Line Detection by Hough transformation

09gr820

April 20, 2009

1 Introduction

When images are to be used in different areas of image analysis such as object recognition, it is important to reduce the amount of data in the image while preserving the important, characteristic, structural information. Edge detection makes it possible to reduce the amount of data in an image considerably. However the output from an edge detector is still a image described by it's pixels. If lines, ellipses and so forth could be defined by their characteristic equations, the amount of data would be reduced even more. The Hough transform was originally developed to recognize lines [5], and has later been generalized to cover arbitrary shapes [3] [1]. This worksheet explains how the Hough transform is able to detect (imperfect) straight lines.

2 The Hough Space

2.1 Representation of Lines in the Hough Space

Lines can be represented uniquely by two parameters. Often the form in Equation 1 is used with parameters a and b.

$$y = a \cdot x + b \tag{1}$$

This form is, however, not able to represent vertical lines. Therefore, the Hough transform uses the form in Equation 2, which can be rewritten to Equation 3 to be similar to Equation 1. The parameters θ and r is the angle of the line and the distance from the line to the origin respectively.

$$r = x \cdot \cos\theta + y \cdot \sin\theta \Leftrightarrow \tag{2}$$

$$y = -\frac{\cos\theta}{\sin\theta} \cdot x + \frac{r}{\sin\theta} \tag{3}$$

All lines can be represented in this form when $\theta \in [0, 180[$ and $r \in \mathbf{R}$ (or $\theta \in [0, 360[$ and $r \geq 0$). The Hough space for lines has therefore these two dimensions; θ and r, and a line is represented by a single point, corresponding to a unique set of parameters (θ_0, r_0). The line-to-point mapping is illustrated in Figure 1.

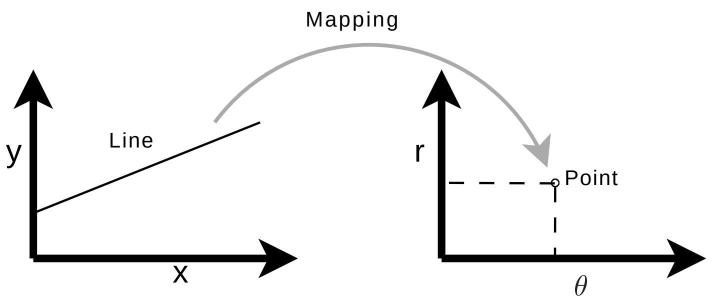
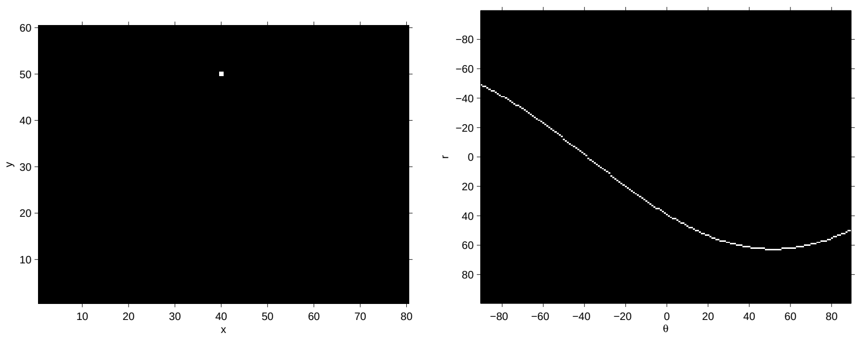


Figure 1: *Mapping of one unique line to the Hough space.*

2.2 Mapping of Points to Hough Space

An important concept for the Hough transform is the mapping of single points. The idea is, that a point is mapped to all lines, that can pass through that point. This yields a sine-like line in the Hough space. The principle is illustrated for a point $p_0 = (40, 30)$ in Figure 2.



(a) Point p_0.

(b) All possible lines through p_0 represented in the Hough space.

Figure 2: *Transformation of a single point (p_0) to a line in the Hough space. The Hough space line represents all possible lines through p_0.*

3 Algorithm

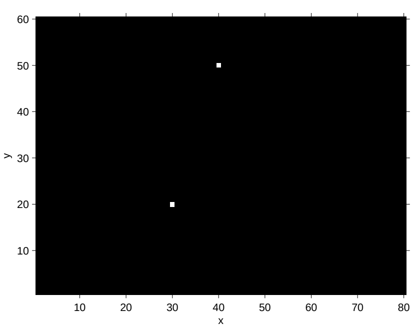
The algorithm for detecting straight lines can be divided into the following steps:

1. Edge detection, e.g. using the Canny edge detector [2].
2. Mapping of edge points to the Hough space and storage in an accumulator.
3. Interpretation of the accumulator to yield lines of infinite length. The interpretation is done by thresholding and possibly other constraints.
4. Conversion of infinite lines to finite lines.

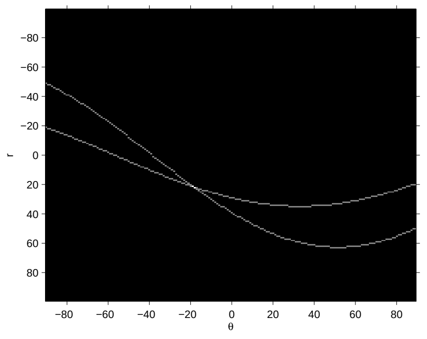
The finite lines can then be superimposed back on the original image. The Hough transform itself is performed in point 2, but all steps except edge detection is covered in this worksheet.

3.1 Transformation to Hough Space

The Hough transform takes a binary edge map as input and attempts to locate edges placed as straight lines. The idea of the Hough transform is, that every edge point in the edge map is transformed to all possible lines that could pass through that point. Figure 2 illustrates this for a single point, and Figure 3 illustrates this for two points.



(a) Points p_0 and p_1.



(b) All possible lines through p_0 and/or p_1 represented in the Hough space.

Figure 3: *Transformation of two points (p_0 and p_1) to two lines in the Hough space. The intersection of the Hough space lines indicates the line that pass through both p_0 and p_1.*

A typical edge map includes many points, but the principle for line detection is the same as illustrated in Figure 3 for two points. Each edge point is transformed to a line in the Hough space, and the areas where most Hough space lines intersect is interpreted as true lines in the edge map.

3.1.1 The Hough Space Accumulator

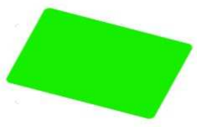
To determine the areas where most Hough space lines intersect, an accumulator covering the Hough space is used. When an edge point is transformed, bins in the accumulator is incremented for all lines that could pass through that point. The resolution of the accumulator determines the precision with which lines can be detected. In this worksheet a resolution of 1 pixel for r and 1 degree for θ has been used.

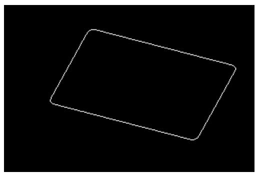
In general, the number of dimensions of the accumulator corresponds to the number of unknown parameters in the Hough transform problem. Thus, for ellipses a 5-dimensional space is required (the coordinates of its center, the length of its major and minor axis, and its angle). For lines 2 dimensions suffice (r and θ). This is why it is possible to visualize the content of the accumulator.

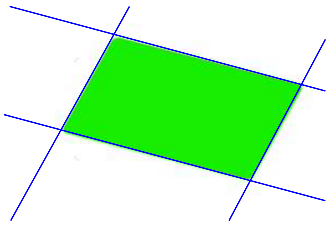
3.2 Detection of Infinite Lines

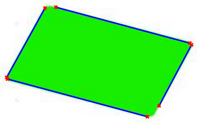
Infinite lines are detected by interpretation of the accumulator when all edge points has been transformed. An example of the entire line detection process is shown in Figure 4.

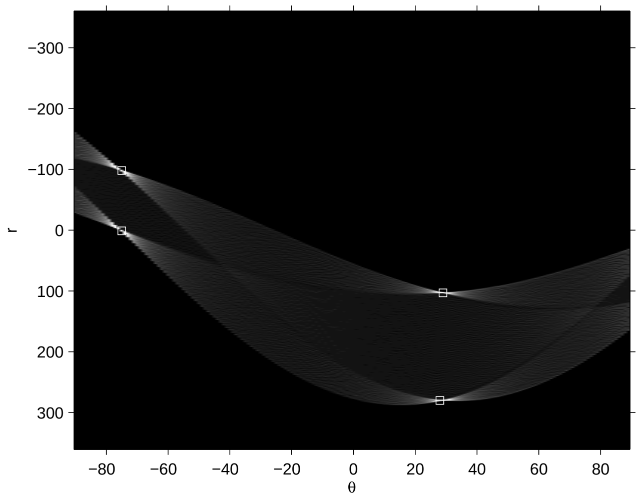
The most basic way the detect lines is to set some threshold for the accumulator, and interpret all values above the threshold as a line. The threshold could for instance be 50% of the largest


(a) Original


(b) Edges


(c) Infinite Lines


(d) Finite Lines



(e) Hough Transform

Figure 4: *Line detection using the Hough transformation. The lines detected in the source image (Figure 4a) are marked with white boxes in the Hough transform (Figure* **??**.

value in the accumulator. This approach may occasionally suffice, but for many cases additional constraints must be applied. As it is obvious from Figure 4e, several entrances in the accumulator around one true line in the edge map will have large values. Therefore a simple threshold has a tendency to detect several (almost identical) lines for each true line. To avoid this, a *suppression neighborhood* can be defined, so that two lines must be significantly different before both are detected.

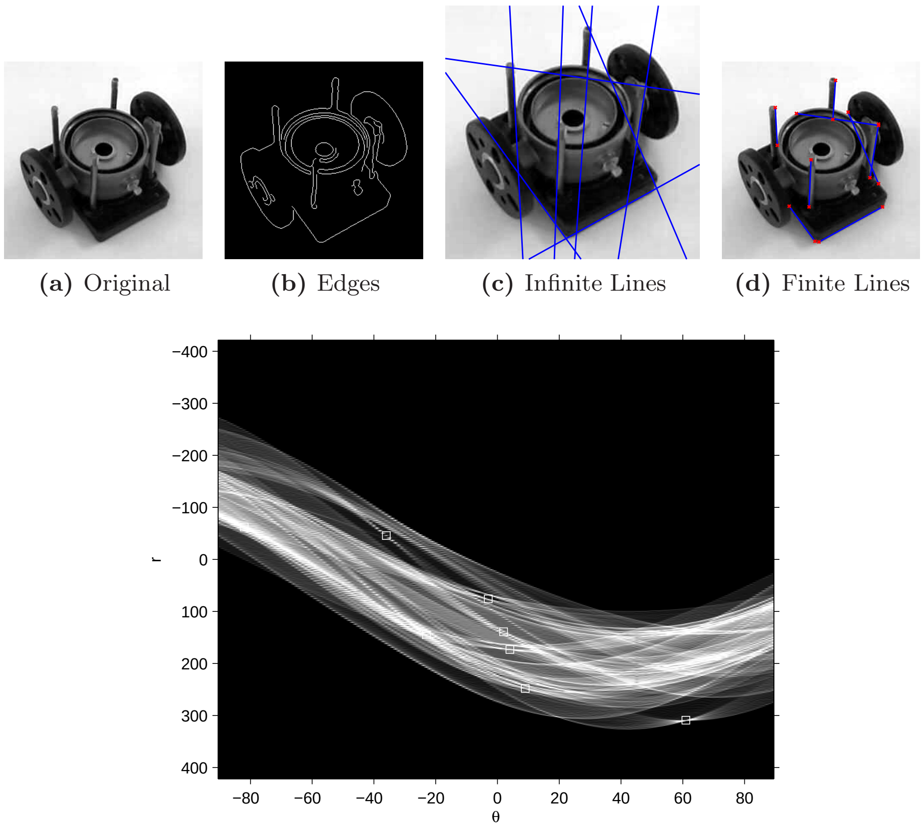
3.3 Finite Lines

The classical Hough transform detects lines given only by the parameters r and θ and no information with regards to length. Thus, all detected lines are infinite in length. If finite lines are desired, some additional analysis must be performed to determine which areas of the image that contributes to each line. Several algorithms for doing this exist. One way is to store coordinate information for all points in the accumulator, and use this information to limit the lines. However, this would cause the accumulator to use much more memory. Another way is to search along the infinit lines in the edge image to find finit lines. A variant of this approach known as the *Progressive Probabilistic Hough Transform* is discussed in Section 6.

4 Evaluation on Real Images

Figure 5 shows the Hough transform applied to a partly assembled pump from Grundfos. In Figure 5d 6 true straight lines has been detected, while two detected lines does not correspond to true straight lines. The algorithm has been "fooled" by the many ellipses in the edge map.

It is not immediately possible to avoid false detections while preserving most true detections through tuning of the algorithm.



(a) Original **(b)** Edges **(c)** Infinite Lines **(d)** Finite Lines

(e) Hough Transform

Figure 5: *Line detection on a real image using the Hough transformation.*

5 Matlab "Implementation"

Matlab has been used to generate the images used in this worksheet. The following functions has been used:

- **hough:** Performs the Hough transform on a binary edge image, and returns the accumulator. The resolution of the accumulator used in this worksheet is 1 for both r and θ.
- **houghpeaks:** Detects lines by interpreting the accumulator. In this worksheet the threshold was set to 20% of the maximum value in the accumulator, and the suppression neighbourhood was set to approximately 5% of the resolution of r and θ respectively.

- **houghlines:** Converts infinite lines to finite lines. In this worksheet, the minimum length of a line was set to 30 pixels, and the algorithm was allowed to connect lines through holes of up to 30 pixels.

6 Progressive Probabilistic Hough Transform

The Hough transform is not a fast algorithm for finding infinite lines in images of a certain size. Since additional analysis is required to detect finite lines, this is even slower. A way to speed up the Hough Transform and finding finite lines at the same time is the Progressive Probabilistic Hough Transform (PPHT) [4]. The idea of this methood is to transform randomly selected pixels in the edge image into the accumulator. When a bin in the accumulator corresponding to a particular infinite line has got a certain number of votes, the edge image is searched along that line to see if one or more finite line(s) are present. Then all pixels on that line are removed from the edge image. In this way the algorithm returns finite lines. If the vote threshold is low the number of pixels to evaluate in the accumulator gets small. The algorithm can be outlined as follows [4]:

1. Create a copy (**IMG2**) of the input edge image (**IMG1**).
2. If **IMG2** is empty then finish.
3. Update the accumulator with a randomly selected pixel from **IMG2**.
4. Remove the pixel from **IMG2**.
5. If the bin with the largest value in the accumulator (**BINX**) that was modified is lower than the threshold, goto point 1.
6. Search in **IMG1** along a corridor specified by **BINX**, and find the longest segment of pixels either continuous or exhibiting gaps not exceeding a given threshold.
7. Remove the pixels in the segment from **IMG2**.
8. Clear **BINX**.
9. If the detected line segment is longer than a given minimum length, add it into the output list.
10. Goto point 2

An issue with this algorithm is, that severel runs may may yield different results. This can be the case if many lines share pixels. If two lines cross, the fist line to be detected removes the common pixel (and a band around it) resulting in a gab in the other line. If many lines cross, then many pixels can miss in the last lines, and the votes in the accumulator may not reach the threshold.

In this project only a few important lines are to be detected, and the PPHT is preferred due to its lower computational cost. We use the implementation available in openCV.

References

[1] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. pages 714–725, 1987.

[2] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov. 1986.

[3] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[4] C. Galambos, J. Kittler, and J. Matas. Progressive probabilistic hough transform for line detection. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1554, 1999.

[5] P.V.C. Hough. Method and means for recognizing complex patterns, u.s. patent 3069654. 1962.