Published in Image Processing On Line on 2017–11–28.
Submitted on 2017–08–18, accepted on 2017–11–09.
ISSN 2105–1232 © 2017 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2017.216

A Sub-Pixel Edge Detector: an Implementation of the Canny/Devernay Algorithm

Rafael Grompone von Gioi[1], Gregory Randall[2]

[1] CMLA, ENS Cachan, France (grompone@cmla.ens-cachan.fr)
[2] IIE, UdelaR, Uruguay (randall@fing.edu.uy)

Communicated by Jose-Luis Lisani *Demo edited by* Rafael Grompone von Gioi

Abstract

An image edge detector is described which produces chained edge points with sub-pixel accuracy. The method incorporates the main ideas of the classic Canny and Devernay algorithms. The analysis shows that a slight modification to the original formulation improves the sub-pixel accuracy of the edge points.

Source Code

The ANSI C source code for this algorithm is available from the web page of this article[1]. Compilation and usage instructions are included in a `README.txt`.

Keywords: edge detection; sub-pixel accuracy; Canny; Devernay; edge point chaining

1 Introduction

Edge detection in digital images is one of the oldest problems in computer vision and continues to be a very active area of research [9]. There is a vast literature on the subject and there are probably more algorithms proposed for edge detection than for any other problem in image processing. We refer the interested reader to the excellent review by Papari and Petkov [9].

The most famous method is the one proposed by Canny [3], a name that has become almost a synonym of edge detection. Canny established three quality measures of a given edge detector (good detection, good localization of the edge pixels and unique response of the filter) and found an optimal filter (in the sense of those three criteria) to detect a border modeled as a step signal plus noise. Canny proposed to approximate the ideal filter by the convolution of the image with the first derivative of the Gaussian function and then suppressing the pixels with non-maximal gradient modulus along the direction of the gradient.[2]

[1] https://doi.org/10.5201/ipol.2017.216

[2] In addition to its important contributions, Canny's method is an excellent synthesis of many of the ideas on edge detection proposed before [10]. The non-maximal idea is related to the zero-crossings of the Laplacian as proposed by Marr and Hildreth [8]; the formulations are almost equivalent in 1D. For the 2D case, Canny's method is following Haralick's insight of using zero crossing of the second directional derivative taken in the direction of the gradient [7].

Several improvements were added later to the Canny edge detector. Among them, we will be interested in a simple post-processing proposed by Devernay [5] to obtain sub-pixel accuracy. In a nutshell, the position of an edge point is estimated as the maximum of an interpolation of the values of the gradient norm. The correction is as simple as computing a quadratic interpolation of the values of the gradient norm between three neighboring values in the gradient direction. The method has the advantage of producing an excellent accuracy with a low computational cost.

As part of this work, a detailed analysis of the accuracy of the Devernay sub-pixel step has been performed, which allows to explain an oscillatory artifact occasionally observed. The same analysis suggests a simple modification of Canny's algorithm that improves slightly the result.

An algorithm for chaining the sub-pixel edge points into polygonal curves is also described.

This paper is organized as follows. Section 2 introduces the main ideas of Canny's method. Devernay's sub-pixel correction is then described in Section 3 and its accuracy is analyzed in Section 4. Following the analysis, a modification to the original formulation is proposed in Section 5. The edge point chaining method is presented in Section 6. A detailed description of the full algorithm is provided in Section 7 and its computational complexity is analyzed in Section 8. Section 9 illustrates the strengths and limitations of the method with experiments on images. Finally, Section 10 concludes the paper.

2 The Canny Method

A significant part of Canny's groundbreaking work [3, 1, 2] concentrates on finding an optimal filter for detecting edges in one-dimensional signals. The sought filter should provide local maxima as close as possible to the center of the true edges without producing false detections or multiple responses. After giving a mathematical formulation to these requirements, Canny solved the optimization problem leading to the optimal filter. Then, Canny showed that the optimal filter can be well approximated by the first derivative of a Gaussian and proposed to use the latter in order to benefit from its computational efficiency.

Canny proposed to use the same filter for two-dimensional signals (images) by analyzing the one-dimensional signal along the direction n, normal to the boundary. Thus, edge points are defined to be the local maxima in the direction n of

$$\frac{\partial}{\partial n} G \star I, \tag{1}$$

where G is a Gaussian kernel and I is the image. This method of keeping only local maxima is called *non-maximum suppression*. Note that due to the separability of the Gaussian kernel, even if G is a 2D isotropic kernel, when restricted to the 1D line along the direction n we get exactly the same formulation as for the one-dimension case: the derivation along the direction n is equivalent to a convolution along n with the first derivative of a Gaussian.

The direction n orthogonal to the boundary can be estimated by the image gradient

$$n \approx \frac{\nabla(G \star I)}{|\nabla(G \star I)|}. \tag{2}$$

The method is completed by a procedure to separate true boundaries from spurious responses produced by noise. The edge strength is

$$|\nabla(G \star I)|, \tag{3}$$

which corresponds to the norm of the derivative along the direction n. Canny proposed to discriminate true edges using two thresholds, H and L, with hysteresis. Local maxima with strength above

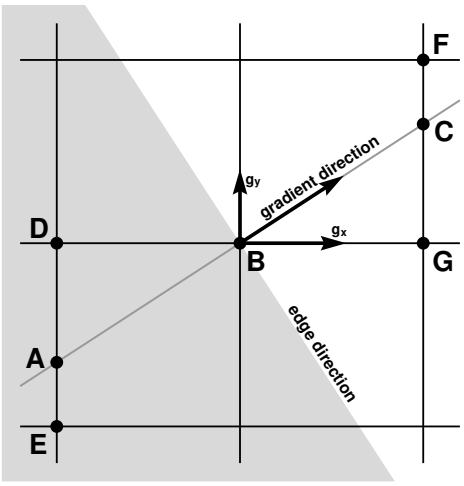


Figure 1: Support of Canny's non-maximum suppression operator.

the high threshold H are immediately validated and added to the output; local maxima connected to validated points are validated if their strength is above the low threshold L.

Canny also discussed how to estimate the noise, the use of multiple scales of analysis, and the use of directional operators. Though interesting, none of these aspects led to a concrete algorithm and are not part of the standard Canny edge detector.

There are two implementation details that define the method: how to choose the maxima points along the direction n in the discrete grid and what is the notion of connectivity used in the hysteresis thresholding step. These two important points are explored in detail in Canny's technical report [1] which is a revised version of his master degree thesis. We will describe the former here; the notion of connectivity is discussed in Section 6.

Canny proposed the following scheme to perform the non-maximum suppression in a 2D grid [1, Sect. 5.3]. Let $||g||$, g_x and g_y denote, respectively, the norm, x-component and y-component of the image gradient $\nabla(G \star I)$. Suppose we want to decide if pixel B is a local maximum of $||g||$ in the direction of the image gradient, see Figure 1. For that aim, $||g(B)||$ needs to be compared to $||g(A)||$ and $||g(C)||$; B is a local maximum when $||g(B)|| > ||g(A)||$ and $||g(B)|| > ||g(C)||$. A simple finite difference scheme can be used to approximate $||g||$ at the positions of the grid, for example for the points labeled B, D, E, F, and G. Values at points A and C cannot be computed in this way. Canny proposed to approximate the values $||g(A)||$ and $||g(C)||$ by a simple linear interpolation of the two nearest known values

$$
\begin{aligned}
||g(A)|| &\approx \frac{g_x(B) - g_y(B)}{g_x(B)} ||g(D)|| + \frac{g_y(B)}{g_x(B)} ||g(E)||, \\
||g(C)|| &\approx \frac{g_x(B) - g_y(B)}{g_x(B)} ||g(G)|| + \frac{g_y(B)}{g_x(B)} ||g(F)||.
\end{aligned} \tag{4}
$$

All in all, Canny's method depends on three parameters: S, the standard deviation of the Gaussian kernel G, and the two thresholds H and L. When a pixel B survives Canny's non-maximum suppression procedure and the threshold with hysteresis step it becomes a Canny edge pixel.

3 The Devernay Sub-Pixel Correction

Canny's method finds edge *pixels*. In other words, edge points are extracted up to the pixel grid precision. In some applications one may require to have the position of the edge points with finer accuracy. For that aim, Devernay [5] proposed a very elegant addition to the original Canny algorithm

to produce sub-pixel accuracy edge points: the position of an edge point is refined as the maximum of an interpolation of the gradient norm; the correction is as simple as computing a quadratic interpolation of the gradient norm between three neighboring positions along the gradient direction.

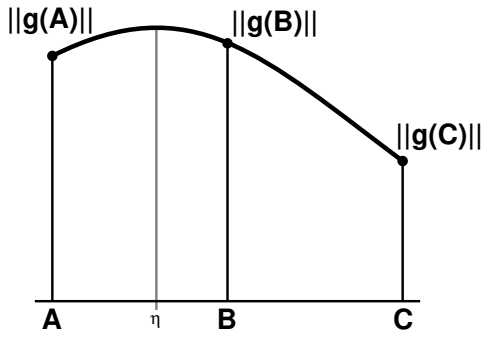


Figure 2: Profile of the norm of the image gradient along the direction of the image gradient (roughly orthogonal to the edge).

Figure 2 shows an example of the profile norm of the image gradient along the direction orthogonal to the edge. Canny's method keeps pixel B as an edge pixel because $||g(B)|| > ||g(A)||$ and $||g(B)|| > ||g(C)||$. However, there may be an intermediate position η between A and C where the norm of the image gradient is larger than in those points. Point η would correspond better to the position of the edge. Devernay proposed a method to estimate that position and then compute an offset vector, along the direction n, to give the sub-pixel position of the edge point near the edge pixel B.

To limit the computational cost, Devernay proposed to use a simple quadratic interpolation of the gradient norm along the three points used in the Canny operator: $(A, ||g(A)||)$, $(B, ||g(B)||)$ and $(C, ||g(C)||)$, see Figure 2. Solving the maximum leads to an offset of

$$\eta = \frac{1}{2} \frac{||g(A)|| - ||g(C)||}{||g(A)|| + ||g(C)|| - 2||g(B)||} \tag{5}$$

relative to the vector $\overrightarrow{BC}$.

The beauty of Devernay's correction term is that an excellent sub-pixel accuracy is obtained by a simple and cheap computation involving values already obtained by the Canny operator. The accuracy obtained will be analyzed in the next section and a small improvement is proposed in Section 5.

4 Devernay's Sub-Pixel Accuracy

There are four sources of error in the Devernay sub-pixel correction:

1. error due to the assumption of a parabolic gradient profile;

2. error due to the computation of the gradient by a finite scheme;

3. error due to the interpolation of the gradient magnitude at off-the-grid points by the Canny scheme;

4. numerical error.

We will analyze these errors for *Gaussian edges*, which have a profile given by $\alpha \cdot G_\sigma \star Y$, where G_σ is a Gaussian kernel of standard deviation σ, Y is the step function defined as

$$Y(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{6}$$

and α is the step size of the edge in gray level values. Figure 3-left shows a Gaussian edge profile with a sub-pixel offset γ relative to the grid. The Devernay scheme computes the position of the edge using the parabolic approximation as shown in Figure 3-right. A quadratic interpolation between points $(-1, a)$, $(0, b)$ and $(1, c)$ leads to a maximum of the parabola located at point $\eta = \frac{1}{2}\frac{a-c}{a+c-2b}$. The derivative of the Gaussian edge is $\alpha \cdot G_\sigma$ so the values to be used in the Devernay method are

$$a = \alpha \cdot G_\sigma(-1-\gamma), \quad b = \alpha \cdot G_\sigma(-\gamma), \quad c = \alpha \cdot G_\sigma(1-\gamma), \tag{7}$$

which leads to the following estimation of the sub-pixel offset

$$\eta(\sigma, \gamma) = \frac{1}{2}\frac{G_\sigma(-1-\gamma) - G_\sigma(1-\gamma)}{G_\sigma(-1-\gamma) + G_\sigma(1-\gamma) - 2G_\sigma(-\gamma)}. \tag{8}$$

Note that the edge step size α is canceled as it appears multiplying both the numerator and the denominator, so the error is independent of the edge step. The error of approximating the Gaussian gradient by a parabolic gradient is thus given by

$$\rho(\sigma, \gamma) = |\eta(\sigma, \gamma) - \gamma|. \tag{9}$$

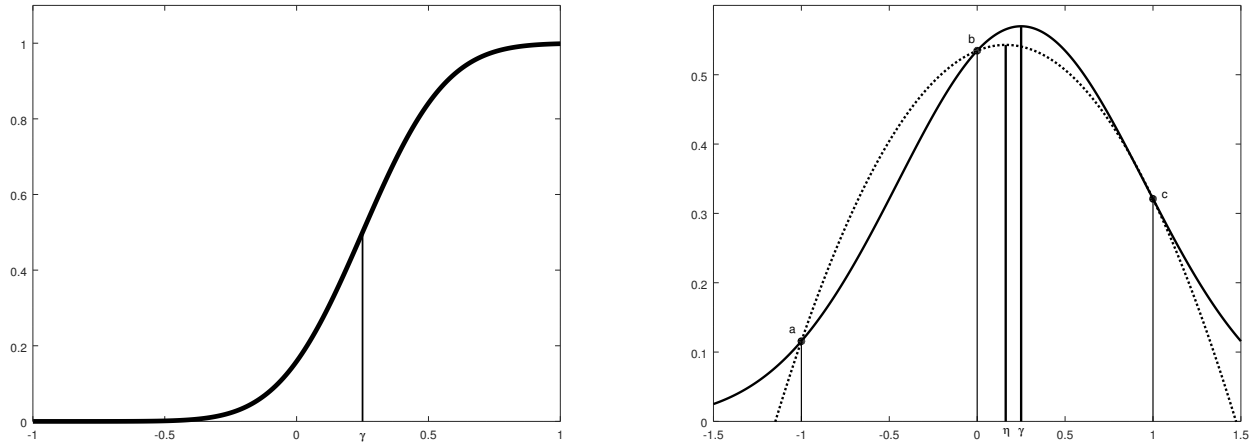


Figure 3: **Left:** A Gaussian edge profile with a sub-pixel offset γ relative to the grid. **Right:** Error in the Devernay sub-pixel correction term due to the assumption of a parabolic gradient profile (dotted line) instead of the true one (solid line), in this case a Gaussian. The error is given by $|\eta - \gamma|$, the difference between the true sub-pixel offset and the one estimated using the Devernay method.

Figure 4 shows a plot of this theoretical error (using the exact Gaussian function) as a function of σ and γ. As expected, the error is zero when the edge is centered in the pixel grid ($\gamma = 0$). The error is also zero when the edge is centered exactly in the midpoint of the grid ($\gamma = 0.5$), which corresponds to $b = c$ (or $a = b$), and the parabolic interpolation gives the right offset. The error is larger for offset values between zero and 0.5 and decreases as σ grows, as expected, because the approximation should work well for very blurred edges. Sharp edges can be modeled by Gaussian edges with σ values between 0.5 and one.

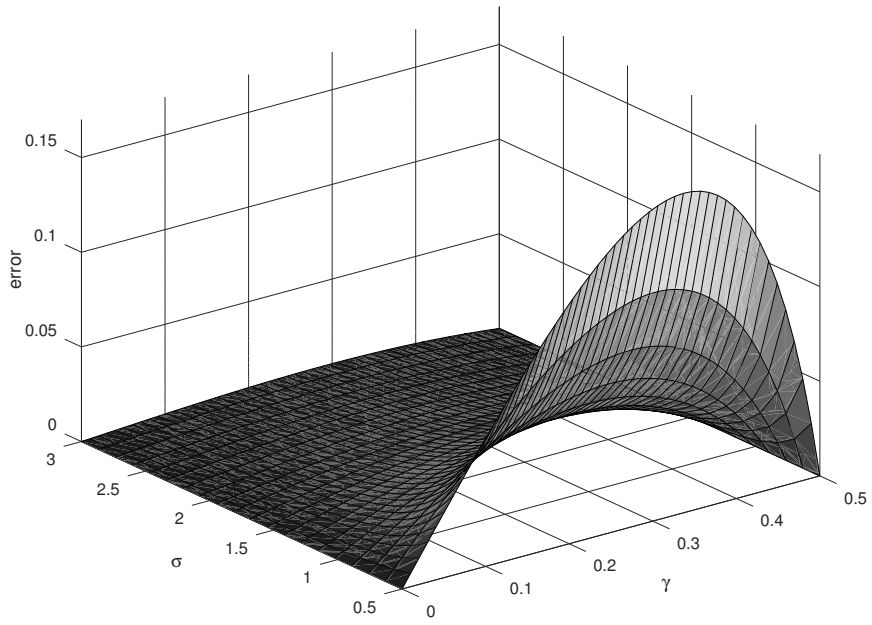


Figure 4: Error of the parabolic approximation of a Gaussian edge as a function of the standard deviation σ of the edge and the sub-pixel offset γ relative to the grid. All axes are given in pixel units.

Equation (8) also allows us to consider the second kind of error, the one due to the use of a centered difference scheme to compute the derivatives. For that, the values $G_\sigma(-1-\gamma)$, $G_\sigma(-\gamma)$ and $G_\sigma(1-\gamma)$ must be replaced by the absolute value of the finite difference of the Gaussian edge values as would be done by the centered difference scheme. An empirical evaluation of Devernay's method can be performed by synthesizing images containing a Gaussian edge. Using floating point variables for the pixel values, the results of this evaluation are accurate up to the floating point machine precision.

A first set of experiments were performed with synthesized horizontal or vertical edges. In this case, horizontal or vertical, there is no need for Canny's scheme to interpolate the gradient magnitude in off-the-grid points; we are thus avoiding the third kind of error, to be analyzed later. Figure 5 shows the empirical error (black dots) as a function of the offset γ for Gaussian edges with two different blurs, $\sigma = 0.5$ and $\sigma = 1$. In both cases the black line corresponds to the theoretical error ρ computed using Equation (9). The gray line is the same computation but using a centered difference scheme to compute the derivative instead of using the exact value from the Gaussian function. Neglecting the numerical error, the gray line provides an accurate prediction of the empirical error (black dots). Notice that the empirical error is smaller than the theoretical parabolic approximation error (black line); this counter-intuitive fact is due to finite difference errors partially compensating the error of the parabolic approximation on Gaussian edges, see Figure 6 for an illustration. Figure 7 shows the same plot as in Figure 5 but for $\sigma = 10$; as the edge is more blurred and the transition more smooth, the finite scheme gives a better approximation of the derivative and both, the theoretical and the empirical error are very similar (the gray line is almost invisible behind the black line). Note also that the error is much smaller than the one obtained for less blurred images, as shown in Figure 5.

For the second set of experiments the edges are no longer restricted to be horizontal or vertical. The interesting range of angles is from zero to 45 degrees (other angles are equivalent due to symmetry). Figure 8 shows the empirical error found for Gaussian edges with $\sigma = 0.5$ and $\sigma = 1$ as a function of the edge angle and of the sub-pixel offset of the edge. When the edge angle is zero we

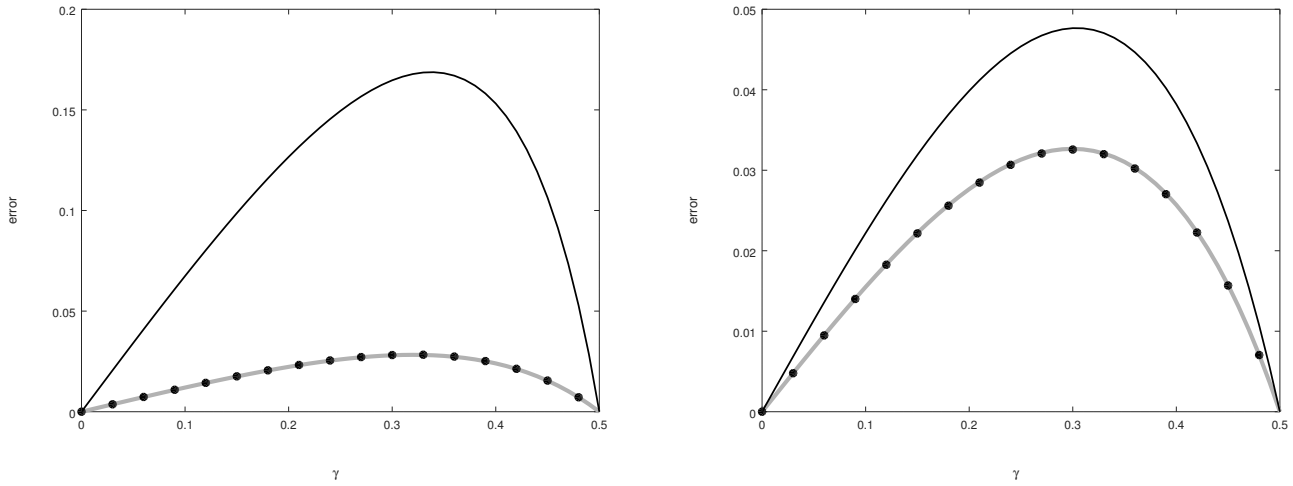


Figure 5: Theoretical error of the Devernay scheme for a Gaussian (black line), the theoretical error when considering the centered difference scheme for computing the derivatives (gray line), and the empirical error (black dots) when applying the Devernay method on synthesized images of horizontal Gaussian edges. Each plot shows the error as a function of the sub-pixel offset γ for fixed standard deviations of $\sigma = 0.5$ **(left)** and $\sigma = 1$ **(right)**. Both axes are in pixel units. Note that the error axis scale is different for each plot.

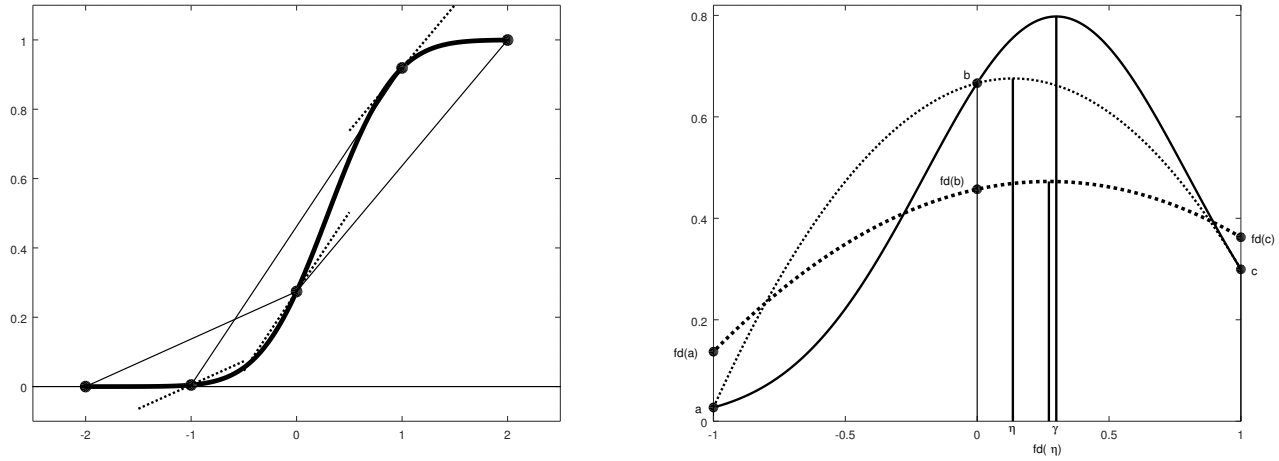


Figure 6: Partial compensation of errors in a Gaussian edge for $\sigma = 0.5$ and $\gamma = 0.3$. **Left:** The Gaussian edge profile (thick line) and the finite difference estimations of the derivatives (dotted lines) by centered difference (thin lines). The finite difference estimations of the derivative are larger that the actual ones at positions -1 and 1; the estimation gives a smaller value at 0. **Right:** The exact gradient profile (solid line) with its maximum at γ; the parabolic approximation using the exact derivative values a, b and c (thin dotted lines) with its maximum at η; the parabolic approximation using the derivatives $fd(a)$, $fd(b)$ and $fd(c)$ estimated by finite differences (thick dotted line) with its maximum at $fd(\eta)$. Note that the errors due to the finite difference estimation of the gradient compensate in part the error of the parabolic approximation, reducing the final error.

obtain exactly the same curves as before (the gray lines in Figure 5). Then, as the edge angle increases, the error increases as well. The extra factor is due to the Canny interpolation of off-the-grid values. As can be seen, for slanted angles the main source of error is Canny's interpolation, which can lead to errors about five times larger for Gaussian edges of $\sigma = 1$.

The error in the Devernay sub-pixel approximation determines the accuracy of the resulting edge points. There is also a particular configuration that leads to "oscillatory" artifacts as illustrated in Figure 9. A synthesized Gaussian edge at 45 degrees is shown in Figure 9A, where $\sigma = 1$ and the offset is $\gamma = \frac{\sqrt{2}}{4}$. Figure 9B shows a detail of the same image and the sub-pixel edges computed

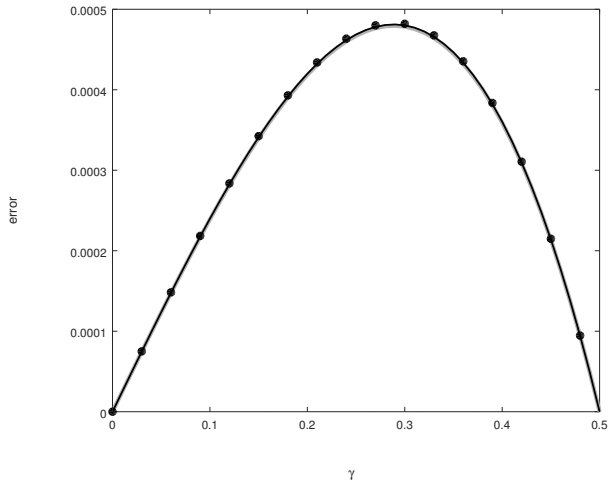


Figure 7: Theoretical error of the Devernay scheme for a Gaussian (black line) with $\sigma = 10$, the theoretical error when considering the centered difference scheme for computing the derivatives (gray line, behind the black line and almost invisible), and the empirical error (black dots) when applying the Devernay method on synthesized images of horizontal Gaussian edges. Both axes are in pixel units.

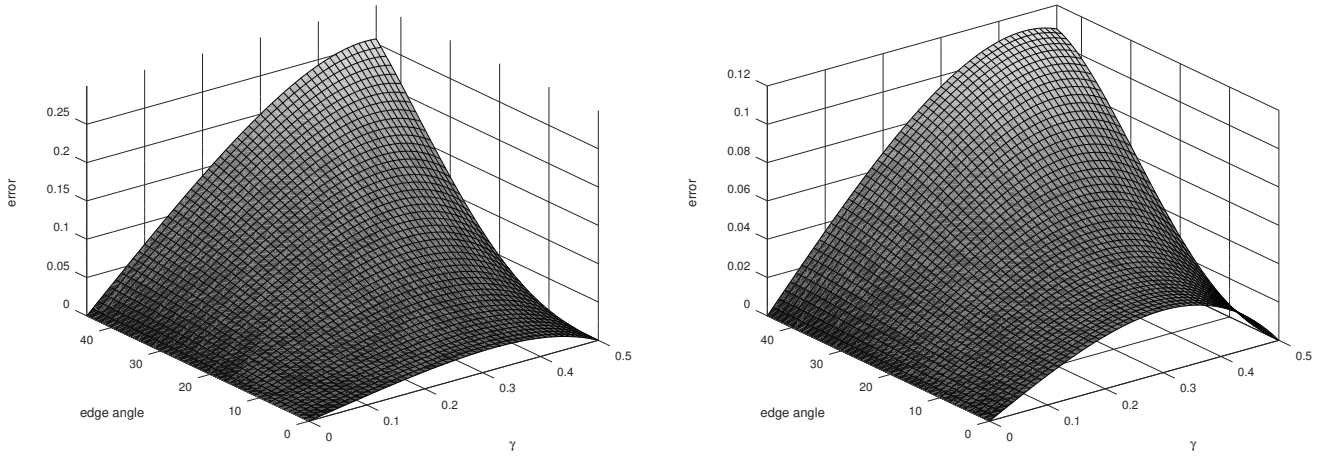


Figure 8: Empirical error of the Devernay sub-pixel method for Gaussian edges as a function of the edge angle and the sub-pixel offset of the edge relative to the grid. The angle zero corresponds to a horizontal (or vertical edge) while an angle of 45 degrees corresponds to the diagonal of the image. Each plot shows the error for a fixed standard deviation of $\sigma = 0.5$ **(left)** and $\sigma = 1$ **(right)**. The error and sub-pixel offset axis are in pixel units; the edge angle axis are in degrees. Note that the error axis scale is different for each plot.

with Devernay's method. An oscillatory pattern is observed. For these particular values, the true edge goes exactly midway between two sets of diagonal positions in the pixel grid. The symmetry of the configuration makes that pixels on both sides of the true edge are local maxima and the sub-pixel errors of the Devernay method are also symmetric, thus creating alternative offsets in opposite directions, see Figure 9C. The result is the oscillatory artifact which appears when similar conditions happen on an image. Figure 9D shows a detail from Devernay's original work, confirming that this problem can happen on real images and that was present in the original implementation of the method.

The magnitude of the oscillatory artifacts depends on the contour angle and shift relative to the pixel grid. In the worst case, the amplitude of the oscillations (from lower to upper envelope) is twice the positioning error, see Figure 9C. Figure 8 shows that these values can go up to a quarter of a

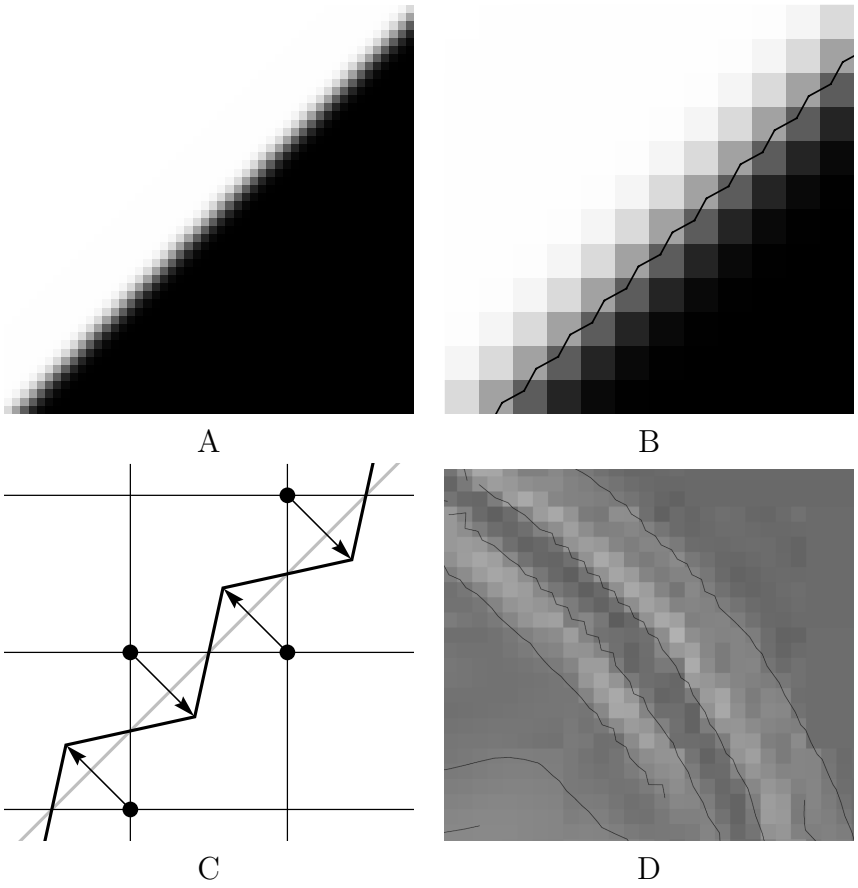


Figure 9: **A** Synthesized Gaussian edge at 45 degrees, with $\sigma = 1$, and an offset of $\gamma = \frac{\sqrt{2}}{4}$. **B** A detail of the synthesized edge in A and the detected edges by the Devernay method. An oscillatory pattern is observed. **C** An scheme explaining the reason of the oscillatory pattern. The true edge is located exactly at mid distance between the two sets of diagonal grid points. By symmetry, all the points marked by dots are local maxima. The arrows show the offsets η computed by the Devernay method. The errors appear in alternate directions, creating the observed oscillations. **D** Detail of the upper-right side of Figure 10 of the original work by Devernay [5], showing that the "oscillatory" patterns were indeed present in the original implementation of the method.

pixel, leading to oscillations of about half a pixel amplitude.

5 A Modified Sub-pixel Scheme

Among the four factors contributing to the error of the Devernay scheme, there is one that can be eliminated by a simple modification of the method. Performing the interpolation always along a vertical or horizontal line, the need to use gradient values in off-the-grid positions disappears and the Canny interpolation is no longer needed.

The Devernay scheme requires that the central point being refined be a local maximum among the three values used. Thus, we must modify the algorithm in order to select the edge points and the direction of interpolation so as to satisfy these conditions. A pixel (x, y) will become a *horizontal* edge point when it is a horizontal local maximum of the gradient modulus ($||g(x-1,y)|| < ||g(x,y)||$ and $||g(x,y)|| > ||g(x+1,y)||$) and the image gradient is more horizontal than vertical at that point ($|g_x(x,y)| > |g_y(x,y)|$). Analogously, a pixel (x, y) will become a *vertical* edge point when it is a vertical local maximum of the gradient modulus ($||g(x,y-1)|| < ||g(x,y)||$ and $||g(x,y)|| > ||g(x,y+1)||$) and the image gradient is more vertical than horizontal at that point ($|g_x(x,y)| < |g_y(x,y)|$). If both conditions are satisfied (which implies $|g_x(x,y)| = |g_y(x,y)|$) we choose to classify it as a

horizontal edge point; this is arbitrary, the opposite convention would be equally good.

The empirical error of the modified scheme can be evaluated in the same way as before. Figure 10 shows the empirical error of the modified algorithm for Gaussian edges; the setting is the same as the one used for Figure 8. As can be seen, the maximal error is noticeable smaller. Figure 11 shows the resulting detection for the same slanted edge as in Figure 9; the oscillatory pattern is no longer observed. The same phenomena can be observed on natural images, as illustrated in Figure 12.

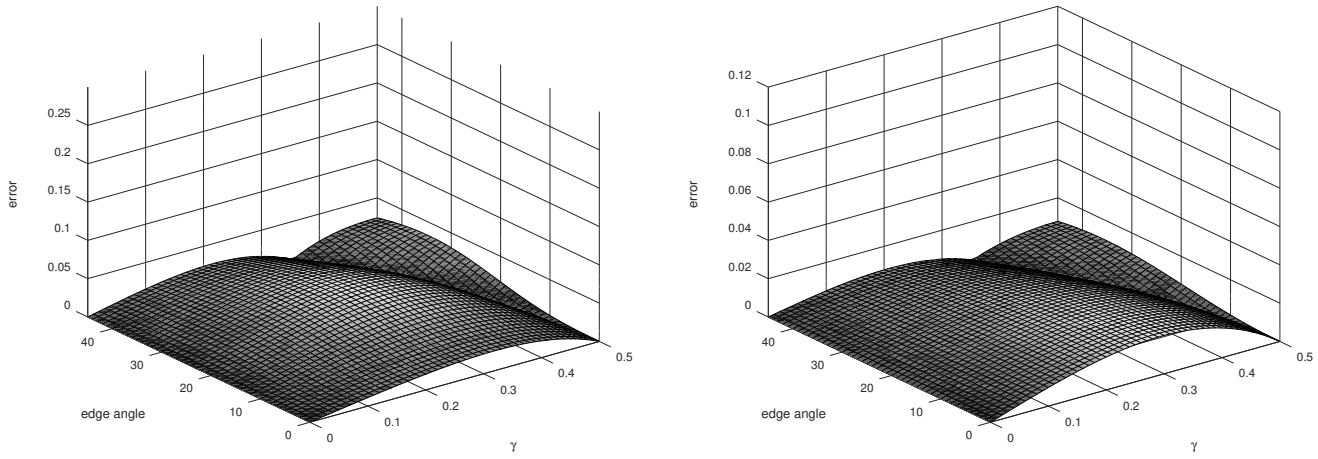


Figure 10: Empirical error for the modified Canny/Devernay sub-pixel method applied to a Gaussian edge as a function of the edge angle and the sub-pixel offset of the edge relative to the grid. These plots should be compared to the ones in Figure 8 corresponding to the original method. The angle zero corresponds to a horizontal (or vertical edge) while an angle of 45 degrees corresponds to the diagonal of the image. Each plot shows the error for a fixed standard deviation of $\sigma = 0.5$ **(left)** and $\sigma = 1$ **(right)**. Error and offset axis are in pixel units. Note that the error axes scale is different for each plot.

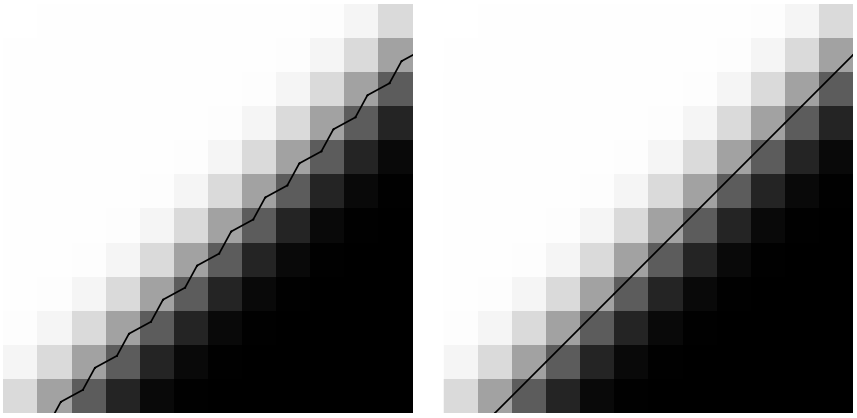


Figure 11: **Left:** The same oscillatory pattern shown in Figure 9B. **Right:** The result of the modified Canny/Devernay algorithm. Note that the oscillatory artifact is no longer present.

6 Edge Point Chaining

Each edge point is computed independently from the others and the ones that belong to the same edge need to be chained to form the curves corresponding to the contours of the image. Because the Devernay scheme provides sub-pixel accuracy for the points, a simple procedure can be used for the chaining: connect an edge point to the nearest other edge points not farther than a certain tolerance.

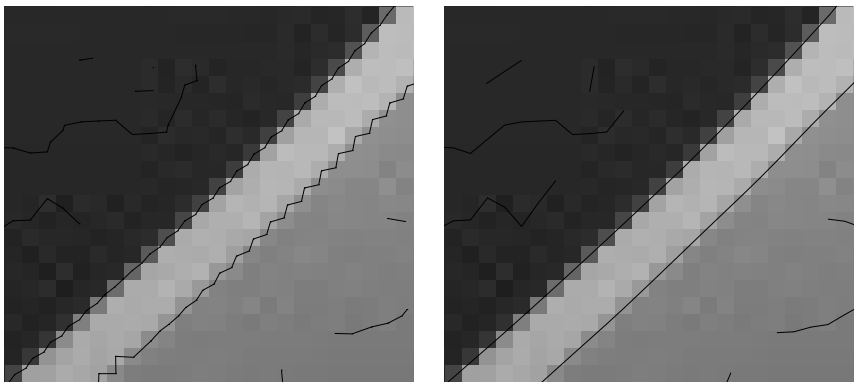


Figure 12: The oscillatory pattern observed on part of a natural image. **Left:** The oscillatory pattern produced by the original formulation of the Devernay sub-pixel correction term. **Right:** The result of the modified Canny/Devernay algorithm. Note that the oscillation disappears.

Because each edge point is associated to a pixel, the search for edge points to chain is reduced to the pixels in a 5×5 neighborhood. This corresponds to extending one step the classic 8-connected pixels. Most edge points are chained to another one belonging to the 8-connected neighborhood. But when noise is present and curves become fragmentary, the extended connectivity helps to extract the right curves.

The first verification that is performed is that edge points to be chained have a similar gradient orientation. To make a chaining from A to B, it is required that the angle between $g(A)$ and $g(B)$ be less than 90 degree. This is verified by the condition $g(A) \cdot g(B) > 0$, where $\cdot$ is the dot product.

An image contour separates a light region from a darker one. The chaining needs to be coherent in the sense that consecutive chains must leave the darker region to the same side of the curve. A simple way of imposing this is to verify that the vector from edge point A to edge point B to be chained is roughly orthogonal to the image gradient in A in one of the two possible directions. As a convention, we will call a *forward* chaining from edge point A to edge point B one in which $\overrightarrow{AB} \cdot g(A)^{\perp} > 0$, where $\overrightarrow{v}^{\perp}$ corresponds to vector $\overrightarrow{v}$ rotated 90 degrees. (The sense of rotation defines the convention, any of the two can be used and are equivalent, but one of them needs to be fixed.) We will call *backward* chaining the reverse direction. We will use the notation $A \rightharpoonup B$ to indicate a forward chaining from A to B. Of course, the same link can be described as backward chaining from B to A.

In this simple setting, each edge point can have at most one forward chaining and at most one backward chaining. Configurations with a shape of a letter Y are thus forbidden. Before chaining edge point A to edge point B, one needs to verify if A has a previously assigned forward chaining and B has a previous backward chaining. In any of those cases, the distance of the proposed chaining is compared to the previous one; if the distance of the proposed chaining is shorter, the previous chaining is cut and the proposed one is added. After that, the corresponding modifications in the linking between edge points are done.

There is an undesirable aspect of this procedure: the result is not independent of the order of exploration. Consider the configuration shown in Figure 13. The distances between points A, B, C and D satisfy $||A - B|| > ||B - C|| > ||C - D||$. Let us consider two possible orders of exploration. If the chaining is evaluated in the following order: A then B then C, we will first chain $A \rightharpoonup B$ when exploring edge point A. Then, when analyzing the backward links of B, we will prefer $C \rightharpoonup B$, and $A \rightharpoonup B$ will be unlinked. Finally, when exploring C, $C \rightharpoonup D$ will be preferred and $C \rightharpoonup B$ will be unlinked. The result is just the link $C \rightharpoonup D$.

If the exploration order is: C then B then A, we will first chain $C \rightharpoonup D$ when exploring C. Then, when exploring the backward connections of B, $C \rightharpoonup B$ will be the preferred link; but because

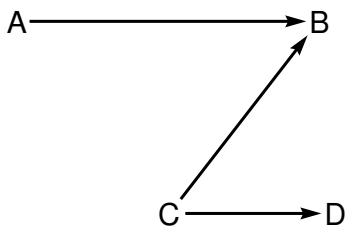


Figure 13: A configuration of edge points which may lead to different chaining results depending on the order of exploration. The potential chains among points A, B, C and D are drawn by arrows. The distances satisfy $||A-B|| > ||B-C|| > ||C-D||$. The resulting chaining when the order of exploration is A then B then C will differ from the result when the order of exploration is C then B then A. See the main text.

$C \rightharpoonup D$ were already linked and with a shorter distance, $C \rightharpoonup B$ cannot be linked. Finally, when exploring A, the link $A \rightharpoonup B$ will be created because there is no better backward linking of B. The result are two links: $C \rightharpoonup D$ and $A \rightharpoonup B$.

We did not find yet a simple algorithm to solve this problem. By simple we mean an algorithm without two passes or the need to re-evaluate the chaining of points when links are cut. For most edge points, there is only one possible chaining and this problem does not arise. Nevertheless, this dependence on the order of exploration does happen occasionally and a better solution would be desirable.

7 Algorithm

Algorithm 1 provides a general description of the Canny/Devernay edge detector in four steps: compute the image gradient, compute the edge points, chain the edge points and apply the Canny threshold with hysteresis. A detailed pseudo-code specifies each of the four steps. Overall, the method takes as input an image represented as a matrix of gray-level values at each position. Three parameters are required, the standard deviation of the Gaussian filter S used to blur the input image and the two thresholds on the image gradient modulus L and H. The result of the method is a list $\mathcal{E}$ of chained edge points, provided with sub-pixel accuracy.

The computation of the image gradient is described in Algorithm 2. There are two steps: blurring the image by a convolution with a Gaussian filter of standard deviation S and computing the x and y components of the image gradient by centered differences.

There are many algorithms to compute the Gaussian convolution [6]. Anyone can be used provided it gives a good approximation to the theoretical result. Also, the two operations can be combined into efficient operators which perform the convolution and the differentiation at the same time [4]. In our implementation, the filtering is computed by separated horizontal and vertical convolutions using a FIR filter to approximate the 1D Gaussian kernel.

Algorithm 3 is the core of the method and includes the modified criterion to determine the edge

Algorithm 1: Canny/Devernay edge detector

input: An image I, a scale parameter S, and two threshold values H and L
output: A list $\mathcal{E}$ of sub-pixel, chained edge points

```
1 g ← image_gradient(I, S)                          /* Algorithm 2 */
2 E ← compute_edge_points(g)                        /* Algorithm 3 */
3 chain_edge_points(E, g)                           /* Algorithm 4 */
4 thresholds_with_hysteresis(E, g, H, L)            /* Algorithm 5 */
```

Algorithm 2: `image_gradient`(I, S)

input: An image I, a scale parameter S
output: The image gradient vector field g

$I_S \leftarrow I \star K_S$ `/* convolution with a Gaussian kernel */`
for $(x, y) \in I_S$ **do**
  $g_x(x, y) \leftarrow I_S(x + 1, y) - I_S(x - 1, y)$
  $g_y(x, y) \leftarrow I_S(x, y + 1) - I_S(x, y - 1)$
$g \leftarrow (g_x, g_y)$

Algorithm 3: `compute_edge_points`(g)

input: The image gradient vector field g
output: A list $\mathcal{E}$ of sub-pixel edge points

$\mathcal{E} \leftarrow \varnothing$
for $(x, y) \in g$ **do**
  $\theta_x \leftarrow 0$
  $\theta_y \leftarrow 0$
  if $||g(x-1,y)|| < ||g(x,y)|| \geq ||g(x+1,y)||$ **and** $|g_x(x,y)| \geq |g_y(x,y)|$ **then**
    $\theta_x \leftarrow 1$ `/* edge point to be interpolated horizontally */`
  else if $||g(x,y-1)|| < ||g(x,y)|| \geq ||g(x,y+1)||$ **and** $|g_x(x,y)| \leq |g_y(x,y)|$ **then**
    $\theta_y \leftarrow 1$ `/* edge point to be interpolated vertically */`
  if $\theta_x \neq 0$ **or** $\theta_y \neq 0$ **then**
    $a \leftarrow ||g(x - \theta_x, y - \theta_y)||$ `/* Devernay sub-pixel correction */`
    $b \leftarrow ||g(x, y)||$
    $c \leftarrow ||g(x + \theta_x, y + \theta_y)||$
    $\lambda \leftarrow \frac{a-c}{2(a-2b+c)}$
    $e_x \leftarrow x + \lambda\theta_x$
    $e_y \leftarrow y + \lambda\theta_y$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(e_x, e_y)\}$ `/* store edge point */`

points and the sub-pixel correction term. When the image gradient modulus at a given pixel is larger than the gradient modulus to the left and to the right, the pixel is dubbed an horizontal edge point provided the g_x component of the image gradient is larger or equal than the g_y component. Analogously, a pixel whose gradient modulus is larger than the gradient at the pixels above and below is dubbed a vertical edge point if the g_y component of the image gradient is larger or equal than the g_x component.

It can happen that two neighbor pixels have equal image gradient modulus and are both maxima. This can happen, for example, when the edge is exactly between the two pixels. In such cases, as an arbitrary convention, the edge is marked on the left pixel for a vertical edge or on the bottom one for horizontal edges. (Bottom pixel here means that the y value of the pixel is smaller; it is usual, however, to set the y axis of the image such that the upper-left pixel has coordinates $(0, 0)$ and y values grow as we go down the image. On this convention, the edge is marked *upper* and not bottom). This is obtained by the asymmetrical use of $<$ and $\geq$ comparisons for deciding that a pixel is horizontal or vertical maximum of the image gradient.

Finally, the Devernay scheme is used (but always along the vertical or horizontal axis) to compute the sub-pixel position of the edge points. Edge points are added to the output list.

Algorithm 4: `chain_edge_points`$(\mathcal{E}, g)$

input: The image gradient vector field g and a list $\mathcal{E}$ of sub-pixel edge points
output: Modified list $\mathcal{E}$

foreach $e \in \mathcal{E}$ **do**
  $\mathcal{N}_F \leftarrow \{n \in \mathcal{N}_{\mathcal{E}}(e,2) : \ g(e) \cdot g(n) > 0$ **and** $\overrightarrow{en} \cdot g(e)^{\perp} > 0\}$ `/* forward neighbors */`
  $\mathcal{N}_B \leftarrow \{n \in \mathcal{N}_{\mathcal{E}}(e,2) : \ g(e) \cdot g(n) > 0$ **and** $\overrightarrow{en} \cdot g(e)^{\perp} < 0\}$ `/* backward neighbors */`
  $f \leftarrow \operatorname{argmin}_{n \in \mathcal{N}_F} \operatorname{dist}(e,n)$ `/* best forward candidate */`
  $b \leftarrow \operatorname{argmin}_{n \in \mathcal{N}_B} \operatorname{dist}(e,n)$ `/* best backward candidate */`
  if $\varnothing \rightharpoonup f$ **or** $\Big(a \rightharpoonup f$ **and** $\operatorname{dist}(e,f) < \operatorname{dist}(a,f) \Big)$ **then**
    unlink $\star \rightharpoonup f$, if linked
    unlink $e \rightharpoonup \star$, if linked
    link $e \rightharpoonup f$
  if $b \rightharpoonup \varnothing$ **or** $\Big(b \rightharpoonup a$ **and** $\operatorname{dist}(b,e) < \operatorname{dist}(b,a) \Big)$ **then**
    unlink $b \rightharpoonup \star$, if linked
    unlink $\star \rightharpoonup e$, if linked
    link $b \rightharpoonup e$

The chaining procedure is detailed in Algorithm 4. Each edge point e from the list $\mathcal{E}$ is evaluated. In each case, the set $\mathcal{N}_{\mathcal{E}}(e,2)$ of neighbors of the edge point e is computed. Edge points are initially associated to the pixel that is a local maximum of the gradient modulus, either vertically or horizontally. The set of neighbors is defined as the edge points associated to pixels in the 5×5 neighborhood; in other words, the pixels whose maximal coordinate difference is not greater than 2. The number 2 in the notation of the set indicates this distance of the neighborhood.

Two subsets of the set $\mathcal{N}_{\mathcal{E}}(e,2)$ are formed. The first one, $\mathcal{N}_F$, includes the edge points n satisfying two conditions: $g(e) \cdot g(n) > 0$ and $\overrightarrow{en} \cdot g(e)^{\perp} > 0$; those edge points are the candidates for forward chaining. In the second subset, $\mathcal{N}_B$, the edge points satisfy conditions $g(e) \cdot g(n) > 0$ and $\overrightarrow{en} \cdot g(e)^{\perp} < 0$, meaning that those edge points are candidates for backward chaining. The elements of each of those subsets with shorter distance to the edge point e are selected as the candidates for forward and backward chaining, respectively. The last eight steps of the algorithm are just the verification of previous chaining, unlinking previous chains if the new ones are better, and creating the new links.

To complete the description of the methods, Algorithm 5 details the threshold with hysteresis step. A flag is defined for each edge point indicating whether the pixel is already validated or not. These flags are all initially set to a *not-valid* condition. Then, the list of edge points is iterated and each *not-valid* point with image gradient modulus greater or equal to the high threshold H is set as *valid*. Then, starting from that validated point, the curve is iterated along the chains of edge points, first in the forward direction, then in the backward direction, while the chained points are *not-valid* and their gradient modulus is larger than the low threshold L. On each step of the process, the new points found satisfying the L condition are also set as *valid*. When all curves have been iterated, starting from each edge point in the list, the points that still remain as *not-valid* are unlinked and removed from the output list $\mathcal{E}$.

Algorithm 5: `thresholds_with_hysteresis`$(\mathcal{E}, g, H, L)$

input: A list $\mathcal{E}$ of chained edge points, the image gradient vector field g and two threshold values H and L
output: Modified list $\mathcal{E}$

foreach $e \in \mathcal{E}$ **do**
  set e as `not valid`
foreach $e \in \mathcal{E}$ **and** e *is* `not valid` **and** $||g(e)|| \geq H$ **do**
  set e as `valid`
  $f \leftarrow e$ `/* iterate the chain forwardly while over` L `*/`
  while $f \rightharpoonup n$ **and** n *is* `not valid` **and** $||g(n)|| \geq L$ **do**
    set n as `valid`
    $f \leftarrow n$
  $b \leftarrow e$ `/* iterate the chain backwardly while over` L `*/`
  while $n \rightharpoonup b$ **and** n *is* `not valid` **and** $||g(n)|| \geq L$ **do**
    set n as `valid`
    $b \leftarrow n$
foreach $e \in \mathcal{E}$ **and** e *is* `not valid` **do** `/* unlink and remove non validated points */`
  unlink $e \rightharpoonup \star$, if linked
  unlink $\star \rightharpoonup e$, if linked
  remove e from $\mathcal{E}$

8 Computational Complexity

The four main steps of the method are listed in Algorithm 1. The first step consists in computing the image gradient (Algorithm 2). The image is first blurred by Gaussian filtering and then the gradient is computed at each pixel by finite differences. Both operations require a number of operations proportional to the number of pixels. The same is true for the computation of edge points (Algorithm 3), in which the gradient conditions are evaluated for each pixel of the image and the Devernay correction is performed in some of them.

The chaining step (Algorithm 4) evaluates a fixed neighborhood of each edge point. The set of edge points is strictly smaller than the set of pixels. Thus, the complexity of this step remains bounded by the number of pixels.

The final step of thresholding (Algorithm 5) also iterates the set of edge points. Edge points with gradient over the high threshold are followed along the curve. In the main loop, a pixel is visited at most three times: as part of the list of edge points; when following the curve of its backward chained point; when following the curve of its forward chained point. So again, the computation is proportional to the number of edge points and bounded by the number of pixels on the image.

All in all, the computational complexity of the whole method is proportional to the number of pixels

$$\text{Complexity} = O(p) \tag{10}$$

where p is the number of pixels in the image. On current computers (2017) the application of the full algorithm to a 512×512 image takes a fraction of a second.

9 Experiments

This section illustrates the behavior of the method with some experiments, showing the impact of each of the three parameters. In all the examples the images have pixel values between 0 and 255. As a convention (also used in the code associated with this document), a standard deviation value of $S = 0$ indicates that no Gaussian filtering is performed.

The choice of the standard deviation S of the Gaussian filtering implies a compromise between robustness to noise and fidelity to the contours on the image. Figure 14 shows the result of the method for S values of 0, 1 and 2. When noise is not present, the resulting curves will follow the image contours when no filtering is applied ($S = 0$), see the first row example. On the other hand, when filtering is applied (and the more filtering, the larger the value of S) the curves are deformed, as can be noted in the rounded corners of the squares.

The second image is contaminated by a strong noise. The computed edge points follow a random pattern (note that the gradient thresholds were set to zero) and only small fractions of the contour were detected with $S = 0$ (no Gaussian filtering). When the filtering is used, the contour is detected correctly.

The third example shows that the filtering can also result in a deformed output. The image corresponds to a thin line, actually a one pixel width line. When no filtering is used, the two contours correspond to the expected result. When filtering is used, however, the two contours interact, resulting in shifted detections. This is a synthetic and aliased image, but the same effect can be observed in natural images, see the two diagonal lines on the center-top of the fourth example in Figure 14. These examples show that the choice of S is crucial and the best value depends on each particular image.

Figure 15 shows the same effects on a synthetic and noisy image. When no Gaussian filtering is used, the resulting curves are incomplete and fragmented. The algorithm extracts the right curves when using the Gaussian filtering. A similar behavior is observed in Figure 16, where the method is applied to a natural image. In this case the noise is small and an acceptable result is obtained even without filtering. Note how the details of the curves become rounded for large S values.

For the experiments in Figure 17, both thresholds L and H were set to the same value to illustrate the output of the detection without using Canny's hysteresis. A well-chosen value can be effective to remove edge curves due to noise. There is, however, no universal value. The value effective in the previous case is too restrictive for the example shown in Figure 18.

Canny's hysteresis was designed to help handling this problem. Using a high threshold allows the algorithm to spot parts of the main contours. But a high threshold prevents detections on parts of the contours where the contrast is less strong. The result is improved by allowing detections on points with a lower contrast provided that they are connected to strong contours. Figures 19 and 20 show some examples. The use of hysteresis makes the method more robust to the particular values of the threshold parameters. Nevertheless, their choice remains an issue with no universal rule.

Figure 21 shows more example results on natural images. The parameters of the method were fixed $S = 1$, $L = 5$ and $H = 15$. In the two first examples the result is good. Nevertheless, one can see how the texture of rocks and sand of the shore in the second example produces noisy edges. The third example has a strong camera noise due to low light conditions, resulting in spurious detections, noisy curves and curve fragmentation.

We will end this section with a comment concerning the sub-pixel nature of the result. The output of the method is a list of curves defined as chained edge *points*. Note that this is not the same thing as chained edge *pixels*. The sub-pixel accuracy of the edge points turns the resulting polygonal curve into a vectorization of the contours. The resulting curves have much more resolution than the image. Figure 22 illustrates the point. The line width used for drawing is arbitrary. Thus, by drawing with a thin line width one can see how the curve is much more detailed than the pixel

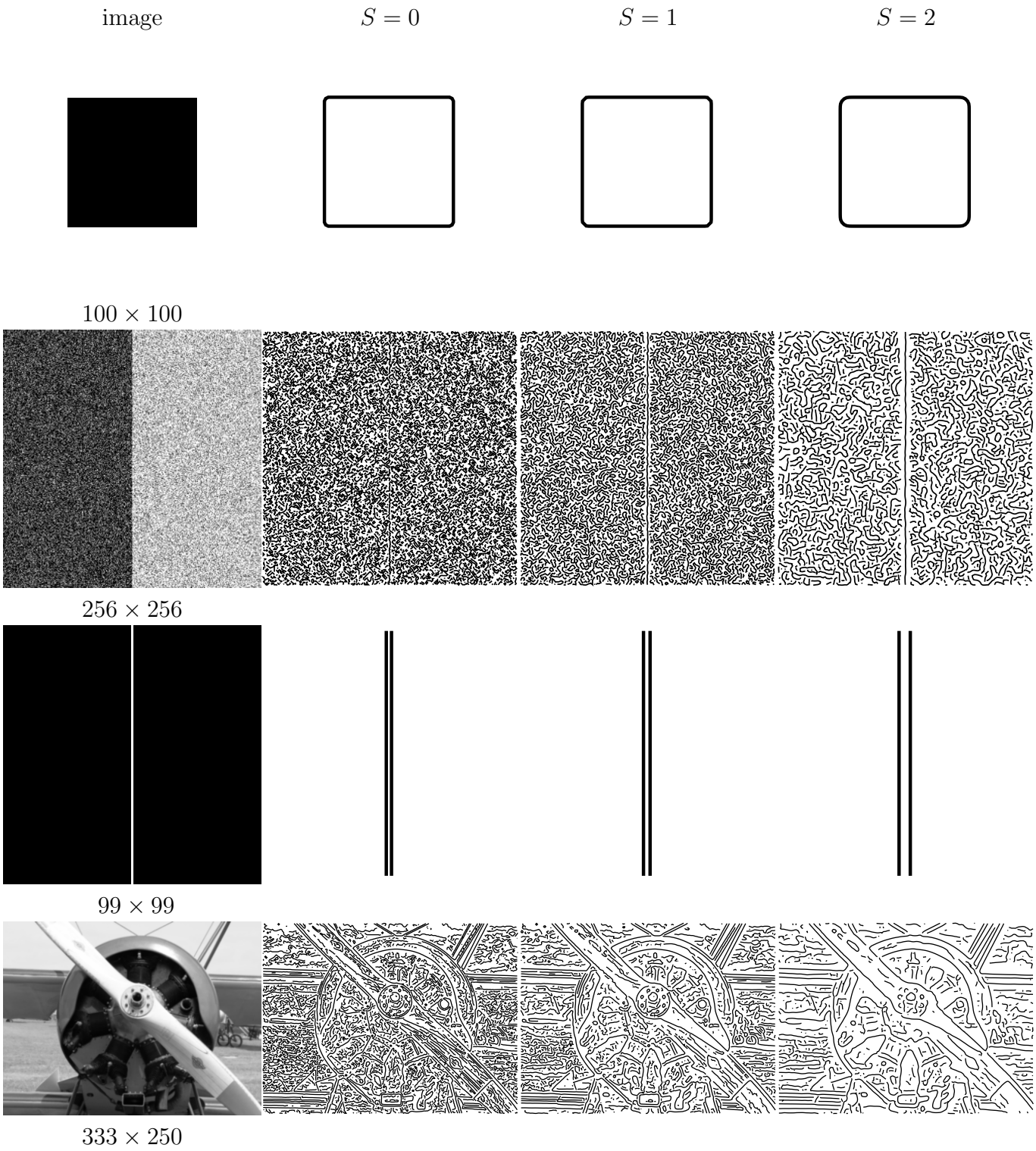


Figure 14: Effect of the Gaussian filtering. The first example (**first row**) shows the rounding of corners. The second example (**second row**) shows how the filtering helps handling image noise. The third example (**third row**) shows how edge points are shifted laterally due to the interaction of near contours with the Gaussian filtering. The last example (**fourth row**) shows the same lateral shift effect on a natural image, see for example the two diagonal lines on the center-top of the image. In all cases the thresholds were set to $L = 0$ and $H = 0$. The image range is $[0, 255]$.

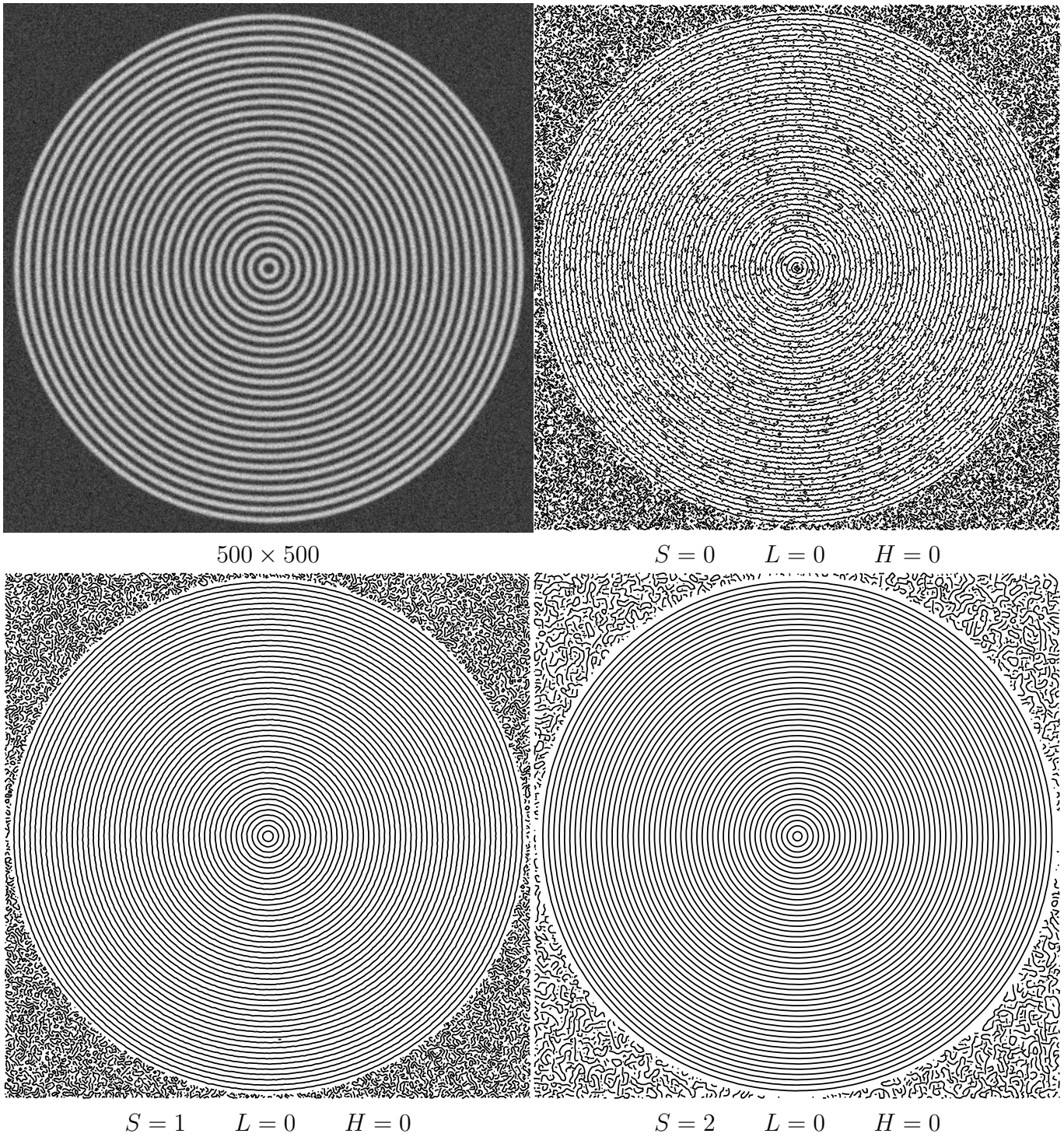


Figure 15: Illustration of the impact of the Gaussian filtering standard deviation S on a synthetic noisy image. The image range is $[0, 255]$.

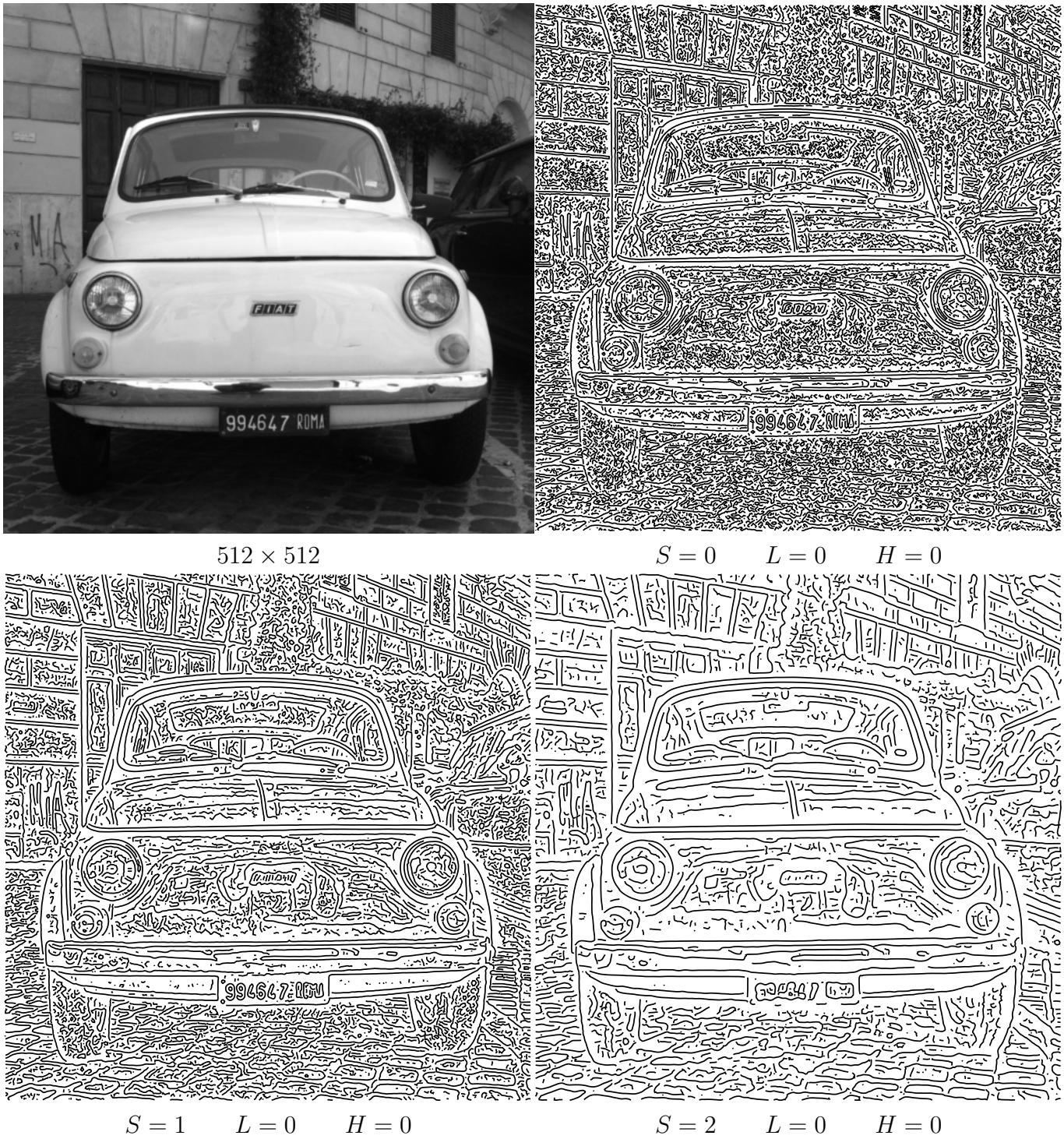


Figure 16: Illustration of the impact of the Gaussian filtering standard deviation S on a natural image. The image range is $[0, 255]$.

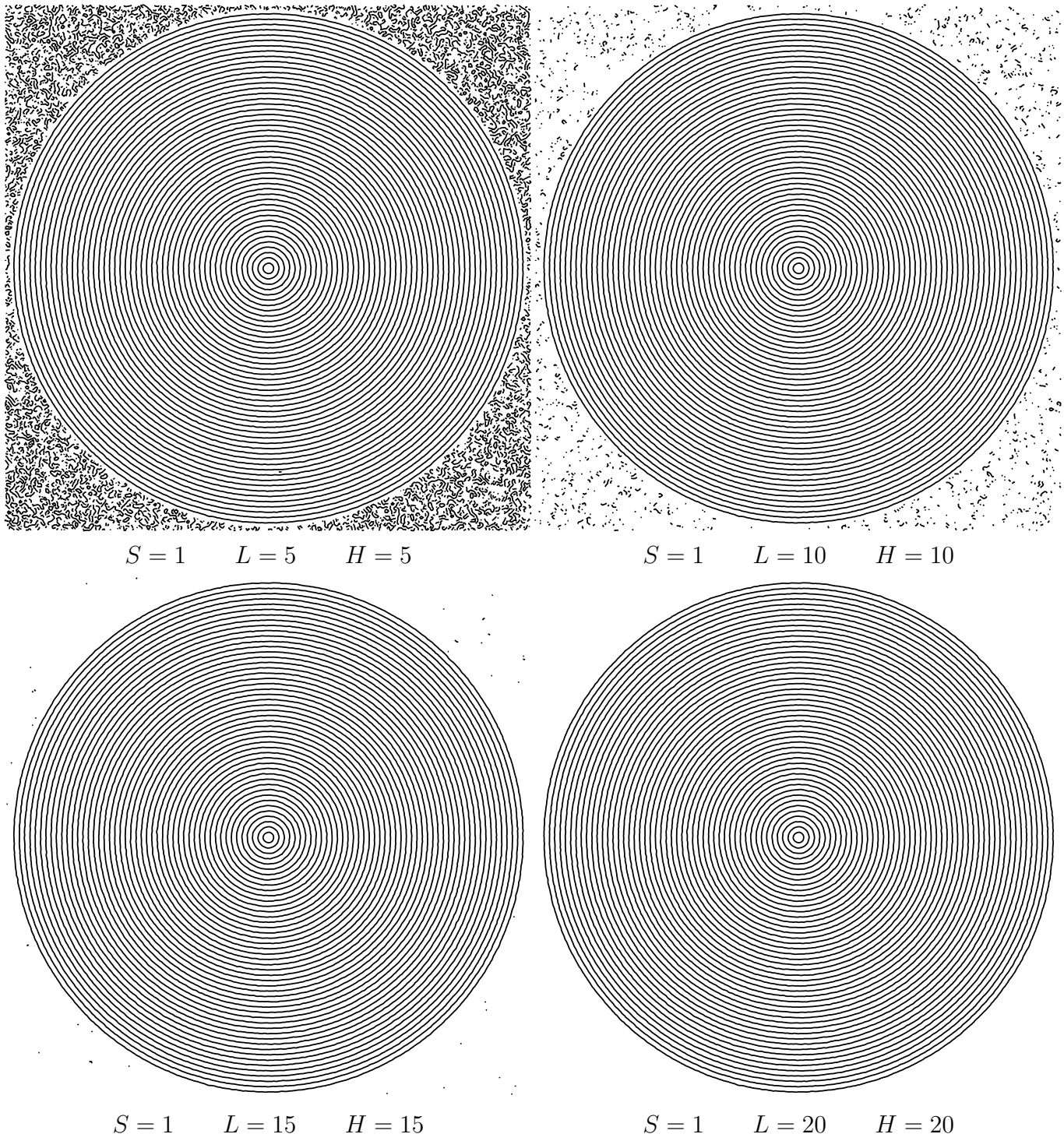


Figure 17: Illustration of the impact of the threshold on the image gradient on the same synthetic noisy image of Figure 15. Here both L and H were set to the same value. The image range is $[0, 255]$.

Figure 18: Illustration of the impact of the threshold on the image gradient on the same natural image of Figure 16. Here both L and H were set to the same value. The image range is $[0, 255]$.

Figure 19: Illustration of Canny's threshold with hysteresis on the same natural image of Figure 16. Here $L = 5$ is fixed and H is set to different values. The image range is $[0, 255]$.

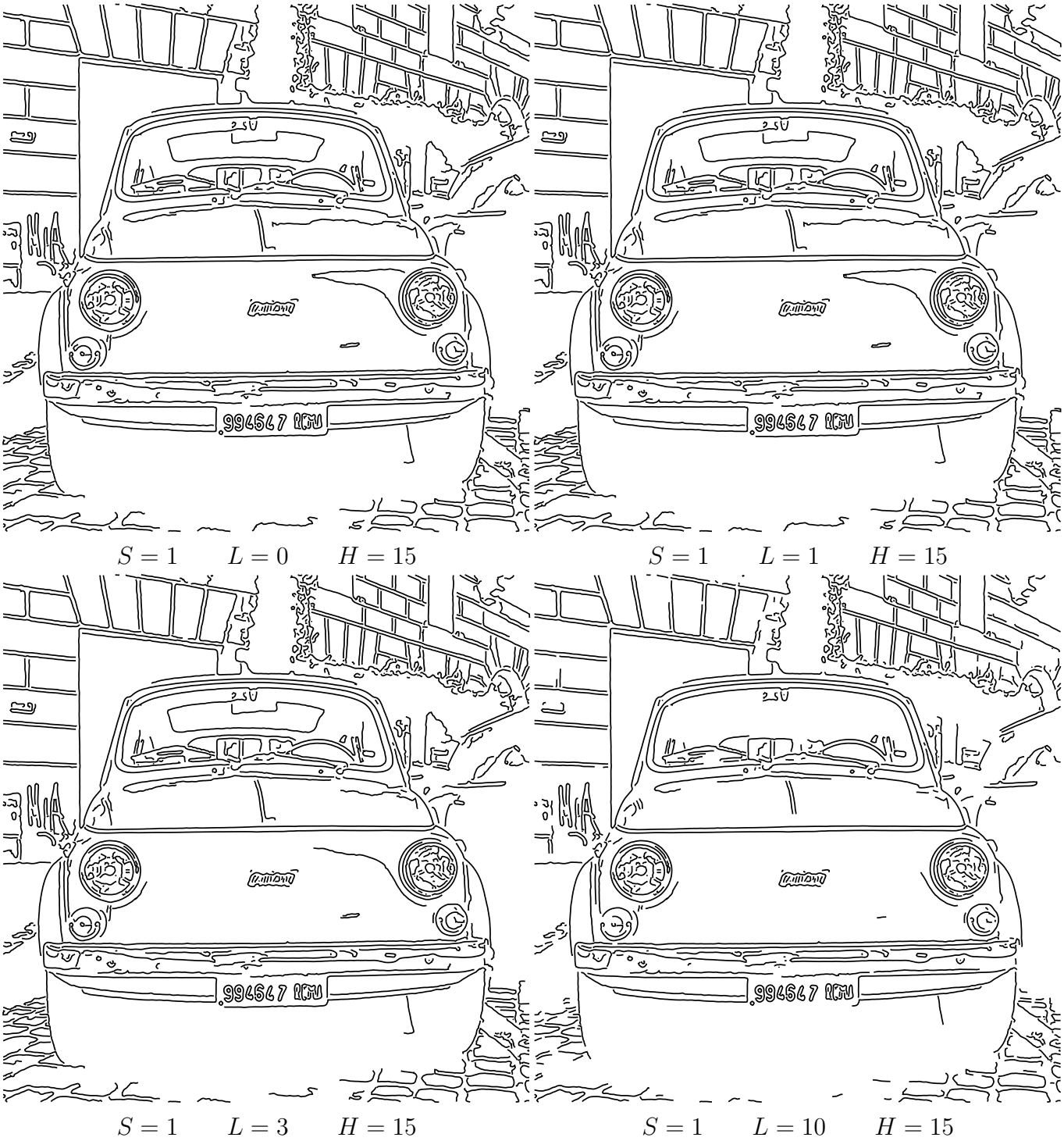
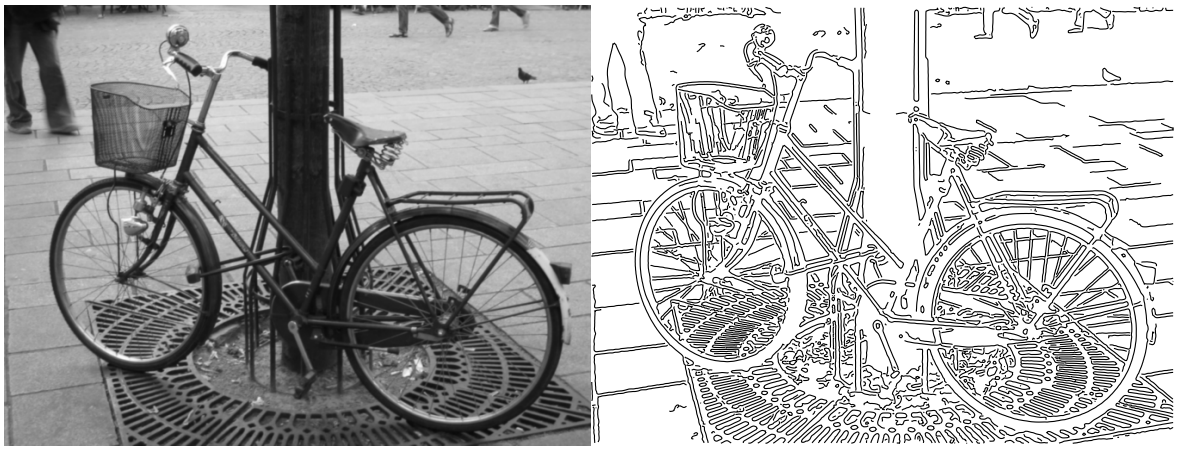
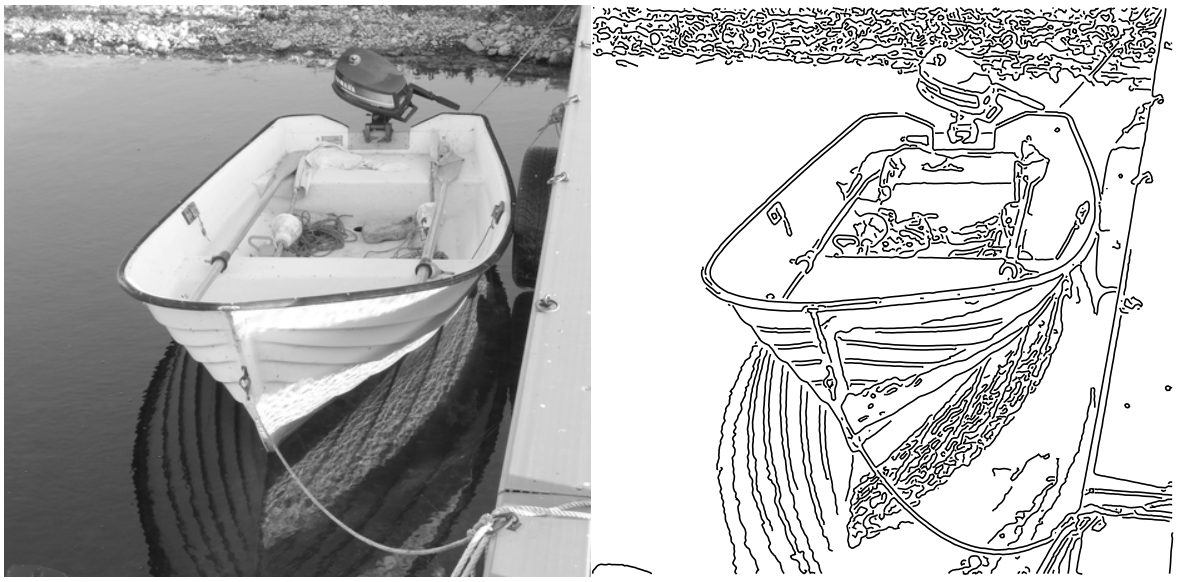


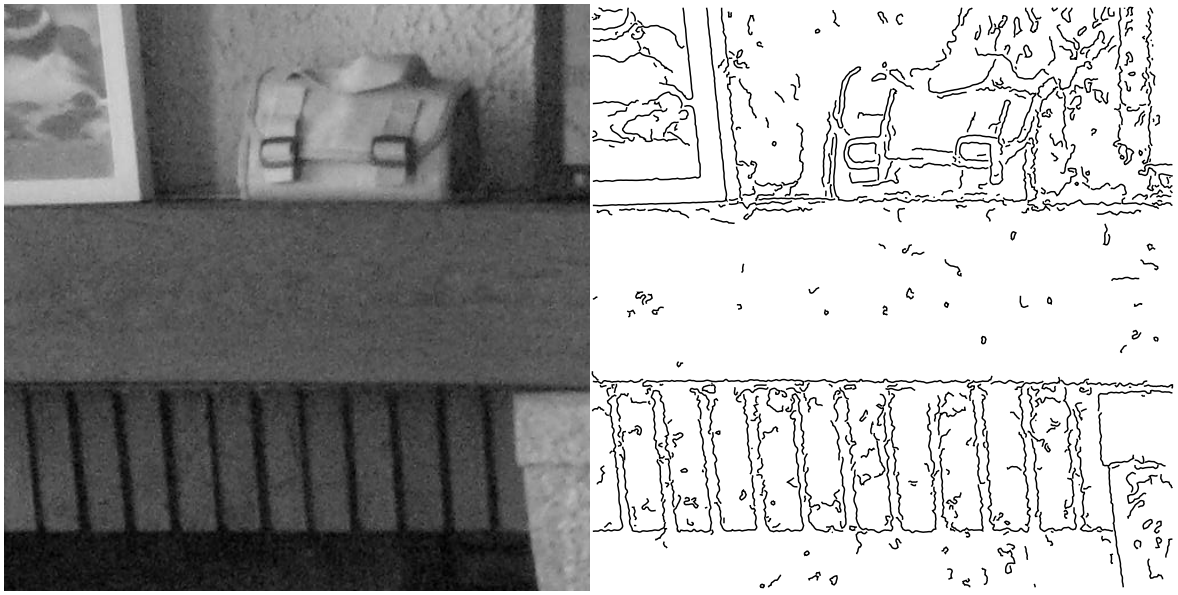
Figure 20: Illustration of Canny's threshold with hysteresis on the same natural image of Figure 16. Here $H = 15$ is fixed and L is set to different values. The image range is $[0, 255]$.



682×512

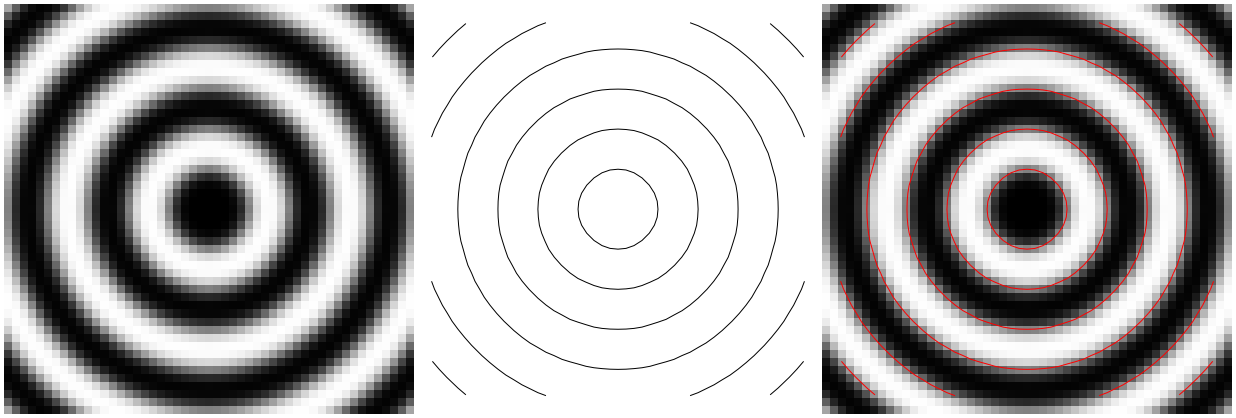


469×458



512×512

Figure 21: More examples on natural images for a particular set of parameters: $S = 1$, $L = 5$ and $H = 15$. All the images have integer pixels within the range $[0, 255]$.



51×51 sub-pixel, chained edge points superposition of both

Figure 22: Illustration of the resulting sub-pixel curves produced by the algorithm. Note that the line width used for drawing the curves is arbitrary; the result of the method is just the coordinates that define the polygonal curve. Due to the sub-pixel accuracy of the method, even if the result is a polygonal curve, it describes well the geometry on the image.

resolution.

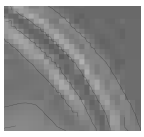
10 Conclusion

An implementation of an image edge detector with sub-pixel accuracy was described. The method is based on the classic Canny and Devernay algorithms. The accuracy was analyzed and a modification proposed to improve the sub-pixel accuracy. The behavior of the method was illustrated on various examples, including natural images and synthetic ones. The experiments also show the effect of the three parameters of the method.

Acknowledgments

We thank Jean-Michel Morel and the anonymous reviewers for valuable suggestions. Work partly founded by BPIFrance and Région Ile de France, in the framework of the FUI 18 Plein Phare project, the European Research Council (advanced grant Twelve Labours n. 246961), the Office of Naval research (ONR grant N00014-14-1-0023), ANR-DGA project ANR-12-ASTR-0035, Duke University and CCDT of Universidad de la República.

Image Credits


A detail of the upper-right side of Figure 10 of [5].
All other images by the authors.

References

[1] J. Canny, *Finding edges and lines in images*, Tech. Report 720, MIT Artificial Intelligence Laboratory, 1983.

[2] ——, *A variational approach to edge detection*, in AAAI-83 Proceedings, 1983, pp. 54–58.

[3] ——, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8 (1986), pp. 679–698. http://dx.doi.org/10.1109/TPAMI.1986.4767851.

[4] R. Deriche, *Using Canny's criteria to derive a recursively implemented optimal edge detector*, International Journal of Computer Vision, 1 (1987), pp. 167–187. https://doi.org/10.1007/BF00123164.

[5] F. Devernay, *A non-maxima suppression method for edge detection with sub-pixel accuracy*, Tech. Report RR-2724, INRIA, 1995.

[6] P. Getreuer, *A survey of Gaussian convolution algorithms*, Image Processing On Line, 3 (2013), pp. 286–310. https://doi.org/10.5201/ipol.2013.87.

[7] R. M. Haralick, *Zero-crossing of second directional derivative edge operator*, in SPIE Proceedings on Robot Vision, 1982, pp. 91–99.

[8] D. Marr and E. Hildreth, *Theory of edge detection*, Proceedings of the Royal Society of London B: Biological Sciences, 207 (1980), pp. 187–217. http://dx.doi.org/10.1098/rspb.1980.0020.

[9] G. Papari and N. Petkov, *Edge and line oriented contour detection: State of the art*, Image and Vision Computing, 29 (2011), pp. 79–103. http://dx.doi.org/10.1016/j.imavis.2010.08.009.

[10] H. Spontón and J. Cardelino, *A Review of Classic Edge Detectors*, Image Processing On Line, 5 (2015), pp. 90–123. https://doi.org/10.5201/ipol.2015.35.