Get It Straight

Image Processing Basics: Hough-transform

The Hough-transform is best known as a method "to find straight lines" in an image. Unfortunately, people who encounter the concept for the first time usually are somewhat irritated by the mathematical formalities necessary to get it straight. The basic idea of the Hough-transform, however, is simple. The simplicity, the elegance and the potential of the method immediately becomes clear when looking at the Hough-transform for circles rather than for straight lines. Once you have caught the wave, you will easily follow the arguments leading to the detection of other analytic curves, including, of course, straight lines – and you will immediately understand the main advantages of Hough-transform, namely, to detect partly occluded shapes and incomplete curves.

The Basic Idea

Imagine a simple binary image like in the left part of figure 1, containing a single, complete circle of black pixels on a white background. Let us first try to find the center of this circle, and let us assume that the radius of the circle is already known. A circle is the set of points which have the same distance to a common point, the distance being the radius and the common point being the center of the circle. To find the center of the circle, you may proceed as follows: for every pixel of the image, beginning at the upper left corner and scanning the image along its lines and columns like with a filter-operation, draw a test-circle with the desired radius around the pixel. Then check the amount of overlap between the test-circle and the "real" circle in the image: walk along the curve of the test-circle, check for black pixels on this circle, simply count the number of these black pixels and store this number in a new image at the current center of the test-circle. For many positions of the test-circle there will be no overlap at all, and the resulting number will be zero; some positions will yield an overlap in one or two pixels, resulting in the numbers one and two; but once you hit the center of the real circle in the image, the result will be the greatest possible number which can be achieved by this procedure. The numbers stored at the pixels of the resulting image may be interpreted as grey-levels, and the center of the circle can easily be identified as a bright spot on a dark background.

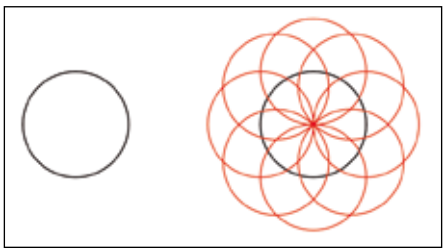
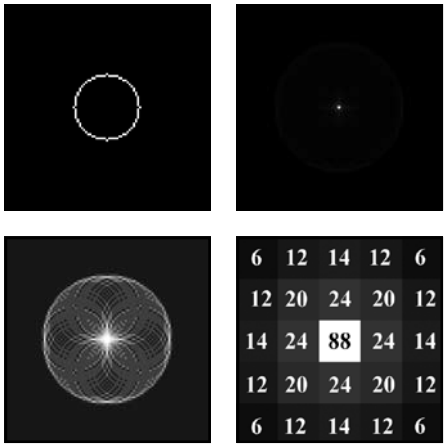


Fig. 1: A circle and some circles with the same radius drawn around points on the boundary

This approach yields good results, but it is not very effective. In many applications, large areas of the background will be examined, although the black pixels of the real circle are far away. Obviously, it would be much better to somehow use the foreground-pixels for guidance instead of scanning the whole image. But how? Now it is time for a great leap: let us put the method upside-

down (or rather inside-out). Instead of drawing a circle around every pixel, foreground or not, and checking for black pixels on the curve, let us draw circles with the desired radius around the black foreground-pixels only and take a look at the resulting image, containing the real circle and all the circles drawn around the pixels on its curve. Some of these latter circles are shown in the right part of figure 1, highlighted by the red colour. All these circles have one common point: the center of the real circle, where they all coincide, and the number of coincidences is equal to the number of pixels on the curve of the real circle. Thus, the number of coincidences is the same number you will get when drawing a circle with the desired radius around the center of the real circle and counting the number of black pixels on the curve of this test-circle, like in the procedure explained before – but much more effective! Think about it for a few seconds, and you will see that the two methods are equivalent, they will yield the same result. Counting the coincidences is very simple: allocate a new image and initialize all grey-levels to zero; go to the position of a black pixel in the original image, draw a circle with the desired radius around this position in the new image, and add 1 to the grey-level of every pixel you touch by drawing this circle in the (new) resulting image; do so for every black pixel in the original image. By this procedure, you will accumulate the coincidences in every pixel, the number of coincidences being represented by the grey-level in the new image, the so-called accumulator-array. Finally look for the brightest spot in the accumulator-array: this is the center of the circle. If the radius chosen is wrong, there will be some coincidences as well, but not as many as with the proper radius. Thus, in the accumulator-array, the centers of the real circles in the image will pop-up as spikes on a noise-floor. This is the basic principle of the Hough-transform.

Now let us skip the assumption that the radius of the circle is already known. In this situation it takes a bit longer to find the position of the circle (and its radius), but just guess and try! Choose a reasonable interval for the radii of the test-circle and proceed as before: for every radius chosen, scan the image for overlap between the test-circle and the real circle. The result will be not a single image, but one image for every radius of the test-circle, a whole stack of images. But, as with the situation where the radius is already known, there is exactly one image out of the stack with the brightest spot (that is, the highest number of overlap) of all spots of all images in the whole stack, the center of the real circle corresponding to the position of the brightest spot and the radius of the real circle corresponding to the radius of the test-circle used for this particular resulting image. Even if you just add up the whole stack of resulting images to a single image, simply adding the grey-levels of corresponding pixels in the stack of resulting images to a single accumulator-image, there will usually remain a



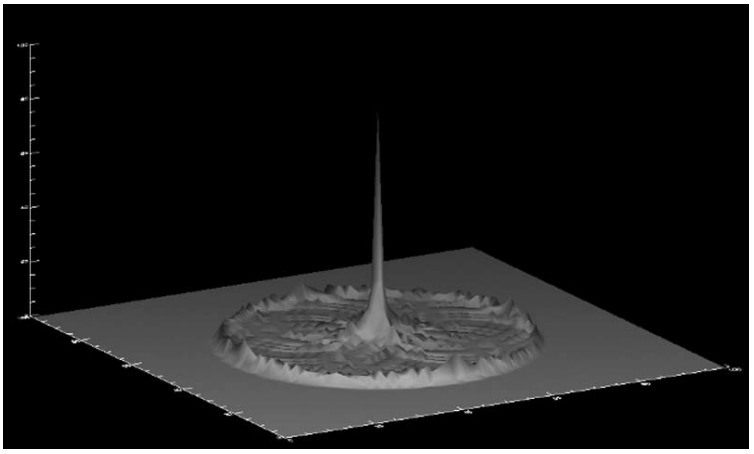


Fig. 2: Hough-transform for a circle a) binary circle in the discrete image plane; b) accumulator-image resulting from Hough-transform; c) accumulator-image, contrast-enhanced; d) central area of the accumulator with grey-levels as insets; e) 3D-plot of the grey-levels of the accumulator

bright spot at the center of the real circle.

A Simple Example

Figure 2 shows the procedure with a synthetic circular structure in a binary image. The pixel-structure due to the discrete image-plain is clearly visible. The corresponding accumulator-image is quite similar to the sketch on the right side of figure 1, showing a prominent spot at the position of the center of the circle. Further structure can be found in the accumulator-image, namely a second circle with a greater radius and some ripple in the background. Figure 2 also shows a contrast-enhanced version of the accumulator-image to amplify these structures. It is easily seen from figure 1 how these features arise. To get a better idea of the numbers, the innermost 5 x 5 pixels of the accumulator are shown in an enlarged view, with their grey-levels as inserts. Finally, a 3D-plot of the grey-levels over the coordinates of the accumulator gives an intuitive view of the signal-to-noise-ratio.

A similar situation may appear in real-world-images like in figure 3. To take advantage of the Hough-transform, it is a good idea to isolate the boundary of objects, yielding best results with boundaries thinned to a width of a single pixel. This has been done in the second image in figure 3. The Hough-transform is then applied to this pre-processed image, resulting in a bright spot in the accumulator at the center of the wheel.

Advantages

It is immediately clear that this method may consume a lot of computing power, but it has some very important advantages. It works not only for a single complete circle, but also for circles where parts of the curve are missing, either as a whole sector or in the form of several small gaps in the curve like in a punctuated circle. The overlap-number, however, will become lower than with full circles, and the discrimination between the brightest spot in the accumulator-image and the background will become less robust and eventually break down once the number of pixels defining the circle becomes too small. Another advantage: it is not necessary that the circles under examination are separated from each other like in blob-analysis; two or more circles may be overlapping and will nevertheless usually be detected.

Figure 4 shows an example for overlapping circles, which already appear as well-prepared boundaries in the original image. Again, a second, post-processed version of the Hough-image is shown to enhance the additional structure in the accumulator. The example in figure 5 shows several coins, most of these objects being clearly separated from each other, but two coins are just touching each other and three others are overlapping, which appear as connected and partly occluded circular objects in the image. Isolating the boundaries and performing the Hough-transform yields the centers of all theses circles as clearly visible bright spots in the accumulator.

There are numerous other examples to show the potential of the method. There may arise some disturbing features, however, in uncontrolled scenes, such as in a search for traffic-signs in a street, and computing the Hough-transform for certain images may well take several seconds even on a state-of-the-art personal computer.

Hough-Transform for Straight Lines

While the basic idea of the Hough-transform for circles and the construction of the accumulator can be intuitively understood and even be implemented straight-forward, the Hough-transform for straight lines needs some mathematical considerations. As with the search for circles, it is not necessary to have complete straight lines in the image. The Hough-transform rather looks

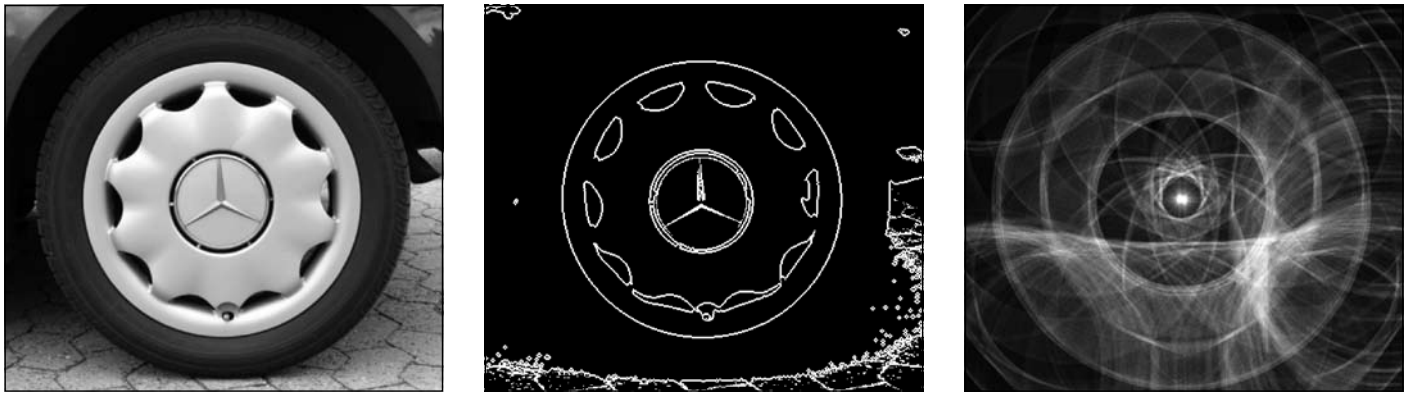


Fig. 3: a) original image; b) boundaries have been isolated and thinned; c) contrast-enhanced version of the accumulator after Hough-transform applied to the boundaries

for the set of pixels in an image which are collinear, that is which are located on a common straight line which may be drawn through these pixels. As with the search for circles, we might draw all possible straight lines through all pixels in the image and count the black foreground-pixels on these lines, thereby getting a figure of merit for discrimination of the desired line. Obviously, this is quite a task. Again, a much better starting-point are the black pixels in the foreground. First imagine that something is already known about the straight line we are looking for, its slope, for instance. Then we might draw a straight line with this slope through every black pixel in the foreground, getting a number of parallel lines. When two or more pixels are collinear, their corresponding straight lines will coincide, the number of coincidences being identical to the number of pixels on this particular line. Unfortunately, with a straight-forward-approach it seems to be quite complicated to check for coincidences between whole lines in an image. But when we look at the well-known representation of a straight line in the xy-plane in the slope-intercept-form, y=mx+b, it becomes clear that out of all the parallel lines with the same slope m those straight lines are identical which have the same intercept b. To check for coincidences between two lines with identical slope m, we thus simply have to calculate their intercepts b_1 and b_2 and compare. For a given point (x,y) on a straight line with slope m, the intercept will be b=y-mx. We may thus proceed as follows: allocate an accumulator-vector with one element for every possible value of b and initialize to zero; go to the position of a black pixel in the original image, draw a straight line with the desired slope m through this pixel and calculate the intercept b; add 1 to the value of the corresponding position in the accumulator; do so for every black pixel in the original image. By this procedure, we will accumulate the coincidences between straight lines through every pixel, the number of coincidences being represented by the values in the accumulator. Finally look for the brightest spot in the accumulator: this is the intercept for the common straight line.

When neither intercept nor slope is known, we can generalize the approach. For every black pixel, there is a whole bundle of straight lines with different slopes and different intercepts which can be drawn through this pixel. Slope and intercept, however, are not arbitrary; there is the relation b=y-mx between b and m for all the straight lines which contain a single pixel (x,y). If we plot the possible values for b and m for a given, fixed pixel (x_1,y_1) in a diagram spanned by m and b, the mb-plane, we will get a straight line, characterized by the equation $b=-x_1m+y_1$, $-x_1$ being the slope and y_1 being the intercept of the line in mb-space. A single pixel (x_1,y_1) is thus transformed to a whole straight line in mb-space. A second single pixel (x_2,y_2) will be represented by a different straight line in mb-space. These two lines in mb-space will intersect at a single point with common values for m and b, thus completely characterizing the straight line in xy-space connecting the two pixels (x_1,y_1) and (x_2,y_2). Obviously,

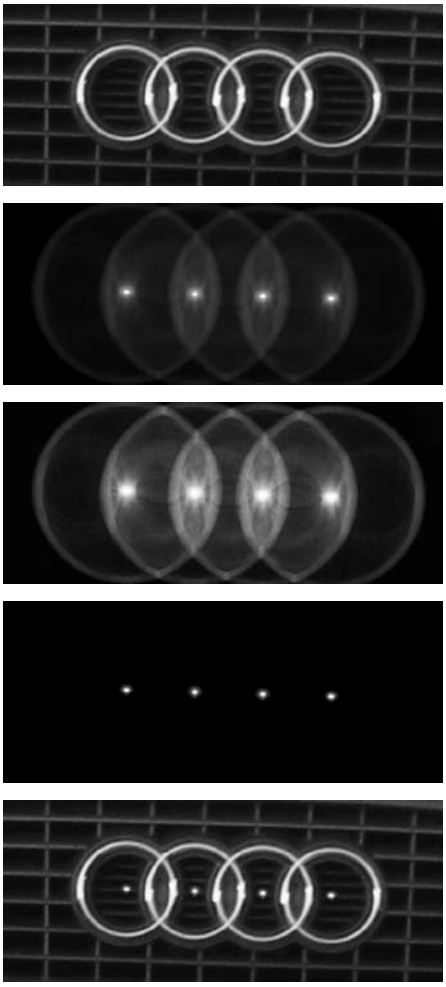


Fig. 4: Hough-transform for overlapping rings a) original image; b) and c) accumulator-images after Hough-transform, raw-image and contrast-enhanced; d) accumulator after thresholding; e) overlay of a) and d) showing the centers of the rings as bright spots

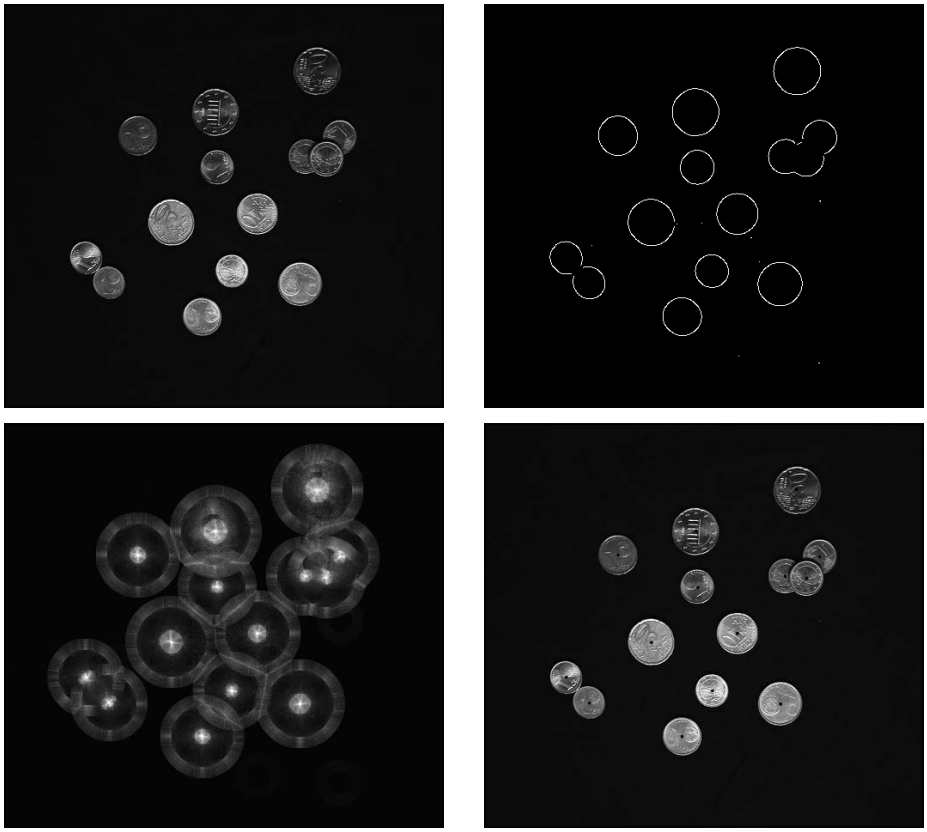


Fig. 5: Hough-transform for partly occluded coins
a) original image; b) boundaries have been isolated and thinned; c) accumulator-image after Hough-transform, contrast-enhanced; d) the original image, showing the result of thresholding of the accumulator as black spots in the centers of the coins

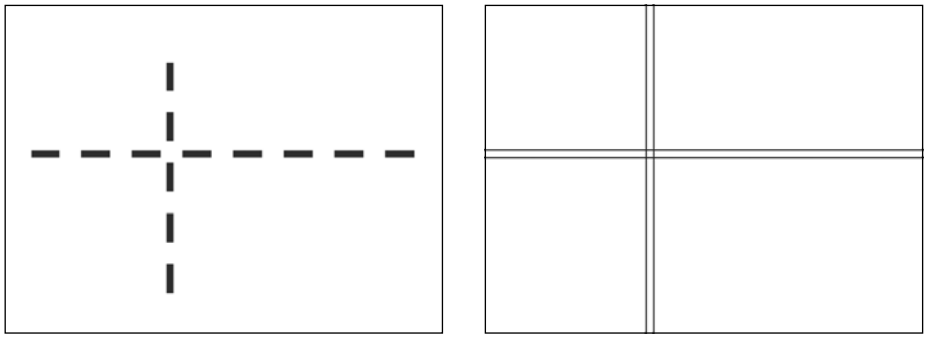


Fig. 6: Hough-transform for straight lines
a) original image; b) straight lines detected with Hough-transform, applied to boundaries

mb-space is an accumulator-array which allows to simply add-up all coincidences between straight lines which appear when drawing the bundles of possible straight lines through all the foreground-pixels in the image. For every black pixel (x,y) in the image, we simply have to draw the corresponding straight line in mb-space; whenever we hit a certain combination (m,b), we add 1 to the corresponding pixel in the accumulator-image. Finally, the brightest spot in the accumulator-image represents the slope m and the intercept b of the desired straight line, and we may go back to the original image and draw this line.

In practice, the Hough-transform for straight lines is not implemented with the parameters slope and intercept, but with the orientation and the distance from the origin according to the normal or Hesse-representation. In this parameter space, a point transforms to a sinusoid rather than to a straight line. Collinearity, however, is again detected by the intersection of these curves in parameter space. The implementation of the Hough-transform for straight lines is not simple, and it is a good idea to consult some standard textbooks [1] [2] to tackle the various further aspects which are far beyond the intentions of this introductory article. Figure 6 shows a simple example for the detection of straight lines in a binary image, omitting the various steps of analyzing the accumulator and just giving the resulting lines in the xy-plane.

Further generalization

The basic idea of the Hough-transform is the construction of an accumulator in a parameter-space. Straight lines are characterized by two parameters, resulting in a two-dimensional parameter space. Other geometric forms such as elliptical curves may be described analytically by more than two parameters. The accumulator-space will thus become more complex and the computing-power necessary to reasonably implement such a method may be beyond current technology, but the concept remains the same. Hough proposed his approach in 1962 [1], probably not even dreaming of the performance of current processors. About 20 years later, in 1981, Ballard [3] published a method for Hough-transform of arbitrary non-analytic shapes. Nowadays, we can get tremendous computational power at reasonable cost, and the Hough-transform for straight lines and circles may well be performed at video-frame-rates in certain applications. The literature is full of such clever, sophisticated algorithms from the 60s, 70s and 80s of the last century, waiting for us to be re-discovered and longing to be implemented on our powerful processing systems.

Acknowledgements

Thanks to diploma-student Sven Schneider for preparation of the images for the Hough-transform for circles, and thanks to my colleague Stephan Neser for pointing out the relevance of reference [3].

References

[1] R. C. Gonzales, R. E. Woods, Digital Image Processing, Addison-Wesley, 1993, p. 432–438
[2] W. Burger, M. J. Burge, Digitale Bildverarbeitung, Springer, 2005, S.152–167
[3] D. H. Ballard, Generalizing the Hough-transform to detect arbitrary shapes, Pattern Recognition Vol. 13, No. 2, p 111–122, 1981

▶ Author
Prof. Dr. Christoph Heckenkamp

Darmstadt University of Applied Sciences, Optical Technology and Machine Vision/Germany
heckenkamp@h-da.de
www.fbmn.h-da.de