Table of Contents

3D Bone Reconstruction by Gabor Transform

```
tr_sh = urlread('http://bit.ly/QrYlmQ');
```

So what is a Gabor Filter?

A Gabor filter is a Gaussian envelope modulated by a complex sinusoid. This filter, developed by Dennis Gabor, resembles 2d visual cortical filters and has been shown to be useful in computer vision, e.g. edge detection and pattern discrimination.

For a 2D gabor filter, the shape of the filter can be varied by altering the size of the envelope with 'sigma', the direction of the sinusoid with 'theta', and the frequency of the sinusoid with 'F'. Below displays the real and imaginary part of a gabor filter.

For more information, start with Wikipedia's Gabor Filter page. http://en.wikipedia.org/wiki/Gabor_filter

```
[x,y]=meshgrid(-50:50,-50:50); sigma = 10; theta = pi/3; F = 0.04;
g_sigma = (1./(2*pi*sigma^2)).*exp(((-1).*(x.^2+y.^2))./(2*sigma.^2));
real_g = g_sigma.*cos((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));
im_g = g_sigma.*sin((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));

figure;
imagesc([real_g im_g]);colormap('gray');axis image;
title('real and imaginary parts of a Gabor filter');
```

Load images and create training and testing images.

```
clc; clear; close all;

scale = 1;

% get training and testing image and vessel location from the above
 images
testingImg = rgb2gray(im2double(imread('AxialTrain.tif')))*255;
testingImg = medfilt2(testingImg, [3 3]);
testingImg = wiener2(testingImg, [5,5]);

testingAns = im2double(imread('AxialGroundTruth.tif'))*255;
testingAns(testingAns==255) = 1;
testingAns(testingAns==0) = 0;

trainingImg = rgb2gray(im2double(imread('AxialTest.tif')))*255;
trainingImg = medfilt2(trainingImg, [3 3]);
trainingImg = wiener2(trainingImg, [5,5]);

trainingAns = im2double(imread('AxialTestTrue.tif'))*255;
trainingAns(trainingAns==255) = 1;
trainingAns(trainingAns==0) = 0;
```

Extract features from training image.

```
% initialize parameters for Gabor transforms
filter_size = 40.*scale;
filter_size_halfed = round((filter_size)/2);
Fs = 0.1:0.1:0.3;
sigmas = [2:2:8].*scale;
thetas=pi/8:pi/8:pi-pi/8;

% initialize array for storing features
features =
 zeros([size(trainingImg),numel(sigmas),numel(thetas),numel(Fs)]);

h1 = figure;
% perform multiple Gabor transforms with varying parameters
for k = 1:numel(sigmas)
for j = 1:numel(Fs)
for i = 1:numel(thetas)


    sigma = sigmas(k);
    F = Fs(j);
    theta = thetas(i);

    % setup the Gabor transform
    [x,y]=meshgrid(-filter_size_halfed:filter_size_halfed,-
filter_size_halfed:filter_size_halfed);
    g_sigma = (1./(2*pi*sigma^2)).*exp(((-1).*(x.^2+y.^2))./
(2*sigma.^2));
    real_g = g_sigma.*cos((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));
    im_g = g_sigma.*sin((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));

    % perform Gabor transform
    uT
 =sqrt(conv2(trainingImg,real_g,'same').^2+conv2(trainingImg,im_g,'same').^2);

    % normalize transformed image
    uT = (uT-mean(uT(:)))./std(uT(:));
    % append tranformed images to 'features'
    features(:,:,k,j,i) = uT;

    % visualize filtered image and the varying filters
    subplot(2,1,1);
    imagesc([trainingImg mat2gray(uT).*255],[0 255]);
    colormap('gray'); axis image; axis off;
    title('testing image and the Gabor transformed image');
    subplot(2,1,2);
    imagesc([real_g im_g]);
    colormap('gray'); axis image; axis off;
    title(sprintf('Gabor filter F:%1.2f t:%1.2f k:
%1.f',F,theta,sigma));

    drawnow;

end
end
end
```

Fit GLM with features and location of the vessels

```
% reshape feature array
szG = size(features);
features = reshape(features,[prod(szG(1:2)),prod(szG(3:end))]);

% fit GLM with the features and the location of the vessels
b = glmfit(features,trainingAns(:),'normal');

% see the output of the model based on the training features
CTrain = glmval(b,features,'logit');
CTrain = reshape(CTrain,szG(1:2));

% visualize
h2= figure;
imagesc([trainingImg trainingAns.*255 CTrain.*255]);
colormap('gray');axis image;
title('testing image, answer, output from GLM');
```

Perform cross validation for Gabor+GLM

```
features =
 zeros([size(testingImg),numel(sigmas),numel(thetas),numel(Fs)]);
for k = 1:numel(sigmas)
for j = 1:numel(Fs)
for i = 1:numel(thetas)

    sigma = sigmas(k);
    F = Fs(j);
    theta = thetas(i);
    % setup the "Gabor transform"
    [x,y]=meshgrid(-filter_size_halfed:filter_size_halfed,-
filter_size_halfed:filter_size_halfed);
    g_sigma = (1./(2*pi*sigma^2)).*exp(((-1).*(x.^2+y.^2))./
(2*sigma.^2));
    real_g = g_sigma.*cos((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));
    im_g = g_sigma.*sin((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));

    % perform Gabor transform
    uT
 =sqrt(conv2(testingImg,real_g,'same').^2+conv2(testingImg,im_g,'same').^2);

    % normalize transformed image
    uT = (uT-mean(uT(:)))./std(uT(:));

    % append tranformed images to 'features'
    features(:,:,k,j,i) = uT;

end
end
end

% feed the features to GLM.
szG = size(features);
features = reshape(features,[prod(szG(1:2)),prod(szG(3:end))]);
Ctest = glmval(b,features,'logit');
Ctest = reshape(Ctest,szG(1:2));

% calculate sensitivity and specificity by thresholding
% the output of GLM 'Ctest' and comparing the thresholded image with
 the answer.
groundBrightness = mean(Ctest(:));

sensitivity = [];
specificity = [];
rgs = 0:0.01:1;
for i = rgs

    tmpBwImg = im2bw(Ctest,i);

    tp = sum(tmpBwImg == 1 & testingAns ==1);
    fn = sum(tmpBwImg == 0 & testingAns ==1);
    tn = sum(tmpBwImg == 0 & testingAns ==0);
    fp = sum(tmpBwImg == 1 & testingAns ==0);

    sensitivity = [sensitivity tp/(tp+fn)]; %true positive rate
    specificity = [specificity tn/(tn+fp)]; %true negative rate

end

% plot roc curve
h3 = figure;
plot(1-specificity,sensitivity,'k-','linewidth',2);
```

```
xlabel('False Positive Rate (1-Specificity)');
ylabel('True Positive Rate (Sensitivity)');
axis([0 1 0 1]);grid on;

% calculate auc.
[fprSort, fprSortInd] = sort([1-specificity],'ascend');
auc = trapz([0 fprSort 1],[0 sensitivity(fprSortInd) 1]);
title(sprintf('ROC curve, AUC: %1.2f',auc));

% get optimal threshold by maximizing Youden's index
[trsh, thInd] = max(sensitivity + specificity - 1);
th = rgs(thInd);
```

Visualize testing image and the detected vessels

Shows the testing image, the output image from the GLM and a thresholded image with a threshold that has a relatively good sensitivity and specificity.

```
h4 = figure;
imagesc([testingImg Ctest.*255 (Ctest > th).*255]);
colormap('gray');axis image;
title('original image, output from GLM, optimally thresholded output
 from GLM');
```

Use the learned model on entire video

```
axialVid = importdata('AxialBone.mat');

testSize = size(axialVid, 4);

segmentedVideoSize = [size(axialVid, 1), size(axialVid, 2), testSize];
segmentedVid = zeros(segmentedVideoSize);

for t = 1 : testSize
 currentFrame = axialVid(:,:,:,t);
 currentFrame = rgb2gray(im2double(currentFrame))*255;

 currentFrame = medfilt2(currentFrame, [3 3]);
  currentFrame = wiener2(currentFrame, [5,5]);

 features =
 zeros([size(currentFrame),numel(sigmas),numel(thetas),numel(Fs)]);
 for k = 1:numel(sigmas)
 for j = 1:numel(Fs)
 for i = 1:numel(thetas)

  sigma = sigmas(k);
  F = Fs(j);
  theta = thetas(i);

  % setup the "Gabor transform"
  [x,y]=meshgrid(-filter_size_halfed:filter_size_halfed,-
filter_size_halfed:filter_size_halfed);
  g_sigma = (1./(2*pi*sigma^2)).*exp(((-1).*(x.^2+y.^2))./
(2*sigma.^2));
  real_g = g_sigma.*cos((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));
  im_g = g_sigma.*sin((2*pi*F).*(x.*cos(theta)+y.*sin(theta)));

  % perform Gabor transform
  uT
 =sqrt(conv2(currentFrame,real_g,'same').^2+conv2(currentFrame,im_g,'same').^2);

  % normalize transformed image
  uT = (uT-mean(uT(:)))./std(uT(:));

  % append tranformed images to 'features'
  features(:,:,k,j,i) = uT;

 end
 end
 end

 % feed the features to GLM.
 szG = size(features);
 features = reshape(features,[prod(szG(1:2)),prod(szG(3:end))]);
 segmentedImage = glmval(b,features,'logit');
 segmentedImage = reshape(segmentedImage,szG(1:2));

 segmentedImage = (segmentedImage > th) * 255;
 segmentedImage = segmentImage(segmentedImage);
 segmentedImage = ExtractNLargestBlobs(segmentedImage, 1);

 se = strel('disk', 4);
 segmentedImage = imclose(segmentedImage, se);
 segmentedImage = imfill(segmentedImage, 'holes');


 segmentedVid(:,:,t) = segmentedImage;

 disp(t);
end
```

References

Soares, Joao VB, et al. "Retinal vessel segmentation using the 2-D Gabor wavelet and supervised classification." Medical Imaging, IEEE Transactions on 25.9 (2006): 1214-1222. http://www.ncbi.nlm.nih.gov/pubmed/16967806

Sandberg, Berta, Tony Chan, and Luminita Vese. "A level-set and gabor-based active contour algorithm for segmenting textured images." UCLA Department of Mathematics CAM report. 2002. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.3145

```
* The above cross validation is only for illustration purposes.
To properly evaluate this method, you can use images
from the STARE or DRIVE projects
```

```
http://www.ces.clemson.edu/~ahoover/stare/probing/index.html
http://www.isi.uu.nl/Research/Databases/DRIVE/
and your desired cross validation methods
http://http://en.wikipedia.org/wiki/Cross-validation_(statistics)
** With the STARE datasets (N = 20) and using a leave-one-out cross
validation, the Gabor+GLM was able to achieve an AUC of 0.94 for
detecting the retinal vessels in the images.
```

Published with MATLAB® R2016b