Correction: Hough transform and line detection

1 Matlab correction

1.1 Contour detection

The first step is to perform contours detections. A classical method is employed here (see Fig.1, Canny edge detection). The important thing is to start by a binary image (binary set of points).

```
% Load an image
I   = double(imread('TestPR46.png'));
I = I(:,:,2); % keep grayscale image

%% performs contour detection
BW = edge(I,'canny');
```

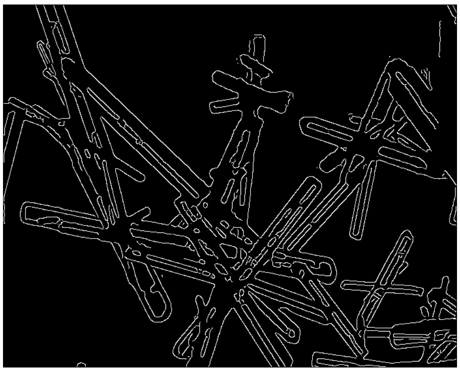

Figure 1: Canny edge detection.

1.2 Hough transform

This code does not make use of the MATLAB® function dedicated to line detection.The result is presented in Fig.2.

First, you can initialize the values. The size of the image is used to determine the maximal ρ value.

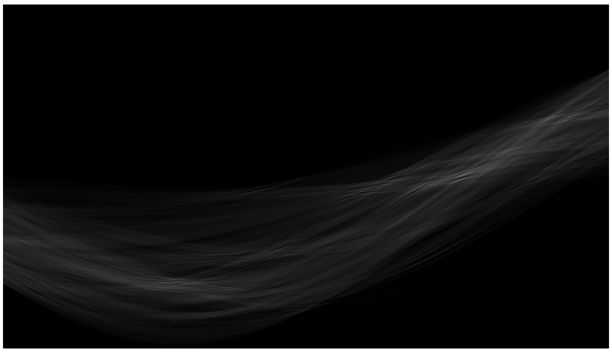

Figure 2: Hough transform.

```matlab
%% Hough transform
angular_sampling = 0.002; % angles in radians
[x, y] = size(BW);

rho_max = norm([x y]);
rho = -rho_max:1:rho_max;
theta = 0:angular_sampling:pi;
H = zeros(length(rho), length(theta));
```

Then, you loop over all the pixels (i, j): in case of a True pixel (BW(i,j)==1, you transform it into a sinusoid function, and increase the rounded values in the H matrix for all discrete values of θ.

```matlab
% performs Hough transform
for i = 1:x
    for j = 1:y
        if BW(i, j)
            for theta_index = 1:length(theta)
                th = theta(theta_index);
                r  = i * cos(th) + j * sin(th);
                rho_index = round(r + length(rho)/2);
                H(rho_index, theta_index) = H(rho_index, theta_index) +
                    ↪ 1;
            end
        end
    end
end
```

1.3 Maxima detection

1.3.1 Basic maxima detection

This version of maxima detection is very simple. However, it does not handle the neighborhood (it has the drawbacks of a basic threshold). One could look at h-maxima operators in order to get blobs instead of points. The threshold value can be tuned to find a given number of lines.

```
%% maxima detection
difference = 50;
M = max(H(:));
maxima = H>(M-difference);

% find the peaks
[indices_rho_peaks, indices_theta_peaks] = find(maxima);
```

1.3.2 Enhanced maxima detection

The MATLAB® version of the maxima detection gives cleaner maxima. Each peak, described by a coordinate ρ, θ, corresponds to a line in the original image.

```
peaks = houghpeaks(H, 5);
indices_rho_peaks   = peaks(:,1);
indices_theta_peaks = peaks(:,2);
```

The following code displays the results in the Hough space.

```
rho_peaks   = rho(indices_rho_peaks);
theta_peaks = theta(indices_theta_peaks);

imshow(H,[]), hold on
title('Hough Transform');
xlabel('\theta (radians)');
ylabel('\rho (pixels)');
plot(indices_theta_peaks, indices_rho_peaks, 'r*');
```

1.4 Lines retrieval

From the coordinates ρ, θ, it is easy to compute and display the different detected lines.

```
%% Find hough lines
x= 1:size(I, 2);
figure, imshow(I,[]), hold on
for i=1:length(rho_peaks)
    y = (rho_peaks(i) - x* cos(theta_peaks(i)) )/ sin(theta_peaks(i));
    plot(y, x);
end
title('detected lines')
```