Meaningful Scales Detection: an Unsupervised Noise Detection Algorithm for Digital Contours

Bertrand Kerautret, Jacques-Olivier Lachaud

Published in Image Processing On Line on 2014–05–07.
Submitted on 2013–02–13, accepted on 2013–04–02.
ISSN 2105–1232 © 2014 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2014.75

Meaningful Scales Detection: an Unsupervised Noise Detection Algorithm for Digital Contours

Bertrand Kerautret[1] , Jacques-Olivier Lachaud[2]

[1] LORIA, Université de Lorraine (France) (bertrand.kerautret@loria.fr)
[2] LAMA, Université de Savoie (France) (jacques-olivier.lachaud@univ-savoie.fr)

Abstract

This work presents an algorithm which permits to detect locally on digital contours what is the amount of noise estimated from a given maximal scale. The method is based on the asymptotic properties of the length of the maximal segment primitive.

Source Code

The implementation of the algorithm is available through the ImaGene[1] library framework. A special version of this library is given without *boost* and *gmp* dependencies. The source code and the online demonstration are accessible at the IPOL web page of this article[2].

Keywords: noise estimation; meaningful scale detection

1 Overview

The estimation of the meaningful scale of a digital contour is a difficult problem that can have important impacts for numerous applications which are dependent of a supervised noise parameter. For instance, it is the case in various curvature estimators [4, 9, 10] or in contour polygonalisation algorithms [1, 11].

This work presents an algorithm which permits to detect locally what is the amount of noise estimated from a given maximal scale. The method is based on the asymptotic properties of the length of the maximal segment primitive [7]. The presented algorithm is related to previous work which was first presented in conference [5] and then further extended [6].

[1] *ImaGene*, {Gen}eric Digital {Ima}ge Library, http://gforge.liris.cnrs.fr/projects/imagene
[2] http://dx.doi.org/10.5201/ipol.2014.75

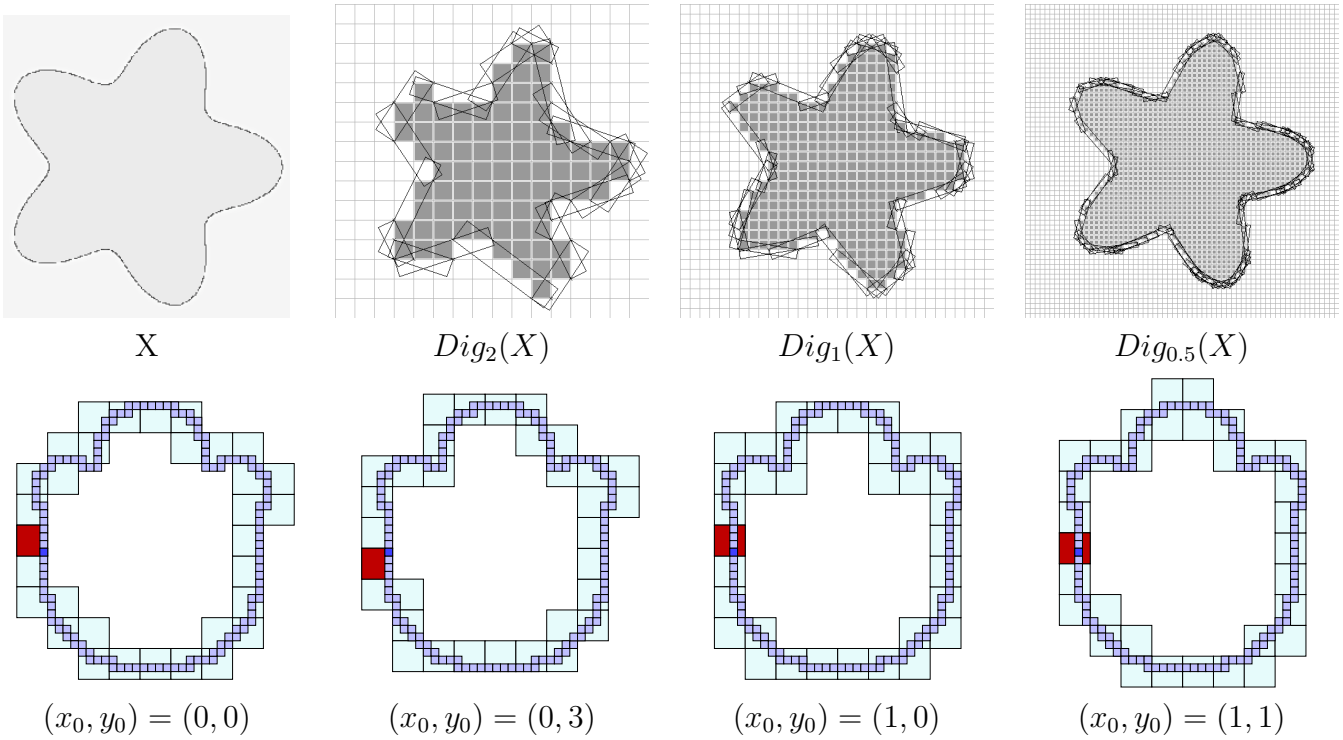


Figure 1: First row: Illustration of the digitization process $Dig_h()$ applied on the real shape X with the set of maximal segments covering the digital contours. Second row: Illustration of several subsampling with the same digitization grid size but with different shifts (x_0, y_0).

2 Method

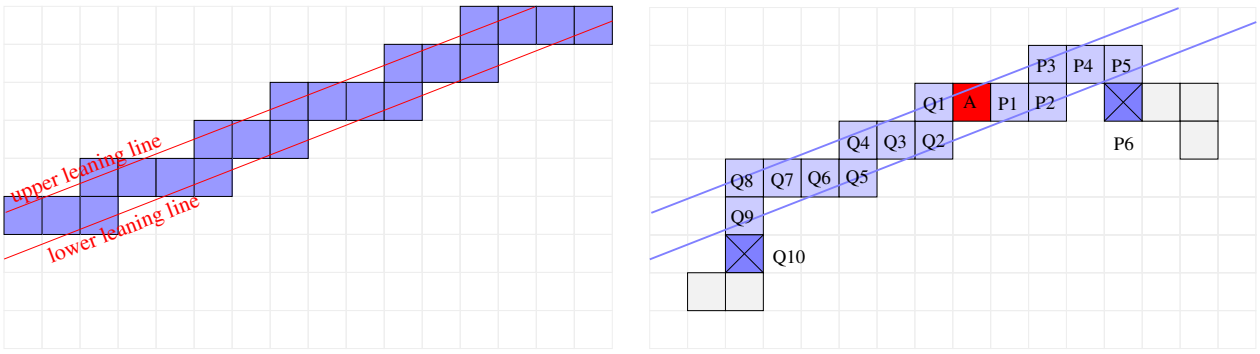
The method is based on the study of the properties of the maximal digital straight segment which is a classic primitive used in the field of discrete geometry. This primitive is for instance used for tangent or curvature estimators. As mentioned in the introduction, the length of the maximal segments is exploited to detect a meaningful scale through a given maximal scale. An illustration representing this primitive with finer and finer scale is given in figure 1 (first row) where maximal segments are represented by the boxes. Before describing the asymptotic property on the length of this primitive, we briefly recall its classic definition.

2.1 Primitive of the Maximal Digital Straight Segment

Definition of a Standard Digital Straight Line (DSL) *A Standard Digital Straight Line (DSL)* is some set $\{(x, y) \in \mathbb{Z}^2, \mu \leq ax - by < \mu + |a| + |b|\}$, where (a, b, μ) are also integers and $\gcd(a, b) = 1$. The real lines of equation $ax - by = \mu$ and $ax - by = \mu + |a| + |b| - 1$ are respectively the lower and upper leaning lines (as illustrated in figure 2 (a)).

Definition of a Maximal Straight Segment A *Digital Straight Segment (DSS)* is a 4-connected piece of DSL. The interpixel contour of a simple digital shape is a 4-connected closed path without self-intersections. A *maximal segment* M of a 4-connected path C, is a subset of C that is a DSS and which is no more a DSS when adding any other point of $C \setminus M$.

The figure figure 2 (b) illustrates the recognition process of a maximal segment starting from the point A. The points P1, Q1, P2, Q2, P3, Q3, P4, Q4, P5, Q5, Q6, Q7, Q8, and Q9 are added alternately to the front and to the back of the current segment.



(a) (b)

Figure 2: Illustration of a Digital Straight Line (a) and example of a maximal segment recognition process (b).

Extraction of Maximal Straight Segments through the ImaGene Library The maximal segment recognition algorithm is included in the ImaGene[3] library. The class C4CTangentialCover[4] is defined to compute all the set of maximal segments (class MaximalSegment[5]) associated to a digital contour (a 4-connected contour represented by the class C4CIterator[6]).

2.2 Asymptotic Properties of Maximal Straight Segments

The previous primitive presents some particular asymptotic properties with its length in number of steps (or number of pixel minus one). The asymptotic behavior is defined by considering different digitization steps h of a real shape X.

More precisely, theorem 5.26 of a research report by Lachaud [7] (in chapter 5) indicates that given a real shape X, if we consider a digitization process $Dig_h(X) = X \cap h\mathbb{Z} \times h\mathbb{Z}$ and if the shape X has a C^3 boundary and is strictly convex, then the smallest discrete length of the maximal segment of $Dig_h(X)$ is some $\Omega(1/h^{1/3})$. The longest discrete length of the maximal segments on the boundary of $Dig_h(X)$ is some $O(1/h^{1/2})$. The experiments of figure 3 (a) confirm such a behavior.

However, since the shapes obtained with smaller and smaller digitization grid sizes are in general not available, we choose another strategy by subsampling the initial shape to obtain coarser and coarser shapes. The second line of figure 1 shows an example of several subsamplings defined with various shifts (x_0, y_0). The lengths are thus defined for a given subsampling (characterized by the size h_i and shift x_0, y_0) by the average. If we denote by $(L^{h_i,x_0,y_0}_j)_{j=1..l_i}$ the discrete lengths of the set of l_i maximal segments covering P, the average length $\overline{L}^{h_i}$ is defined by:

$$\overline{L}^{h_i} = \frac{1}{i^2} \sum_{0 \leq x_0, y_0 < i} \frac{1}{l_i^{x_0,y_0}} \sum_j L_j^{h_i,x_0,y_0},$$

where $l_i^{x_0,y_0}$ represents the number of maximal segments containing the subsampled point P.

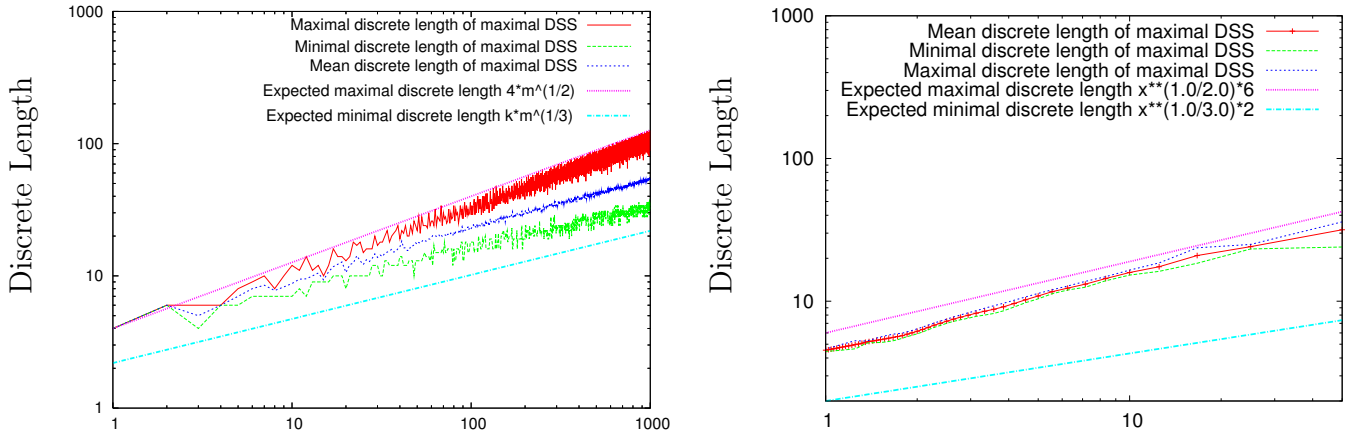
By using this strategy we can observe a similar behavior as shown on the experiments of figure 3 (b).

[3] http://www.lama.univ-savoie.fr/~lachaud/ImaGene/doc/html/index.html

[4] http://www.lama.univ-savoie.fr/~lachaud/ImaGene/doc/html/classImaGene_1_1C4CTangentialCover.html

[5] http://www.lama.univ-savoie.fr/~lachaud/ImaGene/doc/html/structImaGene_1_1C4CTangentialCover_1_1MaximalSegment.html

[6] http://www.lama.univ-savoie.fr/~lachaud/ImaGene/doc/html/classImaGene_1_1C4CIterator.html



(a) Digitization size (h) (b) Subsampling size

Figure 3: Experiments on measures of the length of maximal segments obtained from a flower shape for finer and finer grid size (a). Experiments on measure of the length of maximal segments obtained from a flower shape for different subsamplings (b).

2.3 Meaningful Scale Detection from the Previous Property

To exploit the previous property, we introduce the concept of multiscale profile defined for a contour point. A multiscale profile $\mathcal{P}_n(P)$ is defined as the sequence of samples $(\log(i), \log(\overline{L}^{h_i})_{i=1..n})$. The construction of such a multiscale profile is illustrated in figure 4 (a) with the contour sampled with grid size equal to 8 (represented in light blue).

A **meaningful scale** of a sequence $(S_i, T_i)_{1 \leq i \leq n}$ is defined as a pair (i_1, i_2) such that:

$$\forall i, i_1 \leq i < i_2, \frac{T_{i+1} - T_i}{S_{i+1} - S_i} \leq t_m,$$

and the preceding property is not true for $i_1 - 1$ and i_2. The parameter t_m is related to the previous theorem and this value has few influence towards the detection quality (see annex in the paper by Kerautret et al. [6] or section 5.2). The default value giving slightly better result was set to $t_m = 0$. Note that from this definition the minimal length of the interval (i_1, i_2) is one. Increasing this minimal value does not really improve the meaningful scale detection as you can see in section 5.2.

The **noise level** is defined as the first index for which the multiscale profile is going to decrease (i.e.) i_1. Figure 4 (c) illustrates the multiscale profile obtained on a point of a noisy contour part. The curve is first going the increase and then recover a normal behavior from the scale i_1. The noise level of this point is thus equal to i_1.

3 Algorithm

The algorithm for meaningful scale detection is composed of three main steps:

1. Subsampling the digital source contour to obtain the multiscale representation.
2. Tangential cover computation on all digital subsampled contours.
3. Computation of the multiscale profile and deduction of the noise level.

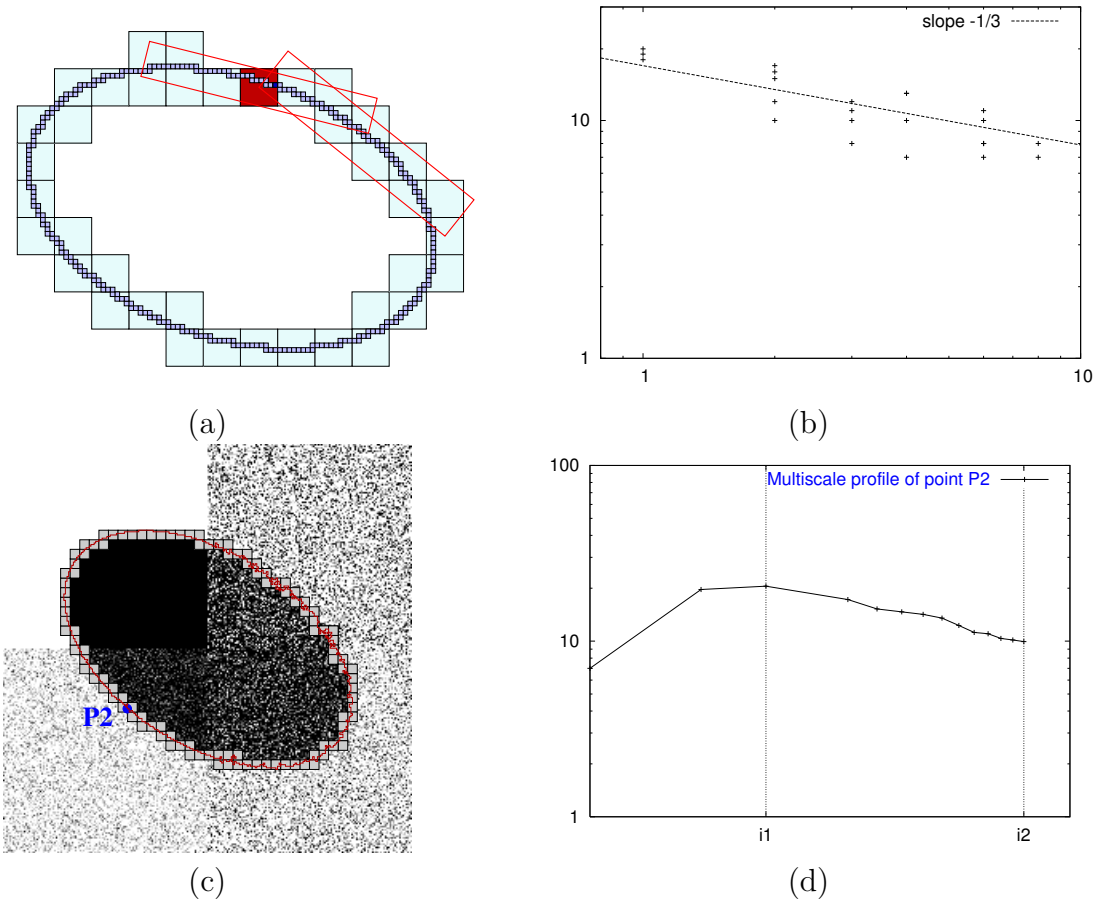


Figure 4: Images (a,b), construction of a multiscale profile with the set of maximal segments covering the red point for the grid size h equals to 8. Images (c,d), illustration of a multiscale profile defined on a noisy contour point P2. The resulting meaningful scale is (i_1, i_2).

3.1 Subsampling the Discrete Source Contour

Different methods are possible for the subsampling step. A common approach is to apply the subsampling by using the binary image of the contour and by covering it with a tilling of squares of size $i \times i$. Such an approach is simple but also presents the first inconvenient to not provide directly the correspondence between the digital contour of O and the digital contour of the subsampling of O (such a correspondence is illustrated in figure 5 (b)). A second inconvenient comes from the topological point of view since a simply connected shape can be transformed into a shape containing holes or vice versa.

We choose another approach by defining a contour based subsampling method. Given a digital 4-connected contour C that is the boundary of the set of pixels O in the cellular model, the subsampling is performed in four steps:

- step 1: the interpixel contour C is shifted toward the inside so that it defines the 4-connected inner border of O. This 4-connected contour of pixels is denoted by C'. It is not necessarily simple and may contain some back-and-forth paths that are oriented toward the exterior of O (*outer spikes*).

- step 2: the pixel contour C' is subsampled as the pixel contour C, composed of the sequence of points $(X_j, Y_j) = ((x_j - x_0) \div i, (y_j - y_0) \div i)$, where C' is the sequence of points (x_j, y_j), (x_0, y_0) represents the origin of the large pixels in $\mathbb{Z}^2$ and $\div$ is the integer division defined by

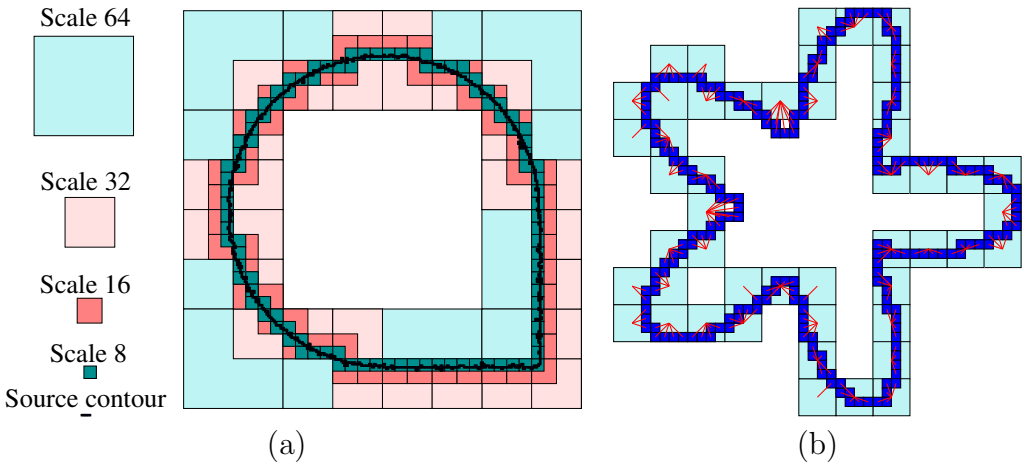


Figure 5: Image (a) illustrates a digital contour subsampling for i=8, 16, 32, 64 and image (b) gives an illustration of the function (represented by red lines) associating each pixel P of C to a point of the subsampled contour.

truncation.

- step 3: Consecutive identical pixels of C'' are merged to get a pixel contour C'''.
- step 4: Outer and inner spikes of C''' are removed (by several linear scans). The obtained contour is shifted toward the outside so that it defines a 4-connected interpixel contour, that is, the boundary of some digital shape.

By following these main steps and from a given maximal scale N, we are able to give as output the set $\mathcal{D}$ of the subsampled digital contours of C (denoted as $\phi_i^{x_0,y_0}$). The function $f_i^{(x_0,y_0)}$ associating each contour point of C''' should also be updated during steps 2, 3 and 4.

Algorithm 1 summarizes the subsampling process which can be obtained with the object MultiscaleFreemanChain[7] of the *ImaGene* framework.

Algorithm 1: Subsampling.
(performed with object `ImaGene::MultiscaleFreemanChain` in the *ImaGene* framework)

input : digital contour C' (defined from previous step 1)
ouput: the set $\mathcal{D}$ of subsampled contours $\phi_i^{x_0,y_0}(C')$ stored in a map $\mathcal{M}$ and $f_i^{(x_0,y_0)}$

```
for integer k = 1 to k < N do
    for integer x_0 = 1 to x_0 < k do
        for integer y_0 = 1 to y_0 < k do
            compute φ_k^{x_0,y_0}(C') and f_k^{(x_0,y_0)} by following the previous steps 2, 3 and 4 ;
            define a key for the resulting map M from the elements (k, x_0, y_0);
            store the contour φ_k^{x_0,y_0}(C') in the map M by using the previous key ;
```

3.2 Tangential Cover Computation and Length Statistic

The set of all the maximal segments of a digital contour can be computed in linear time according the contour size [8]. Such an approach recognizes maximal segments by adding and removing points

[7] http://www.lama.univ-savoie.fr/~lachaud/ImaGene/doc/html/classImaGene_1_1MultiscaleFreemanChain.html

from an initial one. The tangential cover should also be computed on all the subsampled contours.

Once all the maximal segments are computed for a given scale, the statistics of the maximal segment length are stored for each original contour point. The source contour point can be easily recovered by the function $f_i^{(x_0,y_0)}$.

The algorithm is summarized in algorithm 2. The implementation of this algorithm is included in *ImaGene* framework, in the class `MultiscaleProfile` from the directory `ImaGene/helper`.

Algorithm 2: Tangential cover.
(included in class `ImaGene::MultiscaleProfile` from the directory `ImaGene/helper` of the *ImaGene* framework)

input : the map $\mathcal{M}$ of the subsampled contours $\phi_i^{x_0,y_0}(C')$
ouput: a statistics tabular statTab[N][C'.size()] containing for each scale and for each point P of the source contour the set of the length of maximal segment covering P.

```
1 for integer k = 1 to k < N do
2     for integer x_0 = 1 to x_0 < k do
3         for integer y_0 = 1 to y_0 < k do
4             recover the subsampled contour φ_k^{x_0,y_0}(C') from map M with key (k, x_0, y_0) ;
5             compute the set S_k^{x_0,y_0} of maximal segments of the subsampled contour φ_k^{x_0,y_0}(C') ;
6             for each point P of index p of the subsampled contour, store in the tabular L[p] the
              set of lengths of maximal segments of S_k^{x_0,y_0} which cover P ;
7             for integer i=0 to C'.size()-1 do
8                 integer indiceSubsampled = f_k^{(x_0,y_0)}(i);
9                 statTab[k][i].add(mean(L[indiceSubsampled]));
```

3.3 Computation of the Multiscale Profile and Deduction of the Noise Level

Once all the length statistics have been computed for each point of the source contour, we are able to determine a multiscale profile using algorithm 3.

The noise level can then be deduced from each multiscale profile (algorithm 4).

From this last algorithm, we are now able to give for each point of the discrete contour a noise level related to the meaningful scale detection.

3.4 Complexity of the Global Algorithm

The complexity of the global algorithm depends on the algorithm used to extract the set of maximal segments. Such extraction is implemented in linear time [3]. The tangential cover is computed for each scale K which implies K^2 contour shifts. The global complexity is then $O(nK^3)$ with n the number of contour points and K the maximal scale. Note that this complexity could be improved by using a multiscale recognition of the tangential cover [12].

Algorithm 3: Multiscale profile.
(included in class `ImaGene::MultiscaleProfile` from the directory `ImaGene/helper` of the *ImaGene* framework)

input : the statistics tabular statTab[m][i] containing for each contour point of index i ; the set of lengths of maximal segments computed at scale m.
ouput: a tabular tabProfiles[] containing all the multiscale profile $\mathcal{P}_n(P)$ of each source contour point.

```
1  The multiscale profile P_n(P) is represented by using two vectors (X_0,...,X_i,...X_{N-1}) and
   (Y_0,...,Y_i,...Y_{N-1}) associated respectively to the set of the two coordinates of the multiscale
   profile points.
2  for integer i=0 to C'.size() do
3      vector of type double: vectX;
4      vector of type double: vectY;
5      for integer k=0 to N - 1 do
6          vectX.pushback(log(k+1));
7          vectY.pushback(log(mean(statTab[k][i])));
8          MultiscaleProfile MP (vectX, vectY);
9          tabProfiles[i]= MP;
```

Algorithm 4: Noise level estimation.
(included in class `ImaGene::MultiscaleProfile` from the directory `ImaGene/helper` of the *ImaGene* framework)

input : a multiscale profile $\mathcal{P}_n(P)$;
a given threshold parameter t_m set by default to 0 (see section 2.3);
a minWidth parameter set by default to 1 (see section 2.3);
ouput: an integer representing the noise level.

```
1  vector of type double x;
2  vector of type double y;
3  pair of type integer mScale;
4  integer l = 0;
5  fill the two vectors x and y from the coordinates of the multiscale profile P_n(P);
6  for integer k = 1 to x.size() - 1 do
7      float slope = ( y[ k ] - y[ k - 1 ] ) / ( x[ k ] - x[ k - 1 ] );
8      if ( slope > t_m ) || ( (k+1) == x.size() ) then
9          if ( k - 1 - l ) >= minWidth then
10             mScale= make_pair(l + 1, k);
11         l = k;
12 return mScale.first;
```

4 Implementation

4.1 Source code

The C++ source code is provided in the ImaGene library[8]. A lighter version of this library has been constructed to avoid some dependencies with the *boost* and *gmp* libraries. The source code is available at the IPOL web page of this article[9].

The installation uses `cmake` and was tested on *Linux* and *Mac OSX*. You just have to follow the `README` and `INSTALL` files.

4.2 Input Format and Executable Test

Input format. The input contour of the multiscale analysis should be represented in a simple Freeman chain format. Such a representation is defined as follows: `X0 Y0 C0C1C2C2...CN`. The two first integers X0 and Y0 represent the coordinates of the initial contour point. The other numbers `Ci` represent the directions of the move to go from the current point to the next one. The code convention is the following:

- code 0: move left (x+1)
- code 1: move up (y+1)
- code 2: move left (x-1)
- code 3: move down (y-1)

Such contour examples are given in the directory `meaningfulscaleDemo/demoIPOL/Contours` with the ".fc" file extension. The file `meaningfulScaleEstim` located in the directory `meaningfulscaleDemo/demoIPOL` can be used to detect noise from the meaningful scales.

Noise detection test program. To perform the noise detection you can use the following command lines:

```
cd meaningfulscaleDemo/build/demoIPOL
./meaningfulScaleEstim < ../../demoIPOL/Contours/ellipseBruit.fc -↩
   drawXFIGNoiseLevel -enteteXFIG -drawContourSRC 4 1 > tmp.fig
```

The previous command generates an `xfig` file which can be transformed with [10]:

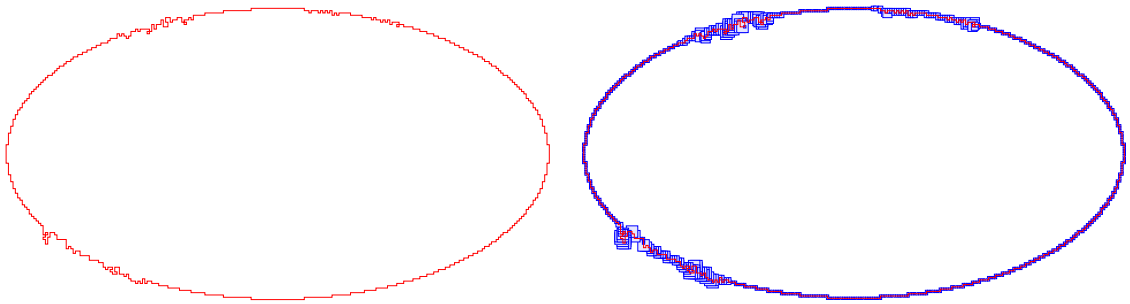
```
fig2dev -L eps tmp.fig tmp.eps
```

You should obtain the results presented in figure 6.

4.3 Including Meaningful Scale Detection in Other Programs

To include the meaningful scale detection in a C++ program with the ImaGene library, you need first to include the following header files:

[8] http://gforge.liris.cnrs.fr/projects/imagene
[9] http://dx.doi.org/10.5201/ipol.2014.75
[10] the `fig2dev` command can be found in the `transfig` package



(a) (b)

Figure 6: (a) Source contour of the Freeman chain `ellipseBruit.fc` located in the directory `demoIPOL/Contours` and (b) Meaningful scale results represented with blue boxes of size equal to the noise level and centered on each contour point.

```
#include "ImaGene/dgeometry2d/FreemanChain.h"
#include "ImaGene/dgeometry2d/FreemanChainTransform.h"
#include "ImaGene/helper/MultiscaleProfile.h"
```

and then follow these steps:

- Step 1: read the Freeman chain from the standard input. And then read the Freeman chain from input:

```
FreemanChain fc;
FreemanChain::read(cin, fc );
```

- Step 2: construction of the multiscale profile.

  This step needs the setting of the parameter associated to the maximal scale for which the shape is analyzed. Since we are just searching for the first meaningful scale, the maximal scale is not really important and depends only of what maximal amount of noise you have to detect (for instance 10 looks sufficient in most cases). The variable `nbIterationSpikeDetection` is used to avoid degenerated contours and a value of 5 appears sufficient for all experiments.

```
FreemanChainSubsample fcsub( 1, 1, 0, 0 );
FreemanChainCleanSpikesCCW fccs( nbIterationSpikeDetection );
FreemanChainCompose fcomp(fccs, fcsub );
FreemanChainTransform* ptr_fct = &fcomp;
FreemanChainSubsample* ptr_fcsub = &fcsub;
MultiscaleProfile MP;
MP.chooseSubsampler( *ptr_fct, *ptr_fcsub );
MP.init( fc, maxScale);
```

- Step 3: estimate and display noise levels.

  The `minSize` and `maxSlope` values should be set to the default value, respectively to 1 and 0.

```
int i=0;
for(FreemanChain::const_iterator it=fc.begin(); it!=fc.end(); ++it){
  uint noiseLevel=MP.noiseLevel(i, minSize, maxSlope);
  cout << "noise level point index" << i << ": " << noiseLevel << ↩
     endl; i++;
}
```

This simple minimal example is given in the meaningfulscaleDemo archive in the file `minimal-CodeNoiseDetection.cxx` of the directory `meaningfulscaleDemo/demoIPOL/`.

5 Examples and Experiments

5.1 Examples on Various Noisy Shapes

We present experiments obtained with various noisy shapes by using the default intern parameters $t_m = 0.0$, `max_scale=15` and minimal meaningful scale size `min_size=1`. Figures 7 and 8 show the results obtained on digital contours extracted from binary and grayscale images. The contours extraction was performed from the multidimensional algorithm based on boundary extraction and contour tracking [2] which is available from the IPOL site[11]. This extraction is also implemented in the *ImaGene* framework and given in the source demo in: `meaningfulscaleDemo/bin/pgm2freeman.cxx`.

5.2 Stability Measure from Internal Parameters

Even if the meaningful scale detection can be considered as parameter free, it contains some internal parameters which have only small influence on the detection quality. To demonstrate this stability, several experiments were performed on the shapes of figure 9 (a,c) for which the default meaningful scale detection is given on images (b,d). First we experiment the influence of the change of each particular parameter and then the change of all parameters together.

Experiments with the change of the threshold of the minimal slope (t_m). As introduced in section 2.3 a threshold t_m was defined to discriminate smooth and noisy contour parts. By default, the value was set to 0 and as it can be seen in figure 10 (two first upper rows), modification of this particular parameter does not really change the global quality of the noise detection.

Experiments by adding a minimal length for the meaningful scale interval (`msMinLength`). In the definition of the meaningful scale of section 2.3, a minimal size of one was introduced to define the interval (i_1, i_2). Figure 10 (two last lower rows) shows the results obtained by changing this definition. As illustrated, the change of the initial definition does not really improve the meaningful scale detection and with too large values of the scale interval some corners tend to be detected as noise.

These experiments with the change of the t_m and `msMinLength` can be reproduced by the following command line from the `meaningfulscaleDemo/build` directory:

```
./demoIPOL/meaningfulScaleEstim  -drawContourSRC 0 1 -meaningfulScale 1 ↩
   -0.3  -enteteXFIG -drawXFIGNoiseLevel  < ../demoIPOL/Contours/↩
   ellipseBruit2.fc  > tmp.fig ;
fig2dev -L eps tmp.fig ellipseBruit2MS1_-0.3.eps;
```

where the option `-meaningfulScale` permits to set the parameters associated to minimal length for the meaningful scale definition and the threshold t_m.

Experiments with the change of the analysis maximal scale (`maxScale`). A last internal parameter is the maximal scale for which the shape is analyzed. Since the noise level is defined as the first scale of the meaningful scale interval, a change of this value has small influence. The experiments

[11] http://www.ipol.im

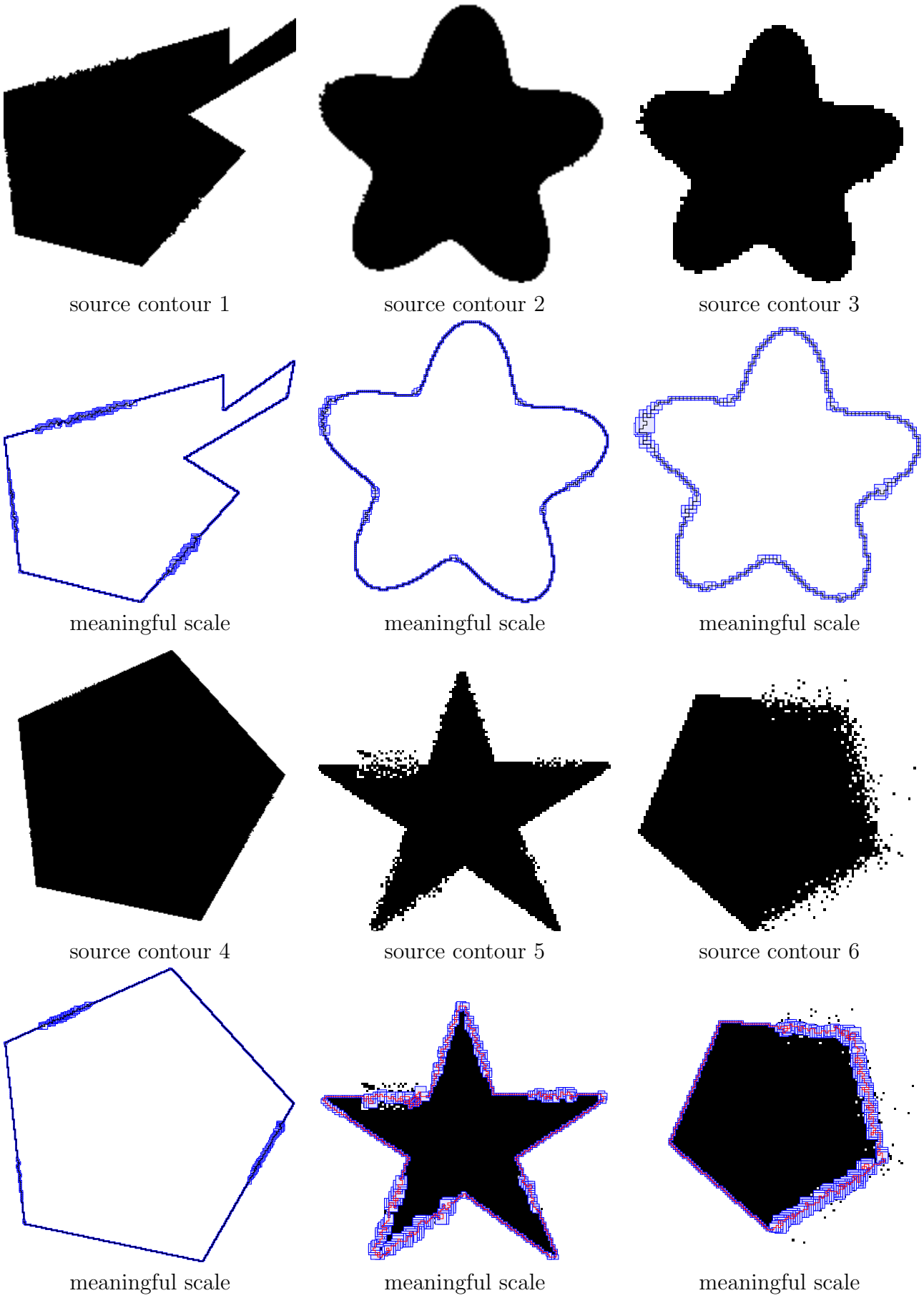


Figure 7: Illustration of the meaningful scale detection on some noisy shapes. The meaningful scales are represented with blue boxes of size equal to the noise level and centered on each contour point.

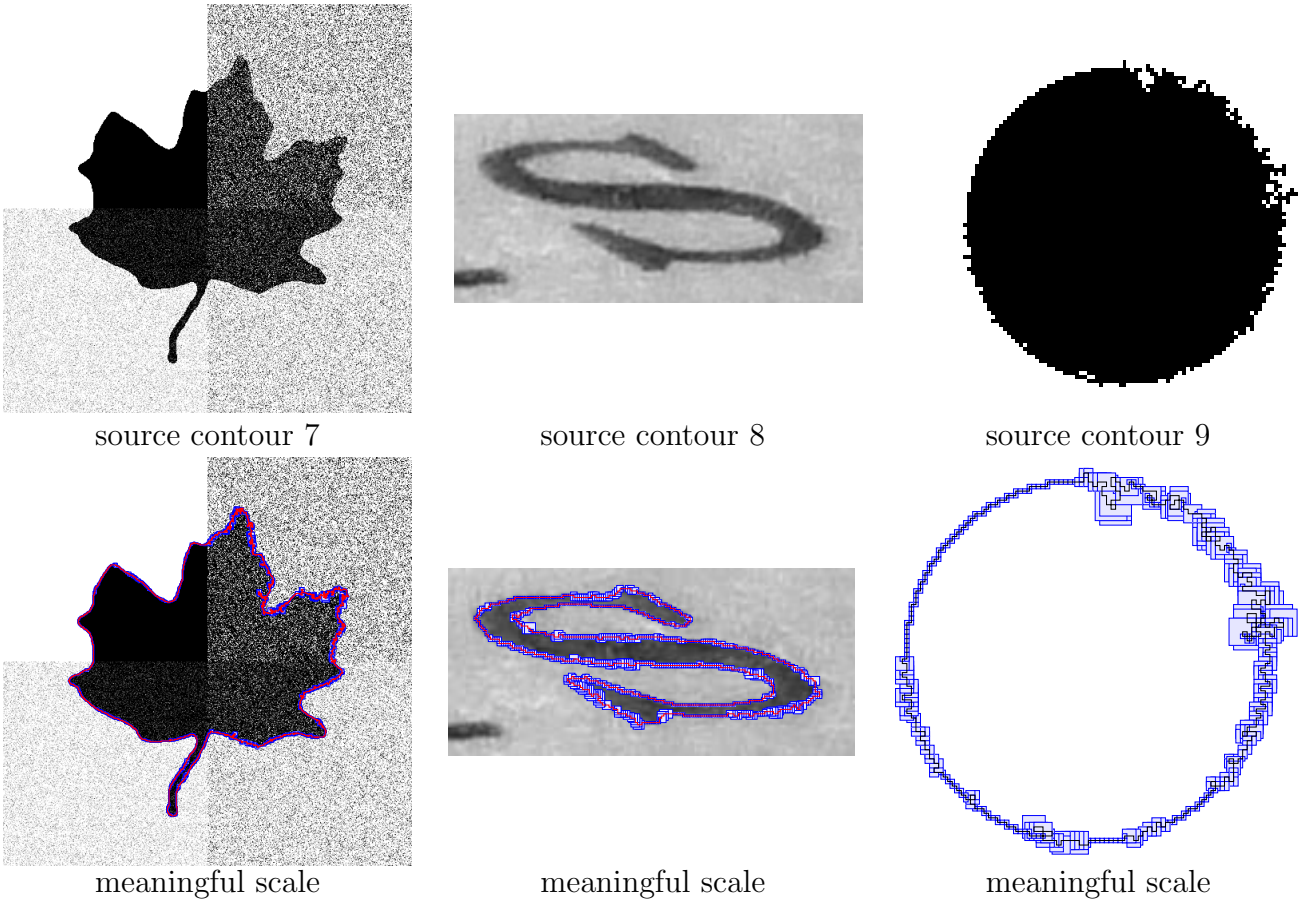


Figure 8: Illustration of the meaningful scale detection on some noisy shapes. The meaningful scales are represented with blue boxes of size equal to the noise level and centered on each contour point.

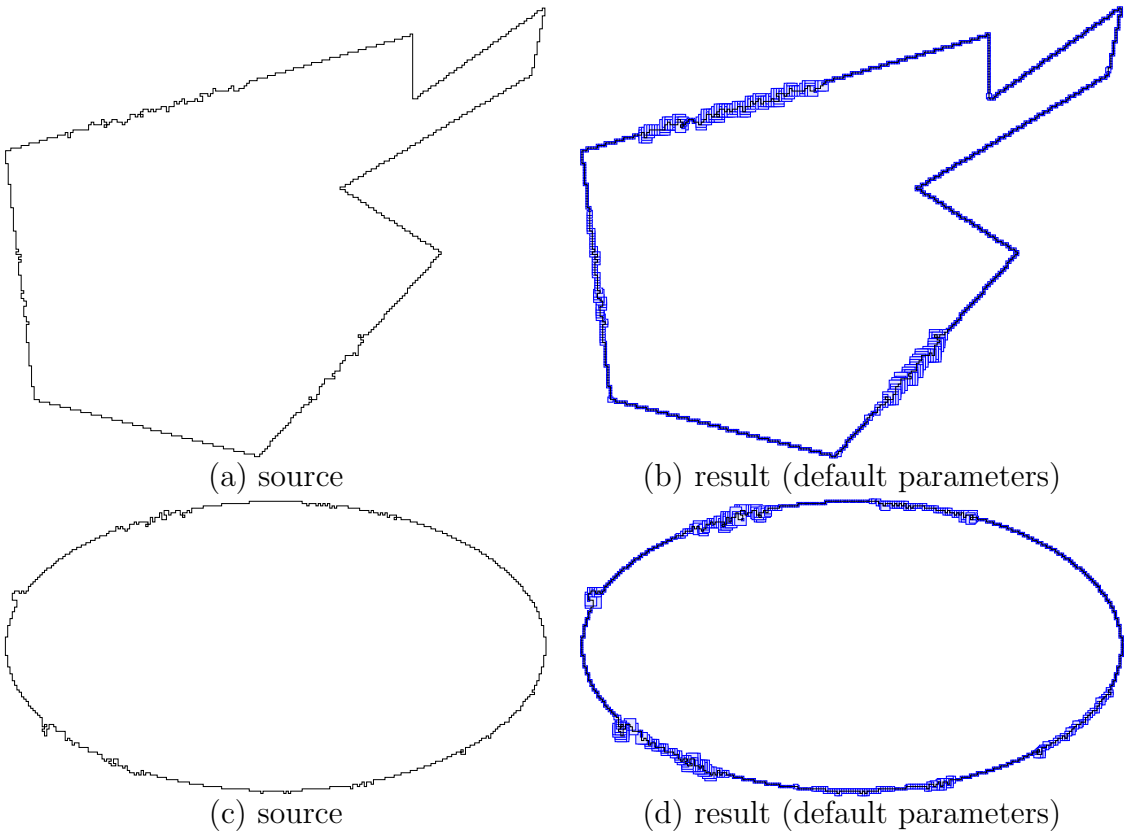
presented in figure 11 (two first upper rows) confirm this fact. Note that the areas displayed in green correspond to the points for which no meaningful scale was found below the maximal scale. In this particular case a box is displayed with the given maximal scale.

These experiments can be reproduced by the following command lines from the `meaningfulScaleDemo/build` directory:

```
./demoIPOL/meaningfulScaleEstim  -drawContourSRC 0 1 -setSamplingSizeMax 10↩
     -enteteXFIG -drawXFIGNoiseLevel  < ../demoIPOL/Contours/ellipseBruit2.↩
   fc  > tmp.fig ;
fig2dev -L eps tmp.fig  ellipseBruitMS1_0.0Max10.eps;
```

where the analysis maximal scale (parameter `maxScale`) is set with option `-setSamplingSizeMax`.

Experiments with the change of all the parameters together. To apply a last test on the independence towards the internal parameters, we apply the meaningful scale detection with various changes on all the three parameters. The lower part of figure 11 shows globally good results even if with the case of parameters (`msMinLength`=4, $t_m = -0.1$, `maxScale`=20) one area between two corners is detected as noise.



(a) source (b) result (default parameters)

(c) source (d) result (default parameters)

Figure 9: Noise estimation with the default parameters on the polygon (a,b) and ellipse shapes (c,d).

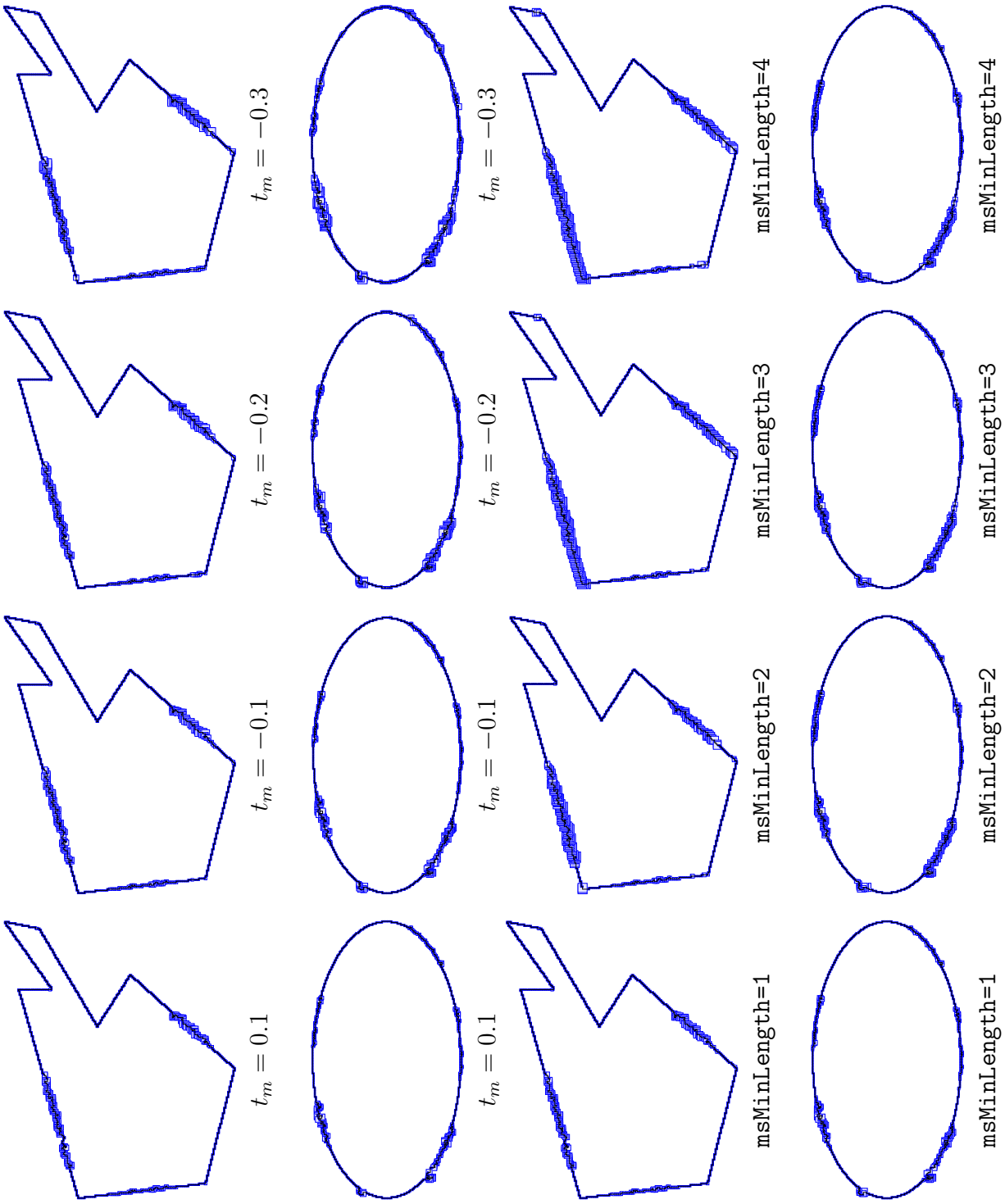


Figure 10: Experiments on the sensibility to the parameters t_m and `msMinLength`.

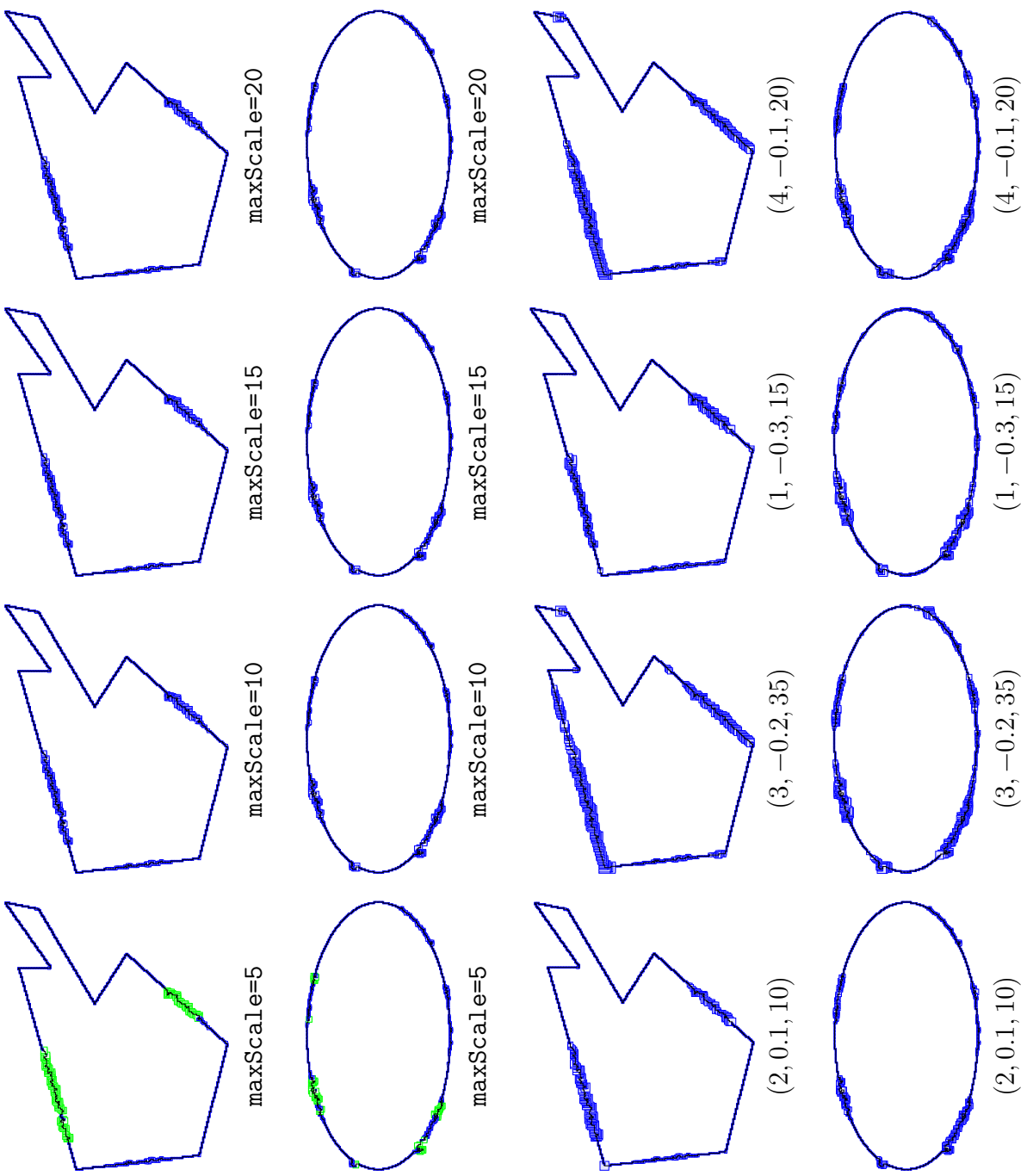


Figure 11: Experiments of the change of the analysis maximal scale (first and second row). The two last rows give results with various changes of all the default parameters together.

Image Credits

All images and contours created by the authors except:

 which is a standard test contour and

 which is a noisy version of a standard test image.

References

[1] P. Bhowmick and B. B. Bhattacharya, *Fast polygonal approximation of digital curves using relaxed straightness properties*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1590–1602. http://dx.doi.org/10.1109/TPAMI.2007.1082.

[2] D. Coeurjolly, B. Kerautret, and J.-O. Lachaud, *Extraction of connected region boundary in multidimensional images*, Image Processing On Line, (2014). http://dx.doi.org/10.5201/ipol.2014.74.

[3] F. Feschet and L. Tougne, *Optimal time computation of the tangent of a discrete curve: Application to the curvature*, in Discrete Geometry for Computer Imagery, Gilles Bertrand, Michel Couprie, and Laurent Perroton, eds., vol. 1568 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1999, pp. 31–40. http://dx.doi.org/10.1007/3-540-49126-0_3.

[4] B. Kerautret and J.-O. Lachaud, *Curvature estimation along noisy digital contours by approximate global optimization*, Pattern Recognition, 42 (2009), pp. 2265–2278. http://dx.doi.org/10.1016/j.patcog.2008.11.013.

[5] ——, *Multi-scale analysis of discrete contours for unsupervised noise detection*, in Proceedings of the 13th international Workshop on Combinatorial Image Analysis (IWCIA), P. Wiederhold and R.P. Barneva, eds., vol. 5852 of LNCS, Springer, November 2009, pp. 187–200. http://dx.doi.org/10.1007/978-3-642-10210-3_15.

[6] ——, *Meaningful scales detection along digital contours for unsupervised local noise estimation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (2012), pp. 2379–2392. http://dx.doi.org/10.1109/TPAMI.2012.38.

[7] J.-O. Lachaud, *Espaces non-euclidiens et analyse d'image : modèles déformables riemanniens et discrets, topologie et géométrie discrète*, habilitation à diriger des recherches, Université Bordeaux 1, Talence, France, 2006. (in french).

[8] J.-O. Lachaud, A. Vialard, and F. de Vieilleville, *Fast, accurate and convergent tangent estimation on digital contours*, Image and Vision Computing, 25 (2007), pp. 1572–1587. http://dx.doi.org/10.1016/j.imavis.2006.06.019.

[9] H. Liu, L. Latecki, and W. Liu, *A unified curvature definition for regular, polygonal, and digital planar curves*, International Journal of Computer Vision, 80 (2008), pp. 104–124. http://dx.doi.org/10.1007/s11263-008-0131-y.

[10] T.P. Nguyen and I. Debled-Rennesson, *Curvature estimation in noisy curves*, in Computer Analysis of Images and Patterns, WalterG. Kropatsch, Martin Kampel, and Allan Hanbury, eds., vol. 4673 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 474–481. http://dx.doi.org/10.1007/978-3-540-74272-2_59.

[11] T.P. Nguyen and I. Debled-Rennesson, *A discrete geometry approach for dominant point detection*, Pattern Recognition, 44 (2011), pp. 32–44. http://dx.doi.org/10.1016/j.patcog.2010.06.022.

[12] M. Said and J.-O. Lachaud, *Computing the characteristics of a subsegment of a digital straight line in logarithmic time*, in Discrete Geometry for Computer Imagery, Isabelle Debled-Rennesson, Eric Domenjoud, Bertrand Kerautret, and Philippe Even, eds., vol. 6607 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 320–332. http://dx.doi.org/10.1007/978-3-642-19867-0_27.