Standard Hough Transform

DIP – Project work 2023

Giulia Ortolani Tobias Baedeker

December 12, 2023

Contents

1 Introduction **2**

2 Theory of Hough transformation **2**

3 Example: Automatic reading of a needle from an image **3**
3.1 Preprocessing of the images 3
3.2 Detect lines of the needle 4
3.3 Converting to kN 5

4 Discussion of strengths and weaknesses **7**

1 Introduction

The Hough Transform is a mathematical technique used in computer vision and image processing to detect shapes, particularly lines and curves, within an image. Developed by Paul Hough in 1962, the Hough Transform provides a systematic and robust method for identifying patterns in noisy or cluttered images.
This paper focuses on illustrating the fundamental concept of the Hough Transform through a specific application: the detection of straight lines in an image.

2 Theory of Hough transformation

Hough transform is a method for straight-line detection in images. This transform is based on the equation

$$x\cos\theta + y\sin\theta = \rho, \tag{1}$$

where, in the Cartesian coordinate system (x, y), ρ is the shortest distance from the origin to the straight line, and θ is the angle between the x-axis and the perpendicular to the line. [1]

- If we fix (θ, ρ), we get the normal equation of a line, where $(\cos\theta, \sin\theta)$ is its normal vector and $|\rho|$ its distance from the origin.
- If we fix (x, y), we get a sinusoid in the (θ, ρ) plane.[2] Which makes sense when considering all the lines which could contain this specific point.

The (θ, ρ) plane is called the **Hough space**, and because of the way it is constructed, each point in this space corresponds to a possible line in the image.
In Digital Image Processing, we first need to find points in the image where there is a significant change in intensity, which often corresponds to the presence of edges or boundaries. For each of these edge points, we compute all possible lines that could pass through that point, in order to obtain a sinusoid in the Hough space. The idea is that if we have many points that stand on the same line in the Cartesian plane, this line corresponds to a point in the Hough plane, precisely it corresponds to the intersection of all the sinusoid generated by the points of the line. The `houghpeaks` function goes to identify the most frequent intersections of these sinusoids, that is the **peaks**. These peaks indicate the most likely lines in the image.

3 Example: Automatic reading of a needle from an image

In Figure 1 we have 16 images of a gauge of a brake dynanometer showing different values. We want to use Hough transform to extract the value of the gauge in each image.

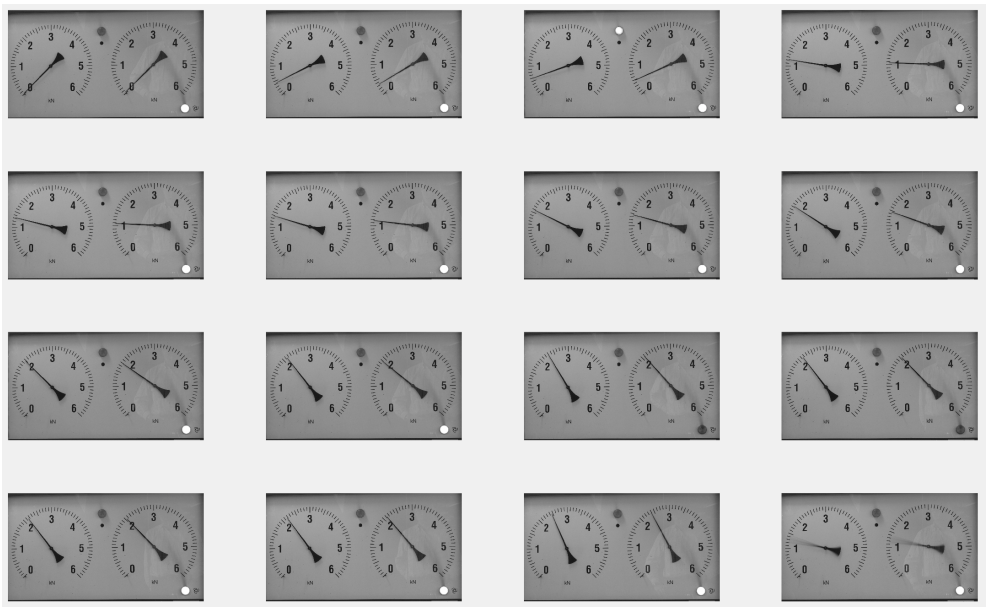

Figure 1: Initial pictures of the gauge

3.1 Preprocessing of the images

The preprocessing process consists of 5 steps. Each step prepares the image a bit more for the Hough Transformation afterwards. A code example for all the different steps can be seen in Figure 3.

1. **Crop the image**
   The first preprocessing step consists of cropping the image, so that the two gauges are now in separate pictures, but both are still fully visible.

2. **Thresholding**
   With a threshold, it is possible to get rid of the background above a certain pixel value. In order to do this, either we binarize our image with `imbinarize` or select some values under which we consider the pixels to be black. In this example, values of 70 and 90 were arbitrarily chosen to satisfy the conditions.

3. **Invert**
   Next, the picture was inverted. This can either be done by using the `imcomplement` function or by subtracting the pixel value from 255.

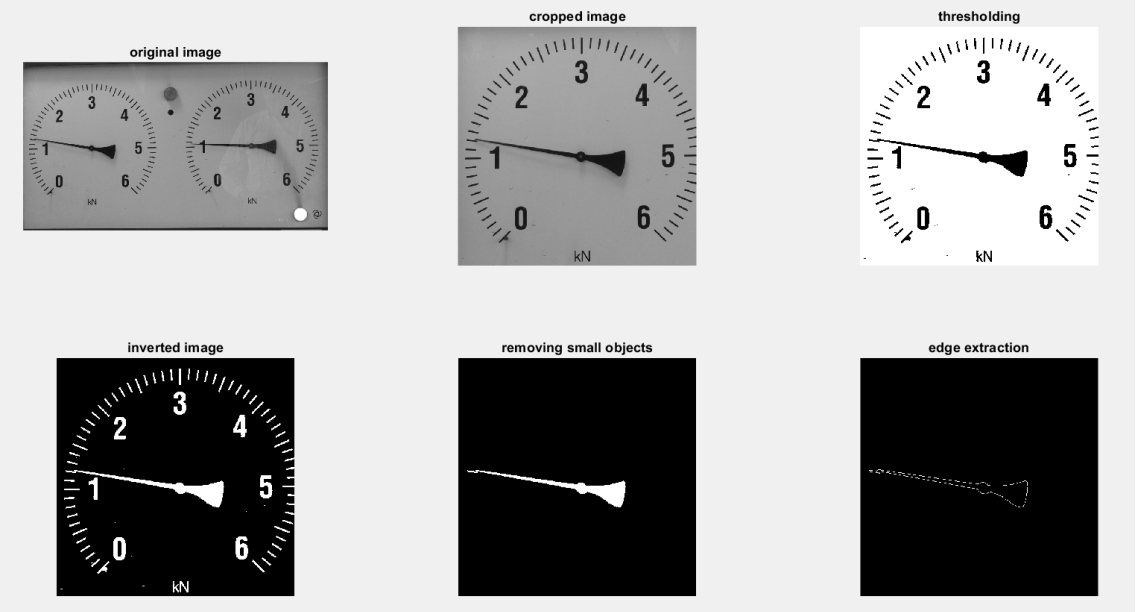

Figure 2: Pictures of Gauges

4. **Remove small Objects**
   To remove small objects in the image, such as the numbers and scale, the function `bwareaopen` was used with a threshold of 1200.

5. **Edge detection**
   The last step of the preprocessing is to only highlight the edges of the needle. Therefore, the `edge` function with the option `'approxcanny'` was used.

```
%% Preprocessing
Imcrop = imcrop(I,[10 70 550 550]); % crop
BW = imbinarize(Imcrop,0.45); % thresholding
invI = imcomplement(BW); % invert
kk = bwareaopen(invI, 1200); % remove small objects
k = edge(kk,'approxcanny'); % edge extraction
```

Figure 3: Preprocessing code

3.2 Detect lines of the needle

To now get the lines of the needles, i.e. the ρ and θ values in the Hough space, we need two Matlab commands. The first command `hough` creates three variables `H`, `R` and `T`.

- `H` is the Hough Transformation matrix of the image;
- `R` and `T` are vectors containing the values of the ρ and θ axis of the `H` matrix.

```
%% hough and houghpeaks
[H,T,R] = hough(k);
figure
imshow(imadjust(rescale(H)),'XData',T,'YData',R,'InitialMagnification','fit')
axis normal
hold on
P = houghpeaks(H,2,"Threshold",0.3*max(H(:)));
plot(T(P(:,2)),R(P(:,1)),'s','color','red');
xlabel('\theta'), ylabel('\rho');
axis on
```

Figure 4: Code that finds the edges of the indicator using Hough transform

This is the first line in the Matlab code in Figure 4.
The second command is `houghpeaks` and it is used to get the indices of the peaks in the `H` matrix. In `houghpeaks(H,2)`, `H` is the Hough Transformation matrix and the number 2 defines how many peaks the command is looking for. Since there is two lines in the needle. the upper and the lower border, it is set to two here. This is shown in the Matlab code in line 6 in Figure 4.
To obtain the actual values for the angle θ of the peak, the index has to be plugged into the `T` vector (line 7 in Figure 4).
At this point, two values were obtained for the angle of the upper and lower needle line. To obtain an accurate measurement, the average was taken to have the needle tip angle.

3.3 Converting to kN

In order to now get the value of the scale in kN, the angle α from the previous chapter calculated with the Hough transformation has to be converted. α reaches from $-90°$ to $90°$.
The angles are spread in such a way that the negative angles are in the second quadrant and the positive in the third quadrant. Now the angle β between the 0 line and the position of the needles in the pictures shall be calculated.

- For angles in the third quadrant, β is given by

$$\beta = \alpha - 43.5° \tag{2}$$

- For angles in the second quadrant, we get β from

$$\beta = (90° - 43.5°) + (90 + \alpha) \tag{3}$$

Note that, in Equation 3, α is always negative. A sketch of the conversion is shown in Figure 5. The final values of the force calculated for the left and right gauge are shown in Table 1.

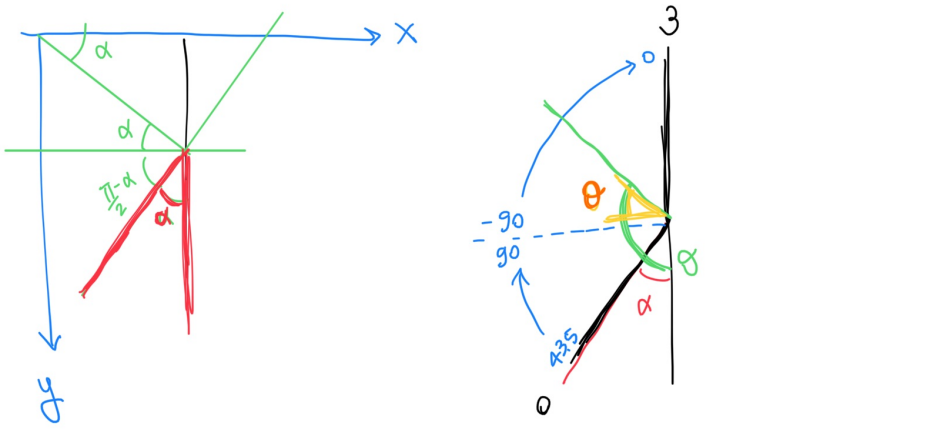

Figure 5: Conversion formula for $\alpha > 90°$

Left gauge in kN	Right gauge in kN
0.000	0.066
0.418	0.352
0.571	0.505
1.231	1.055
1.319	1.099
1.385	1.187
1.604	1.341
1.736	1.473
2.000	1.780
2.154	1.890
2.308	2.066
2.198	2.022
2.198	2.022
2.220	2.132
2.484	2.396
1.330	1.110

Table 1: Rounded values to two decimal places.

4 Discussion of strengths and weaknesses

As every technique the Hough Transformation has its strength and weaknesses for a purpose like this. First of all anyone can see by comparing the values from Table 1 to the initial pictures in Figure 1 that the calculated values are quite precise. A special attendance should be given to the last picture where the needle was quite blurry but still the value of the force was calculated correctly.
In the initial pictures is it noticeable that all the needles show numbers less than 3 kN which is quite handy for the code. This makes the conversion from the angle to the force easier since there is a bijective dependency between the angle and the force. This however changes when the forces are higher than 3 kN. In that case there are 2 forces corresponding to the same angle α which is an additional difficulty that has to be taken into consideration.
The Hough Transform method stands out for its robustness in handling noise and accommodating missing parts of objects within an image. Notably, it excels by eliminating the need for an initial guess about the parameters defining the shape being detected.
On the flip side, the computational intensity of the Hough Transform becomes apparent, particularly when dealing with larger images: the complexity of the algorithm is $\mathcal{O}(A^{m-2})$, where A denotes the size of the image space, and m is the number of parameters involved. [1]

References

[1] "Hough transform". In: (). URL: https://en.wikipedia.org/wiki/Hough_transform.

[2] S. El Mejdani, R. Eglia, and F. Dubeaua. "Old and new straight-line detectors: Description and comparison". In: *Pattern Recognition* (2008).