IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2016–11–18.
Submitted on 2016–04–21, accepted on 2016–10–03.
ISSN 2105–1232 © 2016 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.2016.175

Unsupervised Smooth Contour Detection

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

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

Communicated by José Lezama *Demo edited by* Rafael Grompone

Abstract

An unsupervised method for detecting smooth contours in digital images is proposed. Following the a contrario approach, the starting point is defining the conditions where contours should not be detected: soft gradient regions contaminated by noise. To achieve this, low frequencies are removed from the input image. Then, contours are validated as the frontiers separating two adjacent regions, one with significantly larger values than the other. Significance is evaluated using the Mann-Whitney U test to determine whether the samples were drawn from the same distribution or not. This test makes no assumption on the distributions. The resulting algorithm is similar to the classic Marr-Hildreth edge detector, with the addition of the statistical validation step. Combined with heuristics based on the Canny and Devernay methods, an efficient algorithm is derived producing sub-pixel contours.

Source Code

The ANSI C source code for this algorithm, which is part of this publication, and an online demo are available from the web page of this article[1]. Compilation and usage instruction are included in a `README.txt` file.

Keywords: contour detection; unsupervised; sub-pixel accuracy; a contrario; NFA; Mann-Whitney U test; multiple hypothesis testing

1 Introduction

The detection of contours in digital images is one of the oldest problems in computer vision and still remains unsolved in spite of the impressive progress done in the last fifty years [18]. Indeed, current methods are far from handling correctly the formidable complexity of scenes that human perception solves immediately. Among the difficulties we can mention are the presence of noise along faint contours, subjective contours and the multi-scale geometry of scenes.

No simple definition corresponds to the notion of contour in human vision [18]. Figure 1 illustrates the difficulty with an example of contours labeled by a human. Striking differences of luminance do

[1] https://doi.org/10.5201/ipol.2016.175

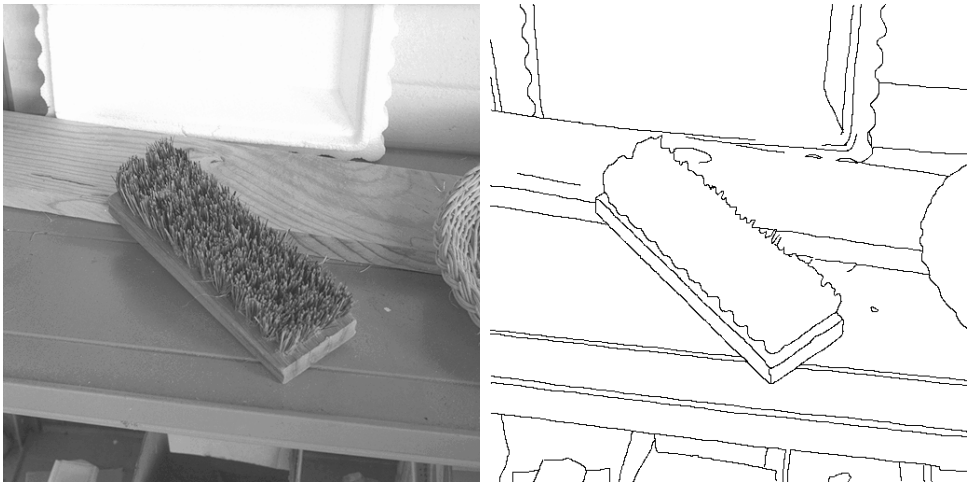

Figure 1: A sample image and the contours drawn by a human (from the RuG image database [8]).

not necessarily lead to contour perception—as the grain of wood and most of the shadows—and a perceived contour does not necessarily correspond to a contrast of luminance—as the one delimiting the bristles of the brush, more a texture change or an illusory contour. The annotation seems to depend also on the level of detail—note that some of the grains of wood were marked while most of them were not and the clearly visible strips of the basket were not marked at all. The full solution to the problem probably involves and interweaves the complete, high-level interpretation of the scene. Nevertheless, it seems that it is still possible to make progress concentrating on the purely geometrical aspects of the image and specifically on the detection of contours produced by the presence of local contrast in the luminance of images.

We will use the word *edge* to refer to a single edge point and the word *contour* to refer to a group of connected edge points forming a curve corresponding to a boundary in the image.

There is a vast literature concerning contour detection; we refer the interested reader to the excellent review by Papari and Petkov [18]. We will concentrate here on the works related to our proposal. In a seminal paper [16], Marr and Hildreth proposed to detect edges as the zero crossings of the Laplacian of the image filtered at a given scale. The method was motivated by the investigations of the visual cortex by Hubel and Wiesel [14] who provided a physiological model for simple cells in V1. Some years later, Canny [2] 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 edges modeled as a step signal plus noise. Canny proposed to approximate his 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. The Canny filter became one of the most popular edge detection methods and several improvements were added later. Among them, we will be interested in a simple post-processing proposed by Devernay [4] to obtain sub-pixel accuracy.

These algorithms include some kind of threshold parameter used to discriminate relevant structure from noise. Otherwise small structures and noise would produce zero-crossings in Marr and Hildreth's method. Canny included in his method a threshold with hysteresis on the gradient modulus. In both methods, if the thresholds are set too low the output will include edges produced by noise (false positives) and if the thresholds are set too high, then portions of real edges will not be detected (false negatives). This effect can be seen in Figure 2B–E. In general these thresholds are manually set and may lead to good results on images of a particular size and type. But there are no universal values for them.

In this paper we concentrate on the problem of the detection thresholds. Our aim is to obtain

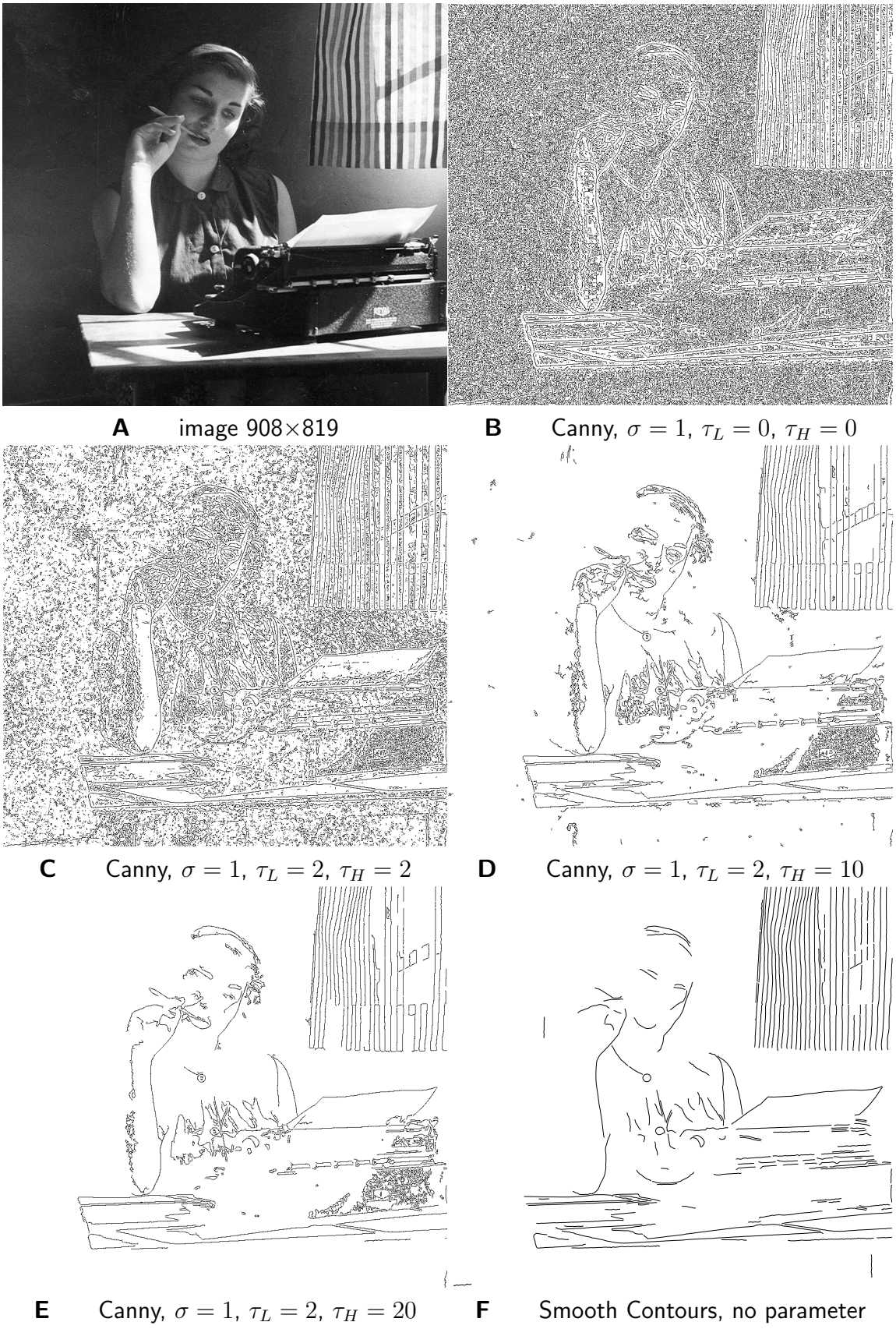

Figure 2: An image (**A**) and the output of Canny edge detector for different threshold values (**B**–**E**). The high threshold τ_H is the main criterion; the low threshold τ_L is used in Canny's hysteresis: local gradient maxima will become edge points when the gradient modulus is over τ_L but only if the pixel is connected to an already validated edge point. **F** shows the result of Smooth Contours, the unsupervised method proposed here.

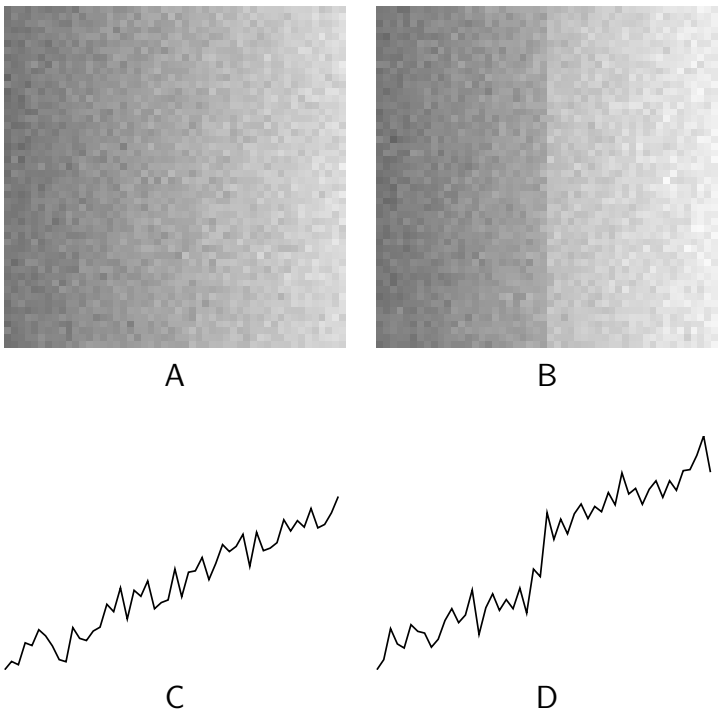

Figure 3: **A:** An example of the null model: noise over a regular gradient. **B:** A similar image but with a contour clearly present. **C:** The luminance profile of one row of the image **A**. **D:** The luminance profile of one row of the image **B**.

contours in a fully automatic way. The core of the proposal is a statistical setting to decide whether a meaningful contrast is present or not, and without the need to estimate the noise level. For this, some constraints need to be imposed. There is a compromise between the complexity of the target structure and the minimal size of the data that can lead to a significant observation. We propose as a good compromise to use a family of circular arc operators as the family of statistical tests. The proposed method detects only *smooth* contours, in the sense of being piecewise approximated by circular arcs. The smoothness is imposed by the fact that the circular arcs approximating a contour must be large enough to satisfy the statistical test. Figure 2F shows a typical result of the proposed method. Some structures are missing, as the irregular contour of the hair or the details of the eyes. However, the algorithm did manage to capture the general geometry of the scene without any parameter tuning.

Evaluating all possible circular arcs on an image would be very expensive in computational time. Besides, many circular arcs overlap everywhere on the image, what would result in redundant detections on the contours. For these reasons, the proposed algorithm starts by using an existing sub-pixel edge detector to propose curve candidates in the form of chained edge points. The statistical test is applied to parts of those curves that can be locally approximated by circular arcs. When the statistical test validates a piece of contour, the corresponding part of the curve is marked. At the end, the output of the algorithm is the union of all the marked pieces of curves. For example, a curve with the shape of a letter S may be validated locally by several statistical tests, each one on a different circular arc; but if the full curve was marked as valid, the full chain of edge pixels will appear at the output. Note that the output is not a set of circular arcs, but a set of polygonal curves defined by sub-pixel points.

The proposed method combines ideas from several important previous works. First, it is based on the edge detection theory of Marr and Hildreth [16] with the addition of the statistical test to

validate whether there is a significant luminance change or not along a candidate contour. The statistical setting follows the *a contrario* approach as proposed by Desolneux, Moisan and Morel [3], which leads to a multiple test procedure as used in statistics [13, 6]. Finally, the curve candidates are obtained from Canny's algorithm [2], including the sub-pixel correction due to Devernay [4].

Our main contribution is the particular statistical test that evaluates image contrast, which makes use of the U test of Mann and Whitney [15]. A second contribution is the integration of the whole setting, including the heuristics to accelerate the algorithm.

This paper is organized as follows. Section 2 introduces in general terms the ideas that led to the proposed method. Then, Section 3 presents a detailed formulation of the statistical setting used to validate contours. The actual algorithm uses heuristics to accelerate the search for contours which are described in Section 4. Section 5 explains the important topic of how to handle quantized pixel values, which is the most common representation of digital images. A detailed description of the algorithm is provided in Section 6 and its computational complexity is analyzed in Section 7. Section 8 shows some experimental results, commenting on the strengths and limitations of the proposed method. Finally, Section 9 concludes the paper and discusses some perspectives for future work.

2 Rationale of the Proposed Method

The *a contrario* framework [3] is a statistical formulation of the *non-accidentalness* principle[2] [1, 20] which states that an observed structure should be considered meaningful only when the relation between its parts is too regular to be the result of an accidental arrangement of independent parts. A fundamental requirement is thus a stochastic model for the data where the sought structure is not present and can only be observed by accident.

The stochastic model (or null hypothesis) is appropriate to the task of interest when it authorizes the maximum possible variability but without including the structure of interest. For contour detection we propose as an appropriate model a noisy soft gradient as can be seen in Figure 3A. No contour is perceived on the image. Both conditions are important, the noise and the regular gradient. We want to prevent detections on zones of the image where the noise dominates the luminance. But we also want to prevent detections on zones of the image with slowly varying luminance (flat zones are particular cases of soft gradients). The image in Figure 2A shows an example: near the window the illumination is irregular, producing slow gradients; the texture of the wall plus the grain of the photo results in image noise. We would expect no contour detection on the wall near the window on that image. The slow gradient of the wall is not completely regular, but it is indeed locally regular, which is the characterization of our null model. These two conditions seem to be sufficient to prevent false contour detections in a wide range of cases, as Section 8 will show.

A local operator is needed to measure the quality of a candidate contour. It should be able to distinguish a real contour from an accidental configuration of the null model. Consider the two images in Figure 3: B contains a contour while A does not. To simplify the explanations, we will concentrate on the 1D signals corresponding to the luminance profiles transversal to the contour candidates, see Figures 3C and D. The two profiles are very similar. How can one be discriminated from the other?

If the noise distribution is well known, contours can be identified as luminance steps larger than the ones expected on the noise. This is roughly the strategy used in the Canny edge detector by imposing a threshold on the image gradient. When the image noise distribution is known or when

[2]In the *a contrario* literature, the *non-accidentalness* principle is sometimes called "Helmholtz's principle". The link to Helmholtz is however unclear. Moreover, there is another principle which is more widely known as Helmholtz's principle: "Objects are always perceived as being present in the field of vision as would have been there in order to produce the same impression on the nervous system, the eyes being used under ordinary normal conditions." [7].

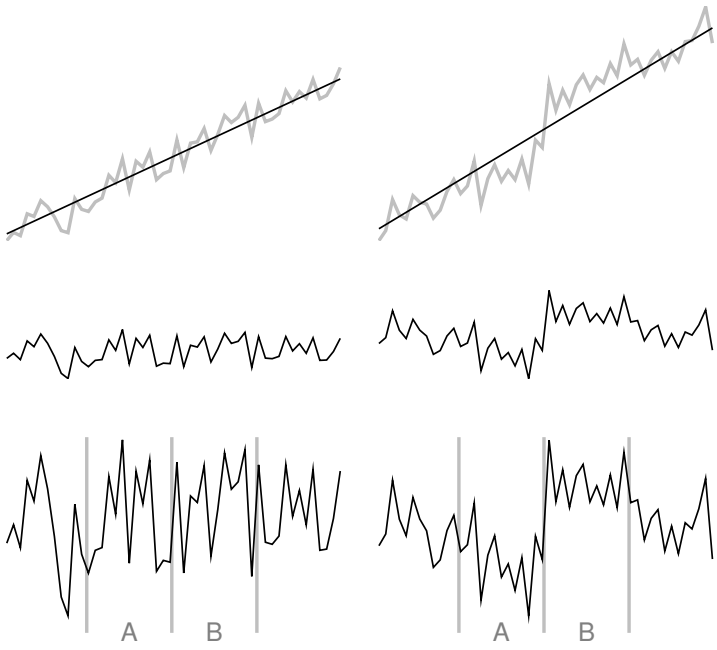

Figure 4: First strategy to cope with regular gradients: estimate the line of best fit and subtract it. Left: a null model signal. Right: a signal with a boundary. First row: the signal in gray and the line of best fit in black. Second row: the result of subtracting the best fit line. Last row: the same signals as in the previous row (the y-axis is exaggerated for visualization) and the local neighborhoods A and B indicated. When a boundary is present the values in region B are larger than in region A, while this is not the case for signals from the null model.

this threshold is manually tuned, this strategy is able to produce good results, see Figure 2E. There are two limitations, however. First, when automatic processing is needed, manual tuning is ruled out. Second, because the operator is too local, noise will occasionally produce luminance variation large enough to overcome the threshold. This situation arises in Figure 2E, see the back of the typewriter for instance.

The proposed operator looks at the neighborhood along both sides of the contour candidate and checks if the values on one side are larger than the values on the other side. This would not work for cases with regular gradients as in Figure 3C: due to the soft gradient, the illumination on one side is indeed larger than on the other side, but no contour is present. To cope with this, the plane that best fits locally the image can be estimated (best fit line in the 1D case) and subtracted, see Figure 4. When no boundary is present, regions A and B at opposite sides of the candidate contour have similar distributions. On the other hand, when a contour is present, the luminance values in region B will be in general larger than in region A.

One drawback of fitting a plane is that it adds a computational cost. More importantly, the case of curved contours is intricate. But the same basic idea can be used in a different way. Instead of fitting a plane, we can compare the signal to a smoothed version of itself, see Figure 5. As the figure shows, the difference in distributions between regions A and B is not as pronounced as in Figure 4, but it is still there when a boundary is present.

Thus, we propose to start by subtracting to the image a filtered version of itself to obtain a difference image $D = J - G \star J$, where G is a Gaussian kernel and J is the input image. This can be seen as applying a high-pass filter to remove the low frequencies which correspond to slowly varying illumination changes, so as to concentrate on the fast transitions. Then, the values between both sides of the contour candidate are compared. The contour is validated if a significant difference in

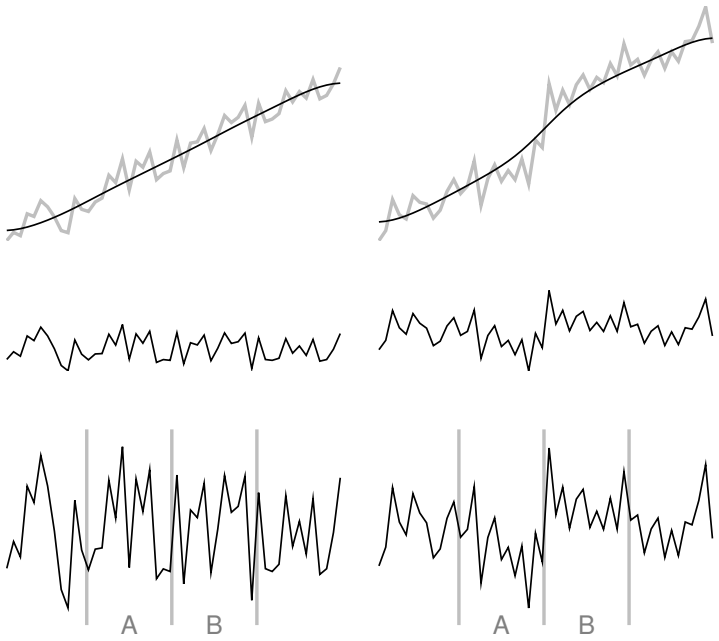

Figure 5: Second strategy to cope with regular gradients: filter the signal and subtract it to the original signal. Left: a null model signal. Right: a signal with a boundary. First row: the signal in gray and its filtered version in black. Second row: the result of subtracting the filtered signal. Last row: the same signals as in the previous row (the y-axis is exaggerated for visualization) and the local neighborhoods A and B indicated. When a boundary is present the values in region B are larger than in region A, while this is not the case for signals from the null model.

distribution is observed. The next section describes the precise formulation.

The methodology just presented can be linked to the edge detection theory by Marr and Hildreth [16]. Based on the study of the visual cortex, they proposed to compute the zero-crossings of the Laplacian of the filtered image. More precisely, their method finds the zero values of $\nabla^2 G \star I$, for a given image I. The first row of Figure 6 shows the images of Figure 3 convolved by $\nabla^2 G$. The second row of the figure shows the same images after a threshold on zero: black pixels had negative values and white pixels had positive values. Thus, the black/white frontiers correspond to the detected contours. Marr and Hildreth also showed [16] that the Difference of Gaussian (DoG) $G_1 - G_2$, with appropriate standard deviations σ_1 and σ_2, provides a good approximation to the Laplacian of Gaussian (LoG) $\nabla^2 G$. Their method is then equivalent to finding the zero-crossings of $G_1 \star I - G_2 \star I$; that is, the image filtered by a Gaussian filter of standard deviation σ_2 is subtracted to the same image filtered by a Gaussian filter of standard deviation σ_1 of smaller value.

As it is evident on the left hand image of Figure 6, without an additional restriction, Marr and Hildreth's algorithm would produce contour detections on noise images. The method proposed here is a way of adding a statistical test to their method to reject this kind of contours that appear accidentally on noise. Because of the filtering, the pixels of $G_1 \star I$ are correlated. The statistical test to be applied is simplified if a sub-sampling is applied to decorrelate the pixels. For example, a white noise image blurred and correctly sub-sampled, will retain the white noise statistics. Let δ be the sampling step which decorrelates the pixels of $G_1 \star I$. We would like to analyze the DoG image down-sampled by a step δ. Equivalently, let J be the sub-sampled version of $I_1 = G_1 \star I$, see Figure 7. Using an appropriate Gaussian kernel G, we would be able to obtain the sub-sampled version of $I_2 = G_2 \star I$ by $G \star J$. Finally, the down-sampled version of DoG is $D = J - G \star J$. The connection to the proposed method should be clear now. If one assumes that the initial sub-sampling

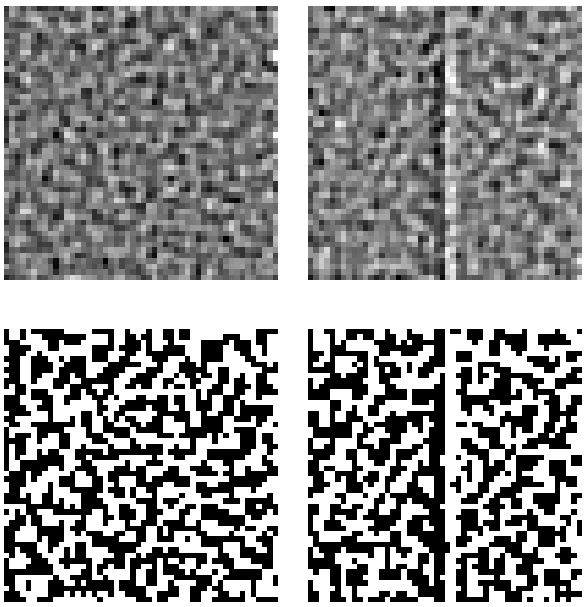

Figure 6: Top row: Result of applying the Difference of Gaussian (DoG) operator to the images in Figure 3 (top row). 2nd row: The same images after a threshold on zero: black pixels had negative values and white pixels had positive values. The black/white frontier corresponds to the detection proposed by Marr and Hildreth at one particular scale.

from I to J was already performed and the input image is J, the proposed method is equivalent to Marr and Hildreth's method.

The standard deviation σ of the Gaussian kernel G does not depend on the scale of the analysis σ_1. Marr and Hildreth [16] have shown that the optimal approximation of $\nabla^2 G$ by a DoG is obtained by a standard deviation ratio $s = \frac{\sigma_2}{\sigma_1} \approx 1.6$. The convolution of two Gaussian kernels of variance σ_a^2 and σ_b^2 is also a Gaussian kernel of variance $\sigma_a^2 + \sigma_b^2$. As a result, I_2 can be obtained by convolution of I_1 with a Gaussian kernel G' with standard deviation $\sigma' = \sqrt{\sigma_2^2 - \sigma_1^2} = \sigma_1 \sqrt{s^2 - 1}$. When performing a Gaussian sub-sampling, the standard deviation σ_1 must be proportional to the sampling step δ: $\sigma_1 = c \cdot \delta$. The choice of the constant c implies a compromise between aliasing and blur. A low value would keep high frequencies, producing aliasing; a high value prevents aliasing at the cost of blurring the image. We will follow the choice of Morel and Yu [17] and use $c \approx 0.8$. The standard deviation of the Gaussian kernel G is $\sigma = \frac{\sigma'}{\delta}$ which results in

$$\sigma = c\sqrt{s^2 - 1} \approx 0.8\sqrt{1.6^2 - 1} \approx 0.9992. \tag{1}$$

Marr and Hildreth proposed to analyze the image at different scales. Our method works at a single scale, the one implicit on the input image J. An analysis at a different scale can be obtained by a preprocessing to produce J from I by Gaussian filtering and then sampling at the desired scale. Section 8 shows some examples. Analyzing the image at multiple scales in an integrated way is an important and difficult extension to be studied in future work.

3 Contour Validation

The proposed method can be described as a variation of Marr and Hildreth's algorithm [16] with the addition of a statistical test for validation. To be validated, a contour must be the frontier separating two adjacent regions of the difference image D, one region with significantly higher pixel

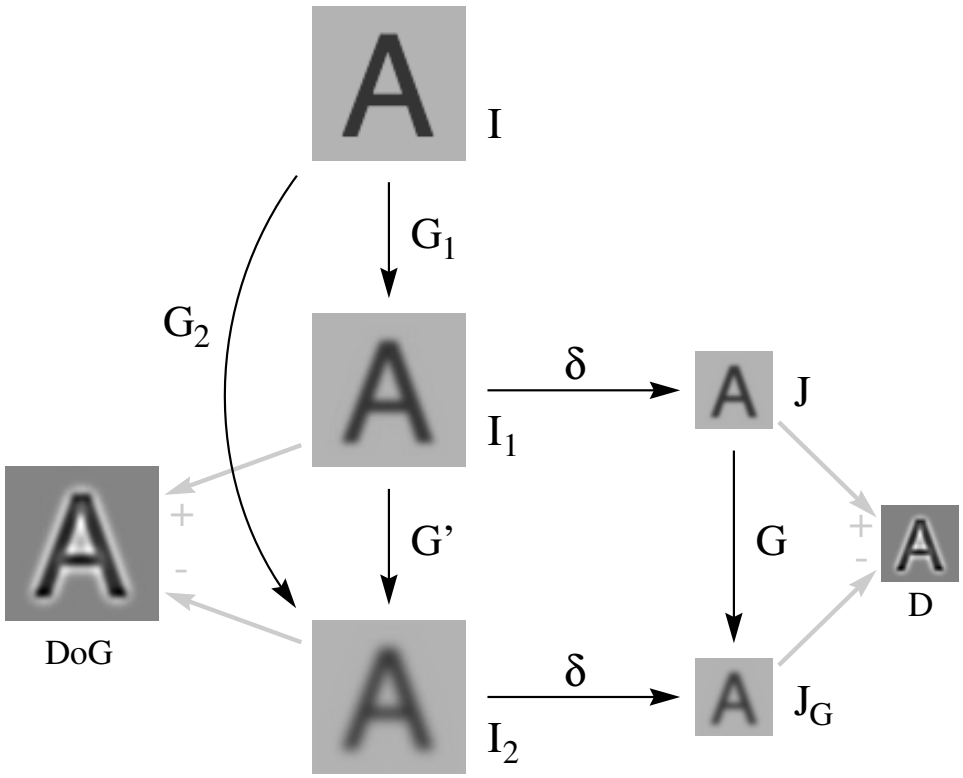

Figure 7: Schematic relation between the proposed method and Marr-Hildreth's method. Marr and Hildreth proposed to use the difference of Gaussian (DoG) to approximate the Laplacian of Gaussian of the image. The input image I is convolved with two different Gaussian kernels G_1 and G_2 to obtain the images I_1 and I_2. The difference of them is the DoG. To decorrelate the pixel values, image I_1 must be sampled using the step δ, with a value proportional to the standard deviation σ_1 of G_1. This results in image J. Sampling I_2 with the same step δ results in image J_G. The difference between them is $D = J - J_G$, which is equivalent to sampling DoG with step δ. The proposed method takes as input the image J, already at the scale to be analyzed. Image J_G is obtained by filtering J with a Gaussian kernel G with a standard deviation σ that is independent of the scale of analysis, as explained in the main text.

values than the other. There are, of course, many possible ways of giving a precise sense to these terms: Candidate frontiers may be any arbitrary curve or they may be restricted to a family of curves. How far from the frontier should the pixel values be compared? The options are also vast for the way of comparing pixel values; when using a statistical test, there are many options for the test and the null hypothesis.

It would be desirable to build a validation step able to cope *directly* with arbitrary curves. However, the more complex the family of curves is, the more likely that an accidental observation is produced just by chance. A compromise is needed between the ability to detect curves of complex geometry and the quantity of supporting data needed to obtain statistically significant observations. As explained below, the compromise appears in our setting in relation to the problem of multiple testing where the number of tests, reflecting the richness of the family, regulates how strict the test must be. As a result, the more complex the family of test is, the larger the minimal curve size that can be detected.

We propose to use a set of circular arcs as the family of contour operators. The experimental section will show that this family is rich enough to handle smooth frontiers in the sense of being locally approximated by circular arcs. At the same time, the family is small enough to allow significant

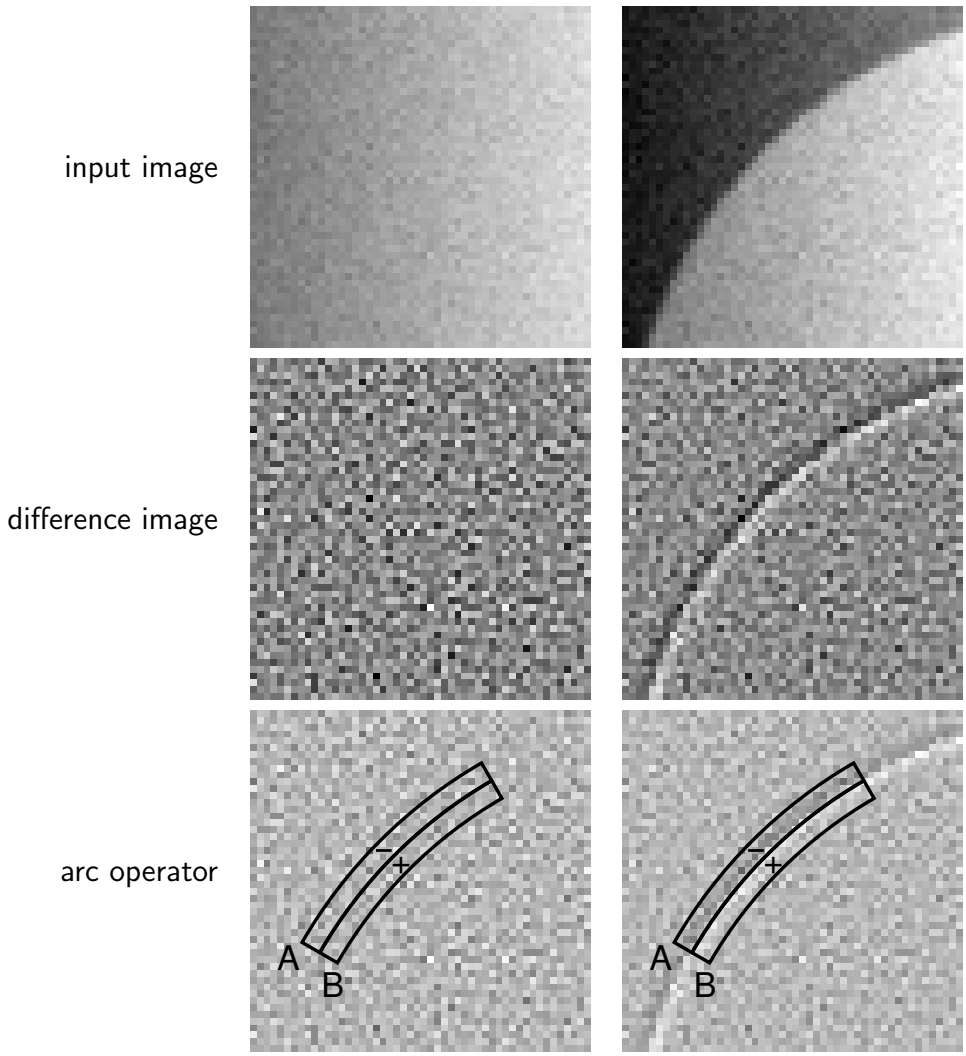

Figure 8: Arc operators on images without (left) and with (right) a contour. The first row shows the input images. The second row shows the difference images D. The last row shows the arc operators superposed to the difference image (in lighter gray-values for visualization). The arc operator consists of two arc rings, delimiting regions A and B. The operator will validate a contour element when the pixels in region B have significantly larger values than the pixels in region A.

validation under very few assumptions. The motivation of this choice is to focus on the simplest operator incorporating curvature. Other choices may be of interest, each one implying different compromises, and will be studied in future work. Figure 8 illustrates what we call an *arc operator*. It is defined by a circular arc on the image domain plus a width w of each of two lateral arc rings, delimiting regions A and B. The rest of this section describes the statistical test associated with each arc operator to decide whether the pixels of region B have significantly larger values than the pixels in region A or not.

In the same spirit as before, we choose to use the simplest possible assumptions for the statistical test. The null hypothesis H_0 is that, in the absence of structure, all pixels of D are independent and follow the same distribution. Thus, when evaluating an arc operator under H_0, the pixels of both regions A and B are independent and follow the same distribution. No further assumption on the particular distribution is made.[3]

[3]It is indeed interesting to use more sophisticated hypothesis about the distribution of values in D. One example

The independence hypothesis is, strictly speaking, not true. Indeed, even if the independence hypothesis is valid for the input image from H_0, a blurred version of itself is subtracted to obtain the difference image D; the pixels of D are no longer independent. Nevertheless, the pixels of D are fairly decorrelated and the independence assumption is approximately valid. More importantly, the assumption allows to perform a very simple statistical test. The experimental results show that the number of false detections is effectively controlled, supporting the use of the assumption.

Given an arc operator of a particular size and position on the image D, two sets of pixel values are defined, $A = \{a_i\}_{1 \le i \le n}$ and $B = \{b_j\}_{1 \le j \le m}$, see Figure 8. To evaluate whether the values of B are significantly larger than the values of A, the statistic U of Mann and Whitney [15, 19] is computed, which counts the number of times that a pixel of region B has a greater value than a pixel of region A. More precisely,

$$u = \sum_{i=1}^{n} \sum_{j=1}^{m} \gamma(a_i, b_j), \qquad \text{where} \qquad \gamma(a_i, b_j) = \begin{cases} 1, & a_i < b_j \\ 1/2, & a_i = b_j \\ 0, & a_i > b_j. \end{cases} \tag{2}$$

The statistic u can take values between zero and nm. Zero means that $a_i > b_j$ for every i, j and does not correspond to the contour being evaluated.[4] A flat zone, with $a_i = b_j$ for every i, j, would lead to a value of $\frac{nm}{2}$. A perfect contour has a value of nm, in which every pixel in B is larger than any pixel in A. So a large value of u corresponds to a contour. But which value is large enough?

The decision is based on a classic statistical hypothesis test using H_0 as the null hypothesis. The null hypothesis is rejected (and the contour validated) when the observed value u is too large so it would be an unlikely realization under H_0. Let U be the result of computing the Mann and Whitney statistic on two sets of pixels $\bar{A}$ and $\bar{B}$, with cardinalities n and m as before, but randomly and independently sampled from the same unknown distribution. Consider the p-value of the test, defined as the probability $P(U \ge u \,|\, H_0)$ of getting by chance a value U equal or larger than the observed value u. The null hypothesis is rejected when this probability is below a significance level α

$$P(U \ge u \,|\, H_0) \le \alpha. \tag{3}$$

For $n, m \ge 8$, Mann and Whitney [15] showed that, under these assumptions, the distribution of U can be well approximated by a normal distribution of mean $\mu_U = \frac{nm}{2}$ and variance $\sigma_U^2 = \frac{nm(n+m+1)}{12}$. We will use the normal approximation to compute the p-value.

When analyzing an image, multiple tests are performed, one for each arc operator. Thus, the problem of multiple testing, well known in statistics [13], needs to be considered. The problem arises from the fact that as the number of hypotheses being tested increases, the likelihood of a rare event also increases, and therefore the likelihood of incorrectly rejecting a null hypothesis. For example, if one test is performed at a level $\alpha = 0.05$, there is a probability 0.05 of incorrectly rejecting the null hypothesis if the null hypothesis is true. However, for 100 tests under the null hypothesis, the expected number of incorrect rejections is 100×0.05 or 5. If the tests are independent, the probability of at least one incorrect rejection is 0.994.

We will follow the *a contrario* methodology [3] which suggests controlling the *expected* number of false detections to a level ε. This boils down to using the classic Bonferroni correction to control the

would be using a null hypothesis H_0 adapted to the noise actually present. This may allow to detect contours with much smaller operators. Or alternatively, may allow to use richer families of curves while keeping the size of the minimal operators. That would come at the cost of parameters to set or estimate. We will explore this kind of alternatives in future work.

[4] A zero value indicates a perfect contour but with opposite contrast. Both contrasts, b/w or w/b, could be handled at the same time by the same arc operator. But in our algorithm, the validation will be applied to curve segments that are part of a larger curve and we want to obtain a coherent contrast all along the curve. Thus, opposite contrasts are handled as different operators. This is just a convention to simplify the general logic.

Per Family Error Rate (PFER) [6]. The level of each test is corrected by the number of tests to be performed. Consider a family of N tests u_1, u_2, ..., u_N, each one corresponding to an arc operator. In each case, the null hypothesis will be rejected when

$$P(U_k \geq u_k \,|\, H_0) \leq \frac{\varepsilon}{N}. \tag{4}$$

This expression lets us understand the compromise between the richness of the family of tests and the strictness of the test. As the family of tests becomes richer, it is more likely that one operator u_i is well adapted to a real contour of the image. By the same token, N becomes larger, imposing a smaller p-value to validate the test, which implies more evidence or larger operators. This is another expression of the classic trade-off between complexity of the model and goodness of fit.

In the *a contrario* literature it is common to define a quantity called Number of False Alarms (NFA) associated to each test

$$\mathrm{NFA}_k = N \cdot P(U_k \geq u_k \,|\, H_0). \tag{5}$$

Consequently, the null hypothesis is rejected (and the contour validated) when

$$\mathrm{NFA}_k \leq \varepsilon. \tag{6}$$

Note that the conditions of Equations (4) and (6) are equivalent. With this setting, it can be shown [3, 6] that the expected number of false rejections (PFER) is controlled[5] by ε:

$$\mathrm{PFER} = E\left(\sum_{k=1}^{N} \mathbb{1}_{\mathrm{NFA}_k \leq \varepsilon} \,\middle|\, H_0 \right) \leq \varepsilon. \tag{7}$$

This provides a meaning to the name NFA. A candidate's NFA corresponds to the global number of false alarms required to validate it. In other words, if a candidate with a value NFA is accepted, one should expect to observe NFA false detections. A large NFA value corresponds to a common configuration under the null hypothesis, giving no hint that a geometrical structure may be present. A small NFA value, in contrast, corresponds to a rare configuration under the null hypothesis; either an exceptionally rare accident happened or a geometrical structure is responsible for the observed data. The smaller the NFA value is, the more likely the observation is to have a causal interpretation.

As usual in the *a contrario* literature [3], we will set $\varepsilon = 1$. This corresponds to accepting, on average, one false detection per image. This usually proves to be a good choice. The experimental section shows some experimental support.

The number of tests N needs to be specified. The family of tests is the set of arc operators considered before. We will compute an approximation to its number. What is really important is the order of magnitude, which allows the thresholds to adapt to the size of the data. What counts are the lateral regions of pixels, which are sets of pixels; thus, the operators will be defined up to pixel precision. The particular parameterization of the family of arc operators used is irrelevant.

We chose to divide the full family of tests into sub-families according to the arc length. This allows us to divide the total number of authorized false alarms ε into equal parts for each length family. Given an input image of size $X \times Y$, we will consider arc lengths from one to $\sqrt{XY}$; each length sub-family will get the same share of authorized false alarms $\frac{\varepsilon}{\sqrt{XY}}$. Because the sub-families of short lengths are smaller, this gives a slight advantage to short lengths (penalizing the longer lengths) while still controlling the total number of false detections to a level ε.

[5]The classic use of the Bonferroni correction is to control the Family-Wise Error Rate (FWER), that is the *probability* of making one or more false rejections. Nevertheless, the PFER is also controlled [6]. In the latter case, because what is controlled is an *expected value* instead of a probability, the level needs not be limited by 1 and any level between 0 and N is possible. What changes is the interpretation and the way of selecting the desired level of the test. In some settings, as in our case, it is natural to fix the mean number of false detections per family of experiments one is ready to accept (i.e., the mean number of false detections per image one is ready to accept). In such cases, Gordon et al. [6] showed that the natural procedure to use is the Bonferroni correction controlling the PFER.

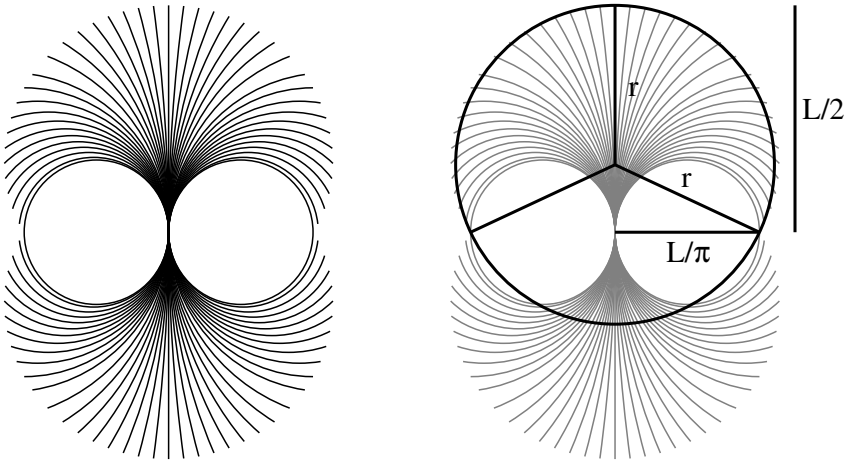

Figure 9: Left: Arc family of a given length L and for a given orientation. Right: When the arc forms a closed circle, its diameter is L/π. An arc is determined by the crossing point with the circle of radius $r = L\left(\frac{1}{\pi^2} + \frac{1}{4}\right)$. (This is deduced by Pythagoras' theorem on the triangle of sides r, L/π and $L/2 - r$.) This spans about 2/3 of the full circle.

We need to complete the approximate counting for each length sub-family. Given a length L, the arc operators still have five degrees of freedom; let us consider the position (two coordinates), the orientation, the curvature and the lateral width. An arc operator will be centered at each pixel of the image, giving a factor XY. The possible orientations are approximated by the perimeter of a circle of diameter L. Thus, we count πL orientations. For a given length, position and orientation there is a family of circular arcs with different curvature, see Figure 9 (left). Discretizing these arcs at approximately pixel precision corresponds to about 2/3 of the perimeter of the circle in Figure 9 (right). This leads to $\frac{2}{3} \times 2\pi \times L\left(\frac{1}{\pi^2} + \frac{1}{4}\right) = \frac{L}{3}\left(\frac{4}{\pi} + \pi\right)$ arcs. As will be described in the next section, several widths w for the lateral regions are tried for each circular arc. Thus, the number W of different widths for the lateral regions must be considered. Finally, for a given length L we count $XYW(4+\pi^2)\frac{L^2}{3}$ arc operators and the test to be performed is

$$P(U_k \geq u_k \,|\, H_0) \leq \frac{\varepsilon/\sqrt{XY}}{XYW(4+\pi^2)\frac{L^2}{3}}. \tag{8}$$

As said before, this is only an approximation (among other things, it does not consider that many of the arcs counted lay outside of the image domain). Putting all together, the effective number of tests is

$$N = \frac{4+\pi^2}{3} WXY\sqrt{XY}L^2, \tag{9}$$

and the NFA expression is

$$\mathrm{NFA}_k = \frac{4+\pi^2}{3} WXY\sqrt{XY}L_k^2 \cdot P(U_k \geq u_k \,|\, H_0). \tag{10}$$

Note that for a given image size there is a minimal operator size that can lead to a validation. Indeed, for a given operator, the lowest p-value that can be observed is for $u = mn$, when all the pixels in region B are larger than the pixels in region A. To simplify the analysis, consider that both

lateral regions (which usually are about the same size) have the same size n. Then, the limit is given by the minimal n that can satisfy the condition of Equation (4)

$$P(U \geq n_{\min}^2 \,|\, H_0) \leq \frac{\varepsilon}{N}. \tag{11}$$

Given that the maximal arc operator width evaluated is $w_{\max}$, the minimal arc length that can produce a meaningful contour is related to $n_{\min}$ by $n_{\min} \approx L_{\min} w_{\max}$. Putting all together,

$$L_{\min} = \min \left\{ L : \frac{4+\pi^2}{3} WXY\sqrt{XY}L^2 \cdot P\Big(U \geq (Lw_{\max})^2 \,\Big|\, H_0\Big) \leq \varepsilon \right\}. \tag{12}$$

As a consequence, contours that are too irregular to be locally approximated by circular arcs of at least the minimal size, will be rejected. This is the precise sense in which the method detects only smooth contours. Section 8 illustrates this with examples.

4 Candidate Selection

The validation procedure just described could be applied exhaustively to the whole family of circular arcs. There are, however, two reasons for not doing so. First, for efficiency considerations: evaluating all the potential candidates leads to a computational burden. If we could perform one of these tests per CPU cycle (and this is very far from being true) at 1 GHz, it would take more than three days to process a 200×200 image. Second, it would lead to many redundant detections because many arcs of the family would overlap over a significant part of each valid contour, leading to multiple detections.

Smooth Contours uses the same strategy of the LSD algorithm, effectively solving both problems [9, section 3.1]. A heuristic method proposes candidates which are then validated or rejected by the statistical procedure derived in the last section. To solve both problems, the heuristic must be fast and prevent redundancy.

Any existing edge detector could be used as a heuristic to generate candidates. We propose to use the chained edge points produced by the Canny [2] algorithm with the detection thresholds set to zero in order to leave the full validation task to the statistical test, see Figure 2B for an example. To get sub-pixel results, the simple correction term proposed by Devernay [4] is included. In a nutshell, the position of an edge point is estimated 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. Finally, the edge points are chained to form curve candidates.

Let us verify that the requirements for solving the two problems are satisfied. First, the combined Canny/Devernay algorithm runs in linear time relative to the number of pixels in the image. Section 7 shows that it leads to a reasonably fast method. Second, due to the *non-maximal suppression* step in Canny/Devernay method, the resulting curves are non-overlapping, preventing redundancy. One may wonder why the original Marr and Hildreth method is not used for the heuristic, but actually Canny gives a better localization of the edges points.

The validation step needs a Gaussian filtering of the image and so does the Canny/Devernay algorithm. The same standard deviation σ is used for both, as computed in Equation (1), so the input image is filtered only once.

The curves produced by the Canny/Devernay method are the candidate contours. Each segment of the curve is approximated by a circular arc. The first, middle and last edge points of the segment are used to compute a circular arc. A verification is necessary to confirm that the arc indeed corresponds to the curve segment. That could be false, for example if the curve segment has the

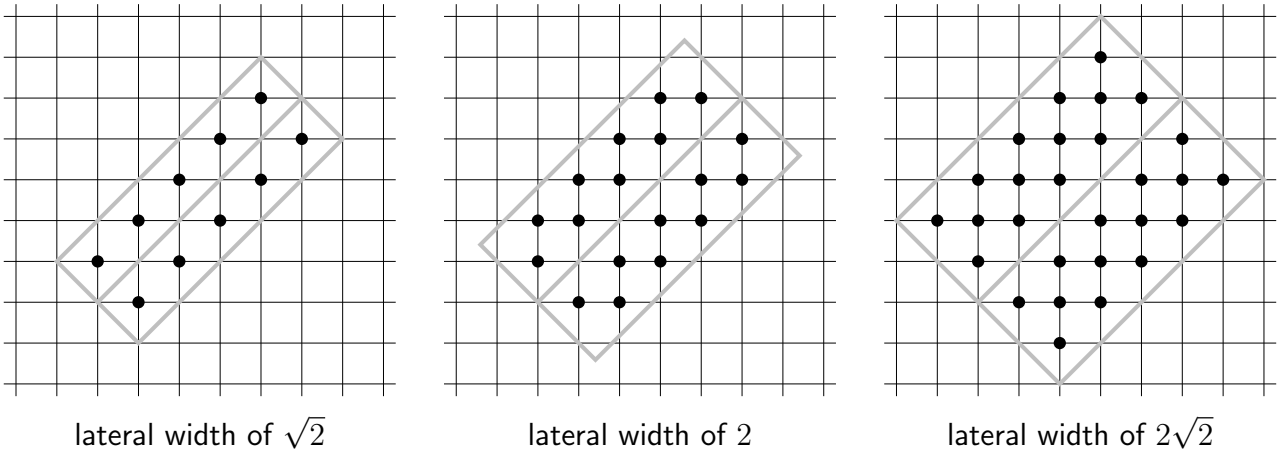

Figure 10: In the worst case for slanted straight contours at 45 degrees, a lateral width of $\sqrt{2}$ ensures at least one row of lateral pixels at each side, a lateral width of 2 ensures at least two rows of lateral pixels, and a lateral width of $2\sqrt{2}$ ensures at least three rows of lateral pixels.

shape of the letter S. The arc is accepted if the offset to the curve segment is never more than σ, the standard deviation used in the Gaussian filtering. If the arc is accepted, the quantity of Equation (10) is computed and the segment is validated as contour when NFA ≤ 1.

The same procedure should be applied to all possible segments of the whole set of curves. Consider again a curve with the shape of a letter S. No arc corresponds to the full curve, but the curve is locally approximated by multiple arcs, each one covering a sector. The whole curve could be validated as a contour if each of its segments passes the test. The curve could be fully rejected too, if none of the segments passes the test. Finally, it is also possible that the curve is only partially validated. This can happen when a part of the curve is not well contrasted to pass the validation step, if some segments are not smooth enough, or a mix of both.

Note that, even if the curves are validated locally by circular arc operators, the output of the method is the union of the curve segments that were validated. Thus, the output is not a set of circular arcs but a set of curves defined by chained edge points. It is of course interesting to produce an output composed of chained circular arcs, what in computer graphics is called the vectorization of a curve. However, in the current algorithm, curves are validated by arc operators with some overlap. Merging those overlapping circular arcs into a smooth result is not an easy task. Many algorithms were proposed in the past to segment a polygonal curve into line and arc segments. And indeed, doing that before the validation may solve two problems at the same time: segmenting the curve in a smart way instead of testing all the curve segments, and providing the output as a vectorization into chained circular arcs. Unfortunately, we have not found yet a satisfactory algorithm for our task. We will focus on this problem in future work. Nevertheless, as will be shown in Section 8, the result is already a vectorization of the curves into polygonal curves of sub-pixel accuracy.

To accelerate the method even more, two additional heuristic criteria are used, with the result that not every segment of the curves is actually tested. Large segments are processed first and then smaller ones. If the edge points at both extremes of the segment are already validated (a previous arc operator was found with NFA ≤ 1) the segment is not evaluated. This usually corresponds to cases where the segment is part of an already validated contour and there is no need to test it again. Inversely, when a segment is well approximated by a circular arc but it is subsequently rejected as a contour, the pixels from the center of the segment up to 3 edge points before the extremes are marked as *used*. A curve segment in which both extreme edge points were previously marked as *used* is not tested. The idea is that if the segment was well approximated by an arc but rejected, a

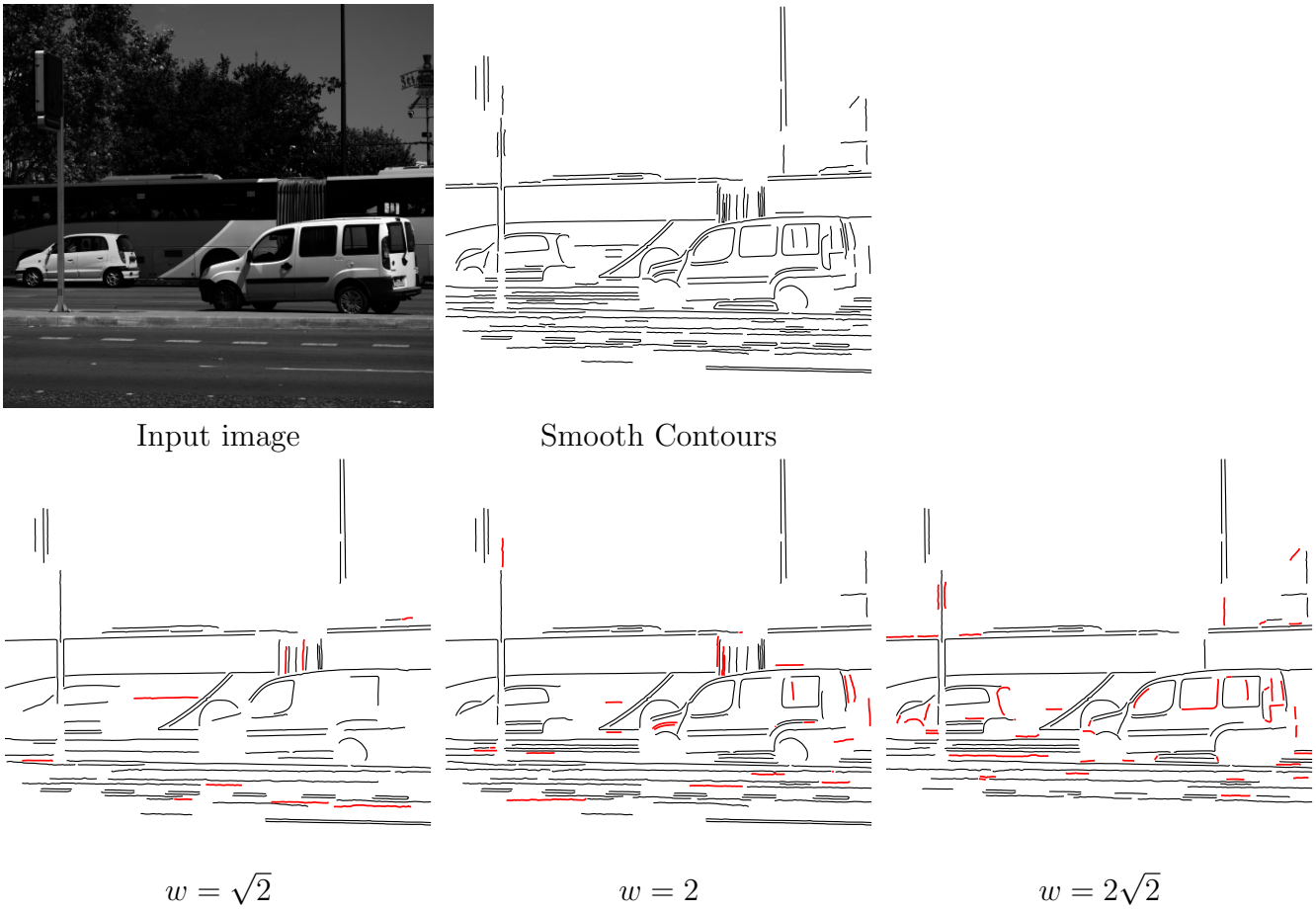

Figure 11: The contribution of each lateral width w to the overall detection. The first row shows the input image and the full result of Smooth Contours. The second row shows the validated contours for each value of the lateral width w. The contours drawn in red were only validated with that particular w.

smaller part of it would be probably rejected as well. This assumption may be false in cases where the segment was rejected for not being globally well contrasted, but a part of it could be better contrasted and potentially validated. These are rare but possible cases in which this heuristic would lead to misdetections. There is no further justification for these heuristics other than the arguments just mentioned. The choices are based on trial and error experimentation until obtaining a good compromise that speeds-up the method while keeping most of the validated contours relative to the exhaustive version.

Let us discuss the choice of the lateral width of the arc operators. The size should be related to σ, the standard deviation used in the Gaussian filtering, which in our case has a value of about one, see Equation (1). If the lateral width is much larger, it would miss the region where the contour is visible; if it is much smaller, there would be no pixels in the region. Increasing the number of lateral widths W also increases the number of tests N, which reduces the details that would be detected. Also, adding more width values implies a computational cost. Empirically, we found a good balance using three different values for the lateral width: $\sqrt{2}$, 2 and $2\sqrt{2}$. The lateral width corresponds to the distance in pixel units to each side of the contour candidate. Given the discrete nature of pixels, and considering slanted contours at about 45 degree, the region within a distance of $\sqrt{2}$ ensures the presence of at least one row of pixels on each side, see Figure 10. Similarly, lateral widths 2 and $2\sqrt{2}$ ensures two and three rows of pixels on each side, respectively. One pixel row is enough to validate

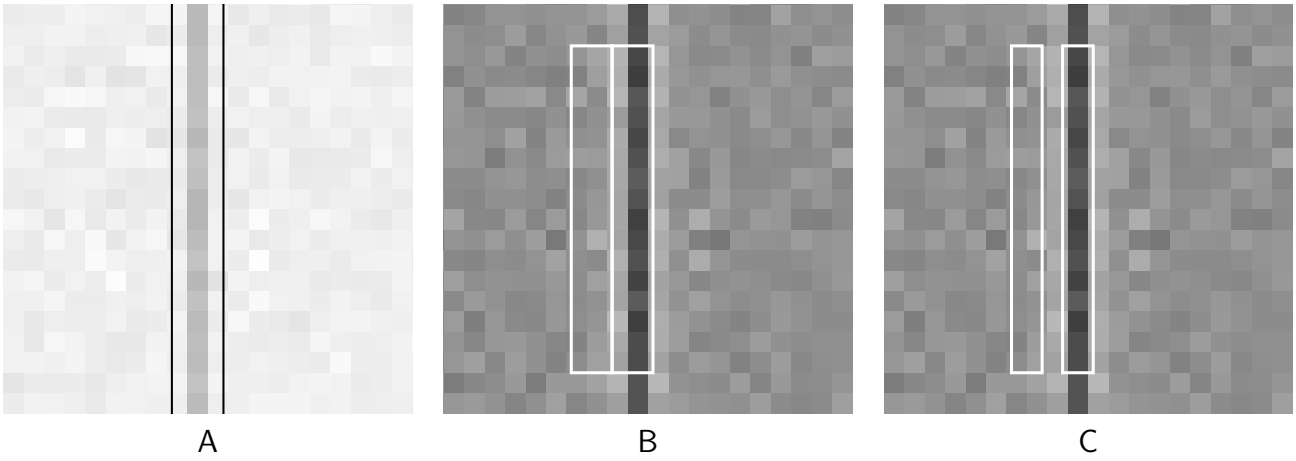

Figure 12: **A:** The Canny/Devernay algorithm usually provides accurate positioning for the edge points. An exception to this happens when thin lines are present as shown here: a gray line, of one-pixel width, is shown over a white background (some noise was added for pixel visualization). Because the line is very thin, the two contours (one at each side) interact during the Gaussian filtering. The resulting contour detection (the two thin black lines) have a lateral offset relative to the expected position. The detected lines are as one would expect from a thicker line. **B:** The difference image D and the operator (in white) corresponding to the left hand contour. Due to the lateral offset, the region that should have dark values also includes as many light ones. **C:** The difference image D and an operator (in white) including a one-pixel width gap between the two regions.

large and well contrasted contours. For low contrast, noisy or short contours, a wider lateral region often helps in the validation. Figure 11 shows an example of the contours validated for each value of lateral width w, highlighting in red the ones only detected for that w. All three values made contributions that would otherwise be missing.

There is an intrinsic problem resulting from using the Canny/Devernay method. Canny/Devernay produces in general accurate edge positioning. There is, however, one condition in which this is not true: when very thin lines are present, see Figure 12A. When a one-pixel width line is present (a dark line over a light background or inversely), the two contours will interact in the Gaussian filtering, leading to a blurred line of larger width. Under these conditions, the contours found by the Canny/Devernay method have a lateral offset relative to the actual position, as shown in the figure. Then, when the arc operator is applied (as shown in Figure 12B, a straight arc in this case), the offset makes that the region that should have dark pixels has as many light as dark ones. The contour is therefore not validated.

This problem does not arise often in natural images, where thin lines with a one pixel width are rare. But the problem appears more often on graphical images, which are a possible target of the proposed method. Thus we devised a simple solution: separating the two regions of the operator by a one-pixel gap, see Figure 12C. Even if the lateral offset is still present, the operator will evaluate pixels keeping the right luminance relation and the contour will be validated. Nevertheless, in non-problematic situations the central pixels may contribute valuable information. For this reason, the two options are kept: operators with and without gap.

From the point of view of the statistical setting, operators with and without the gap are different tests. Thus, when counting the lateral widths, a factor of two will be included to consider these new tests. As a result, we will set

$$W = 6, \tag{13}$$

counting three lateral widths ($\sqrt{2}$, 2 and $2\sqrt{2}$) and each one with and without the gap.

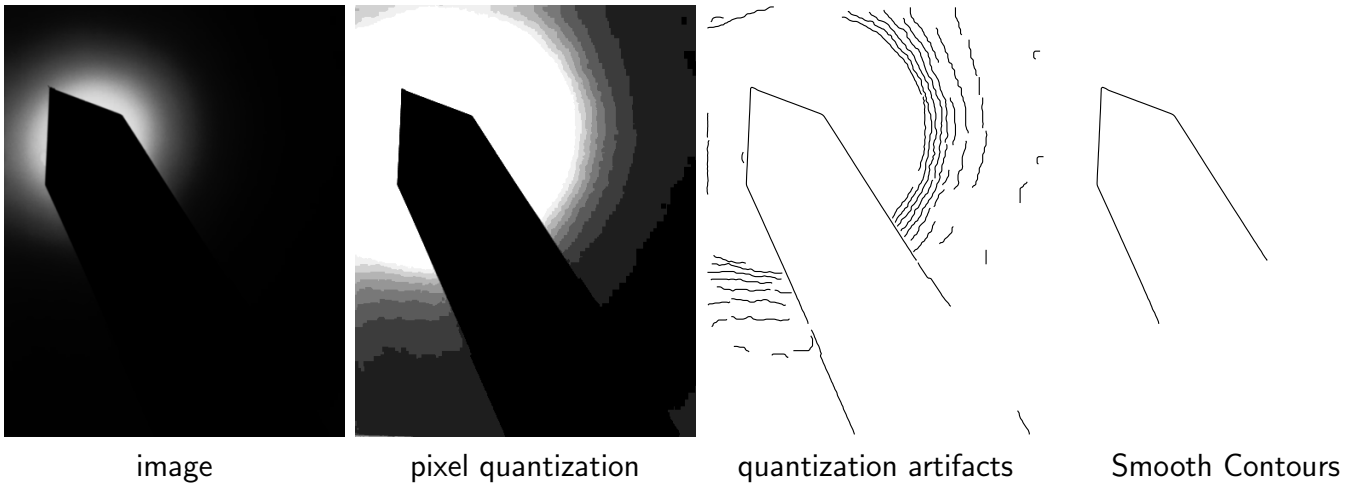

Figure 13: Pixel quantization could produce artifacts in some cases. Left: An image containing almost only soft flat regions. Middle-left: The same image after a contrast change to make visible the effect of quantization. Each zone with constant gray corresponds to pixels with the same value: the black zone on the middle-right are pixels with value zero, the following dark gray band corresponds to pixels with value one, the second gray band corresponds to pixels with value 2, and so on. Middle-right: The almost circular detections are artifacts resulting from the quantization of pixel values. Right: The result of Smooth Contours, using the correcting offset for $Q = 1$ (pixels with integer values) as described in the main text.

This solution is far from perfect. In particular, it does not correct the lateral offset observed on thin lines, which will still be present in the output curves. It would be desirable to find a more fundamental solution to this problem, which may involve improving the Canny/Devernay method. In spite of that, the proposed solution is sound in the sense that it only implies enlarging slightly the family of operators. Moreover, the gap authorizes a transition zone which occasionally improves the detection in some particular cases near the limit of detectability.

More details will be given about the whole candidate selection step in Section 6.

5 Handling Pixel Quantization

The most common digital representation of images is a matrix of integer numbers in the range from zero to 255. This could lead our algorithm to produce some artifacts in soft and regular regions of the image. Figure 13 shows an example. The background, which seems to be black, is actually a soft gradient from black to white, and it is very regular; the level lines are almost circular. In regions that are regular and almost flat, as in the detail shown in the figure, the quantization of pixel values would produce smooth frontiers between regions of the image with values that are consistently different. These are regular regions differing from the *a contrario* model, so it is reasonable for our method to validate contours. Indeed, since the method uses no other information about the pixels than their relative values, such transitions are indistinguishable from, say, visible concentric circles.

The method was designed to work on pixels with real values, where this problem is not present. We provide here a simple workaround which requires knowing the quantization step used in the representation. This is not a serious limitation because assuming a quantization step of 1 is very safe, and in the cases where a different image representation is used one can set this value to the correct one.

The proposed solution is simple. Let Q be the pixel quantization step. That is, the minimal

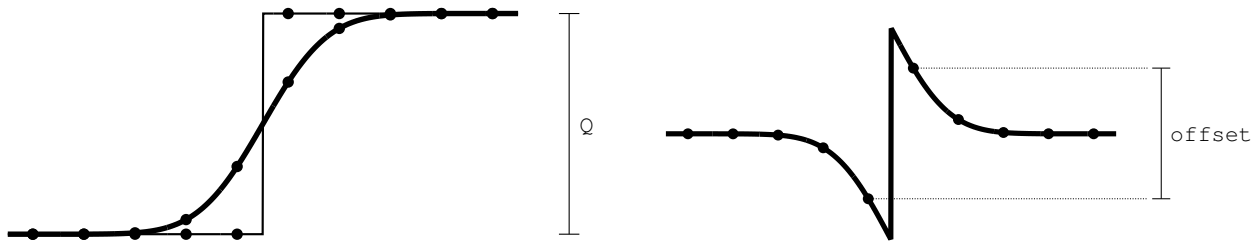

Figure 14: Left: A step of size Q (thin line) and the result after a Gaussian filtering (thick line). The dots represent the samples. Right: The difference between both signals. The offset corresponds to the difference between regions A and B to be observed for a perfect step of size Q.

difference between two possible pixel values. To prevent quantization artifacts, an offset will be added to each of the pixels in the lateral region to the contour candidate that should have lower values (region A in Figure 8). The offset corresponds to the difference between regions A and B that would be observed for a perfect step of size Q. In this way, such a contour will not lead to significant difference in values and the candidate will be rejected. A larger difference is required for validation.

Figure 14 illustrates the profile of a perfect step of size Q and its filtered version by a Gaussian filter (left). The offset is the difference between the two samples adjacent to the contour in the difference of the two signals (right). The value of the offset is twice the value of the filtered sample next to the transition. Its value is obtained by integrating the Gaussian kernel from a half pixel to infinity,

$$\text{offset} = Q \times 2 \times \frac{1}{\sigma\sqrt{2\pi}} \int_{0.5}^{\infty} e^{-\frac{t^2}{2\sigma^2}} dt \approx 0.616793\, Q, \tag{14}$$

where the value was computed for the σ in Equation (1) and its value is proportional to Q. Figure 13 (right) shows the result of applying this strategy.

The value of Q depends on the image representation used. In practice, image compression methods often lead to quantization artifacts with steps as high as two integer values or more. To limit the impact of these artifacts, our implementation uses $Q = 2$ as a default value. When compression artifacts are not present, it is better to use $Q = 1$. Of course, if the method is to be applied on images where the pixels have real values with no quantization, one must set $Q = 0$.

6 The Algorithm

Algorithm 1 describes Smooth Contours. The method takes as input a gray-level image J and its quantization step Q. In the case in which the pixels have integer values, one should set $Q = 1$ or the more conservative $Q = 2$ to cope with image compression artifacts. The output is a list $\mathcal{S}$ of curves, and each curve is represented as a polygonal curve of chained edge points with sub-pixel accuracy.

The algorithm starts by setting ε (line 1), W (line 2), $w_{\max}$ (line 3) and computing the minimal arc length $L_{\min}$ that may lead to a meaningful contour segment as expressed by Equation (12) (line 4 of the pseudocode). Then, the standard deviation σ is computed by Equation (1) (line 5). Its value determines the Gaussian kernel G_σ used for blurring the image (line 6) and the blurred result J_G is subtracted to the input image to produce the difference image D (line 7). The same blurred image J_G is also used as input to the Canny/Devernay algorithm `ChainedEdgeCurves` to compute the chained edge curves (line 8). The Canny/Devernay algorithm is described in detail in a dedicated IPOL article [12] and will not be described here. Any edge detector can be used if it provides a list of curves $\mathcal{C}$, where each curve is described as a list of chained points. The curves thus obtained are the candidate contours to be validated.

Algorithm 1: Smooth Contours

input: An image J ($X \times Y$) with pixel quantization step Q.
output: A list $\mathcal{S}$ of Smooth Contours (chained sub-pixel edge points).

1 $\varepsilon \leftarrow 1$
2 $W \leftarrow 6$ /* Equation (13) */
3 $w_{\max} \leftarrow 2\sqrt{2}$
4 $L_{\min} \leftarrow \min\left\{L : \frac{4+\pi^2}{3} WXY\sqrt{XY}L^2 \cdot P\Big(U \geq (Lw_{\max})^2 \,\Big|\, H_0\Big) \leq \varepsilon\right\}$ /* Equation (12) */
5 $\sigma \leftarrow 0.8\sqrt{1.6^2 - 1}$ /* Equation (1) */
6 $J_G \leftarrow J \star G_\sigma$ /* convolution with a Gaussian kernel */
7 $D \leftarrow J - J_G$ /* difference image */
8 $\mathcal{C} \leftarrow$ `ChainedEdgeCurves`(J_G)
9 `meaningful`$(\mathcal{C}) \leftarrow$ **false** /* initialize all edge points as not meaningful */
10 `used`$(\mathcal{C}) \leftarrow$ **false** /* initialize all edge points as not used */
11 **foreach** *curve* $c \in \mathcal{C}$ **do**
12 **for** $i \leftarrow 1$ **to** `Length`(c) **do**
13 **for** $j \leftarrow$ `Length`(c) **downto** $i + L_{\min}$ **do**
14 **if not** `used`(c_i) **or not** `used`(c_j) **then**
15 $a \leftarrow$ `CircularArc(CurveSegment(`c,i,j`))`
16 **if** $\max_k \left\{ \min_l \{\text{dist}(c_k, a_l)\} \right\} \leq \sigma$ **then** /* is a smooth segment? */
17 `used(CurveSegment(`$c,i{+}3,j{-}3$`))` $\leftarrow$ **true**
18 **for** $w \in \{\sqrt{2}, 2, 2\sqrt{2}\}$ **do**
19 **for** $gap \in \{0, 1\}$ **do**
20 **if** `ArcNFA`$(a, w, gap, D, Q) \leq \varepsilon$ **then** /* Algorithm 2 */
21 `meaningful(CurveSegment(`c,i,j`))` $\leftarrow$ **true**
22 `used(CurveSegment(`c,i,j`))` $\leftarrow$ **true**
23 $\mathcal{S} \leftarrow \{s \subset c \in \mathcal{C} :$ `meaningful`$(s)\}$ /* keep meaningful parts of curves */

Two status flags are associated to each point in the candidate curves: `meaningful` and `used`. The former indicates whether a point was already validated as part of a meaningful contour; it will be used at the end to select the parts of the curves that must be copied to the output. The second flag indicates if the point was already used in a completed validation procedure (whether a rejection or successful validation); it will be used as part of the heuristics to avoid re-evaluating points. Both flags are initialized to false for all the points (lines 9 and 10).

The evaluation loop iterates on each of the candidate curves (line 11). For each one, every possible curve sub-segment will be considered. The segment is determined by two variables i and j, each one of them will be assigned the number of one of the points in the chain. The point with number i is the initial point of the segment and is tried from the first position of the curve (number 1) to the last position of the current curve (its length) in increasing order (line 12). The point with number j ends the segment and is tried starting from the end of the curve and back to the beginning until the length of the segment $(j - i)$ reaches the minimal length $L_{\min}$ (line 13); there is no need to test shorter segments as they cannot lead to a meaningful contour.

Closed curves are represented in the same way as open curves, but with the initial point duplicated again at the end. This simple strategy allows for a common handling of both cases. While usually

Algorithm 2: `ArcNFA`: compute the NFA of a circular arc

input: A circular arc a, a lateral width w in pixels, the size of the operator *gap* in pixels, the difference image D, a quantization step Q.

output: An NFA value.

1 $N \leftarrow \frac{4+\pi^2}{3} WXY\sqrt{XY}L_a^2$ `/* number of tests, Equation (9) */`
2 $A \leftarrow$ `LowArcRing(`a,w,*gap*,D`)`
3 $B \leftarrow$ `HighArcRing(`a,w,*gap*,D`)`
4 $A \leftarrow A + 0.616793\,Q$ `/* offset to handle pixel quantization, Equation (14) */`
5 $u \leftarrow 0$
6 **foreach** $a \in A$ **do**
7 **foreach** $b \in B$ **do**
8 **if** $a < b$ **then**
9 $u \leftarrow u + 1$
10 **else if** $a = b$ **then**
11 $u \leftarrow u + 0.5$
12 $n \leftarrow \#A$ `/* size of region A */`
13 $m \leftarrow \#B$ `/* size of region B */`
14 $\mu_U \leftarrow \frac{nm}{2}$
15 $\sigma_U^2 \leftarrow \frac{nm(n+m+1)}{12}$
16 $\text{NFA} \leftarrow N \frac{1-\text{erf}\left(\frac{u-\mu_U}{\sqrt{2\sigma_U^2}}\right)}{2}$ `/* erf is the standard error function */`

effective, in particular cases this may lead to sub-optimal results. For example, arcs encompassing the curve cut (the duplicated initial and end point) are not evaluated. Parts of some contours may be lost for this reason.

To accelerate the procedure, segments where *both* end points are marked as *used* (`used=true`) will not be evaluated: they probably correspond to a part of a larger segment already validated or already rejected and in most cases the same result would be obtained again (line 14).

When at least one of the end points of the segment is not *used*, the points of the segment from i to j are identified by `CurveSegment` and a circular arc to approximate it is computed by `CircularArc` (line 15). `CircularArc` computes the only circular arc passing by the segment end points and the middle point, that is defined as the point of the segment in position $i+\frac{j-i}{2}$. There are different possible representations for circular arcs and it is not important which one is used; our implementation uses the center of the circle, its radius and two angles to delimit the arc. The direction is stored to identify what should be the dark side of the contour. Also, if the middle point is within a distance σ of the line joining the two endpoints, it is treated as a straight line segment, which is just a particular case of an arc.

The next step is the verification that the computed arc is indeed a good approximation of the segment. We will say that a segment is smooth when each one of its points are within a distance σ of the arc associated to it (line 16). A segment accepted as smooth will be marked as *used*, except for the three points at the beginning of the segment and the last three points of the segment (line 17). As described in Section 4, this is a heuristic procedure that reduces the computation time while keeping most of the valid contours.

As also explained in Section 4, the statistical test is performed for three values of lateral width (line 18) and for each width the operator is tried with and without a central gap (a *gap* of zero or one pixel, line 19). The NFA is computed by `ArcNFA` (line 20) as it is described below and in

Algorithm 3: Alternative computation of u (alternative to lines 5 to 11 of Algorithm 2)

input: Sets of pixel values A and B.
output: The value of the statistic u.

```
1  n ← #A                                              /* size of region A */
2  m ← #B                                              /* size of region B */
3  S ← sort(A ∪ B)
4  for i ← 1 to (n + m) do
5      rank(i) ← i
6  foreach tied group from a to b do                   /* s_a = s_{a+1} = ··· = s_{b-1} = s_b */
7      for i ← a to b do
8          rank(i) ← (a+(a+1)+···+(b-1)+b) / (b-a+1)
9  sum ← 0
10 for i ← 1 to (n + m) do
11     if s_i ∈ B then
12         sum ← sum + rank(i)
13 u ← sum − m(m+1)/2
```

Algorithm 2. If a NFA value equal or smaller than ε is found, all the pixels of the segment are marked as meaningful (line 21) and also as *used* (in this case all of them, including the 3 points at each end, line 22). In our implementation, if the test found a meaningful segment for one width, the w and *gap* loops are ended to save time because it is not necessary to test more arc operators for the same segment.

The last step is just copying each connected chain of meaningful contours to the output list $\mathcal{S}$ (line 23). Note that, as stated before, the output of the algorithm is a set of curves represented as chained sub-pixel edge points and not a set of circular arcs, nor a set of pixels.

Algorithm 2 details the computation of the NFA value, including the u statistic and its p-value. The number of tests is first computed with Equation (9) (line 1). Note that the length L_a of the circular arc a is involved, so N needs to be recomputed in each case.

Now the circular arc operator needs to be applied to the difference image D to obtain the two lateral regions A and B as depicted in Figure 8. The result is two sets of pixels, set A corresponding to what should be the dark side of the contour (`LowArcRing`, line 2), and set B corresponding to the light side (`HighArcRing`, line 3). Sets A and B contain just the values of the pixels inside the corresponding arc rings (repeated as many times as in the arc rings). The two arc rings may be separated by a *gap* (a zero or one pixel separation). The particular implementation depends on the representation chosen for arcs. Line 4 adds the offset computed in Equation (14) to each of the pixel values in region A, which should be the dark side. This offset is added to prevent artifacts due to pixel quantization.

Lines 5 to 11 describe the computation of the Mann-Whitney u statistic. Initially, u is set to zero. Then, every pair of pixels, one from region A and the other from region B, is compared; when the pixel in region B is larger, u is increased one unit; when they are equal, u is increased 0.5 unit. There is an alternative and equivalent method to compute u which is faster for large sets A and B and is described below. To complete the NFA computation, the probability term (the p-value) is computed using the normal approximation (lines 12 to 16), which involves the standard error function

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \tag{15}$$

for which several numerical approximations are known.

Algorithm 3 details an alternative method [19] for computing the statistic u to replace lines 5 to 11 in Algorithm 2. The values of both sets, A and B, are put together and ordered by increasing pixel value by `sort` (line 3). Then, a rank number is assigned to each one, starting from 1 and up to $n+m$, where n is the size of A and m is the size of B (lines 4 and 5). When there are groups of tied values, the ranks are adjusted to an equal midpoint: if the tied group goes from an unadjusted rank value of a to b, the adjusted rank is $\frac{a+(a+1)+\cdots+(b-1)+b}{b-a+1}$, the same for the whole group. For example, if the values in $A \cup B$ are $2, 6, 7, 7, 7, 9, 10.3, 10.3, 15$, the final ranks are $1, 2, 4, 4, 4, 6, 7.5, 7.5, 9$. This adjustment to the ranks is done in lines 6 to 8. Finally, the ranks corresponding to region B are added and u is given by $u = \sum_{i \in B} \texttt{rank}(i) - \frac{m(m+1)}{2}$ (lines 9 to 13). The complexity of the method described previously in Algorithm 2 is $O(nm)$. The bottleneck of the second method is the ordering of the $n+m$ values, which leads to a complexity of $O((n+m)\log(n+m))$. For large sets, the latter is preferred and is the one used in our implementation.

Smooth Contours was designed as an unsupervised method and requires no parameter tuning.

7 Computational Complexity

This section presents the computational complexity analysis of the algorithm. The theoretical analysis will consider the algorithm which uses the Canny/Devernay edge detector to provide candidate curves, but without considering the further heuristics used to reduce the number of curve segments to be actually tested (in other words, without the heuristics based on the flags `used` as described in Section 4). The theoretical complexity will be analyzed in the worst case and in a typical case. Then, the complexity of the complete algorithm, using the full heuristics, will be evaluated empirically.

The first steps of the algorithm, the Gaussian filtering and the chained edge detector, have a computational complexity linear with respect to the number of pixels in the input image. Let us call z the number of resulting curves and let us suppose, to simplify the analysis, that all the curves have the same length l. Each curve is processed by segments. In the worst case, and not using the full heuristics, all the sub-segments are tested. The number of sub-segments of a curve is proportional to the square of the length l, so the total number of segments to process is proportional to $z \cdot l^2$. The processing of each segment is dominated by the computation of the u statistic which, when computed by the method of Algorithm 3, requires sorting the union of both lateral regions. The region size is proportional to the segment length, thus proportional to the length of the curve l. The ordering of l numbers can be done in time proportional to $l \log l$. Thus the computation time is proportional to $z \cdot l^3 \log l$.

Because of the non-maximal suppression step in the Canny/Devernay method, each pixel can be used at most in one curve. So $z \cdot l < p$, where p is the number of pixels in the image (recall that we are assuming that each of the z curves has a length l). The worst computation time is obtained for the case of just one curve using as many pixels as possible. This corresponds to $z = 1$ and l proportional to p. All in all, the worst case computational complexity of the algorithm is

$$\text{Worst Case Complexity:} \qquad O(p^3 \log p), \qquad \text{where } p \text{ is the number of pixels.} \tag{16}$$

The worst case is obtained for very particular images. For example, an extremely elongated image of two rows and $p/2$ columns could lead to a central curve of length $p/2$. An Archimedean spiral covering the whole image would be another case. In more typical cases, however, the length of a curve rarely exceeds several times the diagonal of the image. This case corresponds to l proportional to $\sqrt{p}$, which implies that z is also proportional to $\sqrt{p}$. Then, the typical case complexity of the algorithm is

$$\text{Typical Case Complexity:} \qquad O(p^2 \log p), \qquad \text{where } p \text{ is the number of pixels.} \tag{17}$$

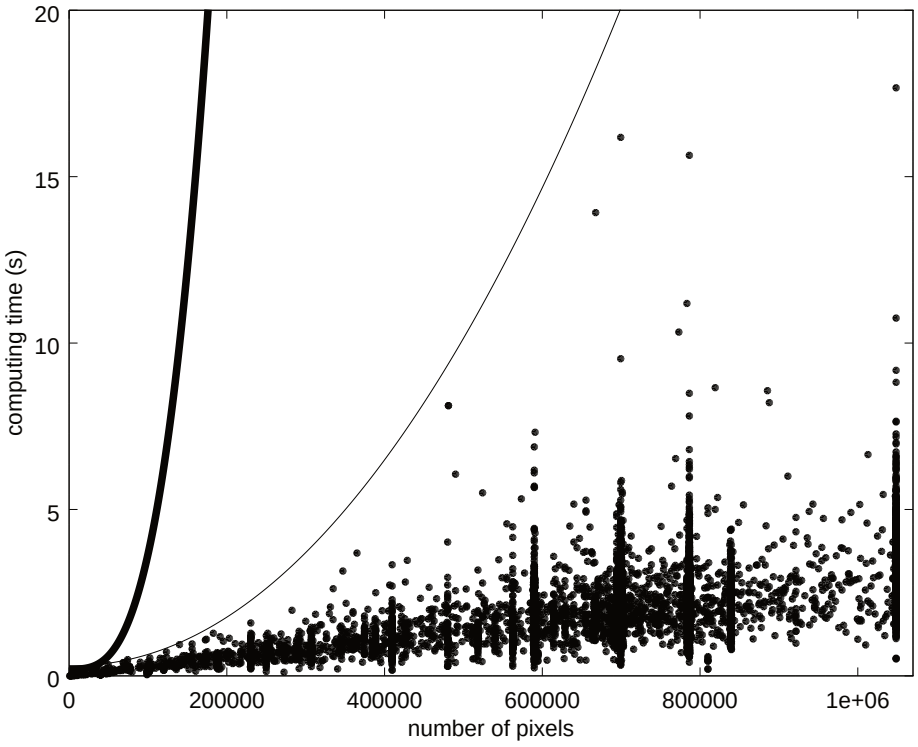

Figure 15: Computation time of Smooth Contours for 10,000 random images from Flickr. Each dot corresponds to one image. The thin line is a possible fit of the theoretical complexity for typical cases. The thick line corresponds to the theoretical complexity in the worst case. In both lines, the proportional factors are arbitrary and its only purpose is to give an idea of the relative rate of growth. The computations were done on a Lenovo X201 with an Intel i5 CPU running at 2.4 GHz.

The heuristics usually manage to accelerate the algorithm considerably. We made an experiment to verify this. Figure 15 shows a plot of the computation time versus the number of pixels for 10,000 images randomly downloaded from Flickr. Each dot represents an image. The thin line is a possible fit of the complexity for typical cases while the thick line corresponds to the theoretical complexity in the worst case. These lines were provided to give an idea of the relative growth rate of time, but their absolute value depends on the proportional constants which here are arbitrary.

The plot shows that for most images the computational time is near-linear with the number of pixels. Thus, in most cases the heuristics result in a reasonable fast algorithm. Nevertheless, one can also see that the heuristics are occasionally less effective. The envelope of the empirical complexity shows a similar profile to the theoretical analysis for typical cases.

8 Experiments

This section illustrates the typical results of Smooth Contours with several examples and discusses the limitations of the method. A better understanding can be obtained by experimenting directly on diverse images using the online demo associated to this publication.

As a sanity check to the *a contrario* methodology used in the design, the first set of experiments concerns noise images where, by design, Smooth Contours should make no detection. Figure 16 shows six such images. The first three satisfy exactly the *a contrario* model H_0 since pixels are

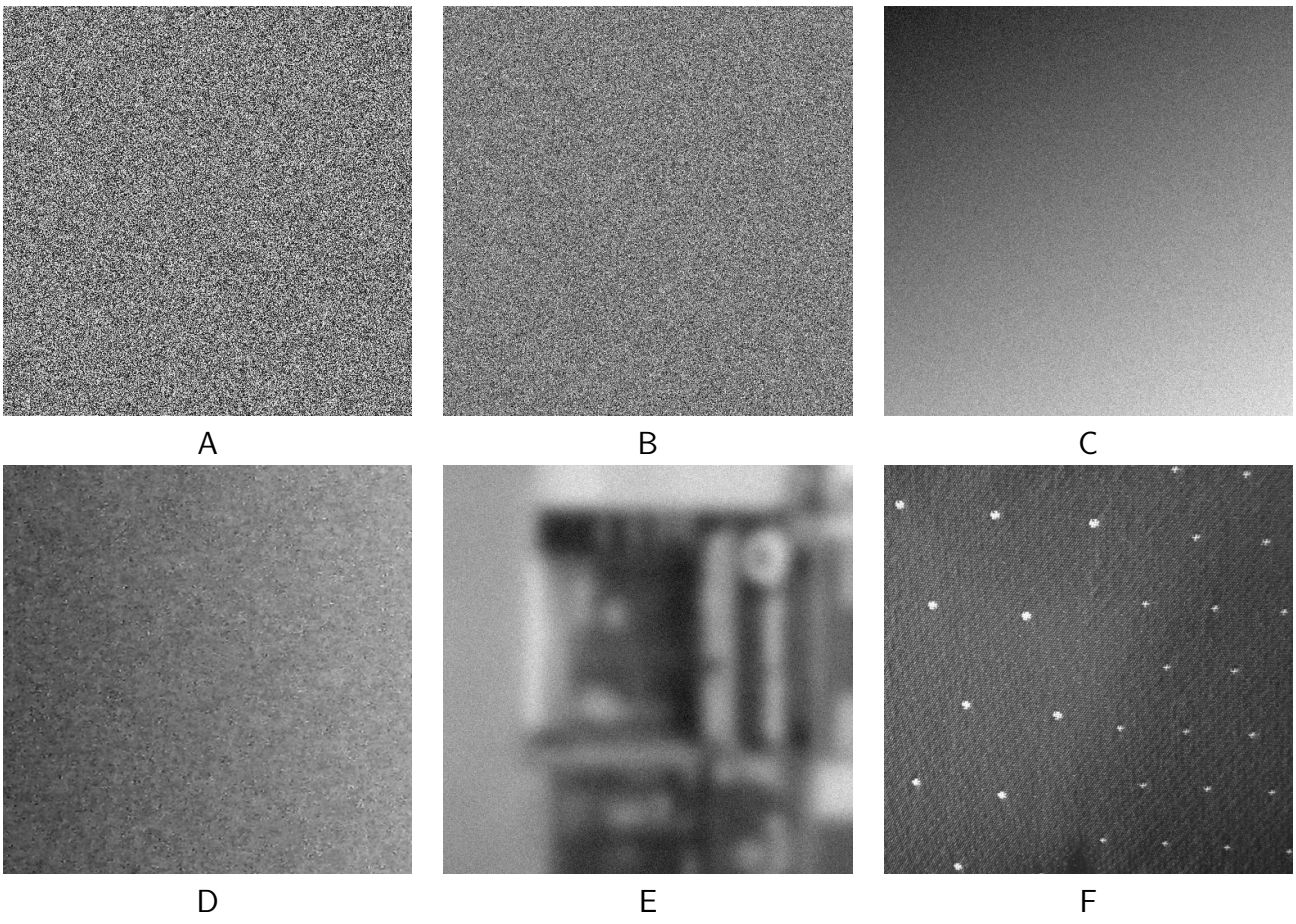

Figure 16: Example images where no detection is produced. **A** Noise image in which pixels are independent and uniformly distributed between zero and 255. **B** Gaussian white noise. **C** Gaussian white noise over a regular gradient. Images from **A** to **C** follow the *a contrario* model H_0 exactly; thus, as expected by design, Smooth Contours produces no detection. **D** An example of real noise from an image. **E** A blurred image with added noise. The model H_0 is locally valid and no detection is produced. **F** A detail of a fabric. The image does not follow H_0 but no detection is produced.

independent and follow the same distribution. The pixels follow a uniform distribution over $[0, 255]$ in **A** and a Gaussian distribution in **B**. In **C** a Gaussian noise was added to a regular gradient. No detection is made on these images and experiments on 1000 random images of each of these kinds showed no detection either. One may wonder why it is not observed in mean $\varepsilon = 1$ detection, as the theory suggests. The reason lies in the heuristics which actually perform a very reduced quantity of the tests relative to N, resulting in a reduced quantity of false detections. This reduction of actual tests does not usually lead to many false positives because the heuristics were designed to propose candidates that have good chances of being good contours. Thus, the reduction of the tests is highly biased towards noise candidates.

One could be tempted to set the number of tests N to the actual number of tests performed. But this would also require changing the *a contrario* model H_0. The heuristics aim at finding good contours, so the lateral regions of actual candidates have complex statistics, very different from H_0. Indeed, the Canny edge detector finds local maxima of the gradient, so the pixel values to each side are expected to be different (although not necessarily significantly different). In the absence of a simple model for these complex statistics, a whole family of potential tests, rich enough to include the actual ones, must be counted.

Figure 17: Some typical results of Smooth Contours.

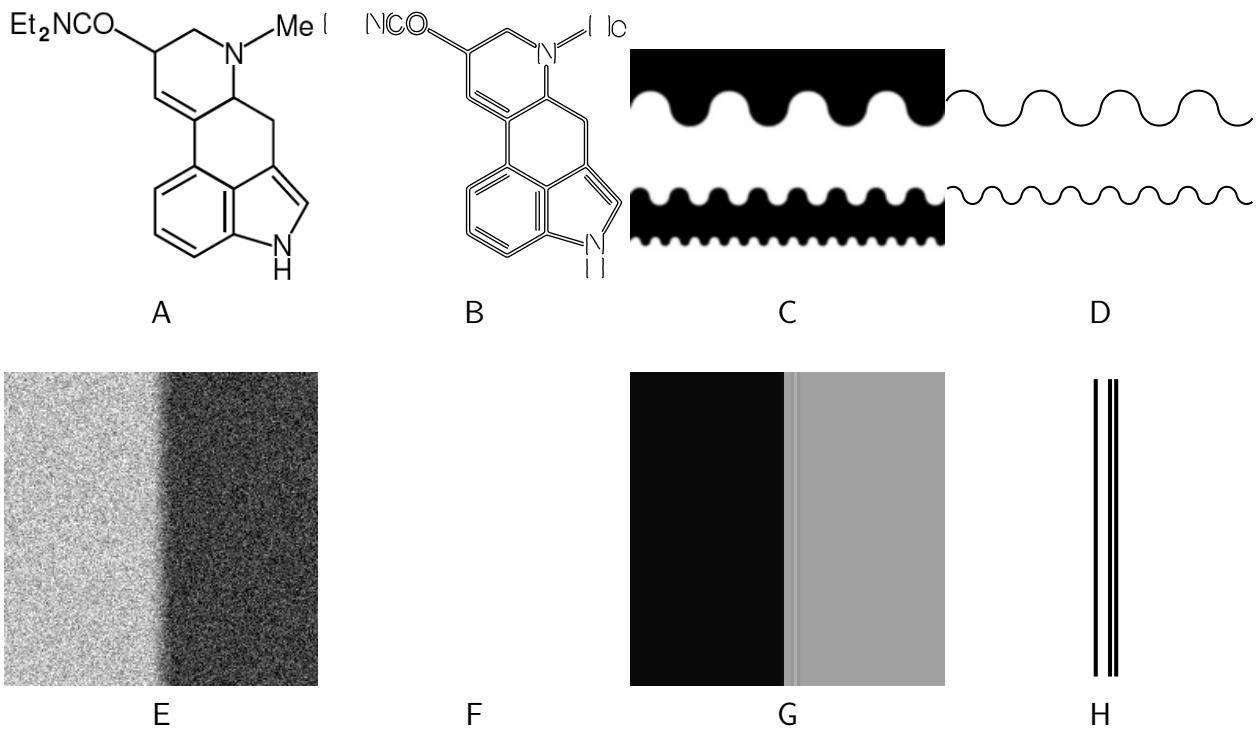

Figure 18: Examples of some features and some limitations of Smooth Contours. **A:** The first example illustrates the fact that Smooth Contours detects transitions from dark to light, so when a dark line is present the result is two curve detections, one on each side as can be seen in **B**. **C:** Three wavelike contours with different period. Because curves are validated by local circular arcs, when the period is small the arc operators may be too small to lead to meaningful contours. This is observed in the lower wavelike contour which is missing in **D**. **E:** A blurred and noisy contour produces no detection in **F**. The contour can be obtained by analyzing the image at a different scale, as shown in Figure 21. **G:** When an image includes artifacts resulting from the Gibbs phenomenon, Smooth Contours will produce detections, as shown in **H**. These detections are surprising but correct.

The second row of Figure 16 shows examples not following H_0 exactly, only approximately. This is usually enough to prevent false detections. The image in **D** is an example of real noise from an image. The model H_0 is locally valid in **E** and again no detection is produced. Finally, **F** shows a detail of fabric where no detection is produced. These examples show that H_0 is an adequate model to prevent contour detection on zones where pixels are not organized according to contours.

The next set of experiments shows typical results on four natural images, see Figure 17. Most complex images include cases of both, the strengths and limitations of the method. In what follows, we will describe with simpler images the particular phenomena, but many of them are also observed in some details of these complex images. At first view, the main geometry of these images was captured by the contours found. On the other hand, there are also many structures that did not lead to detections. To mention the most important, non-smooth contours as the shore on the first image or part of the profile of the Rauk (the rock) on the third one. Small details, where the approximating arc operators are not large enough, are rejected and this leads to missing structures as in the face of the man on the second image or the numbers on the last one. Finally, blurred structures, as the curved shadows on the last image, are also missed.

The Canny/Devernay algorithm used to propose contour candidates is responsible for some of the missing detections. An example is the boundary on the left side of the base of the Rauk (the rock) on the third row of Figure 17. If one candidate were proposed for that part, it would have been

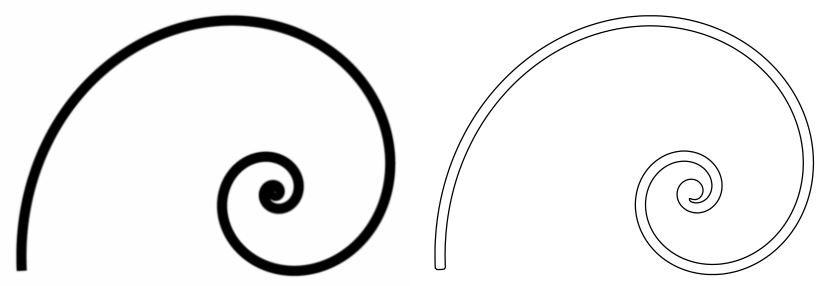

Figure 19: An image showing a logarithmic spiral (left) and the result of Smooth Contours (right). No segment of the curve corresponds exactly to a circular arc. To be validated, however, it is enough that the curve is well approximated locally by arc operators. As a result, the full contour is validated in this case.

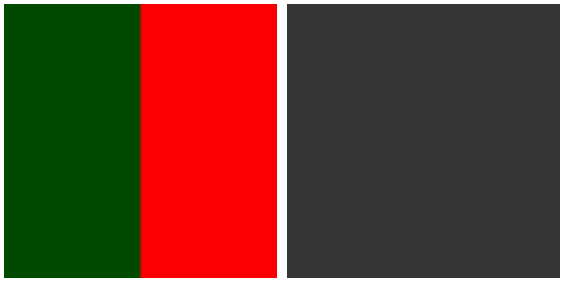

Figure 20: A color image (left) in which, when transformed to gray-level (right) both colors are transformed into the same gray value, thus no contour is visible on the gray-level image and Smooth Contours cannot detect it.

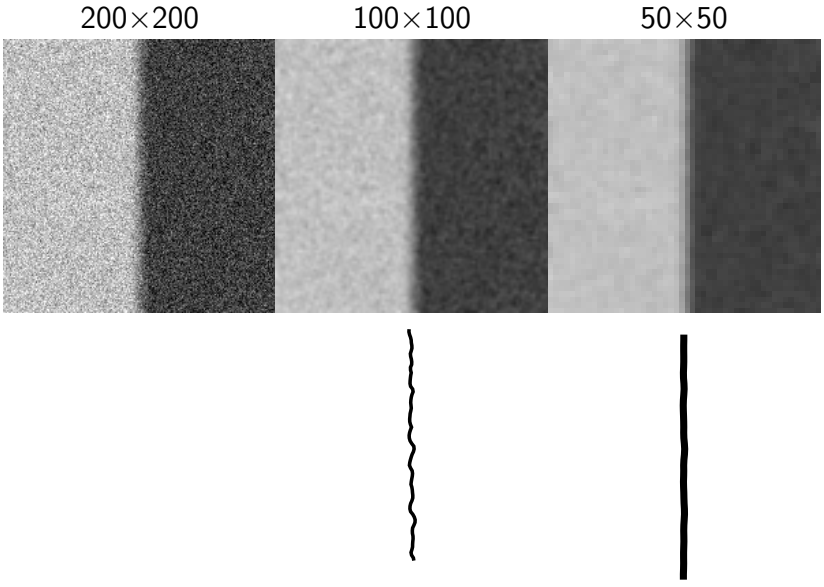

Figure 21: A blurred and noisy contour analyzed at three scales. The images are on the first row and the results are on the second row. Please note that, even if displayed at the same size on this page, the images have different resolution; this difference can be seen in the size of the pixels and on the width of the detected lines. Smooth Contours made no detection at full scale but the boundary is detected at half and quarter scales.

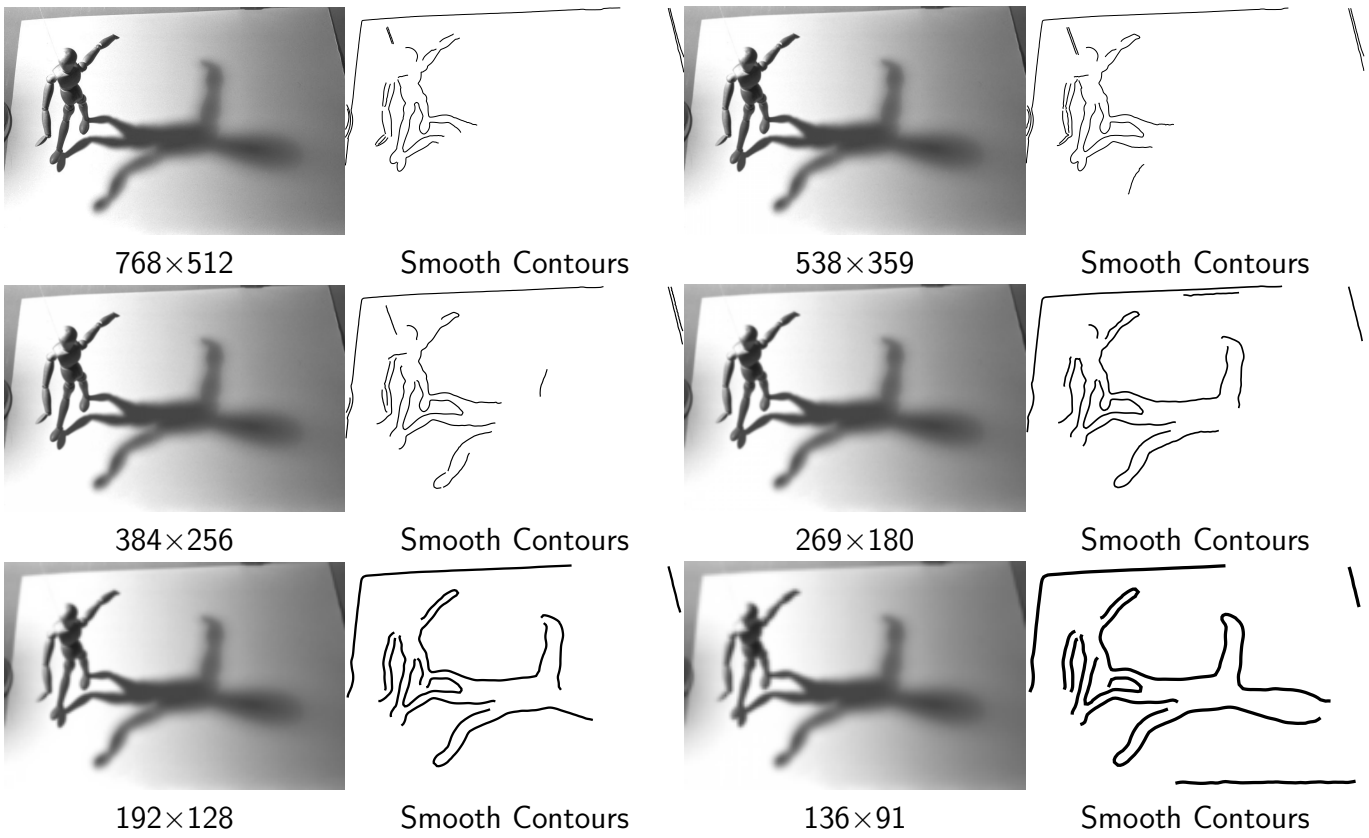

Figure 22: The "mannequin" image analyzed by Smooth Contours at multiple scales. The images are on the first and third columns and the results of Smooth Contours are on the second and fourth columns. Please note that, even if displayed at the same size on this page, the images have different resolution; this difference can be seen in the size of the pixels and on the width of the detected lines. The image was taken from [5].

validated as it is regular and well contrasted. The curve provided by the Canny/Devernay algorithm is cut into three pieces, each one bifurcating into the background. On the right side of the Rauk, however, the contour would not have been validated by the current method even if a candidate was provided. Indeed, the difference between both sides of the boundary is greater in texture than in mean luminance. The use of other statistical tests, the Kolmogorov-Smirnov test for instance, that evaluates whether the distributions are different (and not just whether the mean values differ) may help to improve the results in these situations.

Figure 18 illustrates some features and limitations with geometric examples. The line drawing in **A** results in two contours per line, one on each side, see **B**. Like most edge detectors, Smooth Contours detects transitions from dark to light and lines have two of them, one on each side. Future work will concentrate on an algorithm using similar ideas but focused on detecting dark or light *lines* instead of luminance transitions. One can also observe here the pervasive fact that small details are missing, as the detail of many of the letters. Some of these curves are small, like the inner contour of the lowercase e. A particular case of the same limitation is observed in the bottom of the curtain in Figure 2: the alternation of dark and light color relative to the background cuts the boundary into small pieces at each inversion of contrast (the explanation is the same for the case observed in the bottom right of the propeller in Figure 23 fourth row).

The next image **C** illustrates a case where a long contour is missing due to not being smooth in the particular sense required by the method. On the image one can see three wavelike contours with

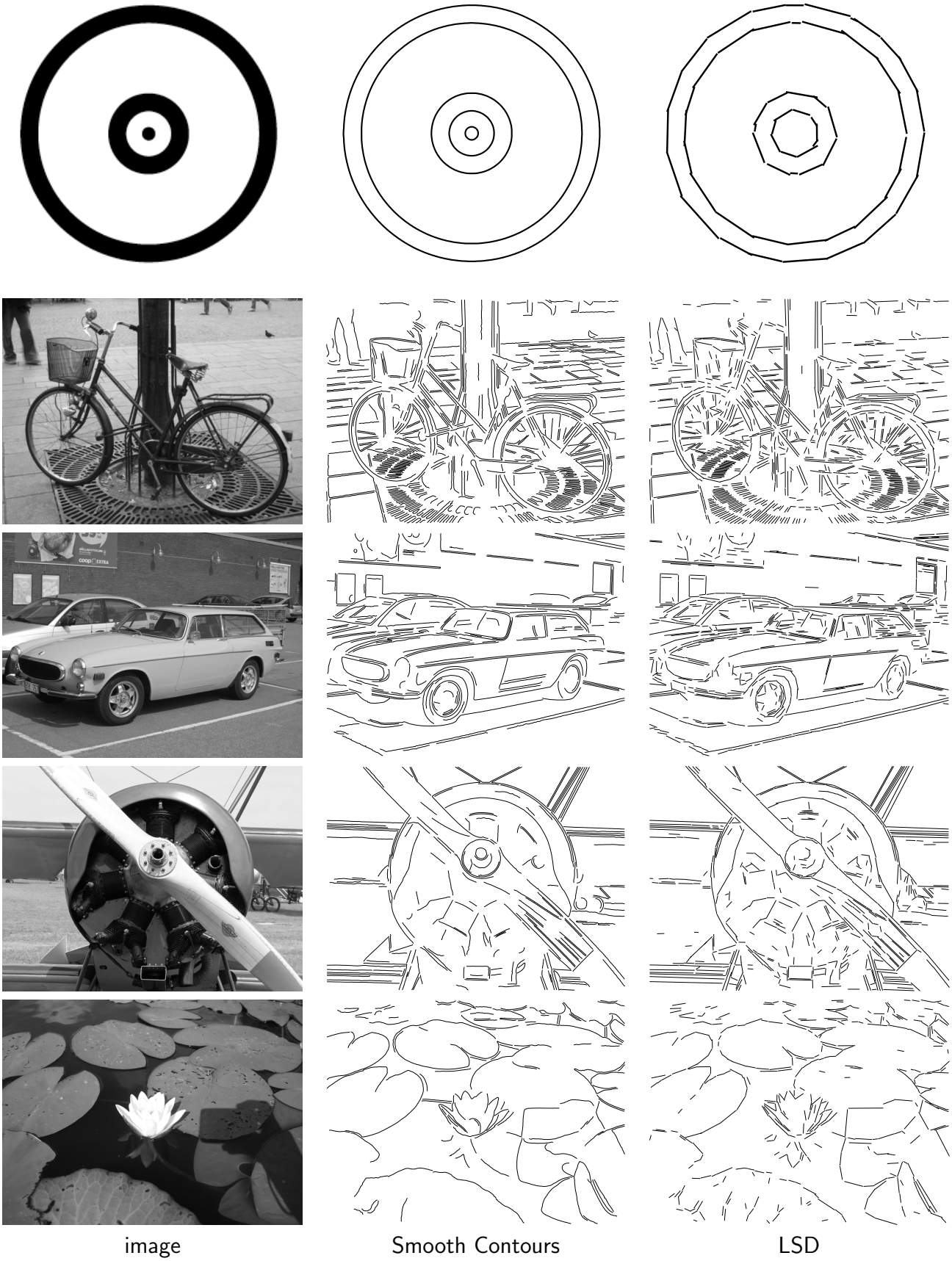

Figure 23: Smooth Contours versus LSD [10, 11]. Both algorithms give similar results, the main difference in the results is that Smooth Contours is able to handle curves while in such cases LSD can only produce a polygonal approximations.

Figure 24: The result of Smooth Contours on the last image of Figure 23 with one of the curves colored in red. As this example shows, the fact that the curve is locally smooth does not prevent it from having some kinks and irregularities.

different period. The two with larger period are detected, see **D**. But the one with a small period can only be approximated by circular arcs that are too small and the validation fails. The reason is that the operator, due to the particular family of curves used, can only observe a very restricted part of the contour. The use of a richer family of curves, Bézier curves for example, may help to improve the detectability. At the same time, a richer family of curves results in a larger number of tests N, which increases the minimal size that can lead to detections, so small details not belonging to larger curves may be lost even more. A compromise is needed. This problem will be studied in future work. Figure 19 shows a logarithmic spiral, which also cannot be cut into circular arcs. Nevertheless, an approximate fit of the arc operator to a curve segment is enough for validation and the full contour is obtained.

Blurred or noisy contours will not be detected, as shown in Figure 18E. Nevertheless, as will be shown below, the contours may be obtained by analyzing the image at a different scale. The images that have been compressed with algorithms like JPEG sometimes present artifacts resulting from the Gibbs phenomenon. The image in Figure 18G is an example. In such cases, Smooth Contours makes correct but sometimes surprising detections, see **H** (as explained before, in an attempt to minimize these artifacts, our implementation uses $Q = 2$ as the default quantization step). Color images also lead occasionally to correct but unexpected results. The algorithm works on gray-level images, so color images need to be converted. In this conversion some contours may be lost, as Figure 20 shows, when different colors are transformed to the same gray-level. The input to Smooth Contours would be the image on the right, so it is natural to get no detection.

The Marr and Hildreth edge detection theory proposed analyzing images at multiple scales. The proposed method assumes that the input image was already down-sampled to the scale to be analyzed. But a multiple scale analysis is still possible. Moreover, doing so helps to detect structures that are noisy or blurry at full scale. Figure 21 shows a simple example. The full scale image contains a blurred and noisy step which is not detected by Smooth Contours (left). The step is detected at half

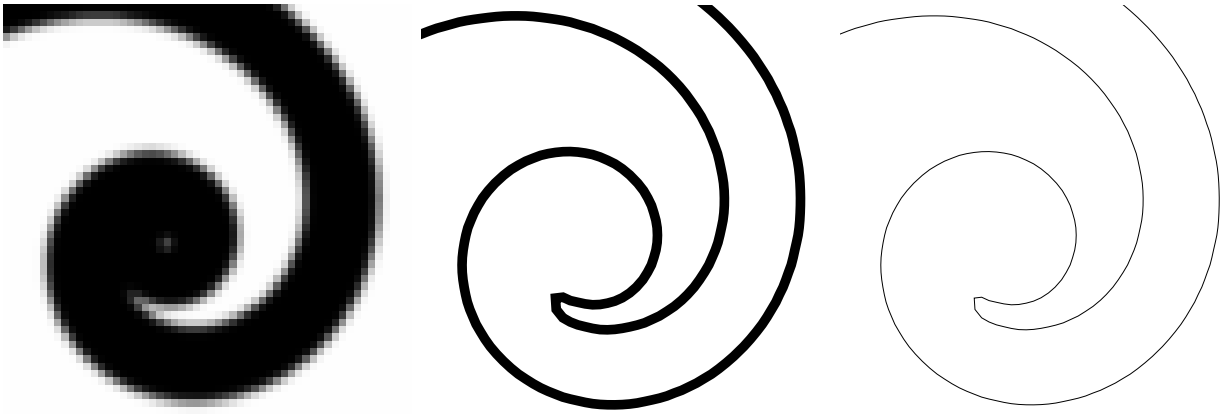

Figure 25: A detail of the experiment on Figure 19 magnified approximately seven times. Left: A detail of the input image. Middle: A detail of the same part of the output of Smooth Contours as shown in Figure 19. The width of the line was magnified accordingly. Note that, due to the sub-pixel accuracy of the method, even if the result is a polygonal curve, it describes well the geometry on the image. Right: The same detail as before but reducing the line width used in the drawing. This shows that the result is indeed a vectorization of the curves by a polygonal curve. Indeed, the resulting curves can be used or drawn at any desired scale or resolution.

scale (center), even if the resulting curve is somehow irregular. At a quarter scale (right) the contour is well detected. Figure 22 shows a classic image in multi-scale analysis, where the shadow of the mannequin is blurred and the more blurred the further away from it. The shadow is not detected at full scale, but it is more and more detected as the resolution is diminished. Some details are also lost as they become small. The task of merging the different scales is left for future work.

Our next set of experiments compares Smooth Contours to the LSD algorithm [10, 11] for Line Segment Detection. The aim of this comparison is to illustrate the similar behavior of the two methods. This is not surprising as both methods share a similar approach, while the latter focuses on a particular kind of contours: straight lines. Even if the details are very different, both algorithms are based on the *a contrario* approach using similar H_0 models and both are unsupervised. Figure 23 shows the results for five images. In most cases the detections are produced by the same regions of the images and differences are observed mainly when curves are present. LSD approximates curves by line segments, producing polygonal approximations, while Smooth Contours finds the curves as lists of chained edge points with sub-pixel accuracy.

We will end this section with two comments concerning the output of Smooth Contours. The method detects contours that are locally smooth. Nevertheless, this does not prevent the resulting curves from having some kinks and irregularities. Figure 24 shows an example. The curve marked in red was detected by Smooth Contours as a full chain. In spite of some irregularities, each point of the curve is part of a segment which was validated by an arc operator. No validated segment spanned from one side to the other of a kink; some validated segments spanned on one side up to the kink while other segments validated up to the kink on the other side. The profile of the obelisk in Figure 13 is a similar example.

The second point concerns the sub-pixel accuracy of the result. The output of Smooth Contours is a list of curves defined as chained edge *points*. But 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 contour. The resulting curve has much more resolution than the image. Figure 25 illustrates the point by showing a detail of the example in Figure 19 magnified about seven times. On the middle, one can see the magnified detail of the result of Smooth Contours as displayed in Figure 19,

so the line width is also magnified about seven times. The line width used for drawing is arbitrary; the result of the method is just the coordinates that define the polygonal curve. Thus, by drawing with a thinner line width[6] (right) one can see how the curve is much more detailed than the pixel resolution. The polygonal curve can be drawn or used at any resolution or scale. One can also see on the figure that the result is not perfect. It would be desirable, of course, to obtain a vectorization in terms of more complex geometric elements, circular arcs or even Bézier curves, which define the curve in a more compact way, using less parameters. Nevertheless, the sub-pixel accuracy polygonal curve is already a vectorization of the contours.

9 Conclusion

We proposed an unsupervised algorithm for smooth contour detection on digital images that produces accurate results and has a small amount of false positive detections. The progress can be ascribed to the association of the edge detection theory of Marr and Hildreth with the general method to eliminate false positives proposed by Desolneux, Moisan and Morel. The experimental section shows promising results but also limitations, illustrating the need of much future work.

There is plenty of room for improvements in the heuristics, both to correct some artifacts but also to improve the speed. The use of other or improved edge detectors to provide the candidate curves may solve some of the limitations of the Canny/Devernay algorithm. In particular, the position problem with thin lines. Also, the potential use of optimal algorithms to cut curves into circular arcs must be explored to replace the current heuristics used to avoid the exhaustive evaluation of all curve segments. This may lead at the same time to a faster algorithm and better vectorization of the output contours as chained circular arcs instead of sub-pixel polygonal curves.

The validation step could be improved by incorporating richer families of curves (Bézier curves for instance), by using different statistical formulations as the Kolmogorov-Smirnov test, or by using a null hypothesis incorporating the actual distribution of noise measured on the image. The right choice of these options may lead to a better balance between fidelity to the contours actually present and the level of details that can be effectively detected. Another potential gain is the ability to detect boundaries between zones with different textures.

On a wider scope, the method needs to be extended into a multi-scale image analysis tool, able to extract automatically structures at different scales as well as selecting the right scale for each one. A further extension would incorporate more general geometric structures such as dark or light lines (instead of luminance transitions), handling T-junctions, spots, etc. The ultimate goal would be to approach the automatic generation of a primal sketch.

Acknowledgments

We thank Jean-Michel Morel, Irwin Scollar, Federico Lecumberry, Viorica Pătrăucean 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), and ANR-DGA project ANR-12-ASTR-0035.

[6]This can be done for any image. Moreover, the implementation of the algorithm that is part of this publication can draw the output curves into a PDF file and the line width can be set to any value with an optional parameter.

Image Credits

  From the RuG image database [8].

 Margaret Randall in Albuquerque (1957) by Eddie Johnson.

 Miguel Colom

 From [5].

All other images by the authors.

References

[1] M. K. Albert and D. D. Hoffman, *Genericity in spatial vision*, in Geometric Representations of Perceptual Phenomena: Articles in Honor of Tarow Indow's 70th Birthday, D. Luce, K. Romney, D. Hoffman, and D'Zmura M., eds., Erlbaum, 1995, pp. 95–112.

[2] J. Canny, *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.

[3] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis, a Probabilistic Approach*, Springer, 2008.

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

[5] J. H. Elder and S. W. Zucker, *Local scale control for edge detection and blur estimation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (1998), pp. 699–716. http://dx.doi.org/10.1109/34.689301.

[6] A. Gordon, G. Glazko, X. Qiu, and A. Yakovlev, *Control of the mean number of false discoveries, Bonferroni and stability of multiple testing*, The Annals of Applied Statistics, 1 (2007), pp. 179–190. http://dx.doi.org/10.1214/07-AOAS102.

[7] R. L. Gregory, *Helmholtz's principle*, Perception, 36 (2007), pp. 795–796. http://dx.doi.org/10.1068/p3606ed.

[8] C. Grigorescu, N. Petkov, and M. A. Westenberg, *Contour detection based on nonclassical receptive field inhibition*, IEEE Transactions on Image Processing, 12 (2003), pp. 729–739. http://dx.doi.org/10.1109/TIP.2003.814250.

[9] R. Grompone von Gioi, *A Contrario Line Segment Detection*, Springer, 2014.

[10] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *LSD: A fast Line Segment Detector with a false detection control*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32 (2010), pp. 722–732. http://dx.doi.org/10.1109/TPAMI.2008.300.

[11] ——, *LSD: a Line Segment Detector*, Image Processing On Line, (2012). http://dx.doi.org/10.5201/ipol.2012.gjmr-lsd.

[12] R. Grompone von Gioi and G. Randall, *A sub-pixel edge detector: an implementation of the Canny/Devernay algorithm*, Image Processing On Line, (In preparation).

[13] Y. Hochberg and A. C. Tamhane, *Multiple comparison procedures*, John Wiley & Sons, New York, 1987.

[14] D. H. Hubel and T. N. Wiesel, *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex*, The Journal of Physiology, 160 (1962), pp. 106–154. http://dx.doi.org/10.1113/jphysiol.1962.sp006837.

[15] H. B. Mann and D. R. Whitney, *On a test of whether one of two random variables is stochastically larger than the other*, Annals of Mathematical Statistics, 18 (1947), pp. 50–60. http://dx.doi.org/10.1214/aoms/1177730491.

[16] 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.

[17] J. M. Morel and G. Yu, *Is SIFT scale invariant?*, Inverse Problems and Imaging, 5 (2011), pp. 115–136. http://dx.doi.org/10.3934/ipi.2011.5.115.

[18] 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.

[19] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, third ed., 2004.

[20] J. Wagemans, *Perceptual use of nonaccidental properties*, Canadian Journal of Psychology, 46 (1992), pp. 236–279. http://dx.doi.org/10.1037/h0084323.