Image processing for Bioinformatics

Laboratory
Feature Extraction

1 Examples, Matlab functions

1.1 Prewitt

Code	Image
%% Prewitt Img = imread('circuit.tif'); Img = imread('arch−window_bw.jpeg'); [m,n] = size(Img); % Img, 'Prewitt', threshold, 'horizontal' or 'vertical' [BW1, threshOut] = edge(Img,'Prewitt'); % Automatically BW2 = edge(Img,'Prewitt',0.3); BW3 = edge(Img,'Prewitt','horizontal'); BW4 = edge(Img,'Prewitt','vertical');	

Table 1: Prewitt

1.2 Sobel

Code	Image
%% Sobel Img = imread('circuit.tif'); Img = imread('arch−window_bw.jpeg'); [m,n] = size(Img); % Img, 'Sobel', threshold, 'horizontal' or 'vertical' [BW1, threshOut] = edge(Img,'Sobel'); % Automatically BW2 = edge(Img,'Sobel',0.3); BW3 = edge(Img,'Sobel','horizontal'); BW4 = edge(Img,'Sobel','vertical');	

Table 2: Sobel

1.3 Canny

Code

```
%% Canny
clc; clear; close all;
Img = imread('arch-window_bw.jpeg');
[m,n] = size(Img);
% Img, 'Canny', threshold of derivative [min max], sigma of Gaussian filter
[BW1, threshOut] = edge(Img,'Canny'); % Automatically
BW2 = edge(Img,'Canny',[0.4,0.5]);
BW3 = edge(Img,'Canny',[0.4,0.5],4);
```

Image

Table 3: Canny

1.4 Hough transform (lines)

Code

```
%% Hough Transform
% https://www.mathworks.com/help/images/hough-transform.html
Img = imread('circuit.tif');
Img = imread('blobs.png');
Img = imread('arch-window_bw.jpeg');
[m,n] = size(Img);

% Find the edges in the image using the edge function.
BW = edge(Img,'Canny');
% Compute the Hough transform of the binary image returned by edge.
[H,theta,rho] = hough(BW);
% Find the peaks in the Hough transform matrix, H, using the houghpeaks function.
P = houghpeaks(H,6,'threshold',ceil(0.3*max(H(:))));

% Find lines in the image using the houghlines function.
lines = houghlines(BW,theta,rho,P,'FillGap',20,'MinLength',10);
```

Image

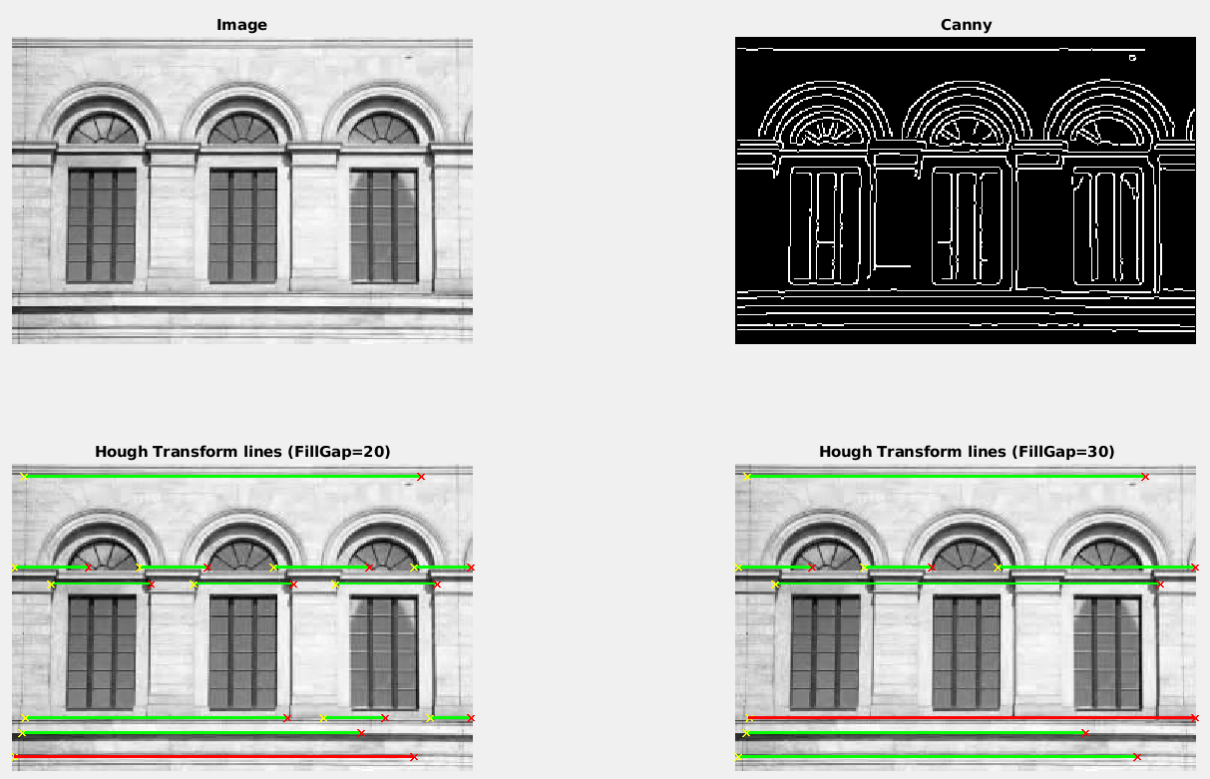

Table 4: Hough transform (lines)

2 Assignment

1. Find a template in an image.

Algorithm 1 Template matching

1: Set template = Img(a:b,c:d) ▷ A section of Img
2: Set avg = mean(Img)
3: Set ImgC = Img - avg
4: Set templateC = template - avg
5: Set matCorrelation = Correlation(ImgC, templateC);

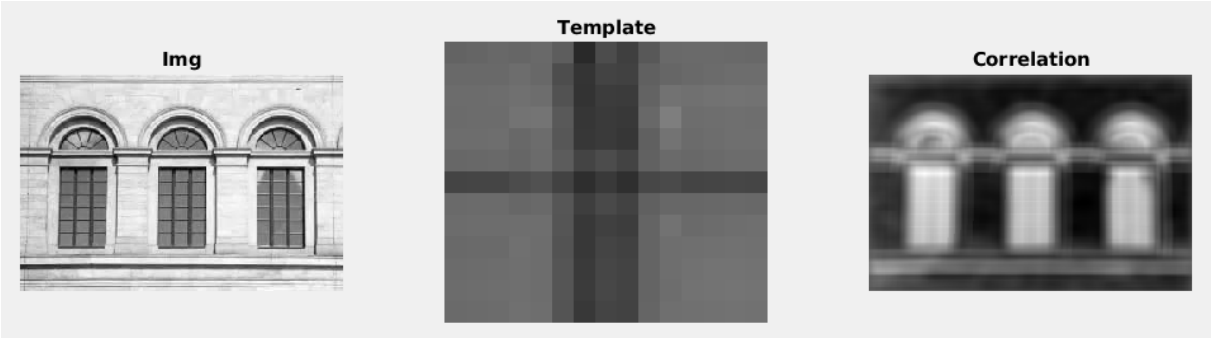

Figure 1: Example 1 of template matching

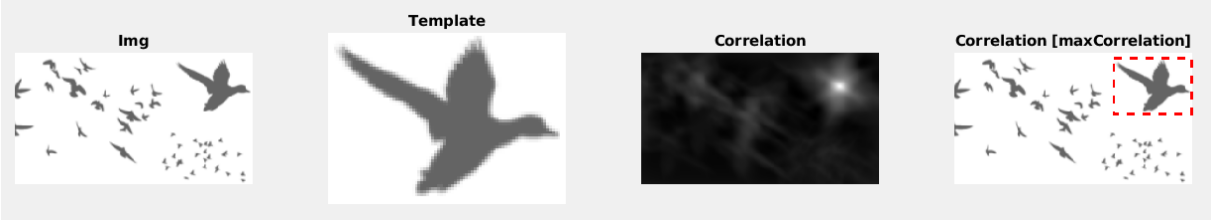

Figure 2: Example 2 of template matching

2. Implement Canny edge detection algorithm

Algorithm 2 Canny

1: Smooth input image with a Gaussian filter
2: Compute the gradient, magnitude and angle, of the image
3: Thin edges by applying non-max suppression to the gradient magnitude
4: Detect edges by using double thresholding

3 Solutions

1. Find a template in an image.

Algorithm 3 Template matching

1: Set template = Img(a:b,c:d) ▷ A section of Img
2: Set avg = mean(Img)
3: Set ImgC = Img - avg
4: Set templateC = template - avg
5: Set matCorrelation = Correlation(ImgC, templateC);

Template matching - Example 1

```
%% Template matching − Example 1
clc; clear; close all;
Img = im2double( imread('arch−window_bw.jpeg') );
template = Img(96:108, 46:60);

avgImg = mean(Img(:));

mat = Correlation_kernel(Img−avgImg, template−avgImg);
minMat = min(mat(:));
mat2 = (mat−minMat) / max(mat(:)−minMat);

nr = 1; nc = 3;
subplot(nr, nc, 1); imshow(Img); title('Img');
subplot(nr, nc, 2); imshow(template); title('Template');
subplot(nr, nc, 3); imshow(mat2); title('Correlation');
```

Template matching - Example 2

```
%% Template matching − Example 2
clc; clear; close all;
Img = im2double( rgb2gray( imread('birds.png') ) );
[m,n] = size(Img);
template = Img(7:67, 173:257);

avgImg = mean(Img(:));

mat = Correlation_kernel(Img−avgImg, template−avgImg);
minMat = min(mat(:));
mat2 = (mat−minMat) / max(mat(:)−minMat);

matMax = max(mat2(:));
[y,x]=find(matMax==mat2);
tmp = matMax−mat2;

nr = 1; nc = 4;
subplot(nr, nc, 1); imshow(Img); title('Img');
subplot(nr, nc, 2); imshow(template); title('Template');
subplot(nr, nc, 3); imshow(mat2); title('Correlation');
subplot(nr, nc, 4); imshow(Img); title('Correlation [maxCorrelation]');
hold on
rectangle('Position',[x−42,y−30,84,60],'LineWidth',2,'LineStyle','−−','EdgeColor','r')
hold off
```

Function Correlation_kernel

```
function mat = Correlation_kernel(Img, k)
  [m,n] = size(k);
  Rm = (m−1)/2;
  kCenterM = ceil(Rm+1);

  Rn = (n−1)/2;
  kCenterN = ceil(Rn+1);

  %Pad Img
  [m,n] = size(Img);
  ImgZ = zeros(m+2∗Rm,n+2∗Rn); ImgZ(1+Rm:m+Rm,1+Rn:n+Rn) = Img;

  [m,n] = size(ImgZ);
  matCorrelation = zeros(m,n);

  for u = 1+Rm:m−Rm
    for v = 1+Rn:n−Rn
      tmp = 0;
      tmp2 = 0;
      for i = −Rm:Rm
        for j = −Rn:Rn
          tmp = tmp + ImgZ(u+i,v+j) ∗ k(kCenterM + i,kCenterN + j);
          tmp2 = tmp2 + ImgZ(u−i,v−j) ∗ k(kCenterM + i,kCenterN + j);
        end
      end
      matCorrelation(u,v) = tmp;
    end
  end
  [m,n] = size(Img);
  mat = matCorrelation(1+Rm:m+Rm,1+Rn:n+Rn);
end
```

2. Implement Canny edge detection algorithm

Algorithm 4 Canny

1: Smooth input image with a Gaussian filter
2: Compute the gradient, magnitude and angle, of the image
3: Thin edges by applying non-max suppression to the gradient magnitude
4: Detect edges by using double thresholding

Function myCanny

```
function BW = myCanny(Img, t1,t2,sigma)
  [m,n] = size(Img);
  Img_blur = imgaussfilt(Img, sigma);

  [Gx, Gy] = imgradientxy(Img_blur);
  [Gmag, Gdir] = imgradient(Gx, Gy);
  gN = nonMaxSupp( Gmag,Gdir );
  gNL = gN>=t1;
  gNH = gN>=t2;

  BW = zeros(m,n);
  for i = 2:m-1
    for j = 2:n-1
      if gNH(i,j)
        BW(i,j)=1;
      elseif gNL(i,j)
        if sum(sum(gNH(i-1:i+1,j-1:j+1)))>1
          BW(i,j)=1;
        end
      end
    end
  end
end
```

Function nonMaxSupp

```
% https://stackoverflow.com/questions/31776434/non-maximal-suppressionthinning-algorithm
function [ marks ] = nonMaxSupp( mag,dir )
  [m,n]=size(mag);
  marks=mag;
  for i=2:m-1
    for j=2:n-1
      if mag(i,j) ~=0
        angle=dir(i,j)+180;
        if (angle>=340 || angle<=22.5) || (angle>=157.5 && angle<=202.5) %horizontal
          if (mag(i,j+1)>mag(i,j)) || (mag(i,j-1)>mag(i,j))
            marks(i,j)=0;
          end
        elseif (angle>22.5 && angle<=67.5) || (angle>202.5 && angle<=247.5) %45
          if (mag(i+1,j+1)>mag(i,j)) || (mag(i-1,j-1)>mag(i,j))
            marks(i,j)=0;
          end
        elseif (angle>67.5 && angle<=112.5) || (angle>247.5 && angle<=292.5) %vertical
          if (mag(i-1,j)>mag(i,j)) ||(mag(i+1,j)>mag(i,j))
            marks(i,j)=0;
          end
        else %135
          if (mag(i-1,j+1)>mag(i,j)) || (mag(i+1,j-1)>mag(i,j))
            marks(i,j)=0;
          end
        end
      end
    end
  end
end
```