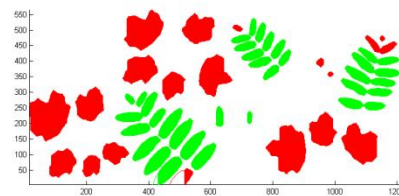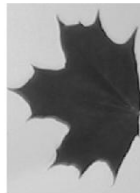# Leaf Classify

QIN, Yujie

03.11.2009

1. Grundlegenden Funktionen

```matlab
%function plotFD(F)
%   F: The (non-normalised) Fourier descriptor of a contour
% Plots the contour U described by F


function plotFD(F)
len = length(F);
if (len > 0)
    U = ifft(F);
    U = [U U(1)];
    plot(U);
end
end


%function [G]=shiftFD(F, x, y)
% F: The (non-normalised) Fourier descriptor of a contour
% x: X-Translation
```

```matlab
% y: Y-Translation
% G: The (non-normalised) Fourier descriptor of the shifted contour
% Translates the contour corresponding to F by (X,Y) Pixels
% This operation does not require an (inverse) DFT!

function [G]=shiftFD(F, x, y)
G=F;
N=length(F);
G(1)=F(1) + N .* (x + y*i);
end



%function [G]=scaleFD(F,scaleFactor)
% F: The (non-normalised) Fourier descriptor of a contour
% scaleFactor: Change in contour scale
% G: The (non-normalised) Fourier descriptor of the scaled contour
% Scales the contour corresponding to F by 100*scaleFactor %.
% This operation does not require an (inverse) DFT!

function [G]=scaleFD(F,scaleFactor)
G=100 .* scaleFactor .* F;
end



%function [G]=resizeFD(F, n)
% F: The Fourier descriptor of a contour
% G: The a resized Fourier descriptor with n elements
% Processes F to obtain G, which has been shortened to contain only n
%   elements (assume F has more than n elements).

function [G]=resizeFD(F, n)
G=F;
len=length(F);
if (len > n)
    T1=fftshift(F);
    p=round((len-n) ./ 2);
    T2=T1(p+1 : p+n);
    G=ifftshift(T2);
end
```

```matlab
end


%function [G]=normaliseFD(F)
% F: The (non-normalised) Fourier descriptor of a contour
% G: The normalised, invariant Fourier descriptor of the contour
% Processes F to obtain G, which is invariant to translation,
%     rotation and scaling of the original contour.

function [G]=normaliseFD(F)
T=F;
%Translation Invariance
T(1)=0;
%Scale Invariance
si=abs(T(2));
T=T ./ si;
%Rotation and changes in starting point
T=abs(T);
%Output the result
G=T;
end


%function [diff]=compareFD(F, G)
% F: Normalised, invariant Fourier descriptor
% G: Normalised, invariant Fourier descriptor
% diff: A measure of the difference between F and G
% Quantifies the difference between F and G (see lecture notes)

function [diff]=compareFD(F, G)
diff = -1;
if (length(F) == length(G))
    diff = sum((abs(F-G)) .^ 2);
end
end


%function [F]=extractFD(U)
% U: A vector of complex pixel coordinates on the contour of an
```

```matlab
%       object (in arbitrary order!)
% F: The non-normalised Fourier Descriptor of the contour
% Extracts the Fourier Descriptor of a contour, described by a vector
%   of coordinates

function [F]=extractFD(U)
%Shift to Center
c=sum(U) ./ length(U);
T=U - c;
%Sort by angle
aT=angle(T);
[aTT, aTI] = sort(aT);
for i = 1:length(aTI)
   sT(i)=U(aTI(i));
end
%Fit the case, which Length of U is not sufficient
if mod(length(aTI),2) == 0
    sT=[sT sT(1)];
end
%Output the result
F=fft(sT);
End
```

2. Fortgeschritten Funktionen

```matlab
%function [F]=getSampleFD(GrayImg, sizeofFD)
% GrayImg: The gray Image, from which this function get Fourier
%          descriptor , which read by imread
% sizeofFD: Length of the Fourier descriptor
% F: The Fourier descriptor from GrayImg with length sizeofFD
%Get Fourier Descriptors from sample image

function [F]=getSampleFD(GrayImg, sizeofFD)
%Turn the gray image into balck/white image
bwimg=im2bw(GrayImg, 0.5);
%Get the edge of the shape from the image
lineimg=edge(bwimg, 'canny');
%Turn the 2D coordinates of edge into 1D imaginary number
```

```matlab
ii=sqrt(-1);
[fx fy] = find(lineimg > 0);
U = fx + fy*ii;
%Extracts the Fourier Descriptor of a contour
[F1]=extractFD(U);
%Normalize the Fourier Descriptor, which is invariant to translation,
%rotation and scaling of the original contour
[F2]=normaliseFD(F1);
%Resize the Fourier Descriptor into assigned length
F=resizeFD(F2, sizeofFD);
end




%function getTargetFDs(GrayImg, Sample1, Sample2, sizeofFD)
% GrayImg: The gray image, which contain many shapes, that will be
%          classified, which read by imread
% Sample1: one gray image of class 1, which read by imread
% Sample2: one gray image of class 2, which read by imread
% sizeofFD: Length of the Fourier descriptor
%Get Fourier Descriptors from Target image GrayImg, classify the
shapes
%into two classes and draw the comparison image.

function getTargetFDs(GrayImg, Sample1, Sample2, sizeofFD)
%Get Fourier descriptor from the two sample images
[SampleFD1]=getSampleFD(Sample1, sizeofFD);
[SampleFD2]=getSampleFD(Sample2, sizeofFD);
%Turn the gray image into balck/white image
bwimg=im2bw(GrayImg, 0.5);
bwimg=flipud(bwimg);
%Isolate the shapes by Morphological Operation
se = strel('diamond',7);
openedimg=imopen(~bwimg, se);
%Get the edges of the shapes from the gray image
lineimg=edge(openedimg, 'canny');
%Mark the edges of the shapes from the gray in different number
labelimg = bwlabel(lineimg);
```

```matlab
ii = sqrt(-1);
figure;
subplot(2,2,1);
imshow(Sample1);
subplot(2,2,3);
imshow(Sample2);
subplot(2,2,2);
imshow(GrayImg);
subplot(2,2,4);
hold on;
%Check each edges from the gray image
for i=1:max(max(labelimg))
    %Get the i-th edge
    [fy, fx] = find(labelimg == i);
    %Turn the 2D coordinates of edge into 1D imaginary number
    U = fx + fy*ii;
    if length(U) > sizeofFD
        %Extracts the Fourier descriptor
        [F1]=extractFD(U);
        %Normialize the Fourier descriptor
        [F2]=normaliseFD(F1);
        %Resize the Fourier descriptot
        F=resizeFD(F2, sizeofFD);
        %Caculate the differences
        diff1 = compareFD(F, SampleFD1);
        diff2 = compareFD(F, SampleFD2);
        %Classify i-th shape by the Differences and plot it
        if diff1 < diff2
            plot(U, 'r');
        else
            plot(U, 'g');
        end
    end
end
axis image;
hold off;
end
```

3. Zusammenfassenden Funktionen

```matlab
%function uebung1()
%This is just for runing the whole process of the task


function uebung1()
leaf1 = imread('leaf_type1.jpg');
leaf2 = imread('leaf_type2.jpg');
leaves = imread('leaves_example.jpg');
sizeofFD = 32;
getTargetFDs(leaves, leaf1, leaf2, sizeofFD);
end
```