Hough transform

In the last exercise, we detected edges in images. Today, we will analyze these edges whether they create a line (it can be used, for example, in road lines detection, see Fig. 1). Why we use only edges to detect the lines? Because the number of pixels to examine is significantly reduced comparing to all image pixels, and the shape is usually given only by its border, which corresponds to the edge points. For the detection of lines, we use Hough transform. The method is based on the pixel transformation from the image space to the parameter space (in the case of line detection, the parameters of line equation).



Figure 1: The input image with detected edges.

Line Detection

Let (x_i, y_i) be a 2D point in a space. The equation of line is defined as

$$y_i = ax_i + b. \tag{1}$$

How many lines go through the point (x_i, y_i) ? The answer is infinity, it only depends on choosing the parameters a and b such that the line equation is fulfilled. If we have a second point, then we have to find the values a and b such that the line connects both the points. We will try to find the lines by transforming the lines from the space xy to the parametric space ab. In this space, the line is represented only by one point. For example, the line y = 2x - 1 is represented by the point (2, -1) in the ab space. It is illustrated in Fig. 2, where you can see three lines in the xy space. These lines go through the point (2, 3) and they are represented by three points in the space ab (the points represent the lines of the corresponding color). Since there is an infinite number of lines going through (2,3), this infinite number of lines creates a line in the ab space - denoted as p in Fig. 2).



Figure 2: Three of infinite number of lines that are going through the point (2,3) in the space xy (left), and the representation of these lines in the space ab (right). Since there is an infinite number of lines going through one point in the xy space, these lines are represented also by a line (denoted by p) in the ab space.

Let (x_j, y_j) be the second point in the space xy. Again, an infinite number of lines is going through this point. How to find out whether one of the lines goes through the (x_i, y_i) ,

too? We transform the lines that go through (x_j, y_j) to the *ab* space. In this space, it is represented by the line denoted by q (Fig. 3 right). If there is an intersection between pand q, it means that there exists the line with the parameters a_p, b_p that goes through the points (x_i, y_i) and (x_j, y_j) . It is shown in Fig. 3 left, where we added the point (4, 4) in the xy space. The green and purple lines go through (4, 4), the blue line goes through through (4, 4) and also through (2, 3). In the *ab* space, the intersection of p and q is at the point a = 0.5, b = 2. It means that the line y = 0.5x + 2 goes through both the points. From this observation, it can be derived that if several points in the xy space lies on one line, it creates the intersection of the lines in the *ab* space.



Figure 3: An illustration of several lines going through the points (2,3) or (4,4). One of these lines goes through both the points. These lines are transformed to the *ab* space, the intesection in the *ab* space gives us the parameters of the equation of the line, in this case a = 0.5, b = 2.

The equation of the lines with the parameters a and b is usual, but, imagine points with a constant value of x, e.g. (2,2), (2,3), (2,5). It is clear that a vertical line goes through these points. It is impossible to define this line with Eq. (1), since the value of a is required to be ∞ . Therefore, in Hough transform, we use the equation

$$\rho = x\cos\theta + y\sin\theta. \tag{2}$$

The solution is the same - the points are transformed from the space xy to the space $\theta\rho$. In this space, the lines going through the point (x_i, y_i) do not create a line but in creates a sin function, see Fig. 4.



Figure 4: The points (x_i, y_i) and (x_j, y_j) in the xy space lie on the one line defined by the Eq. (2) with the parameters θ and ρ (left). The right image shows the $\theta\rho$ space, where the lines going through (x_i, y_i) and (x_j, y_j) are represented by sin function. The intersection of the sin functions (θ', ρ') gives the parameters θ and ρ of the equation of the line that goes through both the points.

If we have image pixels that lie on one line, we find this line by finding the intersection in the $\theta\rho$ space. How can it be realized in the discrete image space? We create a histogram (see https://en.wikipedia.org/wiki/Histogram). This histogram will be two-dimensional and it

will represent the space $\theta \rho$ - let us call such a histogram as accumulator. All the values in the accumulator are set to zero. For each image pixel, we compute the sin function. It means that we get the coordinaters (θ_i, ρ_j) - each pair represents one line in the xy space. Since the parameters can be continuous, it is required to get the discrete values. It can be done in the following way: let the pair (m, n) represents a rectangular area in the $\theta \rho$ space, such that $\theta_{m-1} \leq \theta_i < \theta_m$, $\rho_{n-1} \leq \rho_j < \rho_n$. This step allows us to set the cell in accumulator, where we increment the value by 1. When we process all the pixels, we find the maximum value in the accumulator. This maximum value is the intersection of the sin functions as it is seen in Fig. 5. From the maximum value, we get the values θ and ρ .



Figure 5: The points that are lying on one line create a high value in the accumulator - it is the intersection of the sin functions.

The last task to solve is to define the intervals $(\theta_{min}, \theta_{max})$ a (ρ_{min}, ρ_{max}) for which we create the accumulator. For θ , it is $-90^{\circ} \leq \theta \leq 90^{\circ}$ and $-D \leq \rho \leq D$, where D is the distance between opposite corners in image.

Then, for each image pixel (let us assume only pixels belonging to edges), we iterate the values of θ with a chosen interval step, and for each θ_i we compute ρ_j following the Eq. (2). The results are rounded to the integer number, which gives us the accumulator coordinates, and the corresponding accumulator value is incremented by 1.

Let us have the input image with several pixels that can be connected by a line (Fig. 6a). For each pixel, we generate a sin function in the $\theta \rho$ space (Fig. 6b). Since the pixels can be connected by a line, the *sin* functions have one intersection (Fig. 6c shows the accumulator - the brightest point is the maximum. This intersection is in the point $\rho = 0.0$, $\theta = -45^{\circ}$. Fig. 6d shows the drawn line defined by the equation $0 = x \cos(-45^{\circ}) + y \sin(-45^{\circ})$.



Figure 6: [a b c d] a) Input image containing 7 points that are lying on one line, b) transforming the points to the $\theta \rho$ space, each point is represented by sin function, c) accumulator, the light spot is the intersection of the sin functions = maximum value in the accumulator d) the line with the parameters θ and ρ gained from the accumulator.

In real images, in may happen that there is no line, or that there is more than one line. Therefore, it is not enough to find the maximum value in the accumulator, but it is better to find some significant values (higher than a chosen threshold).

TASK

1. Implement the Hough transform for line detection.

HINTS:

For each image point belonging to the edge (the white point), iterate the value of θ from -M_PI/2 to M_PI/2 with a chosen step (for example, 0.02) and compute the value of ρ using Eq. (2). The values of θ and ρ can be positive and also negative. Therefore, it is good to transform the negative values to positive. The values of the angle θ are from -90 to 90 degrees. If we have the angle in radians, we transform it to the interval 0-180 degrees. It is realized as int itheta = (int)(theta*180/M_PI)+90;. The value of ρ is from the interval $-D \leq \rho \leq D$, where D is the distance between the opposite image corners. For the image of size $M \times N$, $D = \sqrt{M^2 + N^2}$. The transformation to the interval from 0 to 2D can be computed as int irho = (int)rho + D;. Use these new values as the coordinates in the accumulator.

How to draw the output line? If we know θ and ρ , iterate the values from 0 to the image width (x=[0, img.cols]) and for each value of x compute $y = (\rho - x \cos(\theta))/(\sin(\theta))$. To reduce the time consumption, compute the values cos and sin only once before the loop, than use these precomputed values.