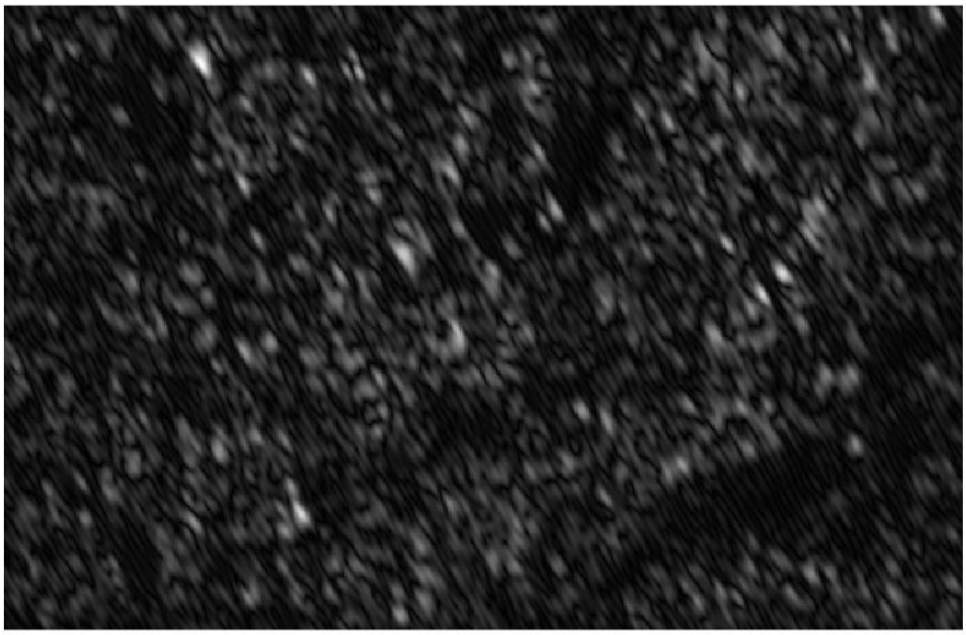

Figure 25. Gabor texture features examples of image 4, scale = 5, orientation = 30

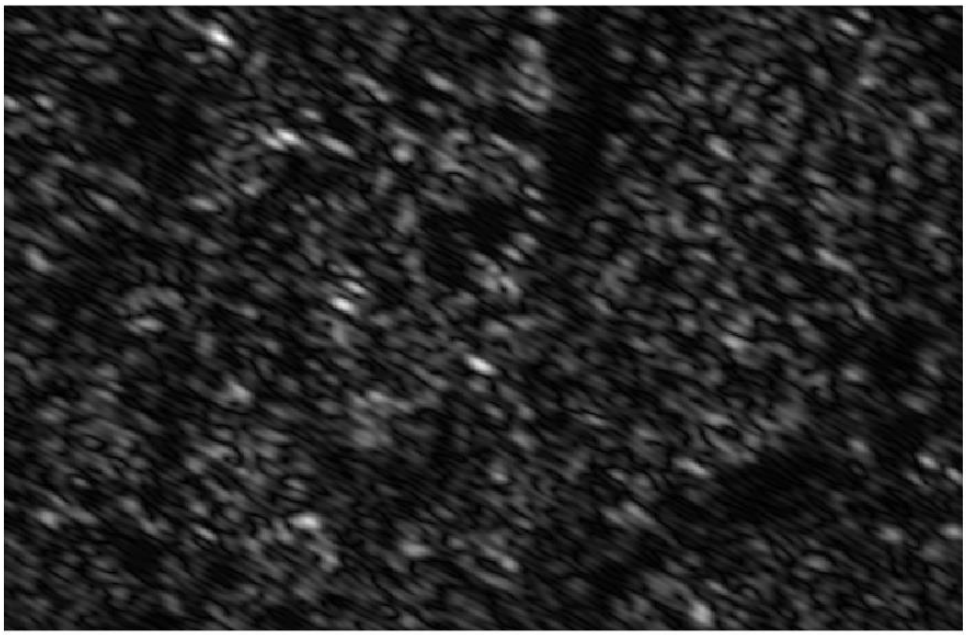

Figure 26. Gabor texture features examples of image 4, scale = 5, orientation = 60

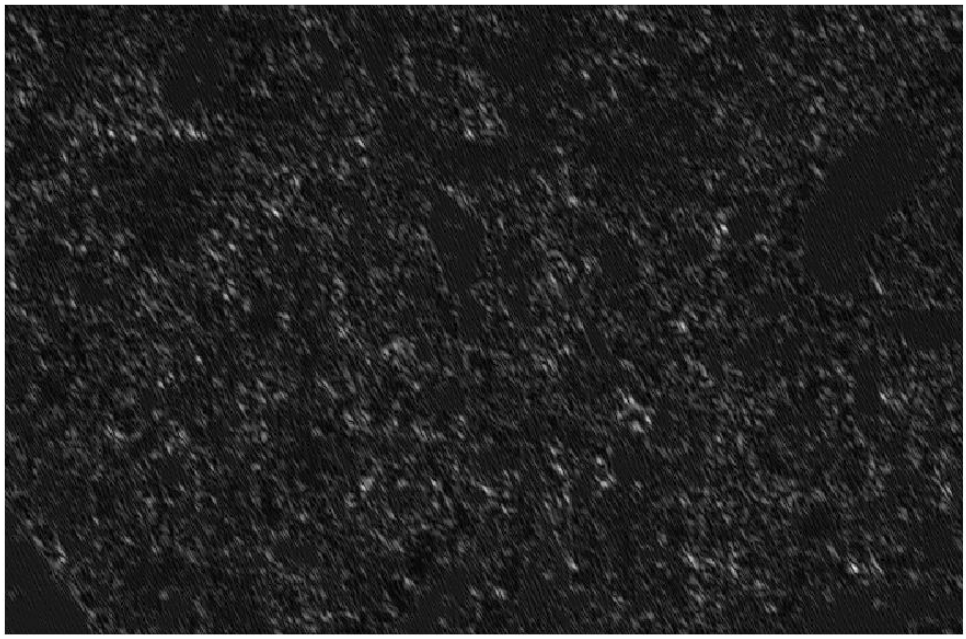

Figure 27. Gabor texture features examples of image 5, scale = 2, orientation = 30

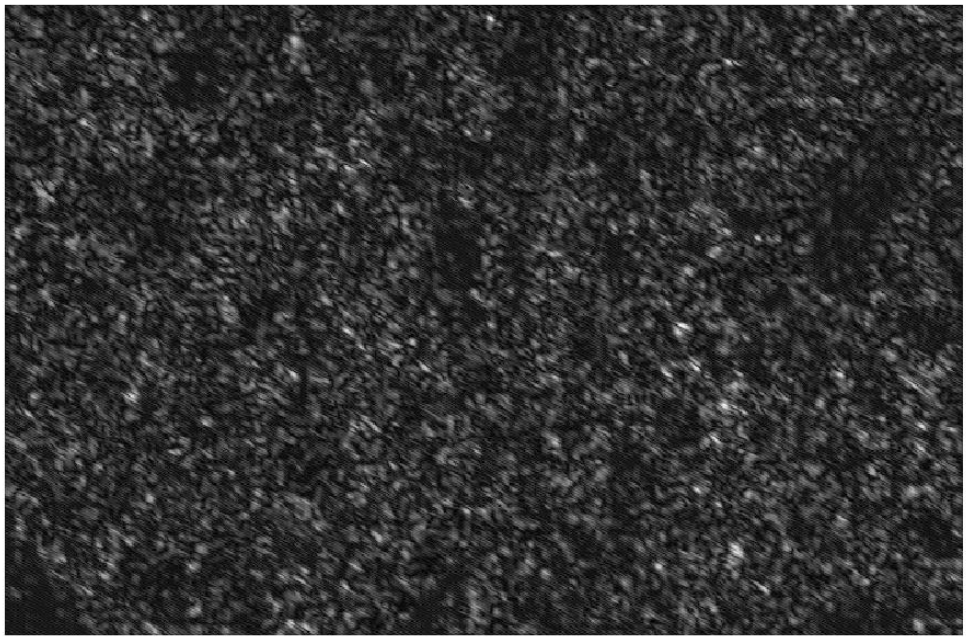

Figure 28. Gabor texture features examples of image 5, scale = 2, orientation = 60

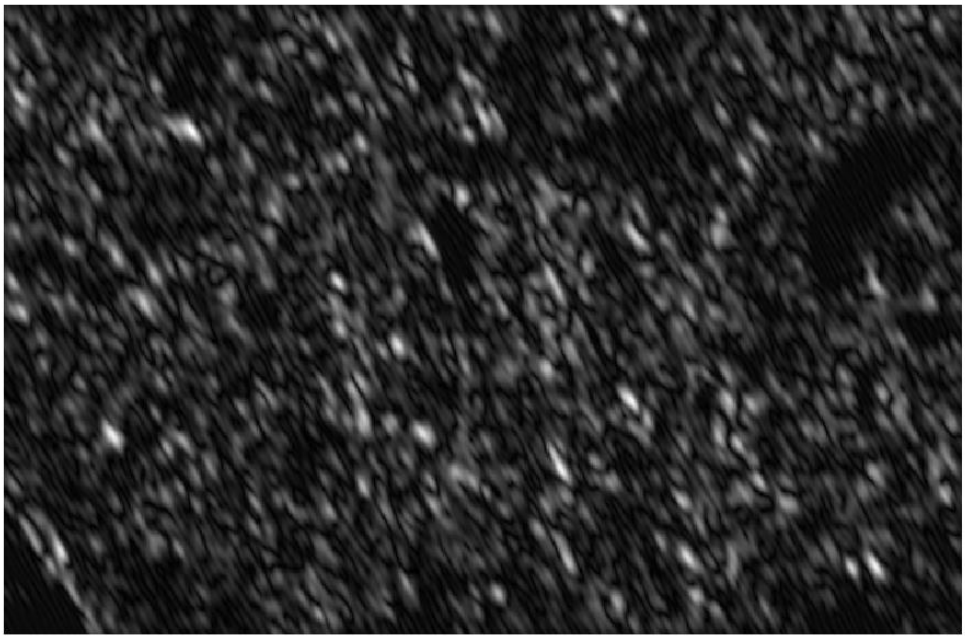

Figure 29. Gabor texture features examples of image 5, scale = 5, orientation = 30

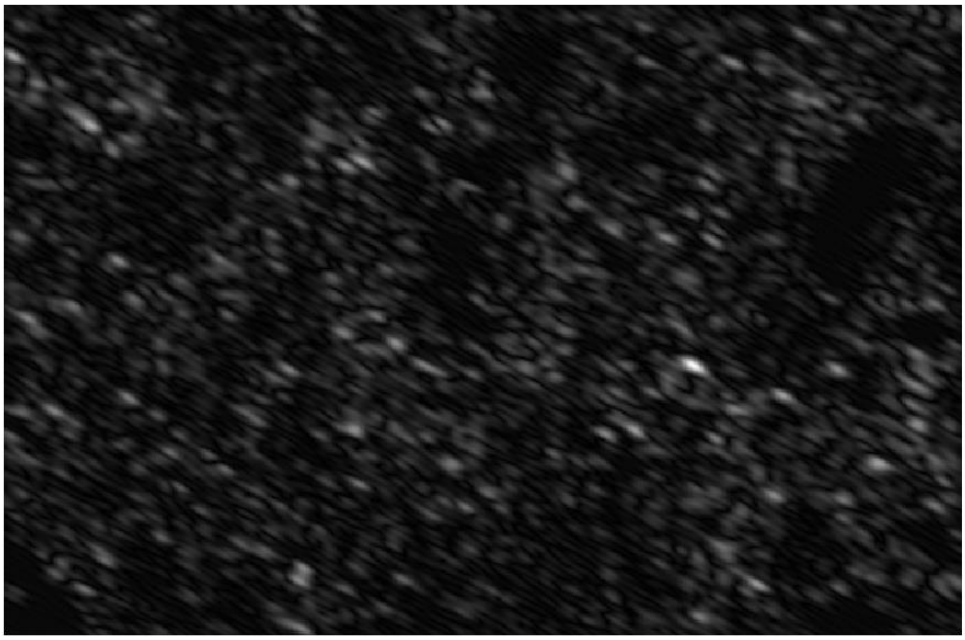

Figure 30. Gabor texture features examples of image 5, scale = 5, orientation = 60

Part 5 – Clustering/Segmentation results for each type of feature vector

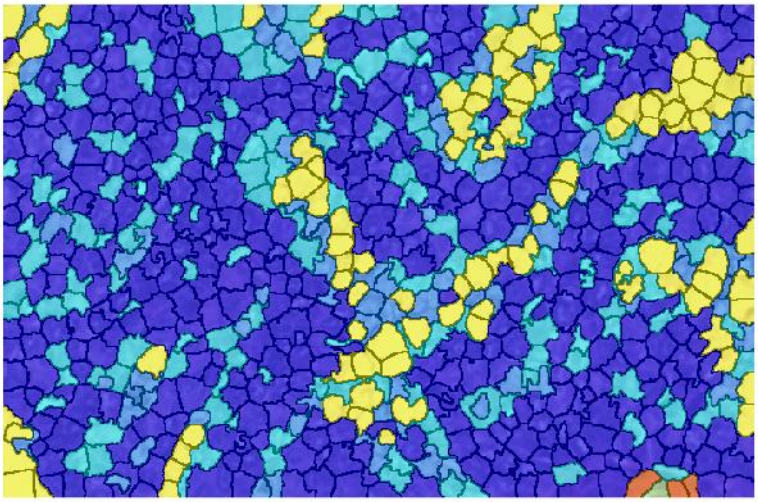

Figure 31. Clustering result of image 1, part 4

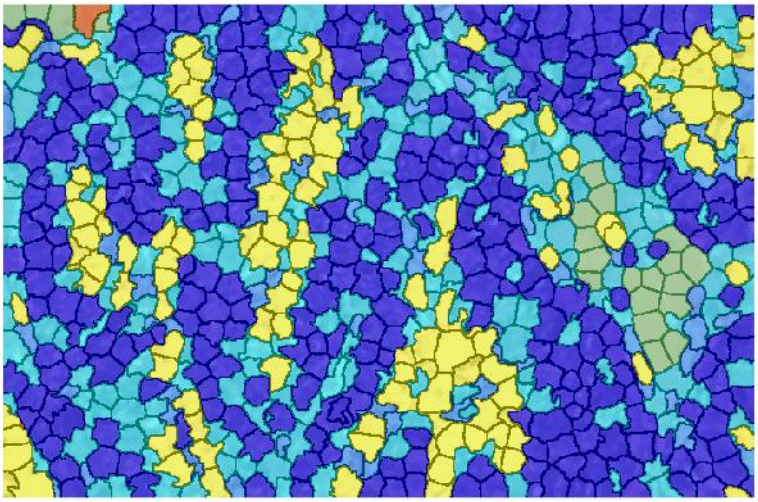

Figure 32. Clustering result of image 2, part 4

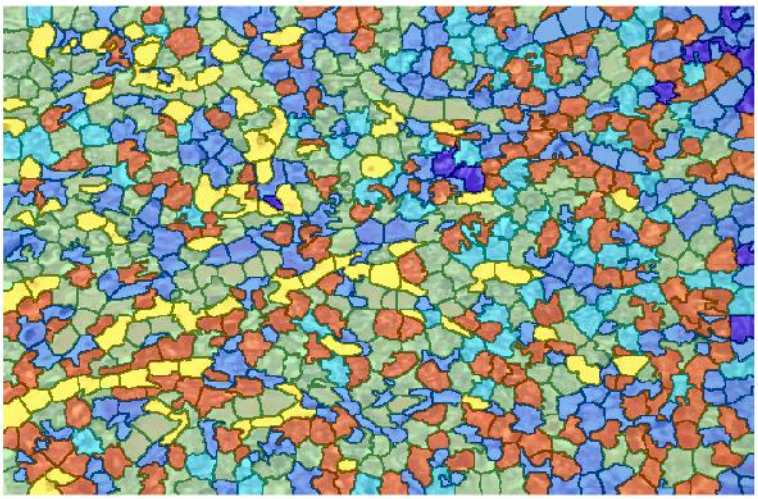

Figure 33. Clustering result of image 3, part 4

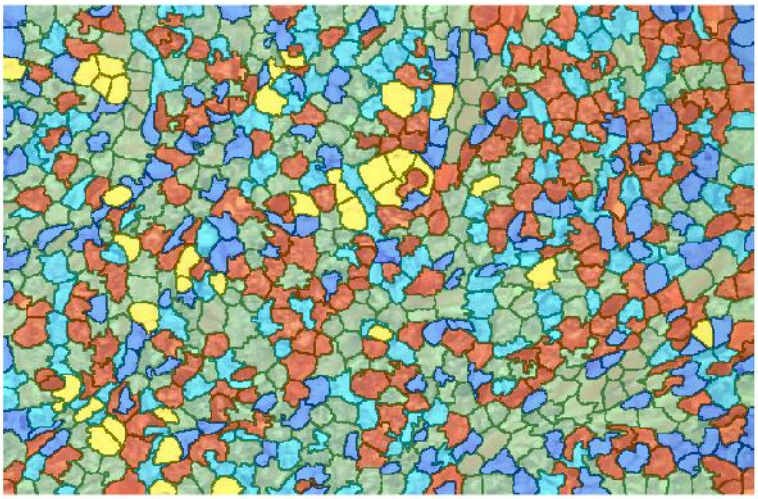

Figure 34. Clustering result of image 4, part 4

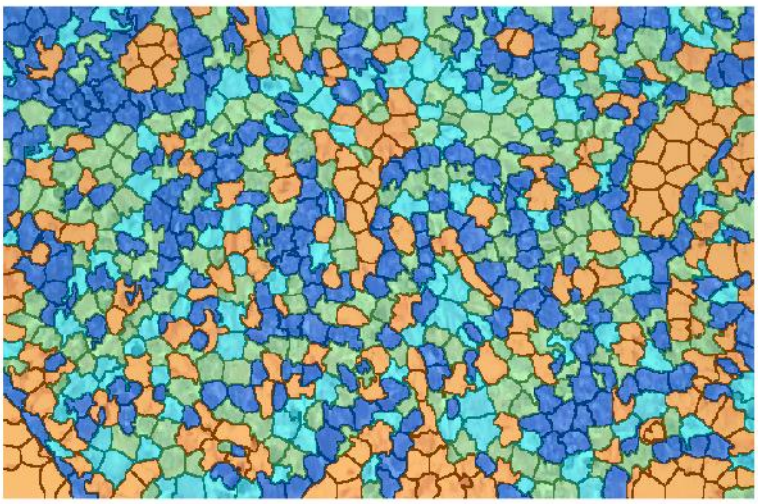

Figure 35. Clustering result of image 5, part 4

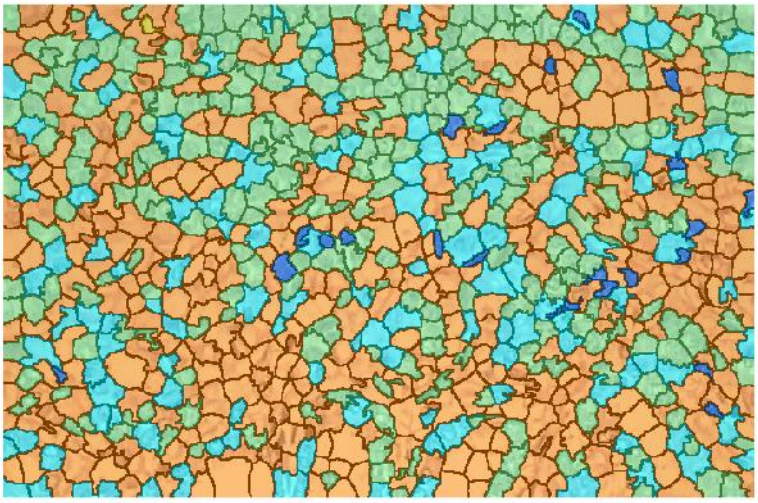

Figure 36. Clustering result of image 6, part 4

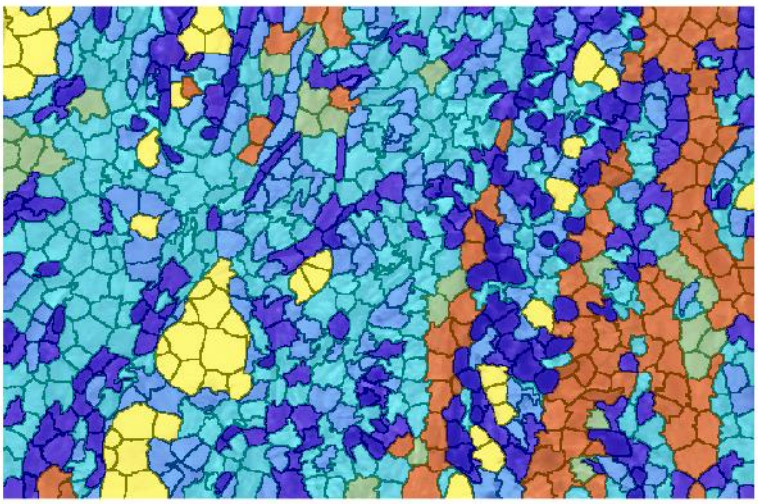

Figure 37. Clustering result of image 7, part 4

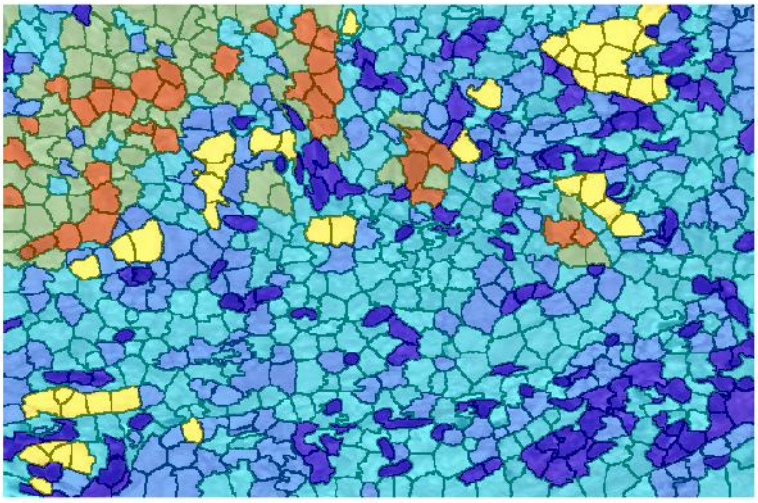

Figure 38. Clustering result of image 8, part 4

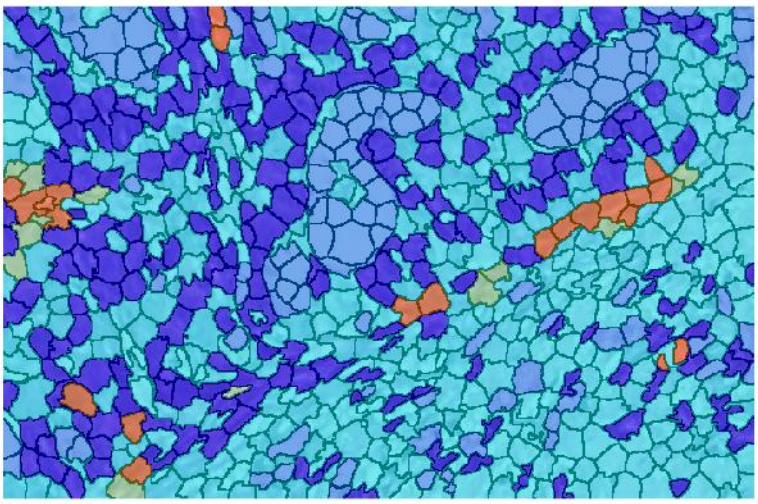

Figure 39. Clustering result of image 9, part 4

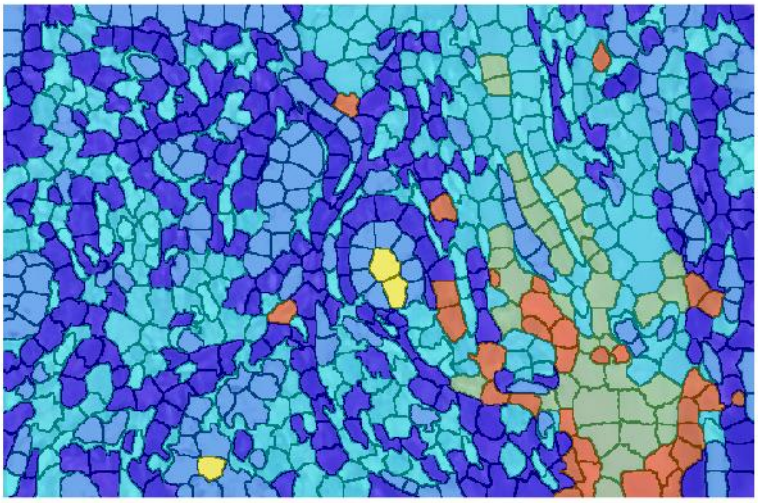

Figure 40. Clustering result of image 10, part 4

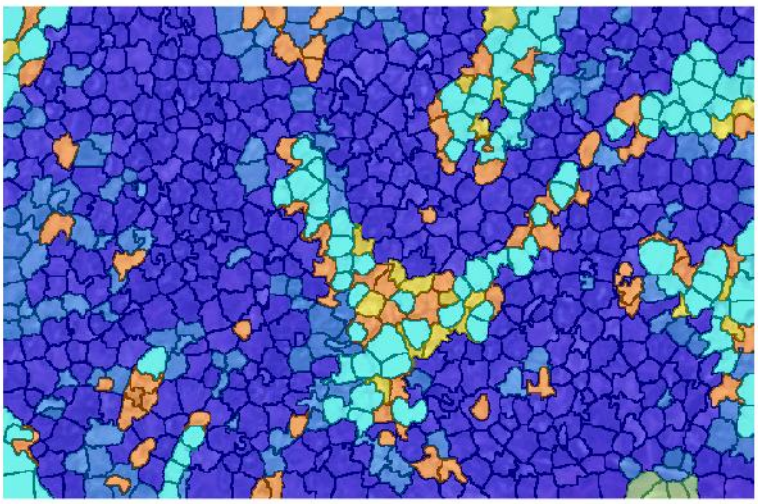

Figure 41. Clustering result of image 1, part 5

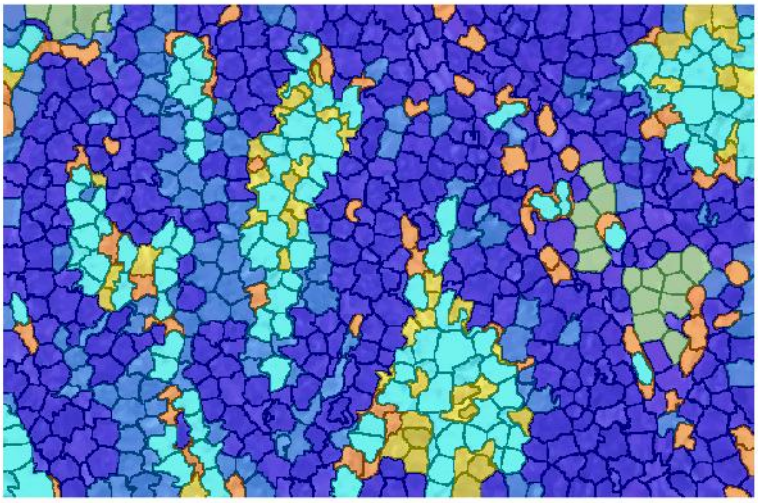

Figure 42. Clustering result of image 2, part 5

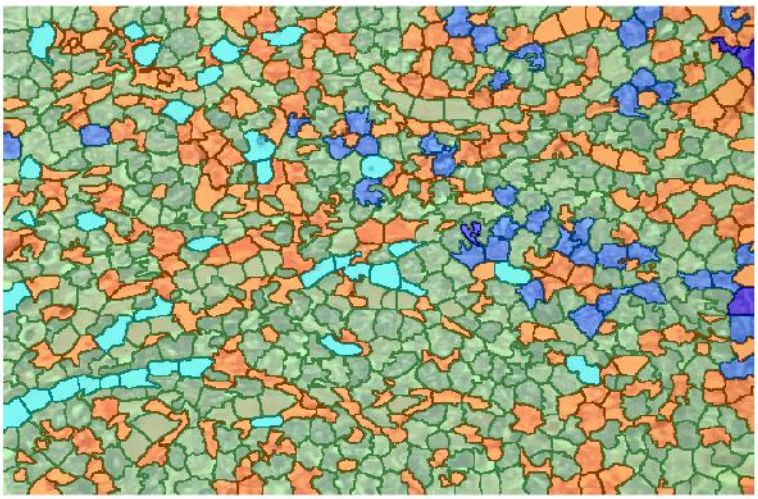

Figure 43. Clustering result of image 3, part 5

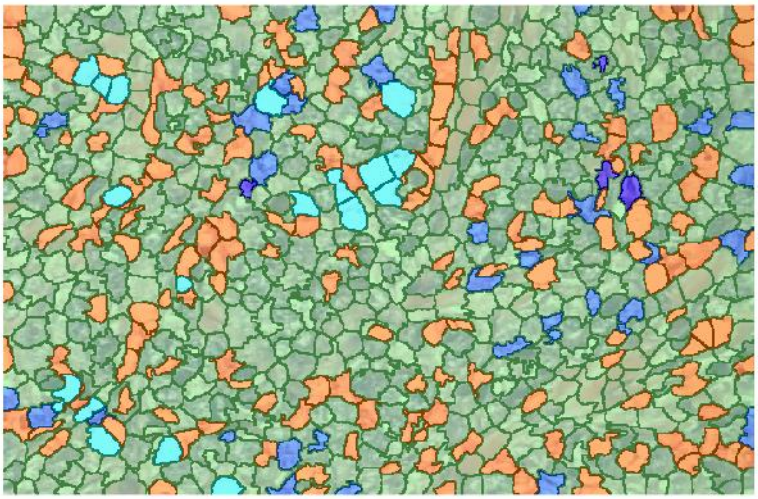

Figure 44. Clustering result of image 4, part 5

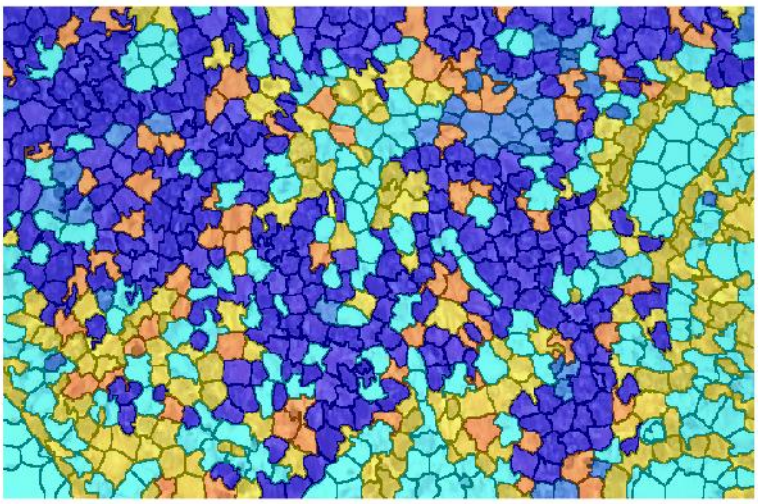

Figure 45. Clustering result of image 5, part 5

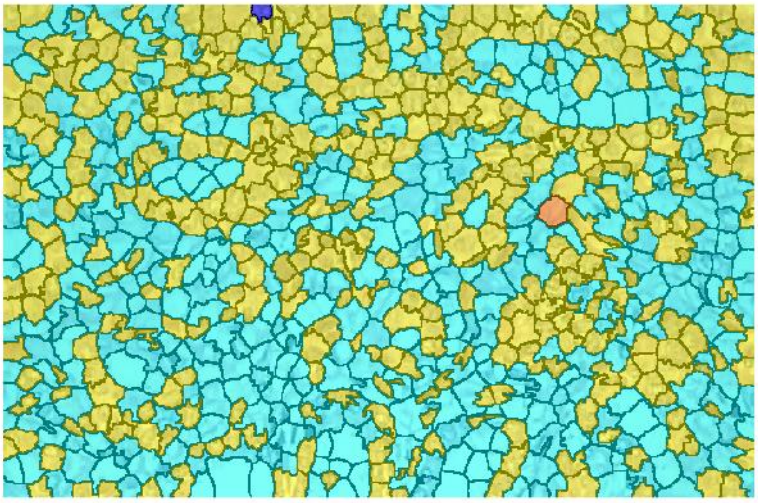

Figure 46. Clustering result of image 6, part 5

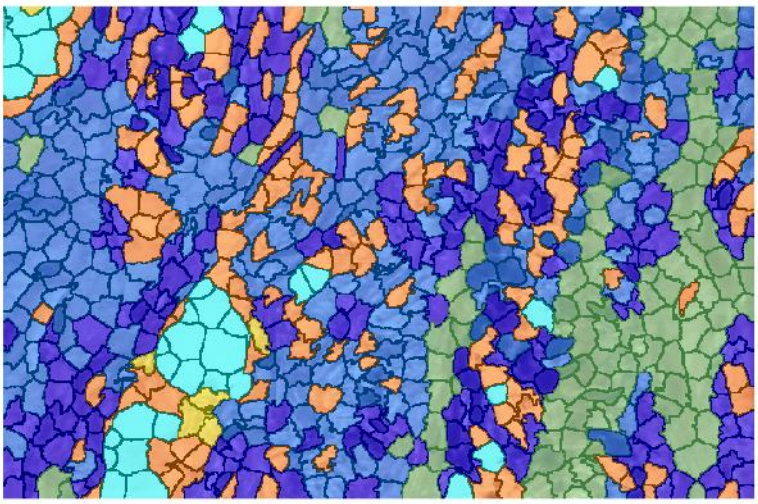

Figure 47. Clustering result of image 7, part 5

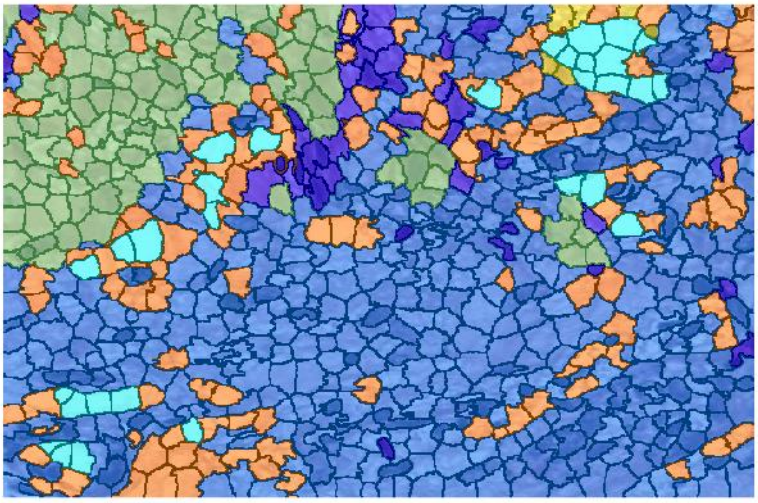

Figure 48. Clustering result of image 8, part 5

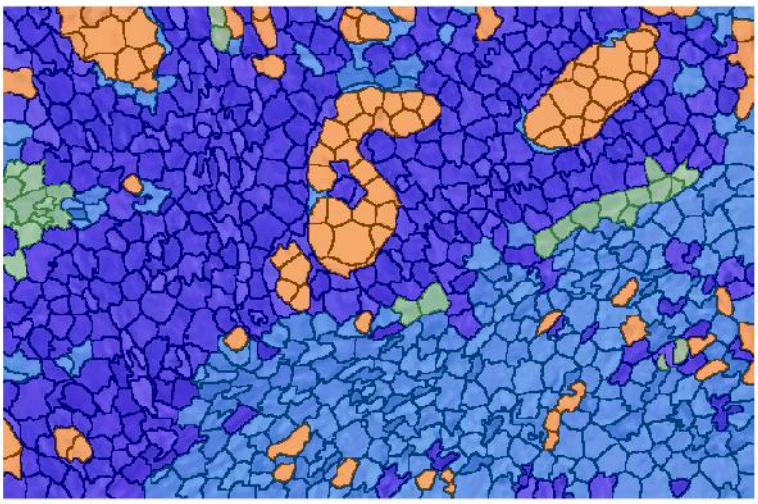

Figure 49. Clustering result of image 9, part 5

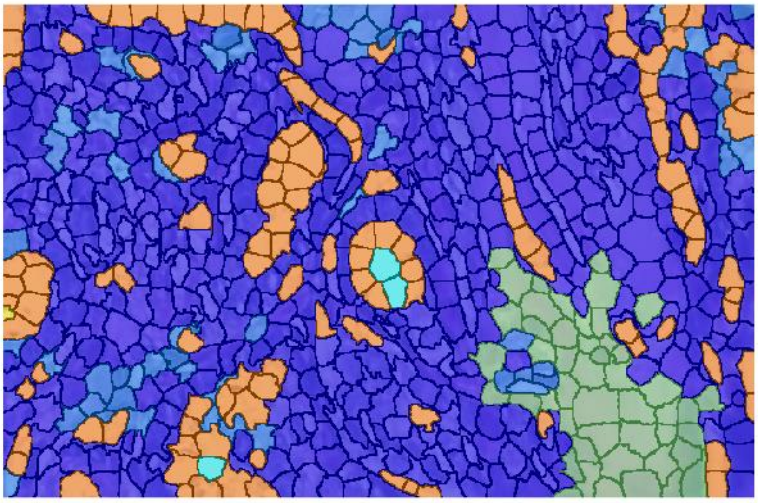

Figure 50. Clustering result of image 10, part 5

Part 6 – Source code

```
% Metehan Kaya - 21401258


close all;
imageAnalysis();


function imageAnalysis()


   allLs = {}; % labels
   allHFs = {}; % histogram features
   allBWs = {}; % boundary masks
   RGBimages = {}; % original images
   histFeature = []; % merged histogram feature matrix
   cntSuperpixel = []; % # of superpixels
   inputPathSuffix = [ "01.png" , "02.png" , "03.png" , "04.png" , "05.png" , "06.png" , "07.png" , "08.png" , "09.png" , "10.png" ];


   % Calc necessary part4 feature matrix
   for imageId = 1 : 10
      dataFilePath = strcat( 'images/' , inputPathSuffix(1,imageId) );
      [ histImageFeature , L , imageRGB , BW ] = getHistImageFeature( dataFilePath , imageId );
      allLs{imageId} = L;
      allBWs{imageId} = BW;
      RGBimages{imageId} = imageRGB;
      allHFs{imageId} = histImageFeature;
      cntSuperpixel = [ cntSuperpixel , size( histImageFeature , 1 ) ];
      histFeature = [ histFeature ; histImageFeature ];
   end


   % --- STEP 4 ---
```

```
    % display part4 figure
    displayFigure( histFeature , allLs , cntSuperpixel , RGBimages , allBWs , 0 );

    % --- STEP 5 ---

    allHCFs = {}; % history complex features
    histComplexFeature = []; % merged complex feature matrix
    for imageId = 1 : 10
        L = allLs{imageId};
        imageRGB = RGBimages{imageId};
        histImageFeature = allHFs{imageId};
        histComplexImageFeature = findComplexFeature( L , imageRGB , histImageFeature );
        allHCFs{imageId} = histComplexImageFeature;
        histComplexFeature = [ histComplexFeature ; histComplexImageFeature ];
    end

    % display part3 figure
    displayFigure( histComplexFeature , allLs , cntSuperpixel , RGBimages , allBWs , 10 );

end

% responsible for displaying part4/5 figures
function displayFigure( histFeature , allLs , cntSuperpixel , RGBimages , allBWs , totalPrev )

    % kmeans
    kClusters = 6;
    tHistFeature = transpose( histFeature );
    [ centers , assignments ] = vl_kmeans( tHistFeature , kClusters );

    clusterMatrices = {};
    totalPrevSuperpixels = 0;
```

```
% pixel -> cluster matrix
for imageId = 1 : 10
   L = allLs{imageId};
   [ height , width ] = size( L );
   clusterMatrix = zeros( height , width , 'uint32' );
   for i = 1 : height
      for j = 1 : width
         labelId = L(i,j);
         clusterId = assignments( 1 , labelId + totalPrevSuperpixels );
         clusterMatrix(i,j) = clusterId;
      end
   end
   clusterMatrices{imageId} = clusterMatrix;
   totalPrevSuperpixels = totalPrevSuperpixels + cntSuperpixel( 1 , imageId );
end

% display 2-layered figures
for imageId = 1 : 10
   if imageId > 0
      BW = allBWs{imageId};
      rbg = RGBimages{imageId};
      clusterMatrix = clusterMatrices{imageId};
      figure;
      imshow( imoverlay( rbg , BW , 'black' ) , 'InitialMagnification' , 67 )
      hold on;
      img = imshow( uint8( label2rgb( clusterMatrix ) ) );
      set( img , 'AlphaData' , 0.5 );
      hold off;
      saveas( gcf , [ 'outputCluster/' num2str( totalPrev+imageId , '%02d' )  '.png' ] );
   end
```

```
    end

end

% calc Nx120 matrix for part 5
function histComplexImageFeature = findComplexFeature( L , imageRGB , histImageFeature )

    % initializations

    [ height , width ] = size( L );
    cntSuperpixel = size( histImageFeature , 1 );

    cnt = zeros( 1 , cntSuperpixel );
    sumRow = zeros( 1 , cntSuperpixel );
    sumCol = zeros( 1 , cntSuperpixel );
    rowMin = zeros( 1 , cntSuperpixel );
    rowMax = zeros( 1 , cntSuperpixel );
    colMin = zeros( 1 , cntSuperpixel );
    colMax = zeros( 1 , cntSuperpixel );
    histComplexImageFeature = zeros( cntSuperpixel , 120 );

    rowMin( 1:1 , 1:cntSuperpixel ) = height + 1;
    colMin( 1:1 , 1:cntSuperpixel ) = width + 1;

    % find properties of superpixels for each of them
    for i = 1 : height
        for j = 1 : width
            id = L(i,j);
            cnt(1,id) = cnt(1,id) + 1;
            sumRow(1,id) = sumRow(1,id) + i;
            sumCol(1,id) = sumCol(1,id) + j;
```

```
        rowMin(1,id) = min( rowMin(1,id) , i );
        rowMax(1,id) = max( rowMax(1,id) , i );
        colMin(1,id) = min( colMin(1,id) , j );
        colMax(1,id) = max( colMax(1,id) , j );
    end
end

% calc matrix
for superId = 1 : cntSuperpixel

    % center coordinate of the superpixel
    centerRow = sumRow(1,superId) / cnt(1,superId);
    centerCol = sumCol(1,superId) / cnt(1,superId);

    % counter for each circle
    markYellow = zeros( 1 , cntSuperpixel );
    markGreen = zeros( 1 , cntSuperpixel );

    % find diameter & radiuses
    diameter = sqrt( ( rowMax(1,superId) - rowMin(1,superId) + 1 ) * ( colMax(1,superId) - colMin(1,superId) + 1 ) );
    yr = 1.5 * diameter;
    gr = 2.5 * diameter;

    rowLow = max( int32( 1 ) , int32( centerRow - gr ) );
    rowHigh = min( int32( height ) , int32( centerRow + gr ) );
    colLow = max( int32( 1 ) , int32( centerCol - gr ) );
    colHigh = min( int32( width ) , int32( centerCol + gr ) );

    % calc counter for each circle
    for r = rowLow : rowHigh
```

```
   for c = colLow : colHigh
      id = L(r,c);
      R = double(r);
      C = double(c);
      dist = ( R - centerRow ) * ( R - centerRow ) + ( C - centerCol ) * ( C - centerCol );
      if dist < yr*yr
         markYellow( 1 , id ) = markYellow( 1 , id ) + 1;
      elseif dist < gr*gr
         markGreen( 1 , id ) = markGreen( 1 , id ) + 1;
      end
   end
end

% denominator threshold
myThreshold = 5.0;

% calc Nx40 matrix
cntYellow = 0;
histYellow = zeros( 1 , 40 );
for i = 1 : cntSuperpixel
   if double( markYellow( 1 , i ) ) >= double( cnt(1,i) ) / myThreshold
      cntYellow = cntYellow + 1;
      histYellow( 1 , : ) = histYellow( 1 , : ) + histImageFeature( i , : );
   end
end
histYellow( 1 , : ) = histYellow( 1 , : ) / cntYellow;

% calc Nx40 matrix
cntGreen = 0;
histGreen = zeros( 1 , 40 );
for i = 1 : cntSuperpixel
```

```
        if double( markGreen( 1 , i ) ) >= double( cnt(1,i) ) / myThreshold
            cntGreen = cntGreen + 1;
            histGreen( 1 , : ) = histGreen( 1 , : ) + histImageFeature( i , : );
        end
    end
    histGreen( 1 , : ) = histGreen( 1 , : ) / cntGreen;

    % copy
    histComplexImageFeature( superId , : ) = [ histImageFeature( superId , : ) histYellow( 1 , : ) histGreen( 1 , : ) ];

  end

end

% get hist 40xN matrix for part 5
function [ histImageFeature , L , imageRGB , BW ] = getHistImageFeature( dataFilePath , imageId )

  % --- STEP 1 ---
  filePath = convertStringsToChars( dataFilePath );
  imageRGB = imread( filePath );
  [ height, width, numBands ] = size( imageRGB );

  % superpixel segmentation
  kSuperpixels = 750;
  [L,N] = superpixels( imageRGB , kSuperpixels );
  BW = boundarymask(L);
  if imageId > 0
    figure;
    imshow( imoverlay( imageRGB , BW , 'cyan' ) , 'InitialMagnification' , 67 )
    saveas( gcf , [ 'outputSuperpixel/' num2str(imageId,'%02d')  '.png' ] );
```

```
end


% --- STEP 2 ---

imageGray = rgb2gray( imageRGB );
%figure
%imshow( imageGray );

wavelength = [ 2 , 4 , 5 , 10 ];
orientation = [ 30 , 45 , 60 , 90 ];
mag = zeros( 4 , 4 , height , width );
phase = zeros( 4 , 4 , height , width );
avgGabors = zeros( N , 16 );

% gabor feature extraction
for i = 1 : 4
  for j = 1 : 4
    [tmpMag,tmpPhase] = imgaborfilt( imageGray , wavelength(1,i) , orientation(1,j) );
    if imageId < 6 && ( i == 1 || i == 3 ) && ( j == 1 || j == 3 )
      figure;
      imshow( tmpMag , [] )
      saveas( gcf , [ 'outputGabor/' num2str(imageId,'%02d') '_' int2str(i) '_' int2str(j) '.png' ] );
    end
    mag(i,j,:,:) = tmpMag;
    phase(i,j,:,:) = tmpPhase;
    gaborAvg = getAverageGabor( tmpMag , N , L );
    for k = 1 : N
      avgGabors( k , 4*(i-1) + j ) = gaborAvg(1,k);
    end
  end
```

```
end

% normalization
for i = 1 : N
   avgGabors(i,:) = avgGabors(i,:) / norm( avgGabors(i,:) );
end

% --- STEP 3 ---

imageRGB_R = imageRGB(:,:,1);
imageRGB_G = imageRGB(:,:,2);
imageRGB_B = imageRGB(:,:,3);

% LAB color space

imageLAB = rgb2lab( imageRGB );
imageLAB_L = imageLAB(:,:,1);
minLAB_L = min( imageLAB_L(:) );
maxLAB_L = max( imageLAB_L(:) );
imageLAB_A = imageLAB(:,:,2);
minLAB_A = min( imageLAB_A(:) );
maxLAB_A = max( imageLAB_A(:) );
imageLAB_B = imageLAB(:,:,3);
minLAB_B = min( imageLAB_B(:) );
maxLAB_B = max( imageLAB_B(:) );

labCounter = zeros( N , 24 );

for i = 1 : height
   for j = 1 : width
      [l,a,b] = getHistLAB( imageLAB_L(i,j) , imageLAB_A(i,j) , imageLAB_B(i,j) );
```

```
            id = L(i,j);
            labCounter(id,l) = labCounter(id,l) + 1;
            labCounter(id,a+8) = labCounter(id,a+8) + 1;
            labCounter(id,b+16) = labCounter(id,b+16) + 1;
        end
    end

    % normalization
    for i = 1 : N
        labCounter(i,:) = labCounter(i,:) / norm( labCounter(i,:) );
    end

    % merge
    histImageFeature = [ avgGabors , labCounter ];

end

% convert LAB values to interval indices
function [l,a,b] = getHistLAB( lValue , aValue , bValue )
    lMin = 0;
    lMax = 100;
    lBand = ( lMax - lMin ) / 8.0;
    aMin = -87;
    aMax = 99;
    aBand = ( aMax - aMin ) / 8.0;
    bMin = -108;
    bMax = 95;
    bBand = ( bMax - bMin ) / 8.0;
    l = ( lValue - lMin ) / lBand;
    l = floor( l );
    l = min( l , 7 );
```

```
    a = ( aValue - aMin ) / aBand;
    a = floor( a );
    a = min( a , 7 );
    b = ( bValue - bMin ) / bBand;
    b = floor( b );
    b = min( b , 7 );
end

% calcs gabor average for each superpixel
function gaborAvg = getAverageGabor( mag , N , L )
    sum = zeros( 1 , N );
    count = zeros( 1 , N );
    [ height , width ] = size( L );
    for i = 1 : height
        for j = 1 : width
            value = L(i,j);
            count( 1 , value ) = count( 1 , value ) + 1;
            sum( 1 , value ) = sum( 1 , value ) + mag( i , j );
        end
    end
    gaborAvg = zeros( 1 , N );
    for i = 1 : N
        if count( 1 , i ) == 0
            fprintf( "Division by 0?\n" );
        end
        gaborAvg( 1 , i ) = sum( 1 , i ) / count( 1 , i );
    end
end
```

Part 7 – Citation for external code

Superpixel

[L,NumLabels] = superpixels(A,N)"

"BW = boundarymask(L)"

"imshow(imoverlay(A,BW,'cyan'),'InitialMagnification',67)"

Gabor Filter

[mag,phase] = imgaborfilt(A,wavelength,orientation)

Part 8 – Discussion

I tested my code based on 3 different parameters which are # of clusters, # of superpixels, and threshold that is for checking if a superpixel is in the circle whose center is the superpixel that it is currently looked for.

# of clusters is one of the important parameters. When it is decreased too much, I lose details. One type of superpixel dominates the image since it does not care about differences between superpixels. As I expected, when it is increased too much, I face with separation of the superpixels that actually belong to the same component. The reason behind this separation is that the code focuses on small details. In other terms, pair of superpixels differ by small differences. Because of these small differences, it is unreasonable to separate them. Therefore, my $kClusters = 6$.

# of superpixels is another important parameter that affects quality of the outputs in terms of details. Similar to effects of # of clusters, when # of superpixels is decreased too much, I lose details. Many irrelevant pixels merge at single superpixel. Therefore, it becomes harder to detect differences. A few types of clusters dominate the image. When it is increased too much, it focuses on small details that are hard to see normally. Many pairs of close superpixels become separated from each other. Therefore my $kSuperpixels = 750$.

Part5 improves the clustering/segmentation results because it also checks surrounding superpixels. If segmentation of superpixels was only based on themselves, it would separate superpixels that are close to each other. I use a denominator to find $threshold = \frac{\#\ of\ pixels\ in\ the\ superpixel}{denominator}$. It is unreasonable to use an absolute threshold for superpixels with different sizes. When I decrease denominator, and that means increasing threshold value, I lose details. Many pairs of superpixels become the same. If denominator is decreased or threshold is increased, I lose superpixels that are close to each other. Therefore, my $denominator = 5$.

To sum up, it's all about a tradeoff between small details and big details.