Feature Detection

Introduction

Feature detection (French: *détection de caractéristiques*) concerns the search for specific marks in an image, such as edges or corners of the objects, geometric shapes (lines, circles…), particular patterns, and so on. In this way, feature detection extracts high-level information from the image. Many applications need feature detection, for example: measuring the detected objects, recognizing a specific object or classifying images according to their characteristics.

Matched Filter

When a pattern (French: *motif*), perfectly known as a sub-image g, is searched for in an image f, then the cross-correlation (French: *corrélation croisée*) between f and g is a very efficient technique. This technique is often known as matched filter (French: *filtre adapté*). The cross-correlation between f and g gives a new image $R_{f,g}$ defined as:

$$R_{f,g}(u,v) = \sum_{m,n} f(m,n)g(u+m,v+n).$$

The cross-correlation can be calculated as a convolution, hence the term “filter” in the name of this technique.

Usually, f and g are normalized into:

$$\tilde{f}(m,n) = \frac{f(m,n) - \mu_f}{\sigma_f} \qquad \tilde{g}(m,n) = \frac{g(m,n) - \mu_g}{\sigma_g}$$

where μ_f and σ_f are respectively the mean and the standard deviation of the image f. This results in the normalized cross-correlation (French: *corrélation croisée normalisée*) which is insensitive to changes in amplitude:

$$\tilde{R}_{f,g}(u,v) = \sum_{m,n} \tilde{f}(m,n)\tilde{g}(u+m,v+n).$$

Fig. 87 gives an example of matched filter.

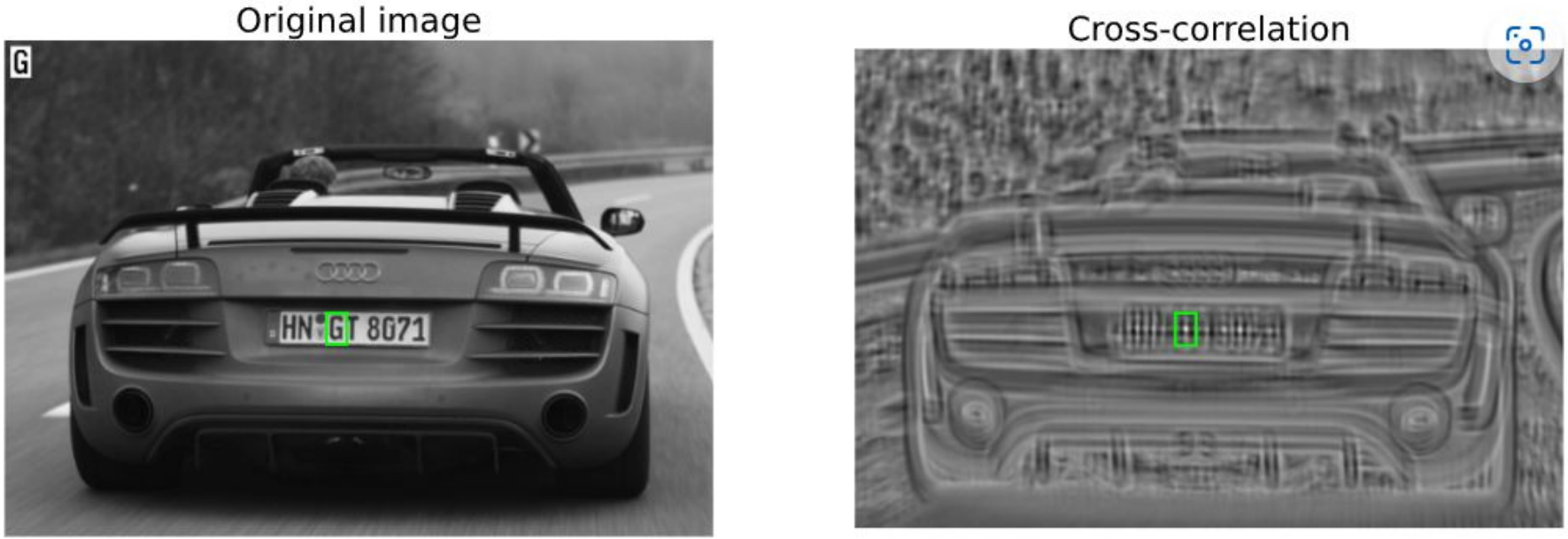


Fig. 87 Normalized cross-correlation with the pattern shown top-left (the letter G). #

As seen in Fig. 88, the major limit of the matched filter is that it is sensitive to variations in orientation, size, etc.

Fig. 88 Normalized cross-correlation with the pattern shown top-left (the digit 0).

To overcome this limit, one can apply several matched filters, each representative of all the variations of the patterns. However, this idea is very time-consuming!

Edge Detection

An edge (French: *contour*) in an image is the frontier that delimits two objects. Therefore, edge detection is useful for identifying or measuring objects, or segmenting the image.

The advantage of using the derivatives

Edges are characterized by a rapid variation in the intensity of the pixels. Fig. 89 represents the brightness profile along a horizontal line in the image. One clearly sees that the outline of the industrial piece shows a sudden decrease in the brightness of the pixels.

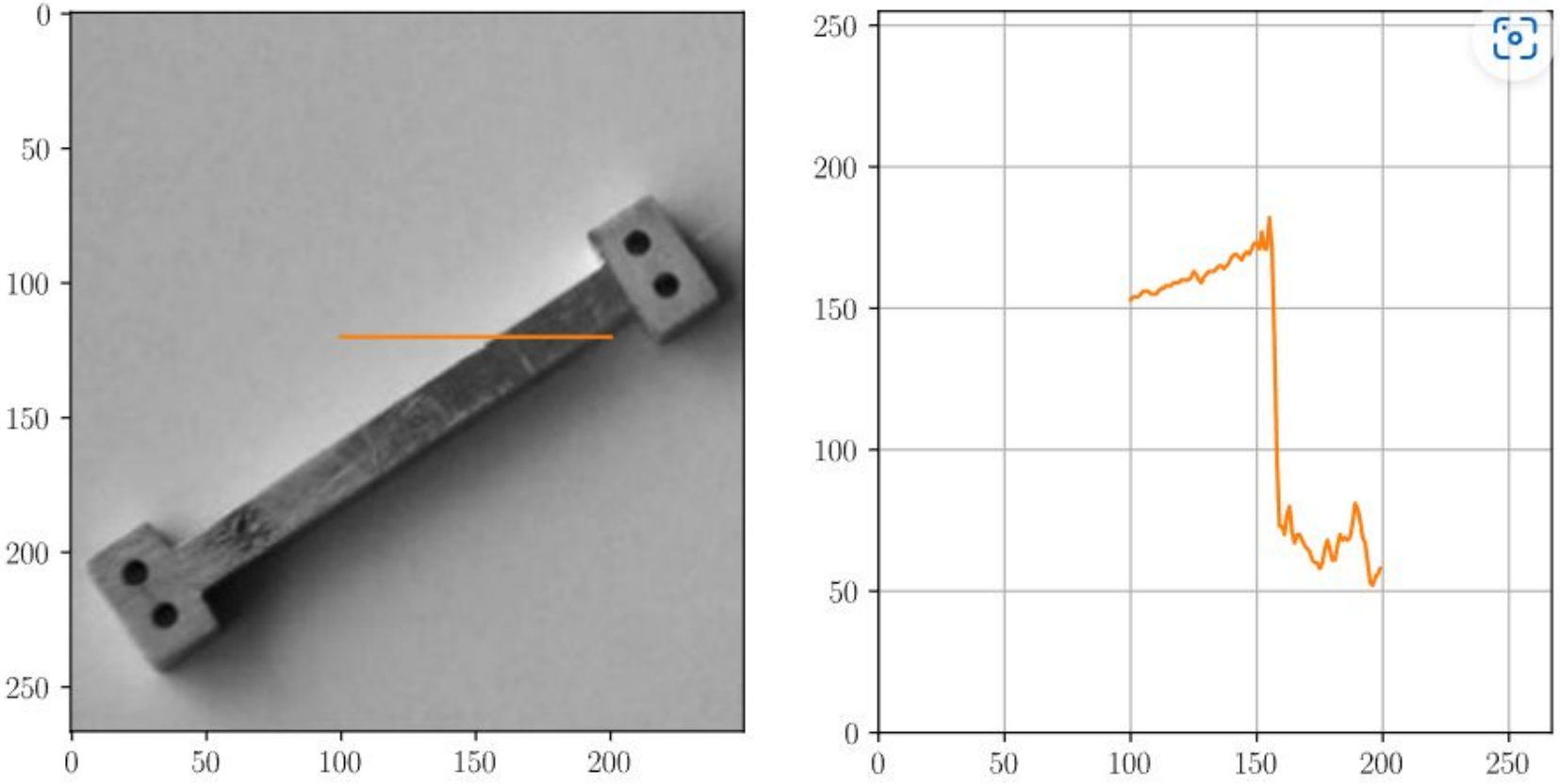


Fig. 89 An image and the luminosity profile on it. #

From this example, it appears that derivation is an efficient tool for highlighting the edges: an edge can be detected by analyzing the first derivative of the intensity profile, taken perpendicular to the edge. Similarly, an edge can be detected by determining the zero crossing of the second derivative.

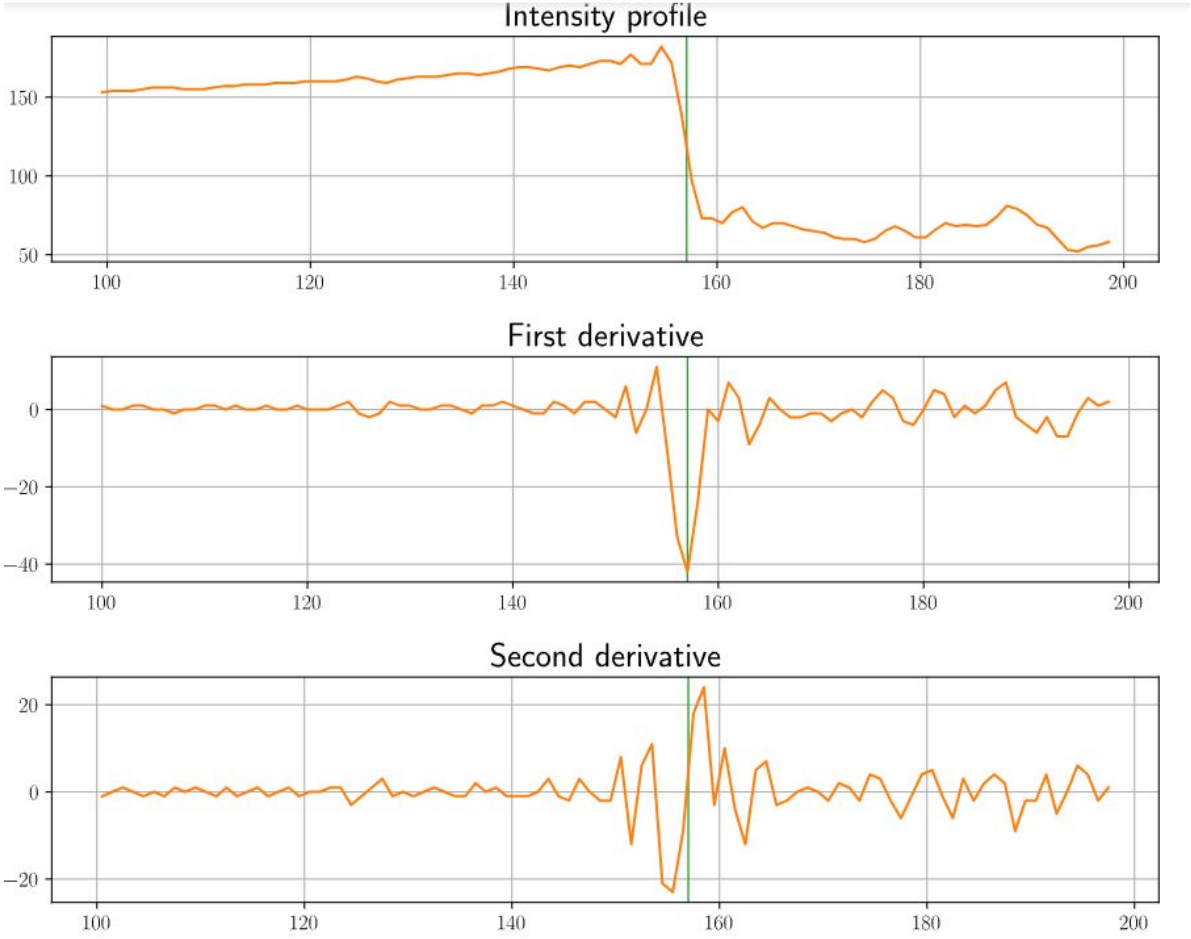


Fig. 90 The profile of Fig. 89 and its derivatives. #

Because an image depends on two dimensions, the derivatives must be calculated according to the two axes. For example, the first derivative of an image f consists of the two terms $\frac{\partial f(x,y)}{\partial x}$ and $\frac{\partial f(x,y)}{\partial y}$. In addition, the derivatives are discrete differences in the case of digital images.

Thus the first derivative, also called “gradient” (French: *gradient*), is defined by:

$$\nabla f = \begin{pmatrix} f(x+1,y) - f(x,y) \\ f(x,y+1) - f(x,y) \end{pmatrix}$$

In the same way, the second derivative, called “Laplacian” (French: *laplacien*), is defined by:

$$\Delta f = \begin{pmatrix} f(x+1,y) - 2f(x,y) + f(x-1,y) \\ f(x,y+1) - 2f(x,y) + f(x,y-1) \end{pmatrix}$$

Gradient Operators

Gradient operators are very simple methods for detecting edges. They use the first derivative and can be calculated using a convolution. Indeed, the first derivative along the x axis of an image f can be written as a convolution product:

$$f(x+1,y) - f(x,y) = \sum_m \sum_n h_x(m,n) f(x-m, y-n)$$

where h_x is a convolution kernel such that:

- $h_x(0,0) = -1,$
- $h_x(-1,0) = +1,$
- and $h_x(m,n) = 0$ elsewhere.

Thus, the kernel h_x is:

$$h_x = \begin{pmatrix} +1 & -1 \\ 0 & 0 \end{pmatrix}$$

Similarly, the first derivative along the y axis is written as the convolution of f with the kernel h_y:

$$h_y = \begin{pmatrix} +1 & 0 \\ -1 & 0 \end{pmatrix}$$

Note that the row of 0 in h_x and the column of 0 in h_y allows having kernels of the same size. These two kernels are very the very basic gradient operators and, in practice, variants of them are used.

Roberts operators

The Roberts operators are along the diagonals [Roberts 1965]:

$$h_x = \begin{pmatrix} +1 & 0 \\ 0 & -1 \end{pmatrix} \quad h_y = \begin{pmatrix} 0 & +1 \\ -1 & 0 \end{pmatrix}$$

Prewitt operators

Another variant are the Prewitt operators [Prewitt 1970] which deal with a kernel with an odd size:

$$h_x = \begin{pmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{pmatrix} \quad h_y = \begin{pmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

Sobel operators

Finally, the Sobel operators [Sobel 1968] are a smoothed version of the Prewitt operators. Indeed, the coefficients reproduce a convolution by a Gaussian filter, which tends to play the role of a mean filter to attenuate noise:

$$h_x = \begin{pmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{pmatrix} \quad h_y = \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Note that the sum of the elements is zero, as the Roberts and Prewitt operators.

In Fig. 91, one can see that h_x detects horizontal edges. For example, the top and bottom of the clock are clearly detected. The top edge is white because it corresponds to an edge from black to white. On the contrary, the bottom edge is black. On the other side, h_y detects vertical edges.

Fig. 91 Horizontal and vertical Sobel operators. #

Magnitude and angle

From the operators above, we also define:

- the magnitude of the edge, which can be interpreted as the "fusion" of the horizontal and vertical Sobel operators:

$$M = \sqrt{(h_x * f)^2 + (h_y * f)^2}$$

  In Fig. 92, all the edges appear with the same color, whatever their orientation.

- the angle of the edge:

$$A = \arctan\left(\frac{h_y * f}{h_x * f}\right)$$

In Fig. 92, the color of the edges corresponds to the angle. For example, an angle of 0 is red, and an angle of $\pi/2$ is blue.

Fig. 92 Magnitude and angle of the Sobel operator. #

Thresholding

In some cases, it would be useful to detect the most important edges. To do that, one can threshold the magnitude to keep only the large values of the gradients. As you can see in Fig. 93, the threshold makes the second hand disappear. Indeed, the magnitude is lower than other parts of the image because the second hand is gray and not black.

Fig. 93 Thresholding the magnitude of the Sobel operator makes the second hands disappear! #

The impact of noise

Noise in an image essentially brings great variations in brightness between pixels. Therefore, the gradient operators is based on the derivative are very sensitive to the noise, as seen in Fig. 94. Then it may be useful to denoise the image before edge detection. In figure Fig. 95, a small mean filter is used before the Sobel filter: the result is much cleaner than without the use of the mean filter.

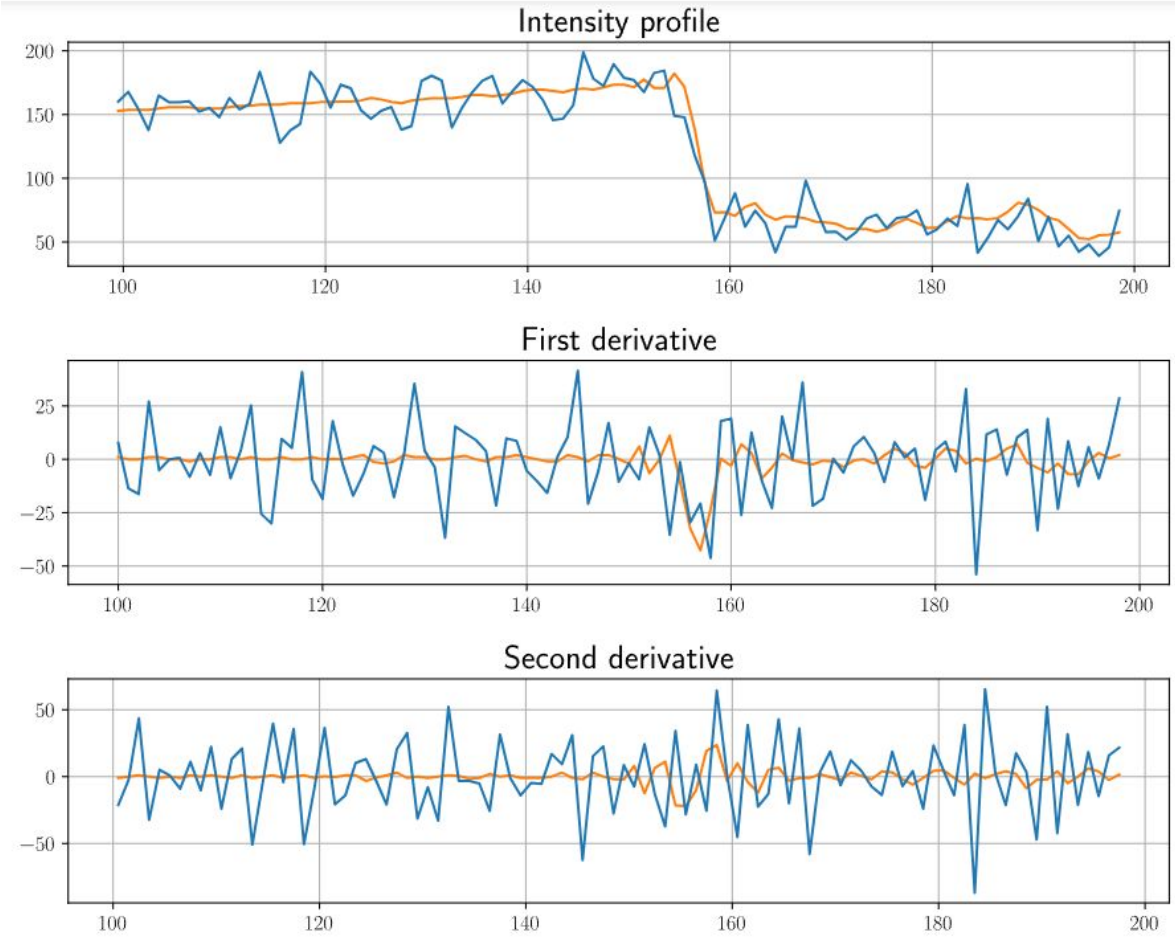


Fig. 94 Consequence of noise on the brightness profile and its derivatives

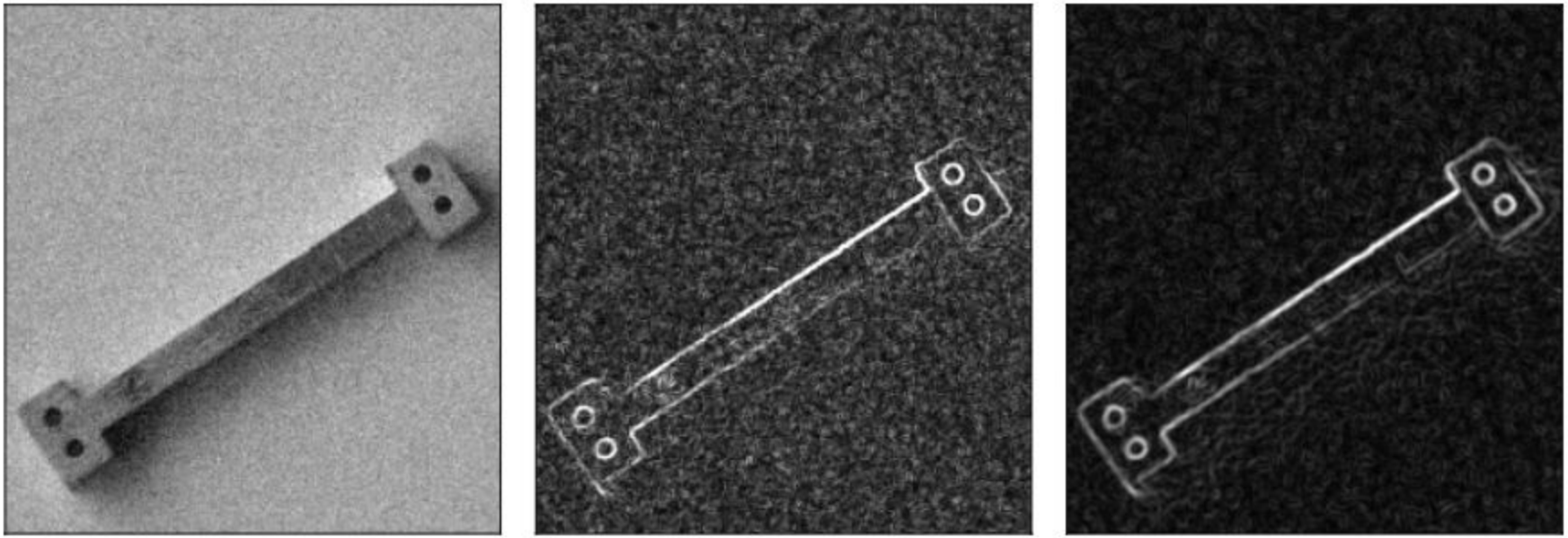


Fig. 95 Left: noisy image. Center: Sobel operator (magnitude). Right: Sobel operator applied on the result of a $\mathbf{3} \times \mathbf{3}$ mean filter.

Advanced methods

Following the gradient operators, new methods have been proposed to improve edge detection by taking into account the noise and the nature of the edges:

- the Marr-Hildreth detector [Marr & Hildreth 1980],
- the Canny detector [Canny 1986].

The Marr-Hildreth detector consists of:

1. apply a Gaussian filter g on the image f to reduce noise (this is similar to a mean filter),
2. compute the Laplacian (second derivative) ℓ on the softened image (this is implemented with a convolution),
3. determine the zero crossings of the result.

As $\ell * (g * f) = (\ell * g) * f$, the first two steps are merged into a single convolution by $\ell * g$. Hence, the filter $\ell * g$ is the second derivative of a Gaussian:

$$(\ell * g)(x, y) = -\left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right).$$

The filter $\ell * g$ is represented Fig. 96. It is also called "LoG" for Laplacian of Gaussian (no French equivalent) or Mexican hat (for the resemblance to a sombrero).

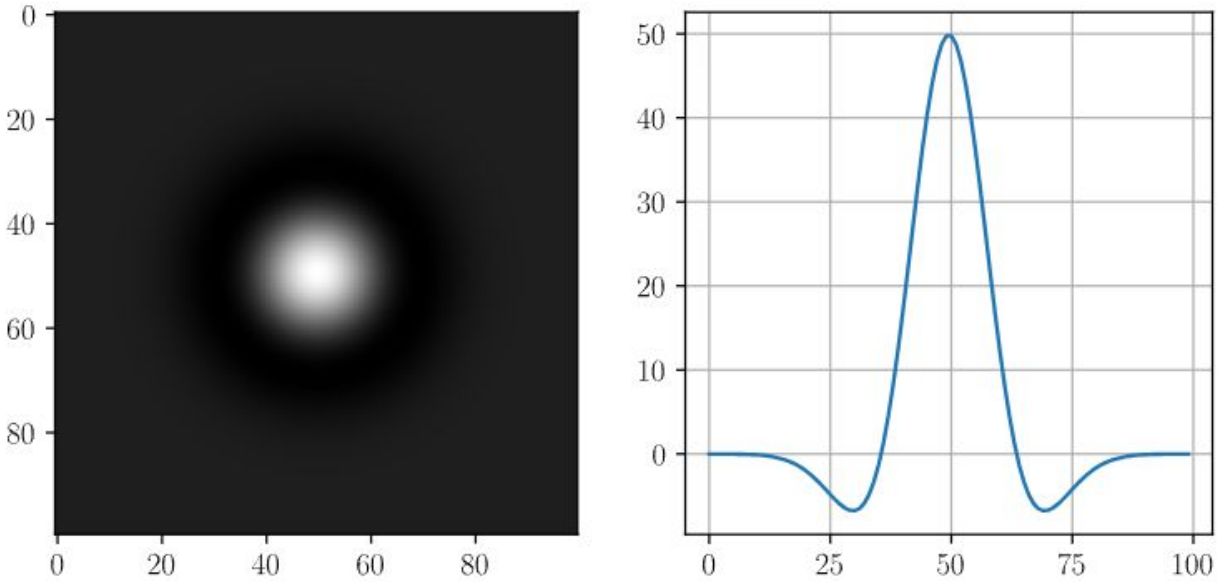


Fig. 96 Laplacian of Gaussian (left: as an image, right: profile along an axis). #

The zero-crossings in the image resulting from the LoG convolution are given by searching for changes of sign in the intensity of two pixels. Fig. 97 gives an example.

Fig. 97 Marr-Hildreth Detector. Left: original image, center: result of the LoG filter, right: detection of the zeros crossings. #

Canny Detector

According to Canny, a good detector should serve the following purposes:

- all edges should be found,
- there should be a minimum of spurious responses,
- the edges must be correctly localized (*i.e.* the distance between a detected point and the true point of the edge must be as small as possible),
- the thickness of the detected edges must be 1 pixel (therefore only one point must be detected for each true point of the edge).

Canny expressed these goals in mathematical form and came up with optimal solutions verifying these goals. The Canny detector algorithm follows the four steps detailed below.

1. The image f is first smoothed with a Gaussian filter to reduce noise. This is done by using a convolution with a Gaussian kernel g to obtain an image $z = f * g$.
2. The gradient of the image is calculated (amplitude and angle):

$$M = \sqrt{(h_x * z)^2 + (h_y * z)^2} \quad \text{and} \quad A = \arctan\left(\frac{h_y * z}{h_x * z}\right)$$

3. Non-maxima are removed from the amplitude. This means that excessively large outlines in M are replaced by thinner outlines. For this, we apply the algorithm below:

    1. For each pixel (x, y) in M:
        1. Choose the direction (vertical, horizontal or one of the two diagonals) the closest to $A(x, y)$
        2. If $M(x, y)$ is lower than one of its neighbors in the chosen direction then cancel the gradient: $M(x, y) = 0$

4. The last step consists of thresholding by hysteresis for the bad edges. Two thresholds are therefore defined ($T_{\text{high}} > T_{\text{low}}$) and the algorithm below is applied:

    1. For each pixel (x, y) in M:
        1. If $M(x, y) > T_{\text{high}}$ then (x, y) is an edge
        2. If $T_{\text{low}} < M(x, y) < T_{\text{high}}$ then (x, y) is an edge if and only if it is neighbor of an edge pixel
        3. If $M(x, y) < T_{\text{low}}$ then (x, y) is not an edge

Fig. 98 gives an example of Canny edge detection.

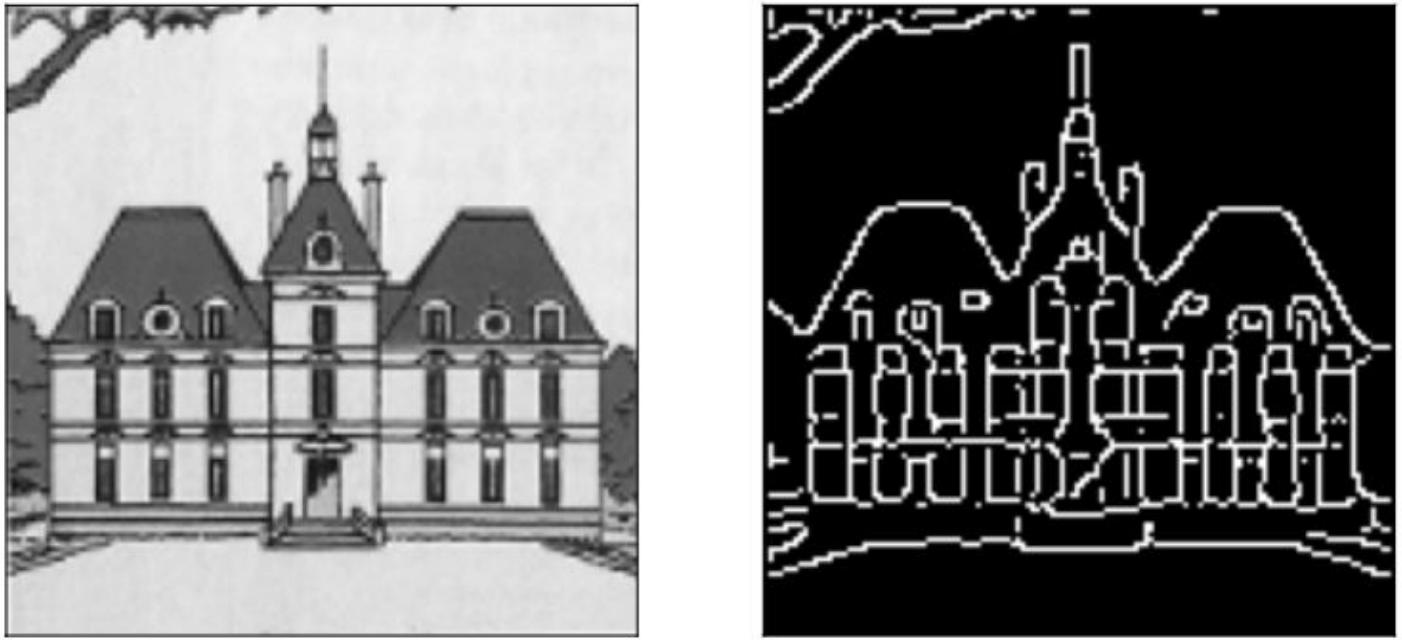
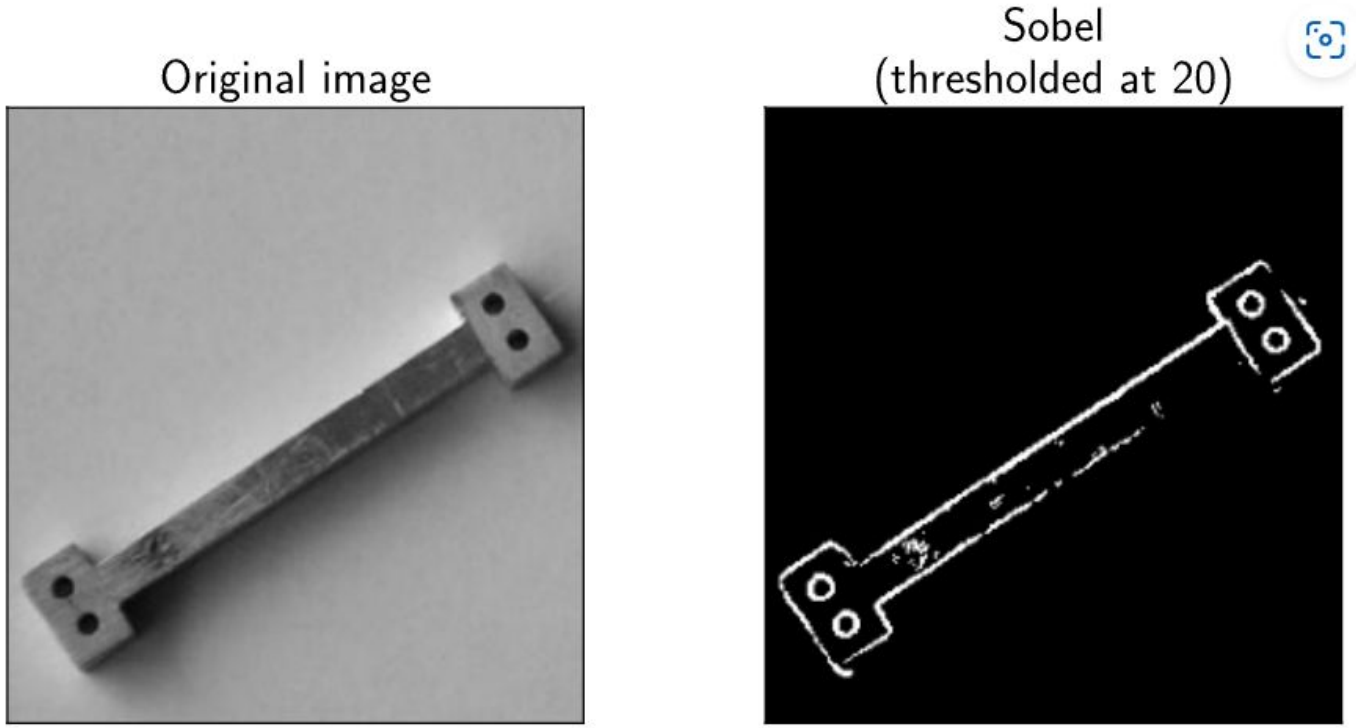


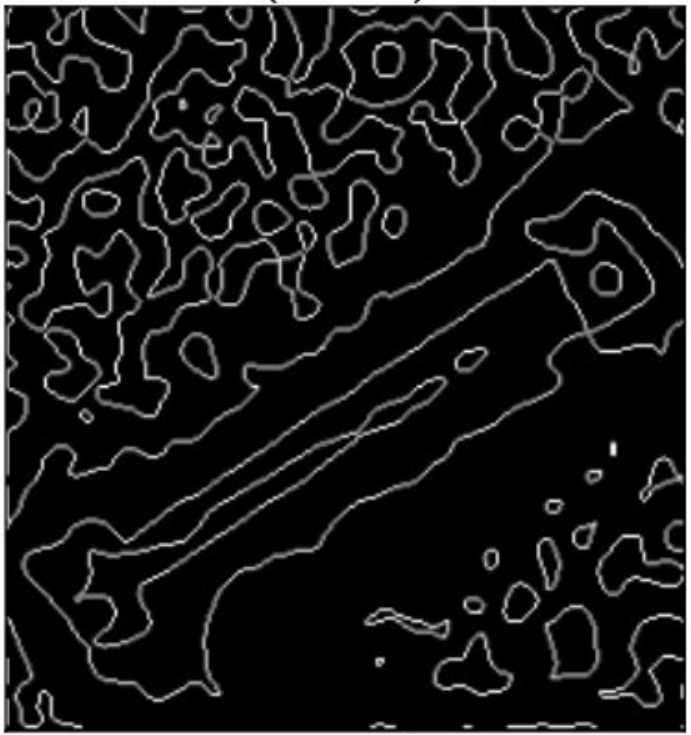
Fig. 98 Canny Detector. #

Comparison

Fig. 99 shows the results of the Sobel, Marr-Hildreth and Canny detectors on an image.



Marr-Hildreth
$(\sigma = 4)$



Canny
$(\sigma = 1.5, T_{low} = 25, T_{high} = 50)$

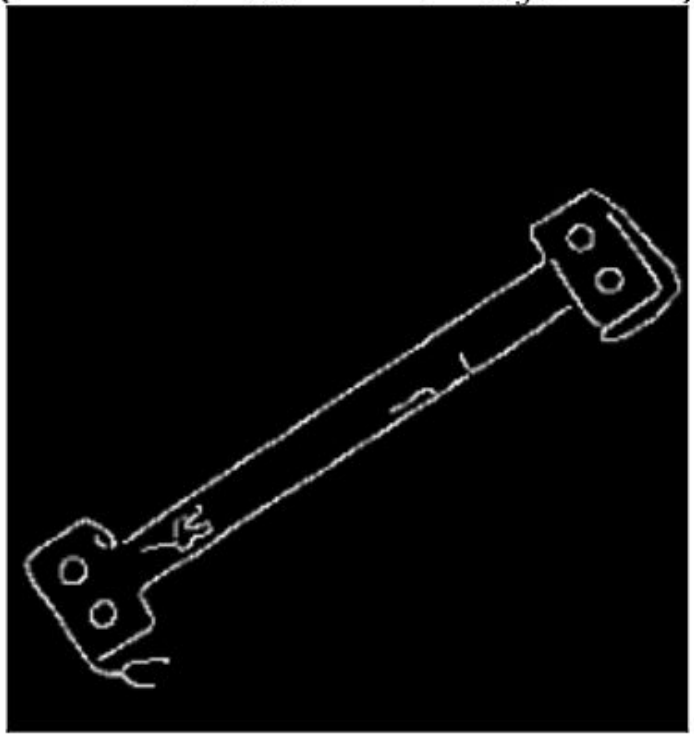


Fig. 99 Comparison of Sobel, Marr-Hildreth and Canny detectors. #

In Fig. 100, which compares Marr-Hildreth and Canny detectors, one can see that the edges detected with Canny detector are better localized.

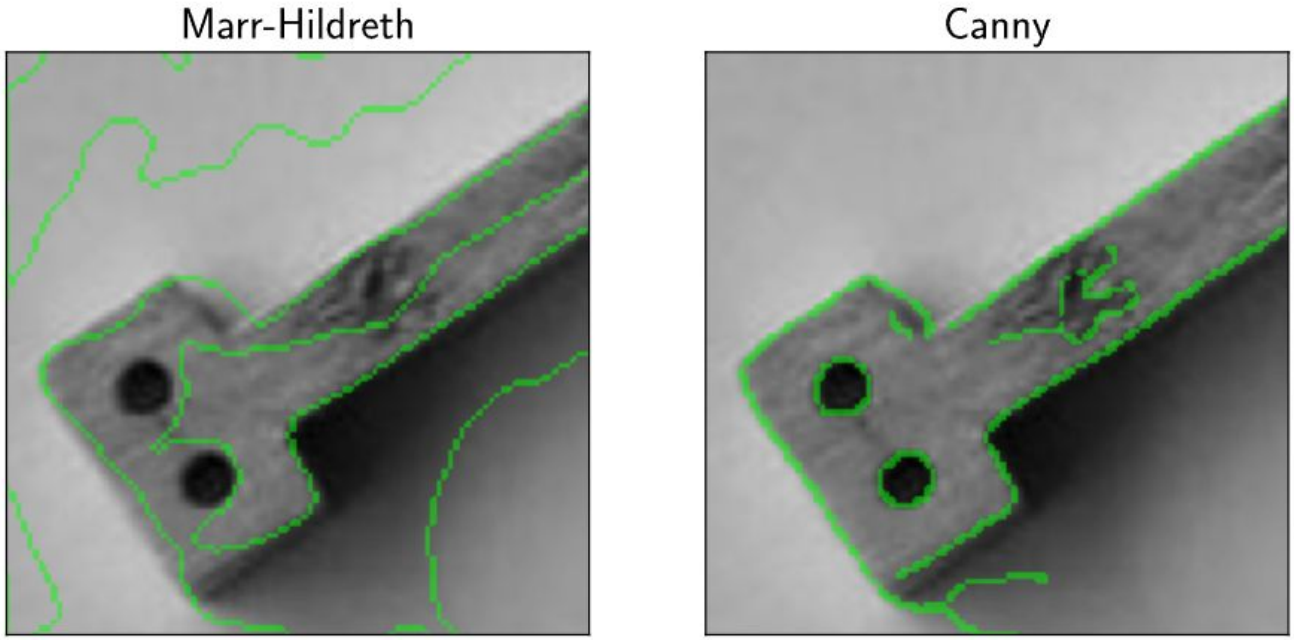


Fig. 100 Comparison between Marr-Hildreth and Canny detectors (zoom). The edges are shown in green. #

Corner detection

To detect the corners of objects in an image, one can start by detecting edges then determining where two edges meet. There are however other methods, among which:

- the Moravec detector [Moravec 1980],
- the Harris detector [Harris & Stephens 1988].

The principle of this detector is to observe if a sub-image, moved around one pixel in all directions, changes significantly. If this is the case, then the considered pixel is a corner.

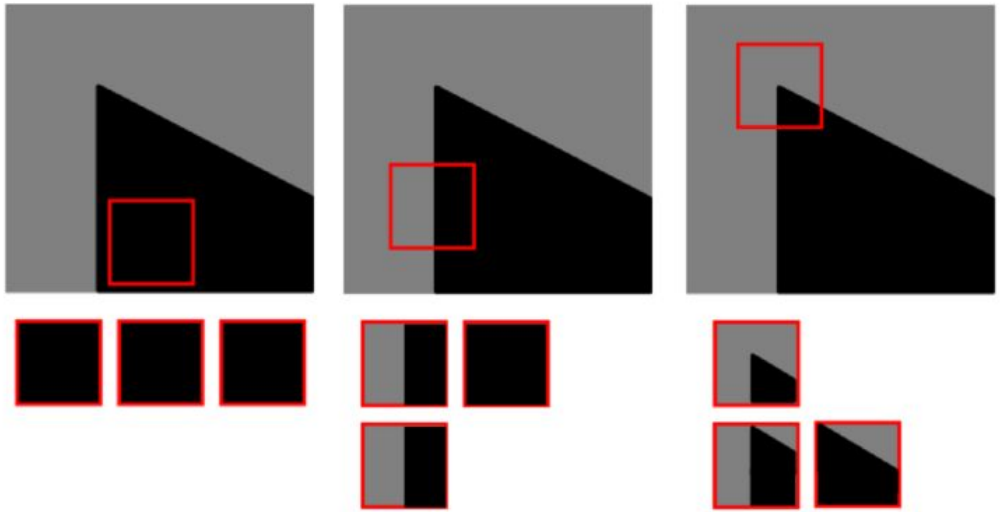


Fig. 101 Principle of Moravec detector. From left to right : on a flat area, small shifts in the sub-image (in red) do not cause any change; on a contour, we observe changes in only one direction; around a corner there are significant changes in all directions. #

Mathematically, the change is characterized in each pixel (m, n) of the image by $E_{m,n}(x, y)$ which represents the difference between the sub-images for an offset (x, y):

$$\forall m, n, x, y \qquad E_{m,n}(x, y) = \sum_{u,v} w_{m,n}(u, v) \big[f(u + x, v + y) - f(u, v) \big]^2$$

where:

- x and y represent the offsets in the four directions: $(x, y) \in \{(1, 0),\ (1, 1),\ (0, 1),\ (-1, 1)\}$,
- $w_{m,n}$ is a rectangular window around pixel (m, n),
- $f(u + x, v + y) - f(u, v)$ is the difference between the sub-image $f(u, v)$ and the offset patch $f(u + x, v + y)$,

In each pixel (m, n), the minimum of $E_{m,n}(x, y)$ in the four directions is kept and denoted $F_{m,n}$. Finally, the detected corners correspond to the local maxima of $F_{m,n}$, that is, at pixels (m, n) where the smallest value of $E_{m,n}(x, y)$ is large.

It turns out that Moravec detector has several limitations. First, w is a binary window and therefore the detector considers all pixels in the window with the same weight. When the noise in the image is high, it can lead to false corner detections. Second, only four directions are considered. Third, the detector remains very sensitive to edges because only the minimum of E is considered. For these reasons, Harris has proposed a detector to overcome these limitations.

Harris detector

To avoid a noisy response, the rectangular window w of the Moravec detector is replaced by a Gaussian window w in the expression of $E_{m,n}(x, y)$.

To extend the Moravec detector to all directions, not limited to the initial four directions, a Taylor series expansion is performed on the shifted sub-image $f(u + x, v + y)$:

$$f(u + x, v + y) \approx f(u, v) + x\,\partial_x f(u, v) + y\,\partial_y f(u, v).$$

Therefore :

$$E_{m,n}(x, y) \approx \sum_{u,v} w_{m,n}(u, v) \left[x\,\partial_x f(u, v) + y\,\partial_y f(u, v)\right]^2$$

This expression can be written in the following matrix form:

$$E_{m,n}(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} M \begin{pmatrix} x \\ y \end{pmatrix}$$

where

$$M = \sum_{u,v} w_{m,n}(u,v) \begin{pmatrix} (\partial_x f)^2 & \partial_x f \, \partial_y f \\ \partial_x f \, \partial_y f & (\partial_y f)^2 \end{pmatrix}$$

Finally, the last limit of the Moravec detector can be avoided by considering a new measure of the presence of a corner: more information about the intensity change in the window can be obtained by analyzing the eigenvalues λ_1 and λ_2 of the matrix M (Fig. 102). Indeed, the presence of a corner is attested if the derivatives of f are very large, then M has large coefficients, and its eigenvalues are also very large.

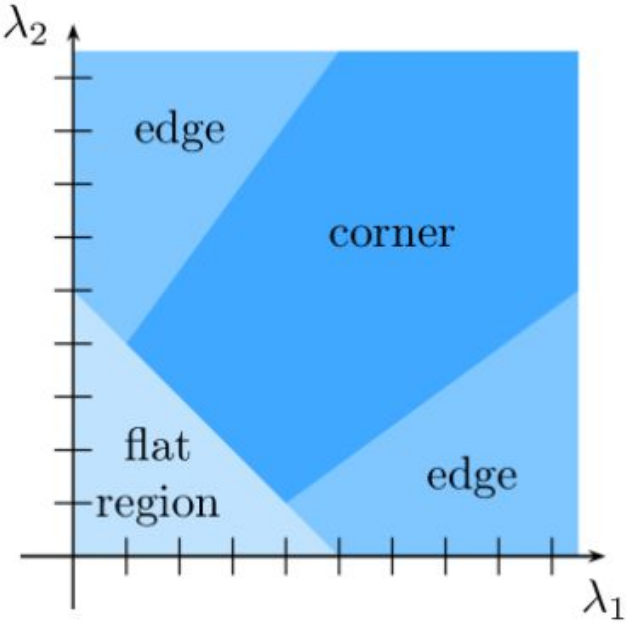


Fig. 102 Decision to be taken in function of the eigenvalues. #

The calculation of the eigenvalues of M can be difficult, so an alternative is to calculate:

$$R = \det(M) - k(\text{trace}(M))^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$

with $0.04 < k < 0.06$.

Thus, the values of R are low in a flat region, negative on an edge, and positive on a corner (Fig. 103).

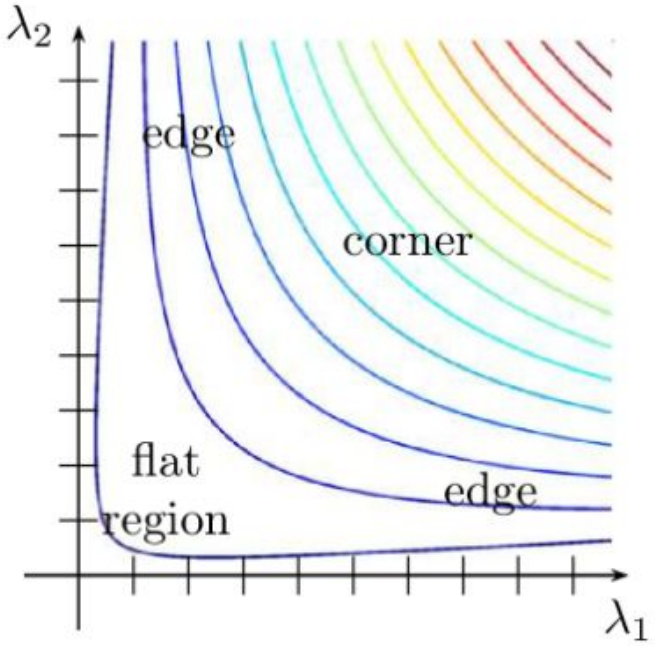


Fig. 103 Decision to be taken in function of R. #

The Harris detector is illustrated on the example of Fig. 104.

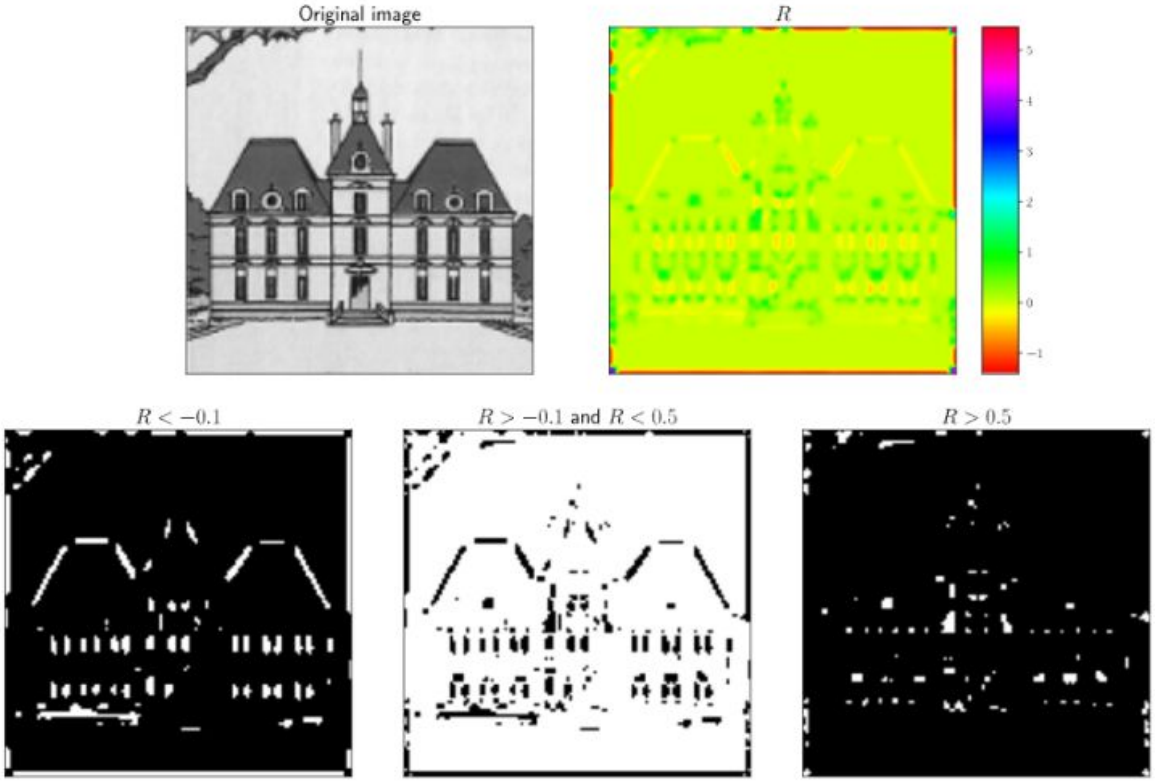


Fig. 104 Harris detector. The binary images represent the negative (contours), weak (flat areas) and positive (corners) values of the

Line detection

Line detection consists of detecting alignments of points on an image of contours. The usual method for line detection is the Hough transform [Hough 1962]. Like the Fourier transform, it transposes the image from the spatial domain to another domain, where the information of interest is represented differently. In this case, the lines in the spatial domain are transformed into points in the Hough domain.

The Hough transform is not restricted to lines and can be used to detect any shape with a mathematical parameterization. The working process remains the same as for line detection, so the sequel focuses on line detection only.

First parameterization

In the spatial domain (x, y), a line is parameterized by its coefficients a and b:

$$y = ax + b.$$

Thus, any line can be represented by the pair (a, b). This is Hough's idea: each line of the image can be represented by a point in the Hough domain (a, b). The Hough domain is also called the parameter space.

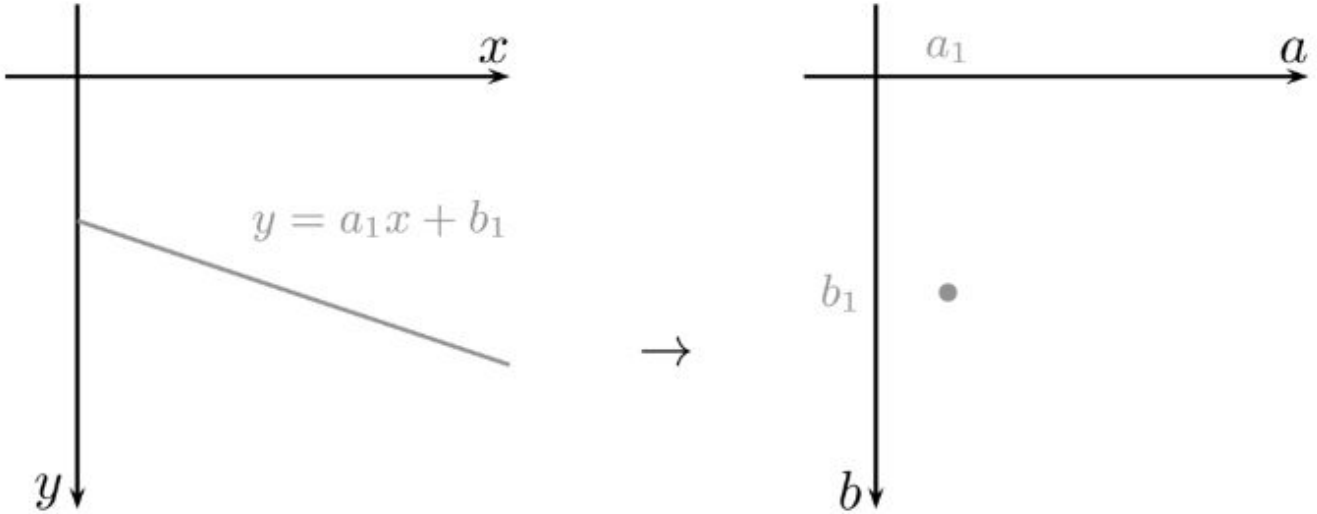
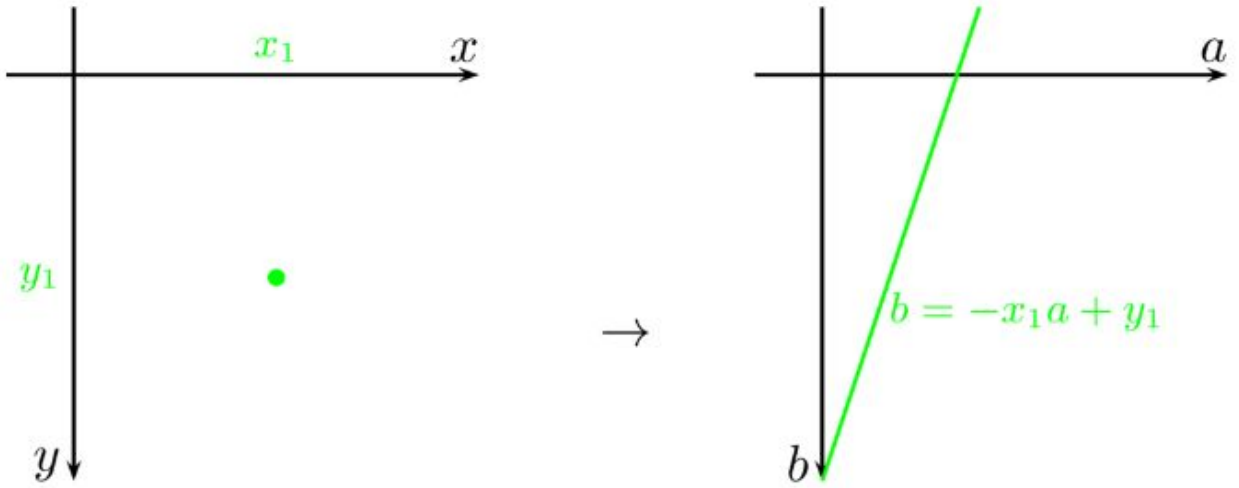
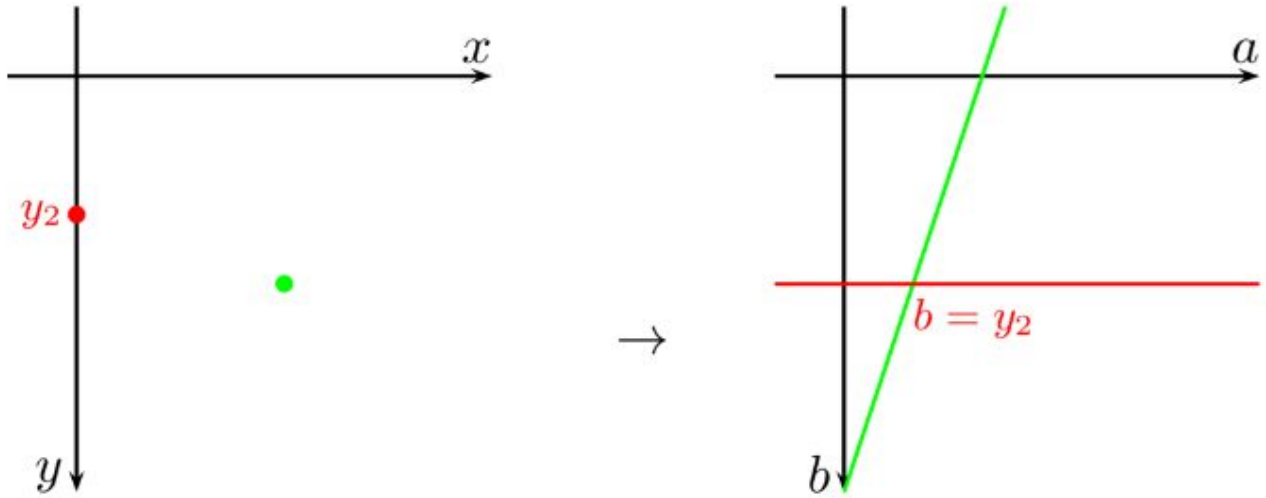


Fig. 105 The Hough transform transforms a line in the image into a point in the parameter space. #

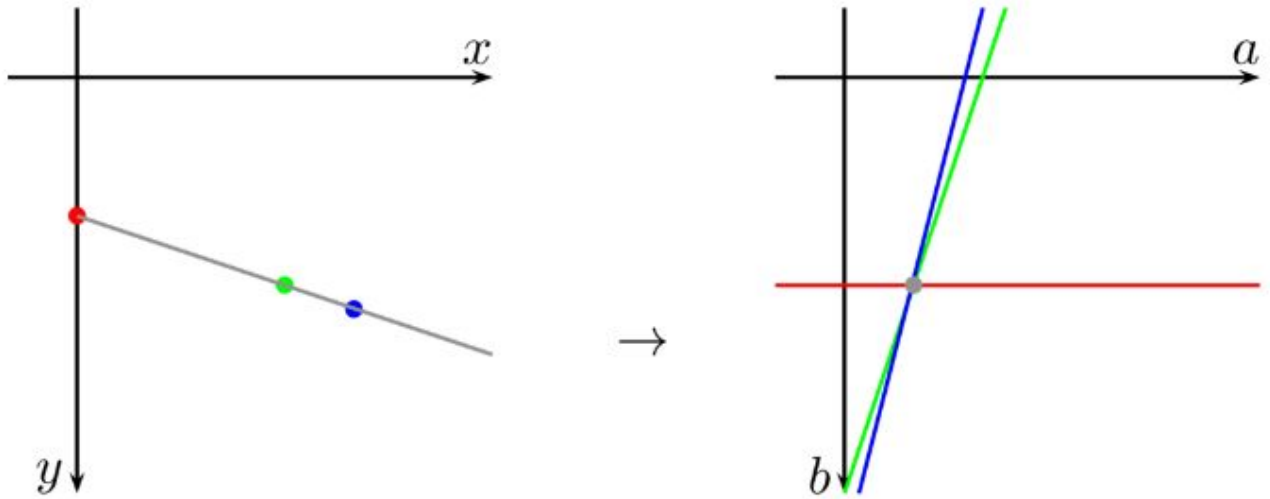
Conversely, a point in the image is represented by a line in the parameter space (which has the equation $b = \alpha a + \beta$).



In particular, a constant line $b = \beta$ in the parameter space corresponds to a point of abscissa $x = 0$ in the image.



Finally, if several points in the image are aligned, their respective lines in the parameter space intersect at a single point, which defines the equation of the line connecting them.



On a computer, the parameter space is finite and discretized (it is called an “accumulator”). Consequently, certain lines of the image cannot be represented there (*e.g.* a vertical line for which $a = \infty$). In consequence, another parameterization of lines needs to be used in practice.

New parameterization

To avoid the aforementioned problem of the parameterization space (a, b), the lines are defined from two other parameters:

- the distance d of the segment connecting the origin with the point closest to the line (this segment is perpendicular to the line),
- the angle θ that this segment makes with the x-axis.

Each line of the image is therefore parameterized by the pair (θ, d) which corresponds to a point in the parameter space (θ, d).

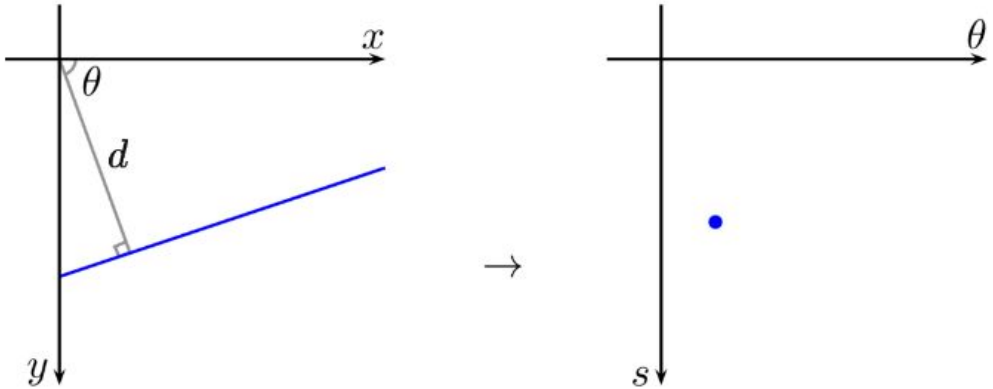


Fig. 106 New parameterization of the Hough transform. #

We can show that for each point (x_i, y_i) of the image, a sinusoid of equation $d = x_i \cos(\theta) + y_i \sin(\theta)$ is associated in the space (θ, d). The sinusoids corresponding to the points of the same line intersect at the point (θ^*, d^*) parameterizing this line.

The Hough transform algorithm is as follows:

Algorithm: Hough transform

1. Get the result of an edge detection
2. Initialize an accumulator (as the discrete space of the parameters)
3. For each pixel in the edges:
    1. Determine the sinusoid corresponding to the points
    2. Increment the accumulator along this sinusoid
4. Search for the maxima in the accumulator
5. Deduce the line parameters

Fig. 107 gives an example of a Hough transform on an image representing a square.

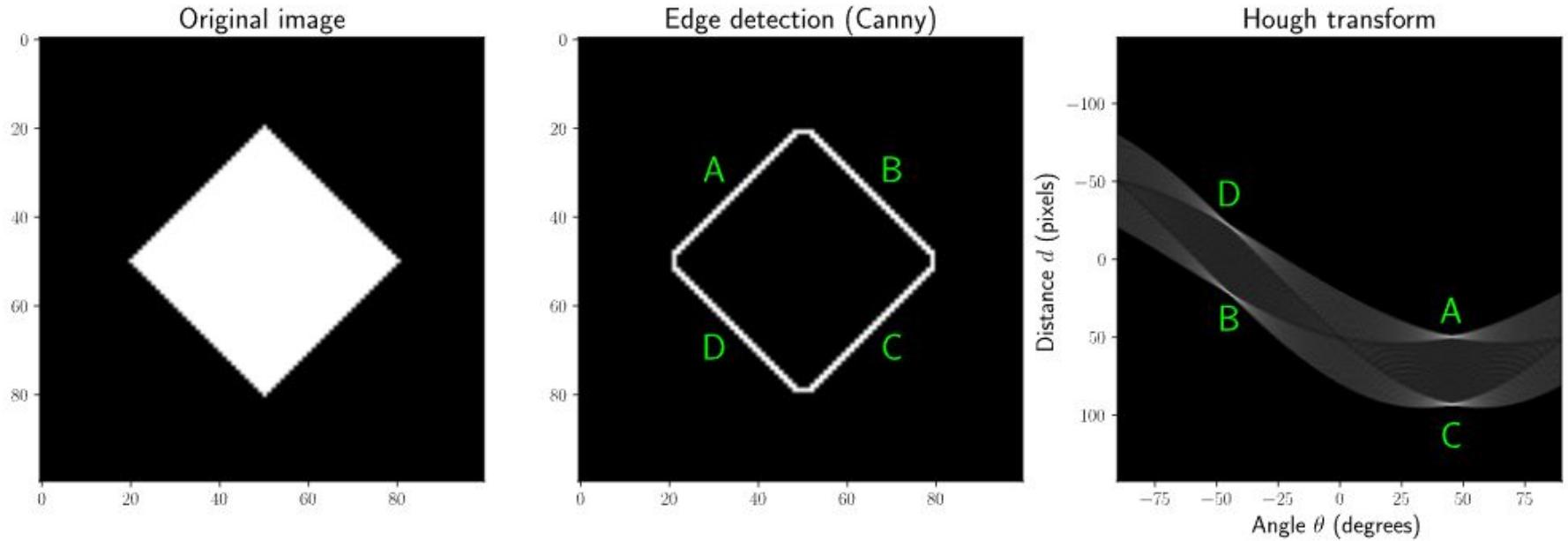


Fig. 107 Hough transform associated with the image on the left. The sinusoids intersect at four (very bright) points, each one is associated with the line with the same letter. #

Fig. 108 gives an example of a Hough transform on an real photograph.

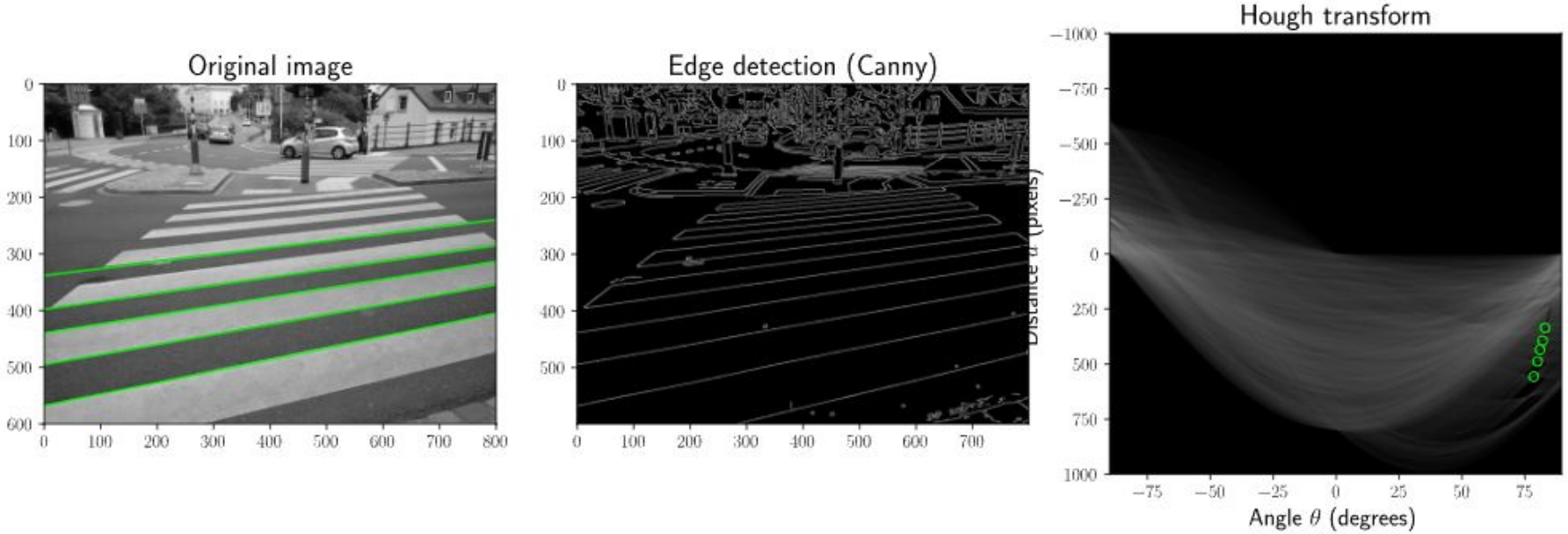


Fig. 108 Hough transform associated with the image on the left. #

Besides, it appears that the Hough transform is robust to noise and to occultation (it can detect partially covered objects)

As said above, the Hough transform can be extended to any parameterized object (circles, ellipses, etc.). For example, a circle is parameterized by three parameters (the center coordinates and the radius), then the corresponding Hough space is three-dimensional.

Because, the dimension of the accumulator is equal to the number of parameters, the main drawback of this method is that the computing time and the memory used quickly become significant.

Conclusion

We have seen in that chapter that methods for finding features are very different and depend on the seeking feature.

- For edges, usual methods are essentially a filtering using the gradient or the Laplacian (Roberts or Prewitt filters, Sobel or Canny detectors).
- For corners, the dedicated methods analyze the variation in intensity in the neighbourhood of the pixels (Moravec and Harris detectors).
- For lines and circles, the image is represented in the space of the parameters of the geometric shape (Hough transform).

References

- J. Canny, “A Computational Approach To Edge Detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence}*, vol.8, 1986.
- C. Harris and M. Stephens “A combined corner and edge detector”, *Alvey Vision Conference*, p. 147-151, 1988.
- P.V.C. Hough, “Method and means for recognizing complex patterns”, US Patent 3,069,654, 1962.
- D. Marr and E. Hildreth, “Theory of Edge Detection”, *Proceedings of the Royal Society of London*, vol. 207, 1980.
- H. Moravec, “Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover”, technical report, Carnegie–Mellon University, Robotics Institute, 1980.
- J.M.S. Prewitt, “Object enhancement and extraction”, *Picture Processing and Psychopictorics*, Academic Press, 1970.
- L.G. Roberts, “Machine Perception Of Three-Dimensional Solids”, *Computer Methods in Image Analysis*, IEEE Press, 1965.
- I. Sobel and G. Feldman, “A 3×3 Isotropic Gradient Operator for Image Processing”, In *Stanford Artificial Intelligence Project*, 1968.

LABTask

Canny Edge Detection Step by Step in Python — Computer Vision | by Sofiane Sahir | Towards Data Science
OpenCV: Harris Corner Detection
Understanding Hough Transform With Python | Alyssa (alyssaq.github.io)