

Automatic Image Analysis

Übung 3

QIN, Yujie

mail@qinyujie.net

15.01.2009

1. calcDiffMatrices Funktion

```
function [distMat, diffX, diffY]=calcDiffMatrices(imgSize)
    mx=imgSize(2)/2;
    my=imgSize(1)/2;
    distMat=zeros(imgSize(1), imgSize(2));
    %work out the distance between all point and the middle previously calculated
    for i=1:imgSize(2)
        for j=1:imgSize(1)
            distMat(j,i)=(j-my)^2+(i-mx)^2;
        end;
    end;
    %shift the matrix so the zero is centered
    distMat=fftshift(distMat);

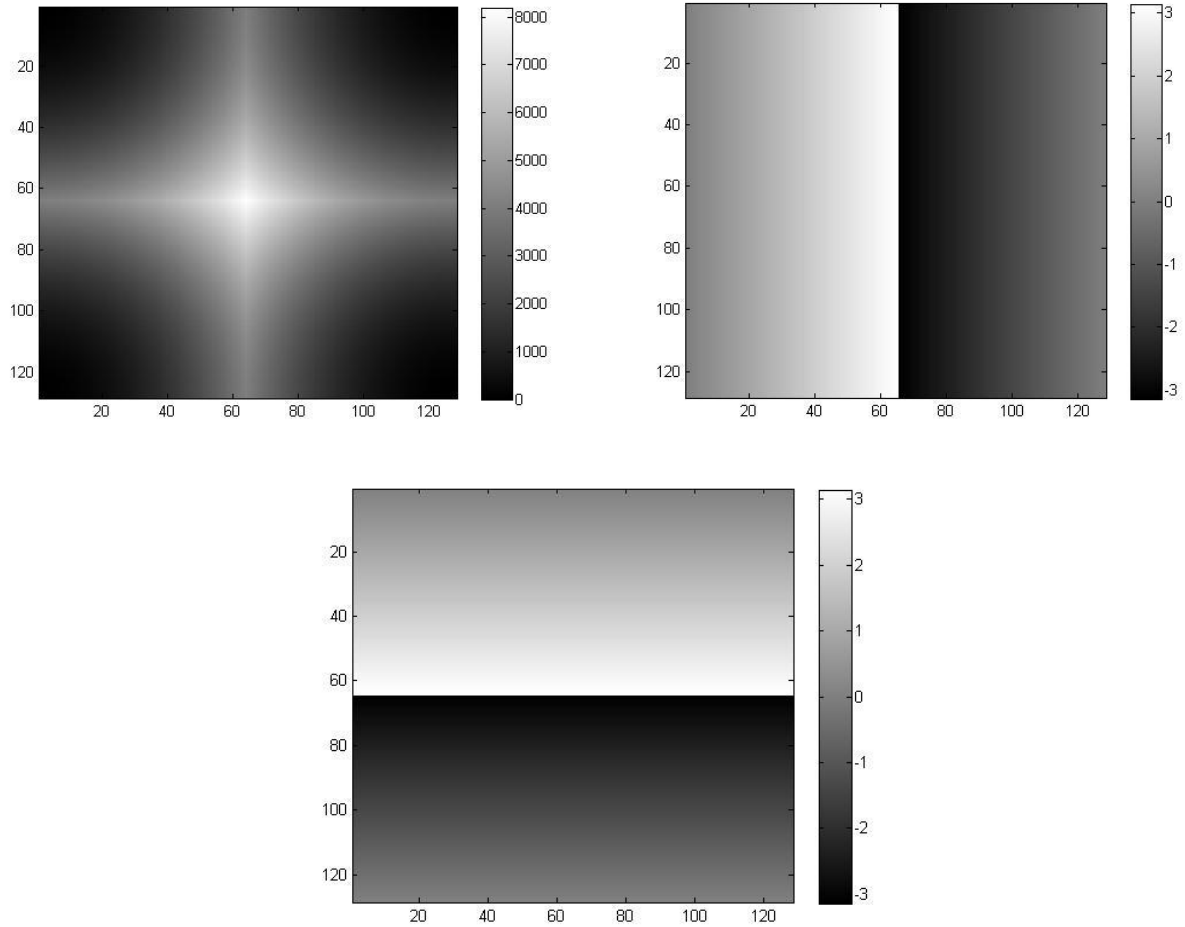
    %works out the magic derivators
    diffX=zeros(imgSize(1), imgSize(2));
    diffY=zeros(imgSize(1), imgSize(2));
    for i=-mx+1:mx
        diffX(:,i+mx)=i;
    end;

    for j=-my+1:my
        diffY(j+my,:)=j;
    end;

    diffX=(2*pi*1i/imgSize(2))*diffX;
    diffY=(2*pi*1i/imgSize(1))*diffY;

    %switch the zeros to the origins of the respective pictures
    diffX=circshift(diffX,[0 mx+1]);
    diffY=circshift(diffY,my);

    %figure, image(distMat,'CDataMapping','scaled'), colormap(gray(256)), colorbar,
    caxis([0 max(max(distMat))])
    %figure, image(imag(diffX),'CDataMapping','scaled'), colormap(gray(256)), colorbar,
    caxis([-pi pi])
    %figure, image(imag(diffY),'CDataMapping','scaled'), colormap(gray(256)), colorbar,
    caxis([-pi pi])
end
```



2. Detect Extrema Funktion

```
function [extremCoord, extremWeight] = detectExtrema(img, scaleRange, scaleSteps,
extremThresh)
    imgSize=size(img);
    %calculate Magic Derivation Matrix
    [distMat diffX diffY]=calcDiffMatrices(imgSize);

    G=ones(imgSize(1),imgSize(2),scaleSteps);
    H=ones(imgSize(1),imgSize(2),scaleSteps,3);
    E=ones(imgSize(1),imgSize(2),scaleSteps);
    %makes a linear subdivision out of the interval of the squared root of t
    st=linspace(scaleRange(1),scaleRange(2),scaleSteps);

    %FFT of the original image
    L=fft2(img);
    for i=1:scaleSteps
        t=st(i)^2;
        %compute Gaussian kernel
        G(:,:,i)=(1/(pi*t))*exp(-distMat/t);

        %works out Hxx
        H(:,:,i,1)=ifft2(L.*fft2(G(:,:,i)).*diffX.^2*t);
        %works out Hxy
        H(:,:,i,2)=ifft2(L.*fft2(G(:,:,i)).*diffX.*diffY*t);
```

```

    %works out Hyy
    H(:,:,i,3)=ifft2(L.*fft2(G(:,:,i)).*diffY.^2*t);
    %works out the determinant
    E(:,:,i) = H(:,:,i,1).*H(:,:,i,3)-H(:,:,i,2).^2;
end;
%works out the extremum relative to the extremum of the determinant
GextremThresh=extremThresh*abs(max(E(:)));
[maxInd1, maxVal1]=findLocalMaxima(E, GextremThresh);
%Blob detection
extremCoord = [];
index = 0;
for i=1:size(maxInd1,1)
    index = index + 1;
    %can't explain why but the extremum don't seem centered without adding 1
    extremCoord(index,1)=maxInd1(i,2)+1;
    extremCoord(index,2)=maxInd1(i,1);
    extremCoord(index,3)=st(maxInd1(i,3));
    extremCoord(index,4)=st(maxInd1(i,3));
    extremCoord(index,5)=0;
    extremWeight(index)=maxVal1(i);
end;
end

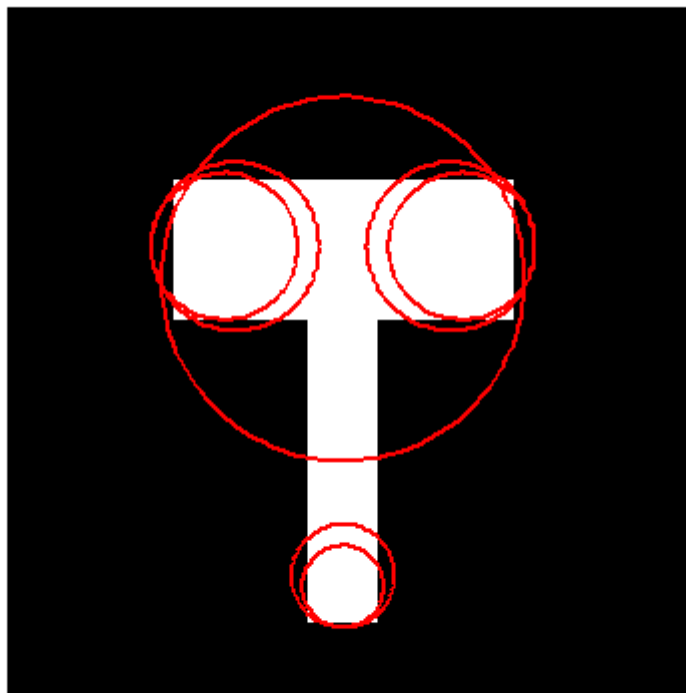
```

Results with:

```

scaleRange = [1.5,80];
scaleSteps = 40;
detectThresh = 0.2;

```



3. Detect Ridges Funktion

```
function [ridgeCoord, ridgeWeight] = detectRidges(img, scaleRange, scaleSteps, ridgeThresh)

imgSize=size(img);
[distMat diffX diffY]=calcDiffMatrices(imgSize);

G=ones(imgSize(1),imgSize(2),scaleSteps);
L=ones(imgSize(1),imgSize(2),scaleSteps,4);
A=ones(imgSize(1),imgSize(2),scaleSteps,4);
B=ones(imgSize(1),imgSize(2),scaleSteps,4);
R=ones(imgSize(1),imgSize(2),scaleSteps);
st=linspace(scaleRange(1),scaleRange(2),scaleSteps);

FIMG=fft2(img);
for i=1:scaleSteps
    t=st(i)^2;
    %compute Gaussian kernel
    G(:,:,i)=(1/(pi*t))*exp(-distMat/t);
    FG = fft2(G(:,:,i));

    %works out Lx
    L(:,:,i,1)=ifft2(FIMG.*FG.*diffX*st(i));
    %works out Ly
    L(:,:,i,2)=ifft2(FIMG.*FG.*diffY*st(i));
    %works out A
    A(:,:,i,1) = ifft2(FG .* fft2(L(:,:,i,1).^2));
    A(:,:,i,2) = ifft2(FG .* fft2(L(:,:,i,1).* L(:,:,i,2)));
    A(:,:,i,3) = A(:,:,i,2);
    A(:,:,i,4) = ifft2(FG .* fft2(L(:,:,i,2).^2));
    %works out B
    B(:,:,i,1) = ifft2(FG .* fft2(L(:,:,i,1))).^2;
    B(:,:,i,2) = ifft2(FG .* fft2(L(:,:,i,1))).* ifft2(FG .* fft2(L(:,:,i,2)));
    B(:,:,i,3) = B(:,:,i,2);
    B(:,:,i,4) = ifft2(FG .* fft2(L(:,:,i,2))).^2;
    B(:,:,i,:) = A(:,:,i,:) - B(:,:,i,:);
    %Compute the ridge strength R:
    for xx=1:imgSize(1)
        for yy=1:imgSize(2)
            tb = [B(xx,yy,i,1), B(xx,yy,i,2);B(xx,yy,i,3),B(xx,yy,i,4)];
            R(xx,yy,i)=trace(tb).^2 - 4*det(tb);
        end
    end
end;

GridgeThresh=ridgeThresh*max(R(:));
%Ridge detection
[maxInd, maxVal]=findLocalMaxima(R, GridgeThresh);
for i=1:size(maxInd,1)
    y=maxInd(i,1);
    x=maxInd(i,2);
    k=maxInd(i,3);
    ridgeCoord(i,1)=x;
    ridgeCoord(i,2)=y;
    ridgeCoord(i,3)=st(k);
end;
```

```
%works out 2x2 Hessian matrix eigen values and vectors
ta = [A(y,x,k,1), A(y,x,k,2);A(y,x,k,3),A(y,x,k,4)];
[V, D] = eig(ta);
lambda = [D(1,1), D(2,2)];
[lambda, index] = sort(lambda, 'descend');
V = [V(:,index(1)), V(:,index(2))];
ridgeCoord(i,4)=st(k)*sqrt(abs(lambda(1)./lambda(2)));
ridgeCoord(i,5)=atan2(V(1,2), -V(2,2));
ridgeWeight(i)=maxVal(i);
end;

end
```

Results with:

```
scaleRange = [1.1, 40];
scaleSteps = 60;
detectThresh = 0.2;
```

