Published in Image Processing On Line on 2011–09–13.
Submitted on 2011–00–00, accepted on 2011–00–00.
ISSN 2105–1232 © 2011 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.2011.cm_fds

Finite Difference Schemes for MCM and AMSS

Marco Mondelli[1], Adina Ciomaga[2]

[1] SSSA, Pisa, Italy (m.mondelli@sssup.it)
[2] CMLA, ENS Cachan, France (ciomaga@cmla.ens-cachan.fr)

Communicated by Antonin Chambolle *Demo edited by* Jose-Luis Lisani

Abstract

This article refers to algorithms based on finite difference schemes for computing mean and affine curvature evolutions of digital images, introduced by Alvarez and Morel [L. Alvarez, J.M. Morel, "Formalization and computational aspects of image analysis", Acta Numerica, pp. 159, 1994]. We discuss consistency, stability and convergence. Our analysis focuses on some possible choices of the parameters, choices that generate multiple variants in the implementations. Meaningful visual examples on how the algorithms actually work are provided.

Source Code

The source code (ANSI C), its documentation, and the online demo are accessible at the IPOL web page of this article[1]. The code is distributed under the license LGPL.

Keywords: image curvature analysis; image morphology analysis; mean curvature motion; affine curvature motion

1 Introduction

Alvarez et al. [1] characterized axiomatically all image multiscale theories and gave explicit formulas for the partial differential equations generated by scale spaces. They showed that all causal, local, isometric and contrast invariant scale spaces are given by curvature evolution equations of the type

$$\frac{\partial u}{\partial t} = g(curv(u), t)|Du|.$$

Two particular motions have become very popular for planar shape deformation and recognition algorithms. One refers to mean curvature flows

$$\frac{\partial u}{\partial t} = curv(u)|Du|.$$

[1] https://doi.org/10.5201/ipol.2011.cm_fds

The other motion regards affine curvature evolutions

$$\frac{\partial u}{\partial t} = curv(u)^{\frac{1}{3}}|Du|,$$

and is preferable to the Mean Curvature Motion for its projective invariance properties.

In the following we refer to algorithms based on finite difference schemes for computing mean and affine curvature evolutions of digital images, introduced by Alvarez and Morel [2]. We discuss consistency, stability and convergence (for definitions see [3]). Our analysis focuses on some possible choices of the parameters, choices that generate multiple variants in the implementations. Meaningful visual examples on how the algorithms actually work are provided.

The algorithms run on both B/W and RGB images. When dealing with true color images, we perform the smoothing on each color channel independently.

1.1 Mean Curvature Motion (MCM)

The Mean Curvature Motion has been studied in great detail for Jordan curves in a series of articles of differential geometry by Gage, Hamilton and Grayson (see for example [4]). In this classical setup, we are initially given a smooth Jordan curve and, as time progresses, we let each point evolve in the direction of the normal with a speed equal to the curvature at that point. It was shown that a curve instantly becomes smooth, shrinks asymptotically to a circle and shows no singularity or self-crossing.

A completely different approach is given in the setting of viscosity solutions theory (see for example [6, 7]), which provides rigorous mathematical justification for level set methods. The undertaking is analytically and numerically subtle principally because the mean curvature evolution equation is nonlinear, degenerate, and even undefined at points where $Du = 0$. Indeed

$$curv(u) = \frac{1}{|Du|^3} D^2u(Du^\perp, Du^\perp).$$

1.2 Affine Morphological Scale Space (AMSS)

Affine curve evolution has been introduced by Sapiro and Tannenbaum [5]. Together with Angenent, they gave existence and uniqueness proofs and showed a result similar to Grayson's theorem: a shape becomes convex and thereafter evolves towards an ellipse before collapsing. The viscosity solution theory developed by Chen, Giga and Goto [7] covers general curvature evolutions, including the AMSS.

It is worth mentioning that this equation is **affine invariant** in the sense that if we take a C^2 function u which is a solution of the AMSS, then

$$v(t, x) = u(|A|^{\frac{2}{3}}t, Ax),$$

is also a solution, where A is a 2×2 matrix with positive determinant $|A|$.

2 Algorithms

2.1 Notations and Common Architecture

The program

- reads an `image.tiff` given as input,

- checks if the image is *true RGB*,
- applies a finite difference scheme for the Mean Curvature Motion at normalized scale R,
- writes the result as a new `image.tiff` and returns it as output.

An image is *false RGB* if it has three identical color channels. In this case, only the array registered in the first channel is processed, the successive ones being discarded.

At normalized scale R every disk with radius less or equal than R disappears.

2.2 Finite Difference Scheme for MCM

We consider a discrete image $u(i,j)$ as an array of $n_{\text{row}} \times n_{\text{col}}$ integer samples taken from the continuous function u. The index i represents the current row number and j the current column number. We implement a finite difference scheme (FDS) taking advantage of the diffusive interpretation of the equation, i.e. the mean curvature can be expressed as the second derivative of u in the direction ξ orthogonal to the gradient

$$u_t = u_{\xi\xi}.$$

In fact

$$u_{\xi\xi} = D^2u(\xi,\xi) = D^2u\Big(\frac{Du^\perp}{|Du|}, \frac{Du^\perp}{|Du|}\Big) = \frac{1}{|Du|^2} D^2u(Du^\perp, Du^\perp) = |Du| curv(u).$$

More precisely, we evaluate numerically $u_{\xi\xi}$ with a linear scheme based on a 3×3 stencil and then we define iteratively the sequence $u_n(i,j)$ by

$$u_{n+1}(i,j) = u_n(i,j) + \Delta t \cdot (u_{\xi\xi})_n(i,j).$$

To overcome the problem at points where the mean curvature evolution is undefined, we diffuse by half Laplacian whenever the gradient is below a threshold T_g (experimentally set to 4)

$$u_{n+1}(i,j) = u_n(i,j) + \frac{1}{2}\Delta t \cdot \Delta u_n(i,j).$$

We point out that in practice, even if $|Du| \neq 0$ but too small in modulus, its direction becomes substantially random because it will be driven by rounding and approximation errors.

As suggested by Guichard, Morel and Ryan [2], if the gradient is not zero we can use a first neighborhood linear scheme for the evaluation of $u_{\xi\xi}$,

$$(u_{\xi\xi})_n(i,j) = \frac{1}{\Delta x^2}\big(-4\lambda_0 \cdot u_n(i,j) \ + \ \lambda_1 \cdot (u_n(i,j+1) + u_n(i,j-1)) \ + \ \lambda_2 \cdot (u_n(i+1,j) +$$

$$+ u_n(i-1,j)) + \ \lambda_3 \cdot (u_n(i-1,j-1) + u_n(i+1,j+1)) \ + \ \lambda_4 \cdot (u_n(i-1,j+1) + u_n(i+1,j-1)) \ ),$$

where Δx is the pixel width and will be set to 1.

The formula can be rewritten in a more synthetic form, where $\star$ stands for a discrete convolution with a variable kernel A and θ is the angle between the gradient and the horizontal axis

$$(u_{\xi\xi})_n = \frac{1}{\Delta x^2} \cdot (A \star u_n) \quad \text{where} \quad A = \begin{pmatrix} \lambda_3(\theta) & \lambda_2(\theta) & \lambda_4(\theta) \\ \lambda_1(\theta) & -4\lambda_0(\theta) & \lambda_1(\theta) \\ \lambda_4(\theta) & \lambda_2(\theta) & \lambda_3(\theta) \end{pmatrix}.$$

We call A a *variable kernel* since the λ_i are not fixed, but depend on θ.

A simple Taylor development shows that in order to obtain **consistency** the following relationships must hold

$$\begin{cases} \lambda_1(\theta) = 2\lambda_0(\theta) - \cos^2\theta \\ \lambda_2(\theta) = 2\lambda_0(\theta) - \sin^2\theta \\ \lambda_3(\theta) = -\lambda_0(\theta) + 0.5(1 - \sin\theta\cos\theta) \\ \lambda_4(\theta) = -\lambda_0(\theta) + 0.5(1 + \sin\theta\cos\theta), \end{cases}$$

where $\sin\theta$ and $\cos\theta$ can be calculated from the following finite difference schemes of the partial derivatives of u,

$$u_x = \frac{2\cdot(u(i,j+1) - u(i,j-1)) + u(i-1,j+1) - u(i-1,j-1) + u(i+1,j+1) - u(i+1,j-1)}{8\Delta x},$$

$$u_y = \frac{2\cdot(u(i+1,j) - u(i-1,j)) + u(i+1,j+1) - u(i-1,j+1) + u(i+1,j-1) - u(i-1,j-1)}{8\Delta x}.$$

We have thus the value of λ_0 as the only free parameter, normally chosen to ensure **stability**. Such a condition implies that all λ_i must be positive. Unfortunately it is impossible to respect these inequalities except for particular values of θ, because

$$\begin{cases} \lambda_1(\theta) \geq 0 \Rightarrow \lambda_0(\theta) \geq \dfrac{\cos^2(\theta)}{2} \\ \\ \lambda_4(\theta) \geq 0 \Rightarrow \lambda_0(\theta) \leq \dfrac{1 - \sin(\theta)\cos(\theta)}{2} \end{cases} \quad \text{while} \quad \frac{\cos^2(\theta)}{2} \geq \frac{1 - \sin(\theta)\cos(\theta)}{2} \quad \forall\ \theta \in \left[0, \frac{\pi}{4}\right].$$

We decide to choose $\lambda_0(\theta)$ somewhere between the boundary functions so that λ_1 and λ_4 become only slightly negative.

The arithmetic mean of these two functions could be a valid candidate

$$\lambda_0(\theta) = \frac{\cos^2\theta + 1 - \sin\theta\cos\theta}{4}.$$

However our final decision is to take

$$\lambda_0 = 0.5 - \sin^2\theta\cos^2\theta,$$

which is the trigonometric polynomial with least degree lying between boundary functions, smooth at 0 and $\pi/4$ and satisfying the following four geometrical requirements:

1. invariance by a rotation of angle $\pi/2$;
2. symmetry with respect to the axes $i + j$ and $i - j$;
3. purely one-dimensional diffusion in the case $\theta = 0$;
4. purely one-dimensional diffusion in the case $\theta = \pi/4$.

2.3 Finite Difference Scheme for AMSS

Following a procedure similar to that adopted for the MCM, we evaluate numerically the operator $|Du|^2 u_{\xi\xi}$ with a linear scheme based on a 3×3 stencil and we define iteratively the sequence $u_n(i,j)$ by

$$u_{n+1}(i,j) = u_n(i,j) + \Delta t \cdot (|Du_n|^2 (u_{\xi\xi})_n)^{\frac{1}{3}}(i,j).$$

If the gradient is not zero we set

$$|Du_n|^2(u_{\xi\xi})_n(i,j) = \frac{1}{\Delta x^2}(-4\eta_0 \cdot u_n(i,j) \ + \ \eta_1 \cdot (u_n(i,j+1) + u_n(i,j-1)) \ + \ \eta_2 \cdot (u_n(i+1,j)+$$

$$+u_n(i-1,j)) \ + \ \eta_3 \cdot (u_n(i-1,j-1) + u_n(i+1,j+1)) \ + \ \eta_4 \cdot (u_n(i-1,j+1) + u_n(i+1,j-1)) \ ).$$

To ensure **consistency** η_i must satisfy

$$\begin{cases} \eta_1(\theta) = 2\eta_0(\theta) - u_x^2 \\ \eta_2(\theta) = 2\eta_0(\theta) - u_y^2 \\ \eta_3(\theta) = -\eta_0(\theta) + 0.5(u_x^2 + u_y^2 - u_x \cdot u_y) \\ \eta_4(\theta) = -\eta_0(\theta) + 0.5(u_x^2 + u_y^2 + u_x \cdot u_y). \end{cases}$$

Instead a **stability** check is more difficult, because of the third root present in the equation defining the AMSS. So we limit to take simply

$$\eta_0(\theta) = |Du|^2 \lambda_0(\theta).$$

One would be tempted to choose for λ_0 the same value as for the MCM, but the arithmetic mean of boundary functions has the great advantage of being of degree 2.

This fact becomes important when we deal with the case $|Du| = 0$, because differently from the MCM, the AMSS is well defined even when the gradient is zero,

$$|Du|^3 curv(u) = D^2u(Du^\perp, Du^\perp).$$

Considering the numerical algorithm, we can see that η_i are well defined and no division by 0 is performed if and only if the degree of λ_0, thought as a trigonometric polynomial in θ, is less or equal than 2.

The results obtained after different choices of λ_0 and after the introduction of a threshold for the gradient will be compared in the next section.

2.4 MCM Implementations and Choice of the Parameters

The various implementations of a finite difference scheme for the Mean Curvature Motion can differ essentially on three aspects:

1. the way we deal with points situated on the edges of the image;
2. the way we overcome instability;
3. the choice of the time step.

A detailed analysis of these three points follows.

2.4.1 Image Extension

In order to apply a FDS to points situated on the borders, one needs to know the gray values of the pixels in the corresponding neighborhoods. However some of these neighbours could not be defined. In the continuous case we extend images by symmetry and periodization. Similarly, in the discrete case, an image can be extended by various procedures:

- **mirror behavior**: the discrete image is symmetrized with respect to $x = 0$ and $y = 0$;

- **semi-mirror behavior**: symmetry is performed with respect to $x = 0.5$ and $y = 0.5$;
- **periodization**;
- **extension by 0** outside the image frame.

The origin is situated in the upper-left corner and the x-axis (respectively the y-axis) runs along the first row (respectively column).

We adopt the mirror behavior because it preserves the continuity of the image and avoids sharp changes at the edges. However, this choice has a major drawback: level lines meeting the frame of the image bend more and more as the scale increases. Actually, the image evolves as if it were the central block of a 9 times bigger image made up by 3×3 flips of the original one.

2.4.2 Instability

As pointed out before, the algorithm is not stable. Numerically, the maximum and minimum values assumed by the pixels exceed the range $[0, 255]$. Nevertheless, in order to visualize the image it is necessary to restore pixel values to the original range. Basically there are four possibilities:

- **saturation at the end**: at the end of the iterative part we evaluate the maximum and minimum values assumed by the pixels and, if the case, set to 0 all the values below 0 and to 255 all the values above 255;
- **saturation at each step**: same operation as above, but performed after each single iteration;
- **renormalization at the end**: at the end of the iterative part we evaluate the maximum and minimum values assumed by the pixels. If they are not in the range, we apply an affine transformation T, such that

$$\begin{aligned} T([min, max]) &= [0, 255] \quad \text{if} \quad min < 0 \quad \text{and} \quad max > 255 \\ T([min, max]) &= [min, 255] \quad \text{if} \quad min \geq 0 \quad \text{and} \quad max > 255 \\ T([min, max]) &= [0, max] \quad \text{if} \quad min < 0 \quad \text{and} \quad max \leq 255 \end{aligned}$$

  An analytic expression for T in the first case is

$$T(x) = \frac{255 \cdot (x - min)}{max - min},$$

  and the other two cases can be easily reduced to the first one, taking respectively $max = 255$ and $min = 0$;
- **saturation at each step**: same operation as above, but performed after each single iteration.

The choice that allows us to remain attached to the original equation as much as possible is the first one. First of all renormalization is nothing but the application of a contrast change to all the pixels, while saturation modifies only the wrong ones, which are supposed to be in small number. Secondly, we prefer performing the operation when the iterative part is done to emphasize the fact that the problem is not algorithmical, but a matter of visualization. Moreover if the normalized scale is sufficiently high, in most cases the global maximum (minimum) after the initial increase (decrease) tends to return smaller than 255 (bigger than 0).

This behavior is testified by the following experiment (Figure 1). We took from the web some images at random and we plotted the maximum and minimum values assumed by the pixels after each iteration of the MCM for a total number of 400 (which means up to a normalized scale of almost

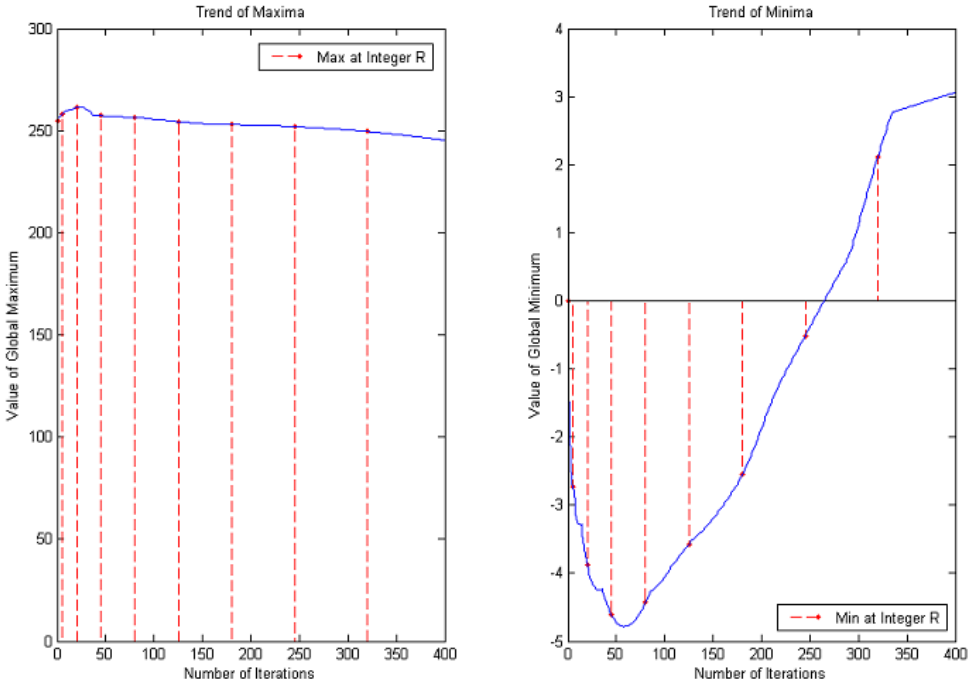

Figure 1: Plot of the maximum and minimum values assumed by the pixels after each iteration of the MCM for a total number of 400 (which means up to a normalized scale of almost 9) for one image taken at random from the web.

9). The numbers of iterations that correspond to integer normalized scales have been displayed in red, in order to be easily recognized.

For RGB images we evaluated the global maximum and minimum of the three arrays.

The maximum increases for the first 20 iterations (normalized scale about 2) and then decreases monotonically returning in range when we reach a normalized scale of about 5. The minimum stays below 0 much longer but at iteration 300 (normalized scale about 8) is again in the correct range.

In Figure 2 we display the plots of the trends of maxima and minima for an RGB image, followed by two B/W ones (figures 3 and 4). In Figure 2 the maximum decreases monotonically. The minimum decays for the first 50 iterations (normalized scale about 3), then it starts increasing and returns to the correct range before iteration 150 (normalized scale about 6). In Figure 3 the maximum remains almost constant while the minimum increases monotonically. In Figure 4, as in the previous figure, the maximum remains almost constant, near the upper bound 255. The minimum increases monotonically and returns quickly in the correct range after the initial decrease.

2.4.3 Time Step

The time step chosen for the experiments is $\Delta t = 0.1$, but the results are about the same if $\Delta t = 0.5$ or $\Delta t = 0.05$, as the comparative graphs in figures 5 to 8 clearly show. A good trade off between results and processing time is to set the value of time step to 0.1. An upper bound for Δt is clearly 0.5, because of the application of the Heat Equation when the gradient is below the threshold, but performances are enhanced if we lower it further.

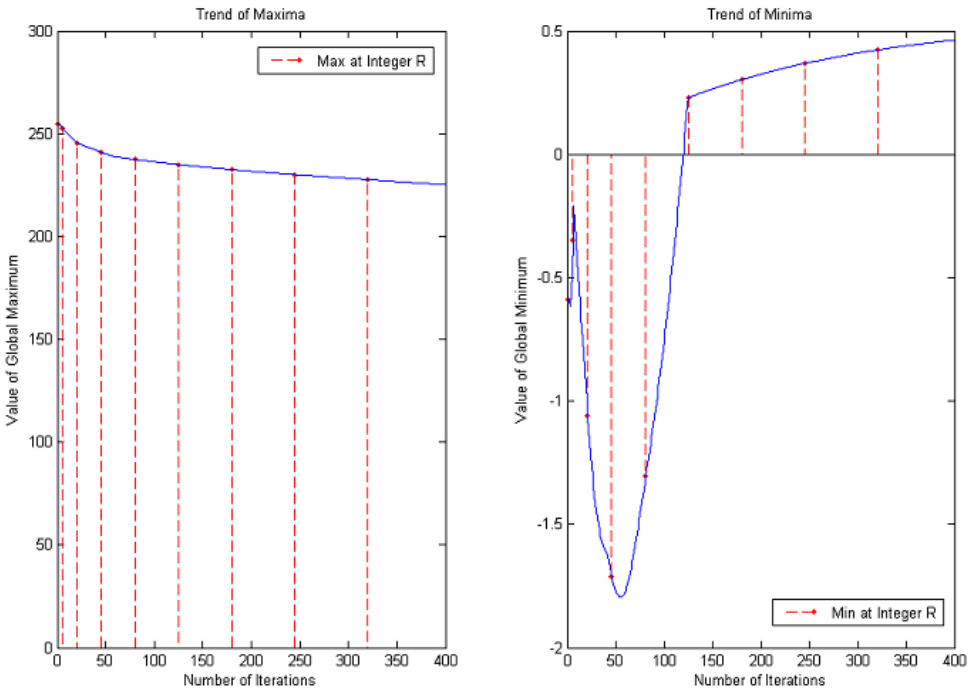

Figure 2: Plot of the maximum and minimum values assumed by the pixels after each iteration of the MCM for a total number of 400 (which means up to a normalized scale of almost 9) for an RGB image taken at random from the web.

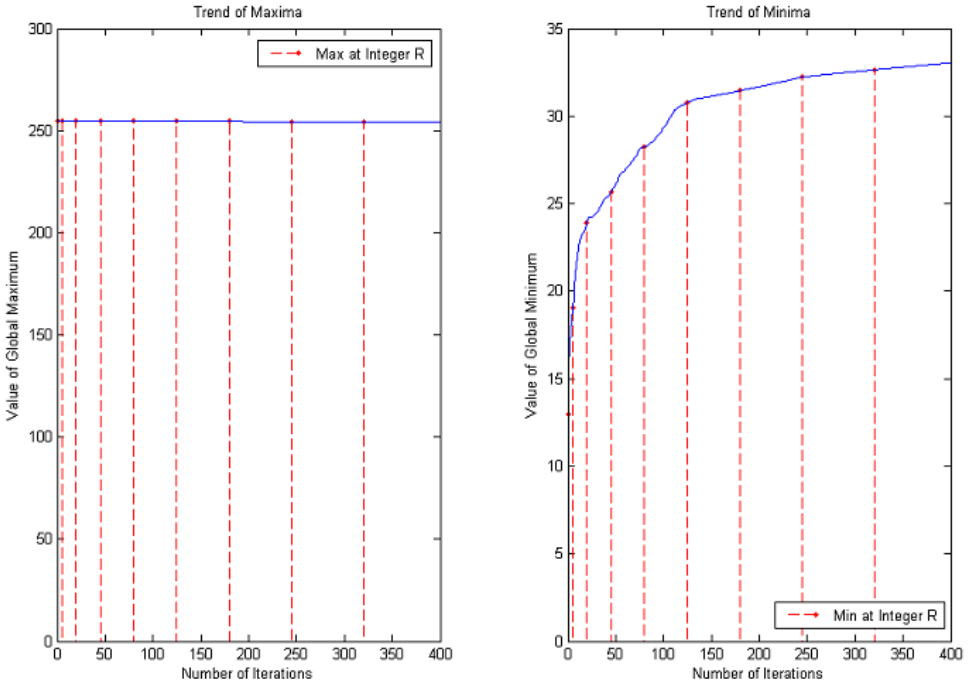

Figure 3: Plot of the maximum and minimum values assumed by the pixels after each iteration of the MCM for a total number of 400 (which means up to a normalized scale of almost 9) for a B/W image taken at random from the web.

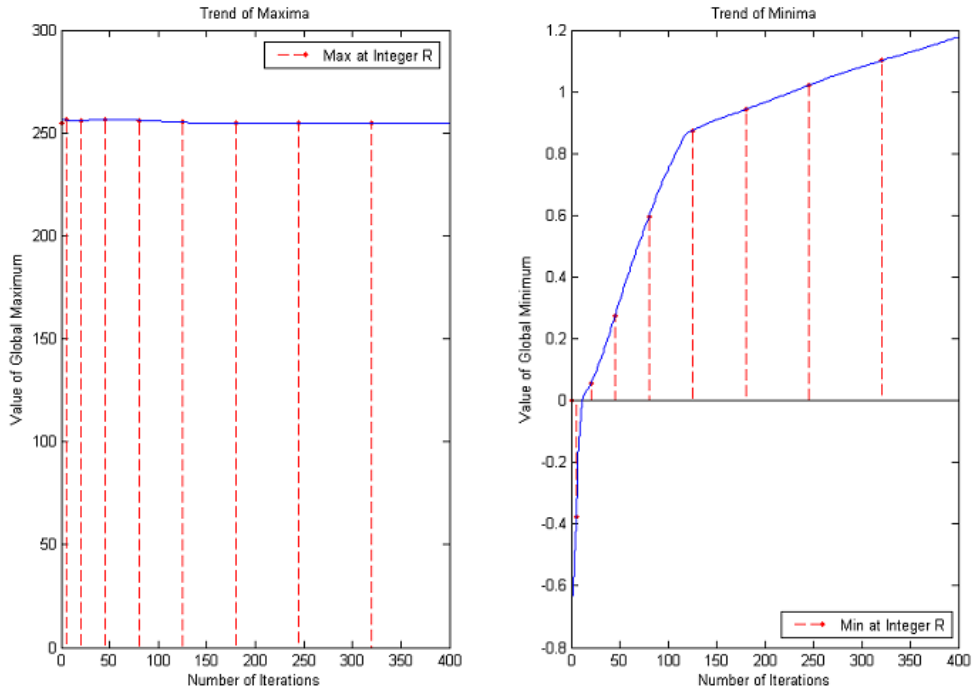

Figure 4: Plot of the maximum and minimum values assumed by the pixels after each iteration of the MCM for a total number of 400 (which means up to a normalized scale of almost 9) for a B/W image taken at random from the web.

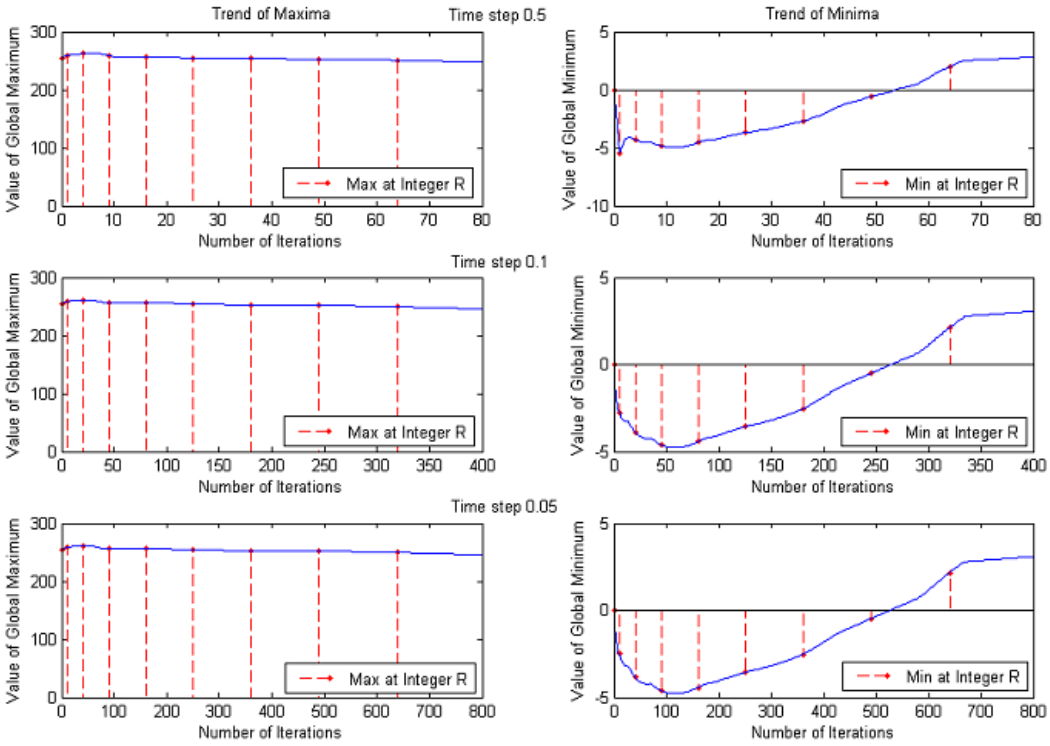

Figure 5: Plot, for different values of the time step, of the maximum and minimum values assumed by the pixels after each iteration of the MCM for one image taken at random from the web.

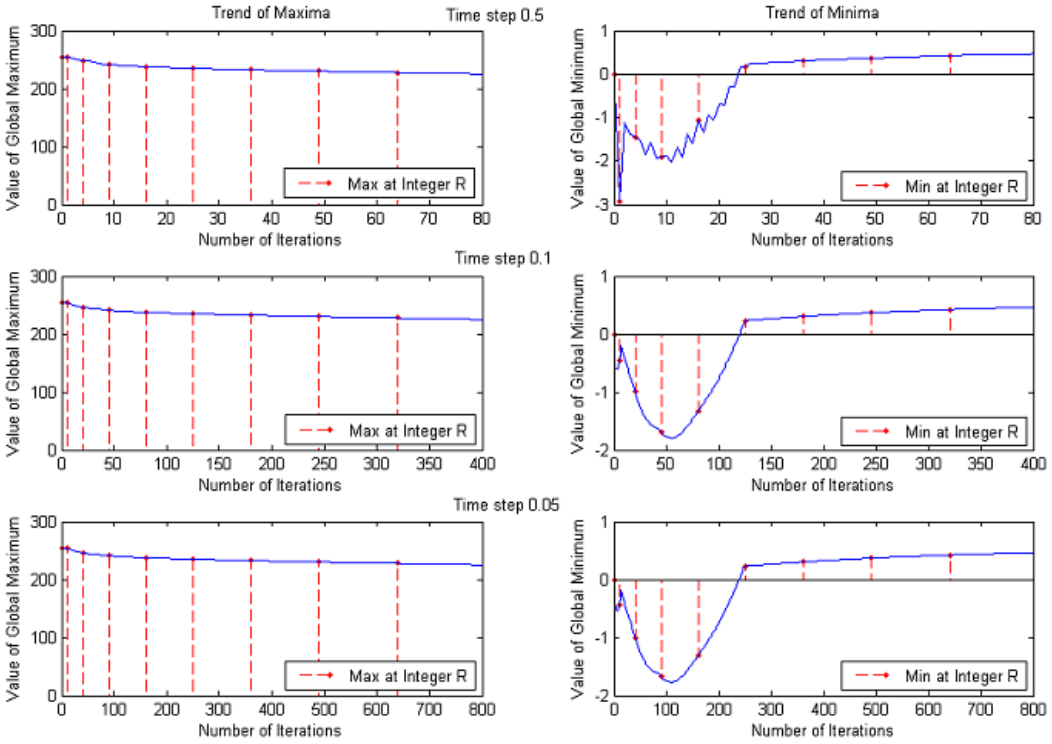

Figure 6: Plot, for different values of the time step, of the maximum and minimum values assumed by the pixels after each iteration of the MCM for an RGB image taken at random from the web.

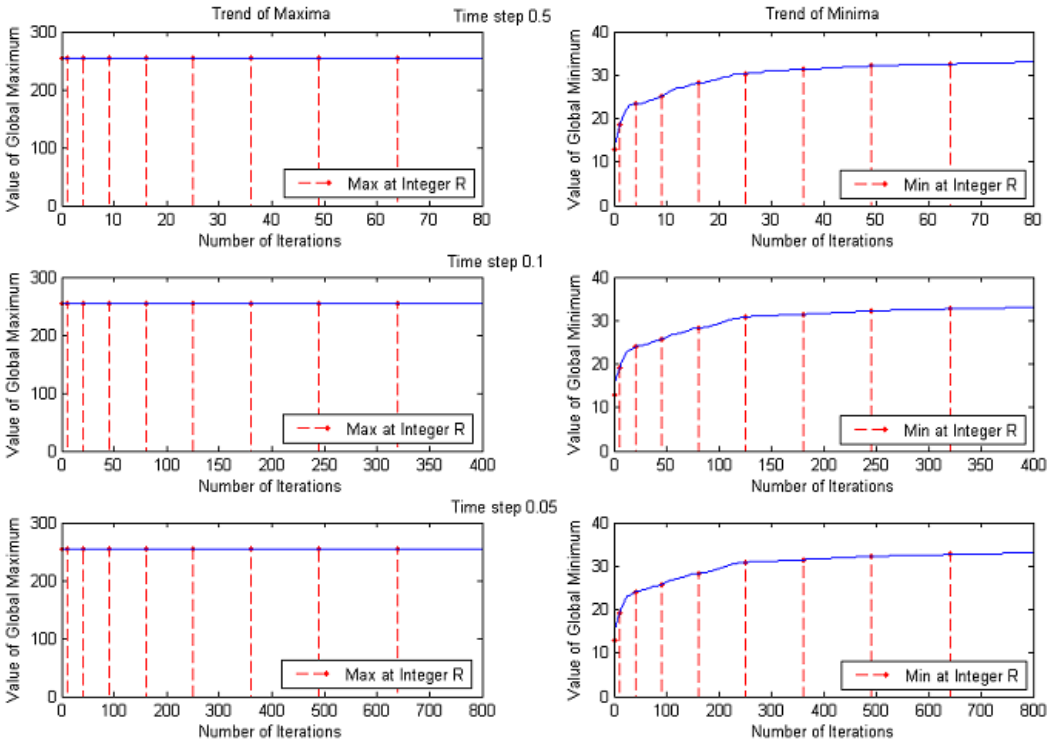

Figure 7: Plot, for different values of the time step, of the maximum and minimum values assumed by the pixels after each iteration of the MCM for a B/W image taken at random from the web.

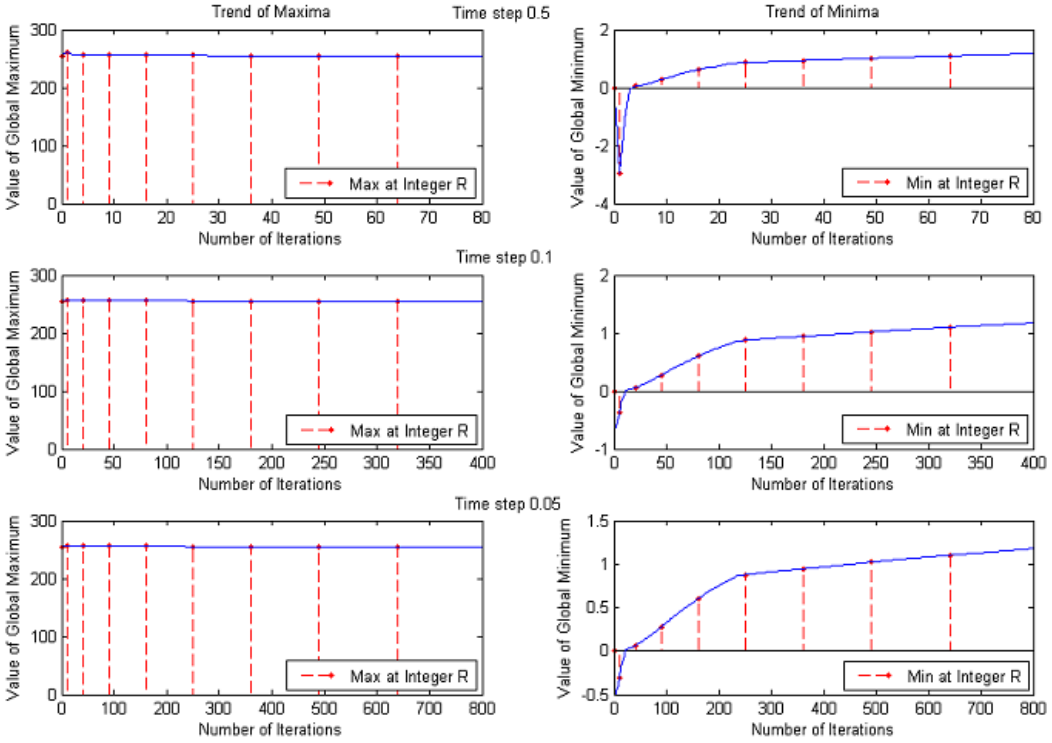

Figure 8: Plot, for different values of the time step, of the maximum and minimum values assumed by the pixels after each iteration of the MCM for a B/W image taken at random from the web.

To decide the best value we performed two additional experiments, analyzing the results for $\Delta t = 0.5$, $\Delta t = 0.1$ and $\Delta t = 0.05$.

Linear test.

We checked if a linear function remains still after the application of the MCM.

An assessment test is to consider a linear image and to evaluate the difference between the original and its MCM evolution (Figure 9). The fewer the differences are, the better the algorithm is. We noticed that:

- different choices of the threshold for the gradient ($T_g = 1$, $T_g = 2$, $T_g = 4$, $T_g = 8$ and $T_g = 16$) and different treatments of the pixels going outside the range $[0, 255]$ (renormalization/saturation at the end/at each step) produce identical results;
- if $\Delta t = 0.5$, 84.7% of the pixels do not change after the application of the algorithm at a normalized scale of 5, while for $\Delta t = 0.1$ or $\Delta t = 0.05$ more than 98.7% of the pixels remain still;
- the two images processed with time steps respectively equal to 0.1 and 0.05 are identical; therefore it would be a waste of time to run the algorithm with $\Delta t = 0.05$;
- when R increases the mirroring effect prevails, making the straight level lines bend.

Radial test.

Similarly, we checked if a radial function remains still after the application of the MCM (see Figure 10).

In this case the *difference image* is not useful, since we expect the level lines to remain circumferences, but to move. In practice we are interested in a *numerical* difference between the evolved level lines and real circles. The smaller the distance, the better the algorithm. To achieve this aim, we computed the difference between the maximum and minimum distance from the center of the image. The level lines too close to the edges were discarded since they would have been strongly deformed because of mirroring effect.

Ideally the difference should be zero, but such an eventuality does not occur even in the case of the original image. In fact the isolevel sets are not closed lines, but closed circular crowns of finite thickness, since there is an inevitable quantization operation when the image is written. The results follow:

- different choices of the threshold for the gradient and different treatments of the pixels out of the range produce identical output images;
- the best results are achieved with $\Delta t = 0.1$;
- when R increases the difference between maximum and minimum distance also tends to grow, but not so much with respect to the initial value (which is a measure of the precision achievable with our radial check). We can notice considerable deformations of the level lines only when they are collapsing or if they are close to the boundaries of the image.

In conclusion, $\Delta t = 0.5$ produces the worst results, while the differences between the other two possibilities are not enough to make us decide for $\Delta t = 0.05$, which will double the number of iterations and therefore processing time.

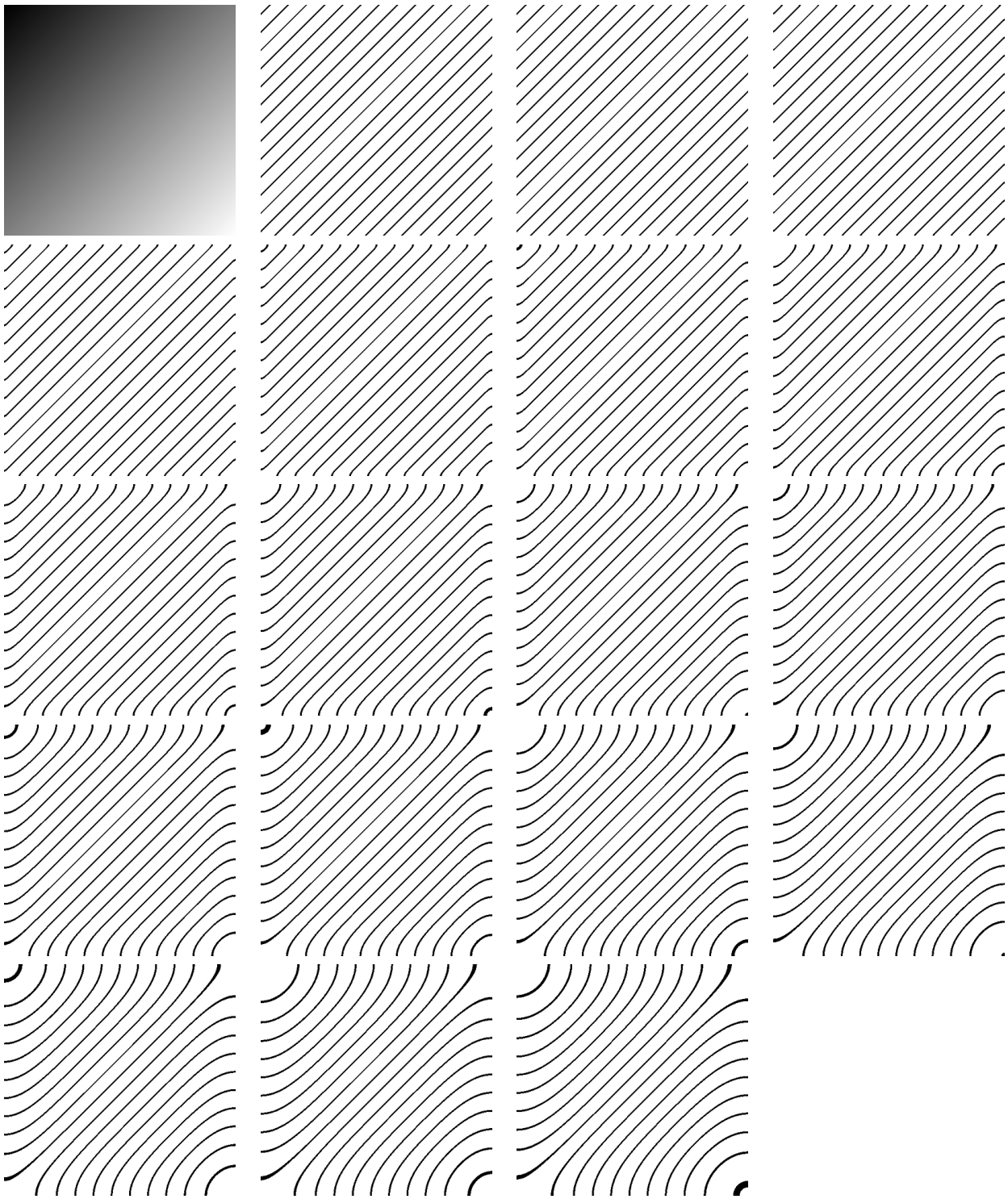

Figure 9: Top left, original image. Rest of images (from left to right and from top to bottom): MCM evolution of some level lines.

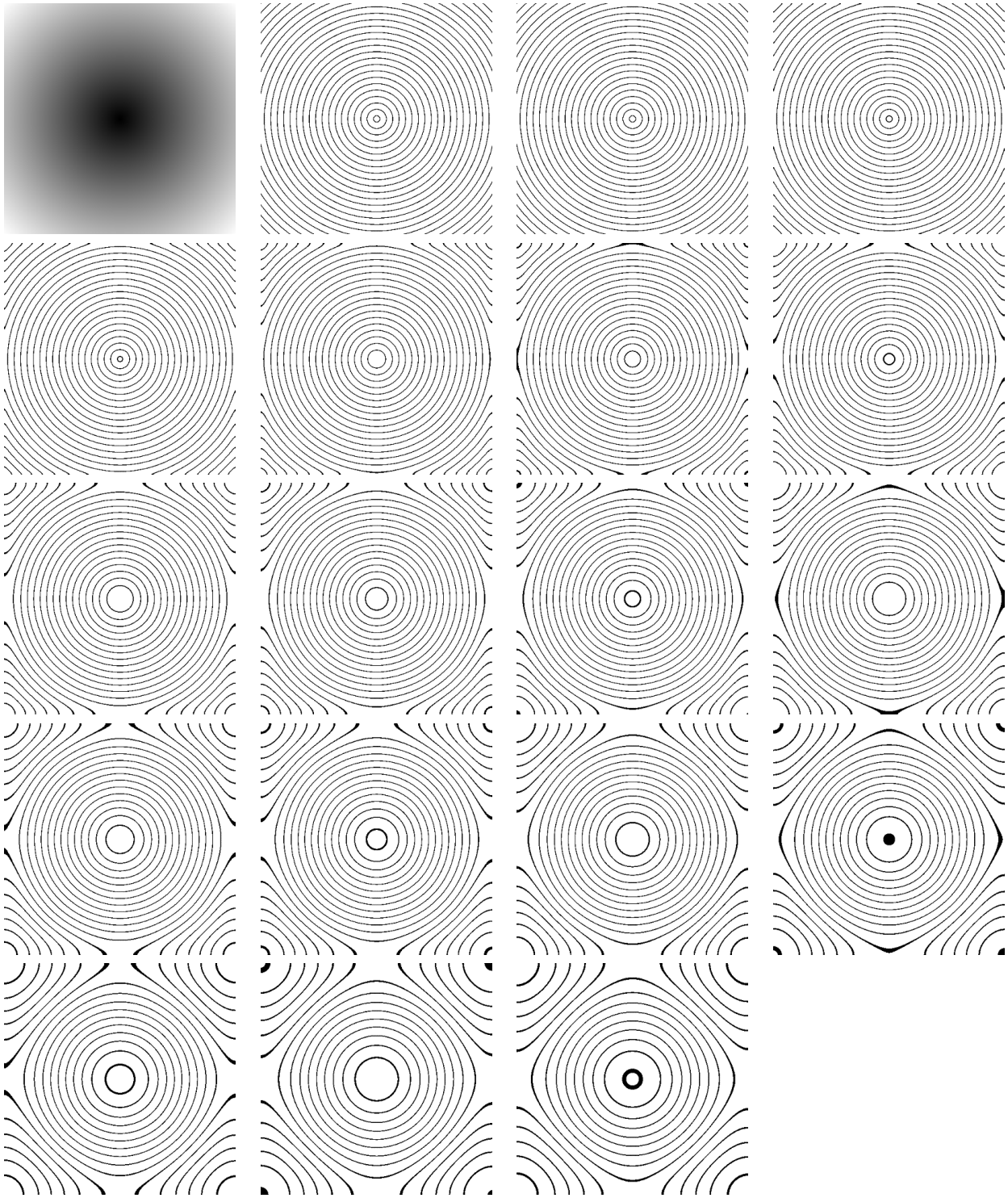

Figure 10: Top left, original image. Rest of images (from left to right and from top to bottom): MCM evolution of some level lines.

2.4.4 Relationship between Normalized Scale and Number of Iterations

We end the analysis regarding the MCM with a brief discussion on the relationship between normalized scale and number of iterations.

Our aim is to see how the formula

$$n_{iter} = \frac{R^2}{2\Delta t},$$

works out in practice.

To this end, we considered a digital image representing a circle (all the pixels whose distance from an assigned center is less or equal than the radius are black, the others are white). We then applied the algorithm until such a circle disappeared after a threshold with $\lambda = 127.5$. This condition is satisfied when the gray value of the center (which is the global minimum) exceeds 127.5.

To minimize the mirroring effect we create bigger images as the radius of the circles increases: the number of rows (and therefore the number of columns, as the images are squares) is ten times the value of the radius and does not go below a minimum of 100. The results are plotted in figures 11 to 14. As can be seen, numerical values stay close to theoretical ones for normalized scales in $[2, 50]$. Then the curve of experimental values starts increasing faster and faster. The gap noticed for R low is probably due to the fact that the *digital circles* are not a good approximation of real circles because of pixelization, while the problems for high values of R are to be found in the finite difference scheme itself and limit the actual range of validity of the formula.

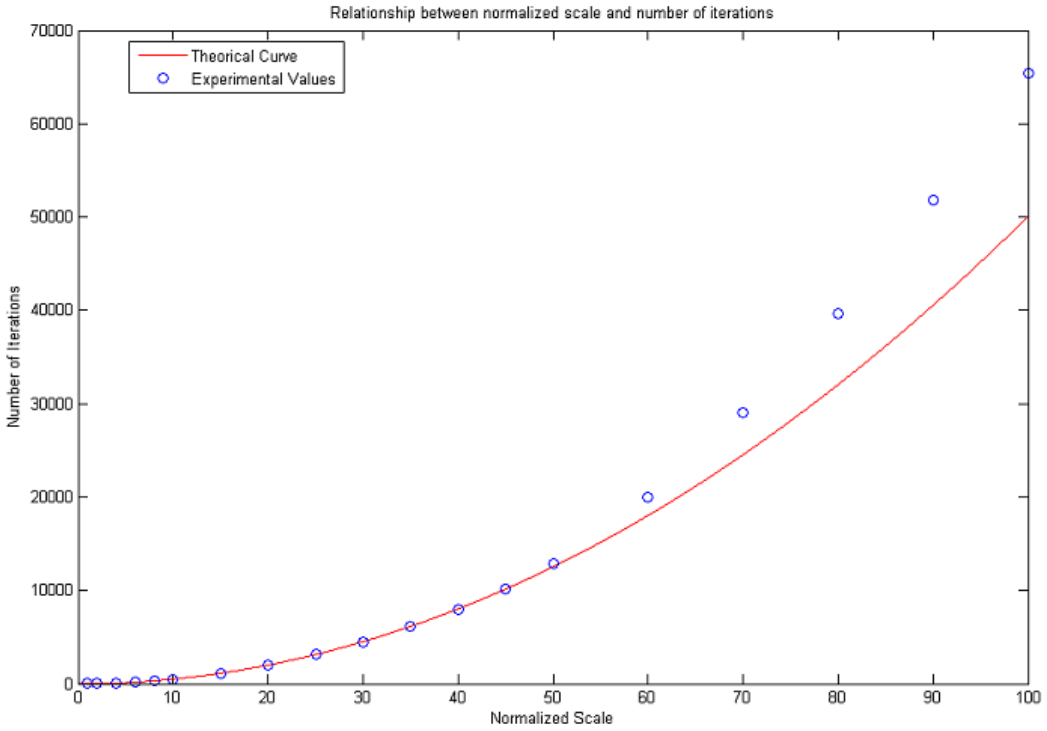

Figure 11: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 100]$

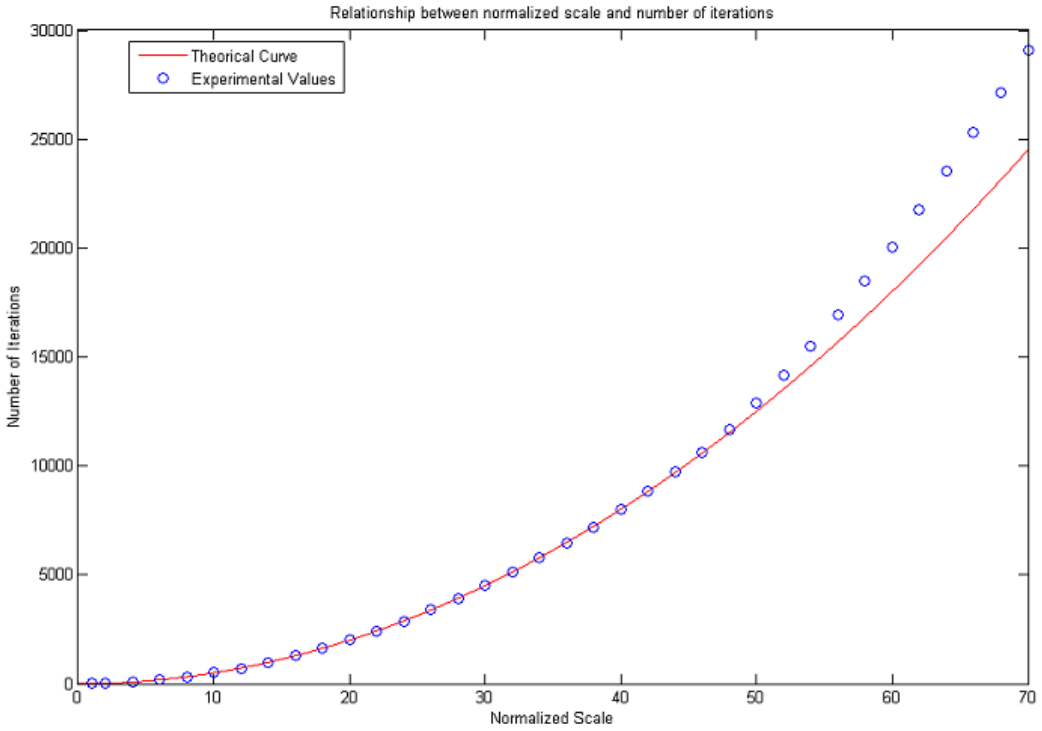

Figure 12: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 70]$

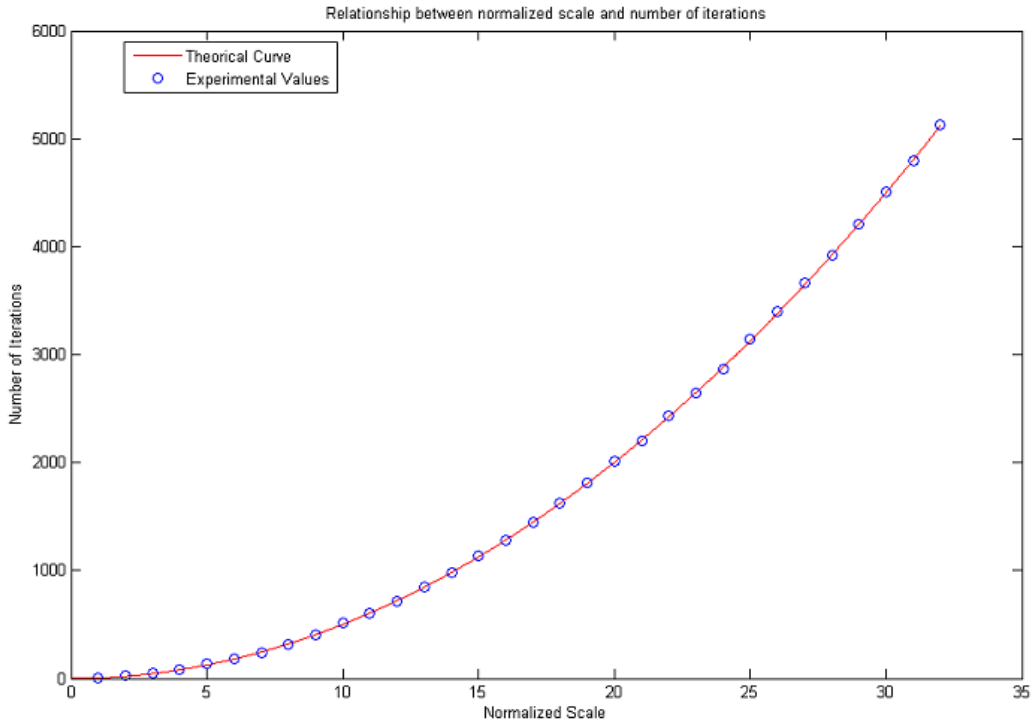

Figure 13: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 32]$

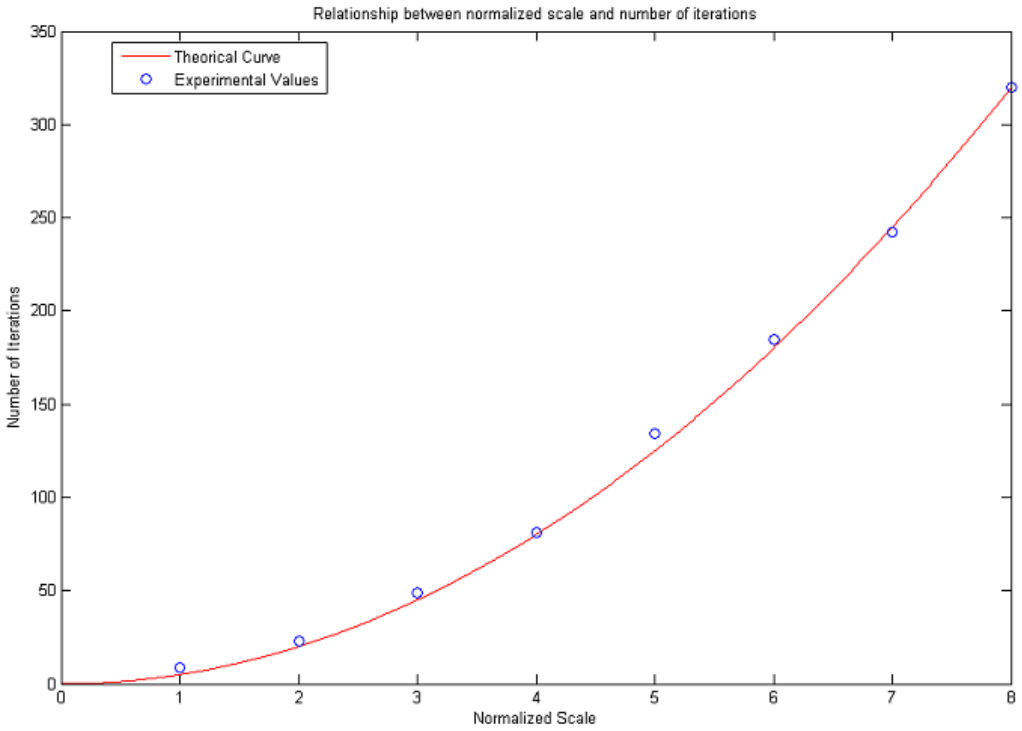

Figure 14: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 8]$

2.5 AMSS Implementations and Choice of the Parameters

Image extension and instability are treated similarly to MCM: we adopt a *mirror behaviour* with *saturation at the end* of the iterative part.

2.5.1 Stability

Setting

$$\eta_0(\theta) = |Du|^2 \lambda_0(\theta) = \frac{2u_x^2 + u_y^2 - u_x \cdot u_y}{4},$$

we remark that the global extrema blow up for any choice of time step, making the scheme unstable.

As before, in figures 15 to 17, we plot the trends of maxima and minima as functions of number of iterations until a normalized scale of almost 9. Red dashed lines connect the normalized scales to the corresponding maximum values.

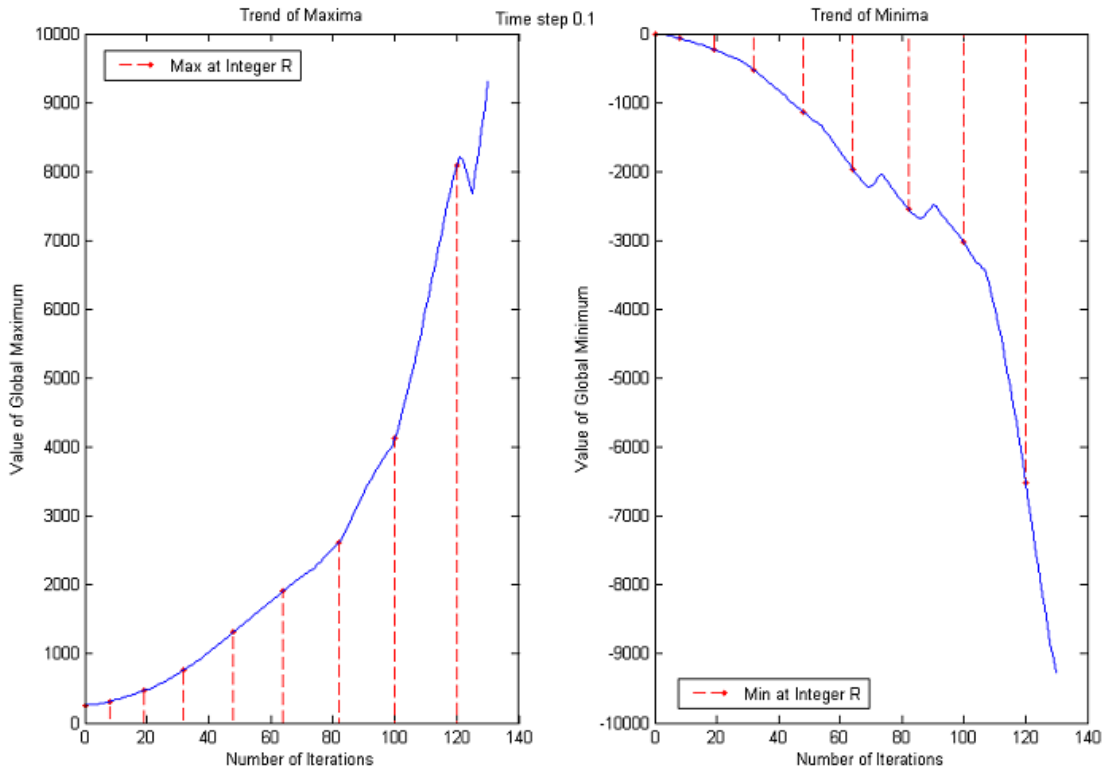

Figure 15: Plot of the maximum and minimum values assumed by the pixels after each iteration of the AMSS, up to a normalized scale of almost 9, with time step 0.1.

Instead, the other choice of λ_0 and consequently of η_0 makes the algorithm *almost* stable.

$$\eta_0(\theta) = |Du|^2 \lambda_0(\theta) = 0.5 \cdot (u_x^2 + u_y^2) - \frac{u_x^2 \cdot u_y^2}{u_x^2 + u_y^2}$$

In this way λ_0 is a trigonometric polynomial of degree 4, so we have to manage separately the case of zero gradient to avoid a division by 0.

Since

$$\lim_{(u_x,u_y)\to(0,0)} \frac{u_x^2 \cdot u_y^2}{u_x^2 + u_y^2} = 0,$$

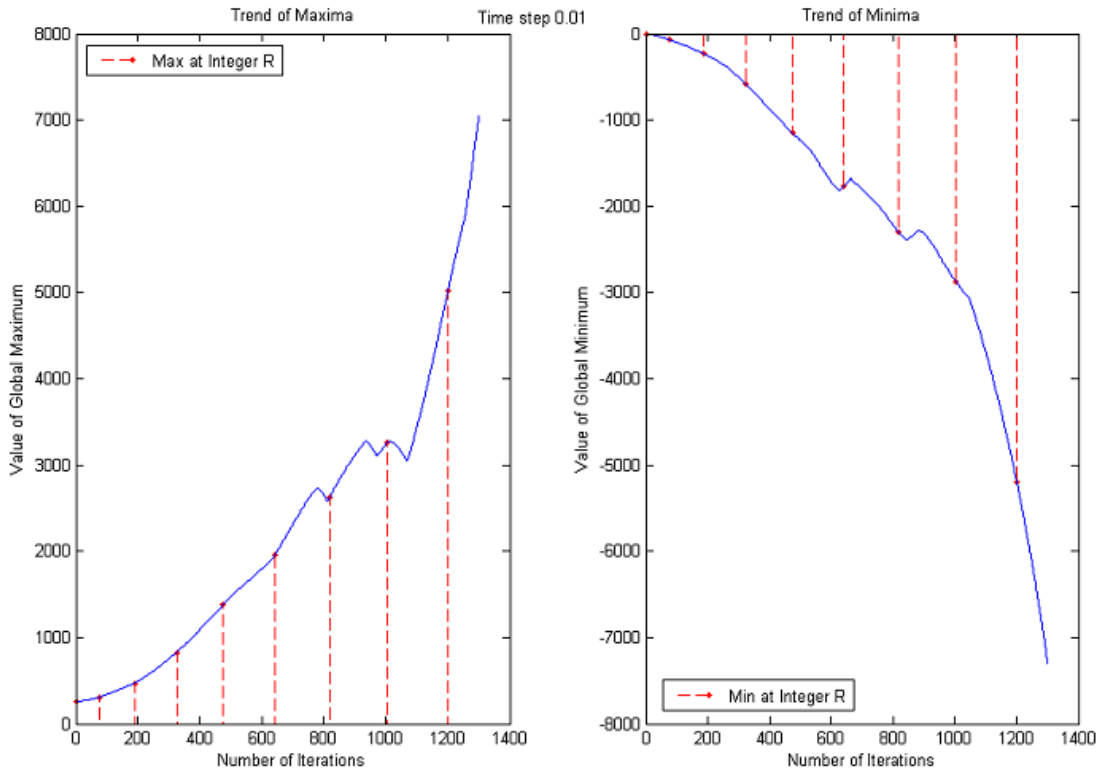

Figure 16: Plot of the maximum and minimum values assumed by the pixels after each iteration of the AMSS, up to a normalized scale of almost 9, with time step 0.01.

the first idea is to put a fictitious extremely low arithmetic threshold, say $T_g = 10^{-6}$, and when the gradient is below it to take simply

$$\eta_0 = 0.5 \cdot (u_x^2 + u_y^2).$$

However this solution is not consistent with the actual behavior of the continuous equation at the extrema: isolated extrema will not evolve and we know that is wrong, since small sets should collapse by curvature motion. To overcome the problem, we replace the AMSS with the Heat Equation when the gradient is too low.

2.5.2 Gradient Threshold

When making a choice of the threshold, we have to ensure two different tasks.

The application of finite difference schemes deals with floats, while for visualization we are obliged to adopt integer values. From a theoretical point of view, a *continuous image* is a smooth function from $\mathbb{R}^2$ into $\mathbb{R}$. To obtain the *digital image* we sample and round. Conversions from real to integer numbers yield rounding errors and make the direction of a gradient small in modulus substantially random. Nevertheless such approximation errors propagate only for the first iterations and then vanish because the whole iterative part is performed with a 4-bytes (float) precision. So eventually the threshold on the gradient is useful only to move isolated extrema.

Experimental data show that the way we pass from the initially higher value to the successive smaller one does not modify final results. So the on-line version presents $T_g = 4$ for the very first iteration and $T_g = 1$ for the following ones. Simple calculations show that the initial threshold $T_g = 4$ implies an uncertainty in the evaluation of the gradient smaller than 15% in the worst case, while the value $T_g = 1$ means that small sets will shrink to points in a reasonable number of iterations.

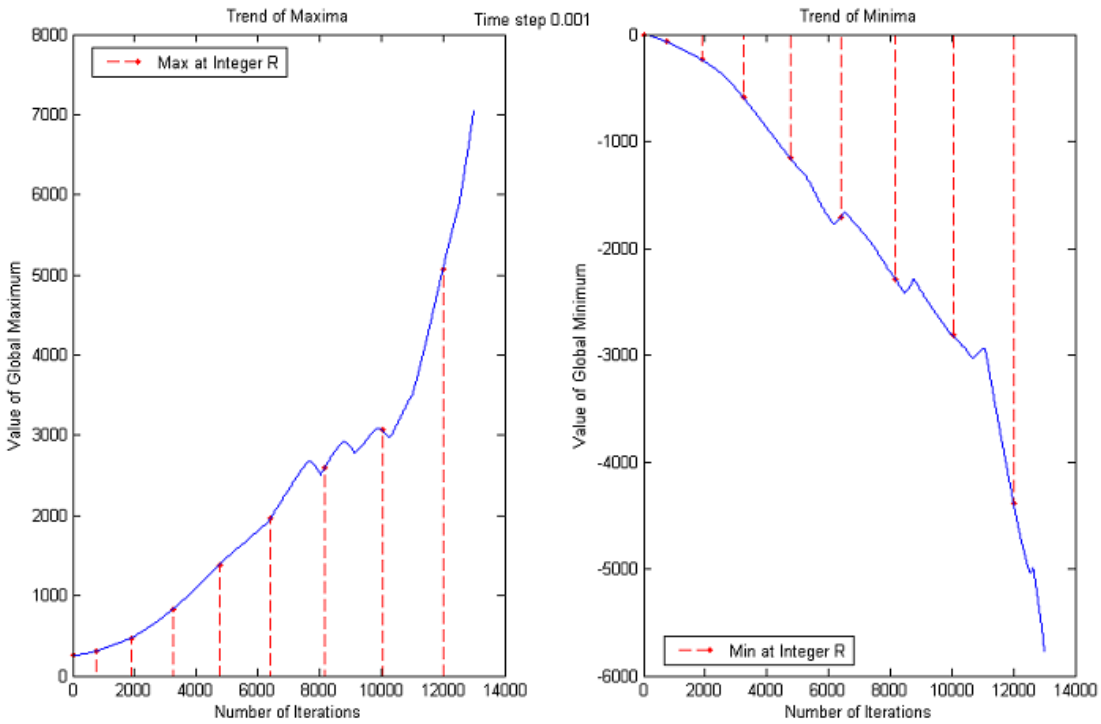

Figure 17: Plot of the maximum and minimum values assumed by the pixels after each iteration of the AMSS, up to a normalized scale of almost 9, with time step 0.001.

If we hadn't added such a geometrical threshold, the final validity check for the theoretical formula that links numbers of iterations with normalized scales would have been impossible.

2.5.3 Time Step

The tests were performed for $\Delta t = 0.5$, $\Delta t = 0.1$ and $\Delta t = 0.05$ and concern stability, linearity, radiality (as previously presented) and ellipticity.

Stability.

A first check allows to exclude immediately $\Delta t = 0.5$, because this value makes the algorithm unstable.

The results are displayed in figures 18 to 21. We are considering ranges up to a normalized scale of 23, that is why we plot the first 100 iterations for $\Delta t = 0.5$, the first 500 for $\Delta t = 0.1$ and the first 1000 for $\Delta t = 0.05$. Analyzing a wider range of numbers of iterations for $\Delta t = 0.5$, we notice that the global maximum continues decreasing and that the values of the minimum oscillate remaining significantly below 0. The other two choices of time step instead yield similar results: for some images the maximum decreases monotonously, while for other ones remains close to the upper bound 255; the minimum, after the initial unstable behaviour, returns always bigger than 0.

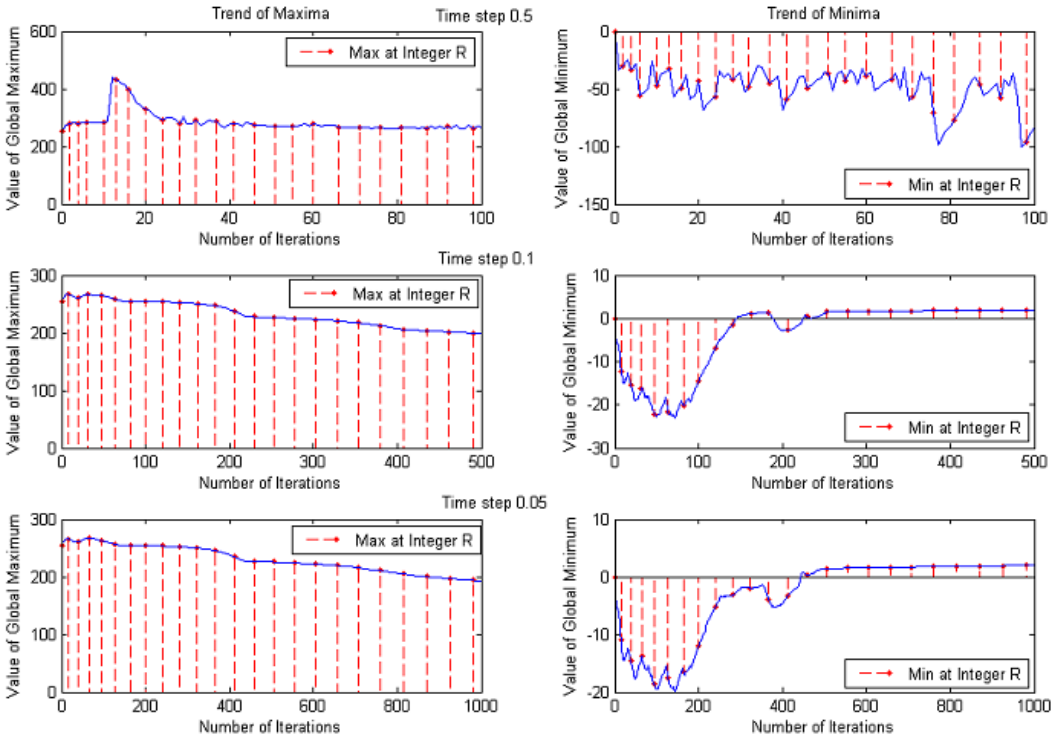

Figure 18: Plot of the maximum and minimum values assumed by the pixels after each iteration of the AMSS for one image taken at random from the web, for different time steps.

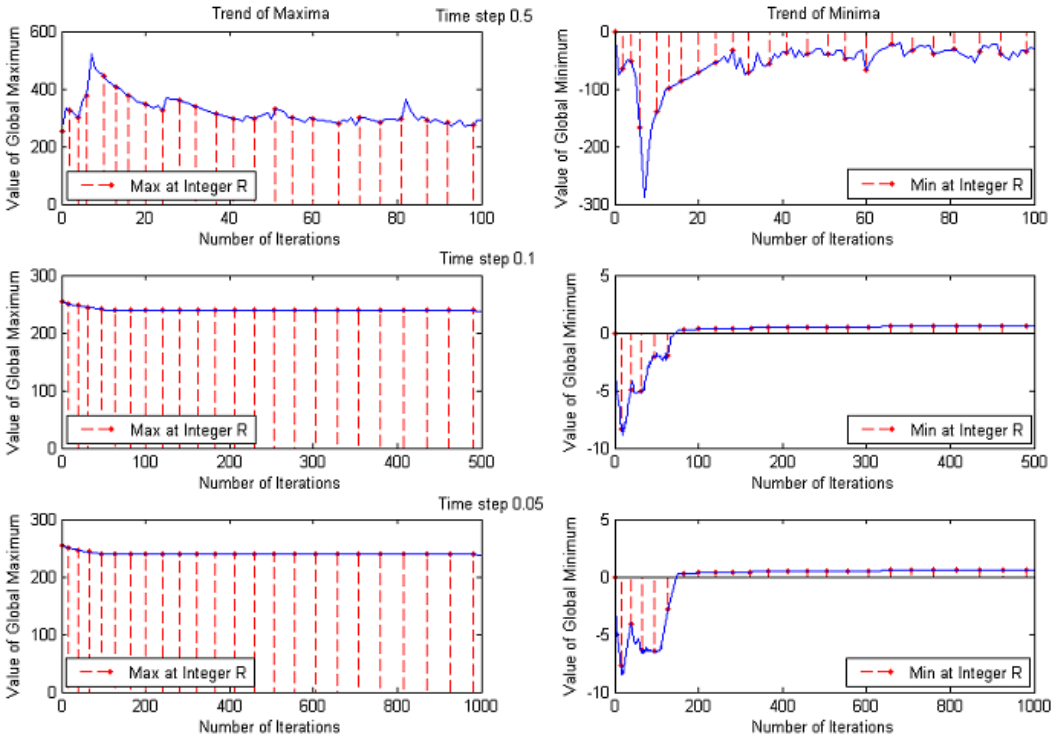

Figure 19: Plot of the maximum and minimum values assumed by the pixels after each iteration of the AMSS for an RGB image taken at random from the web, for different time steps.

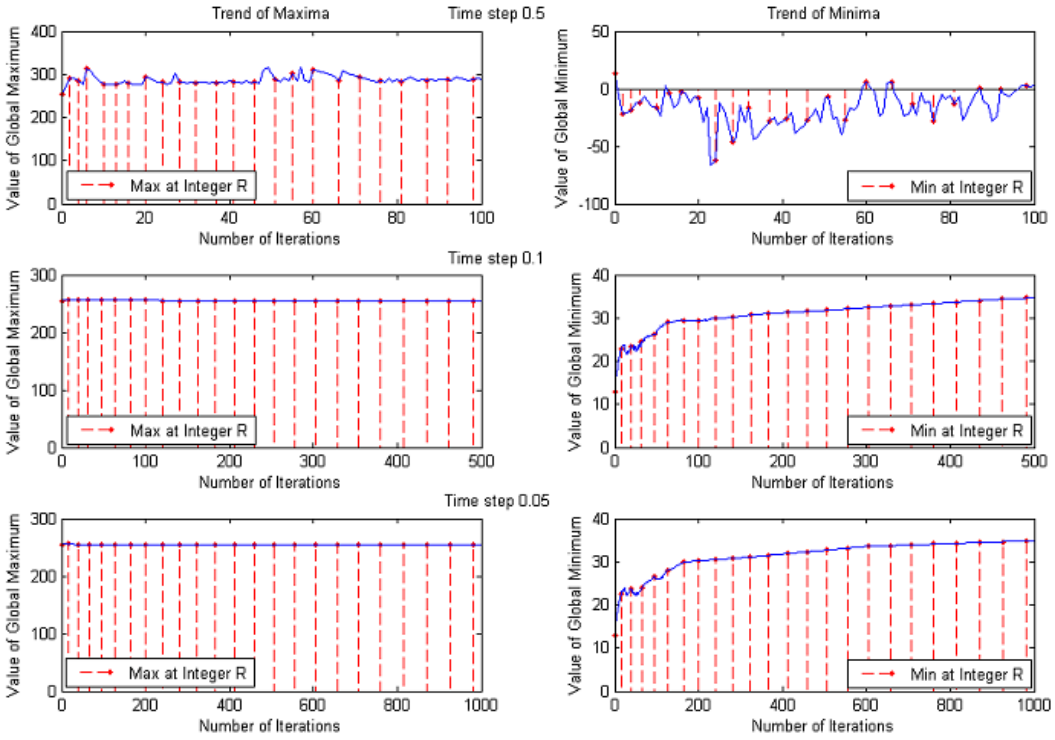

Figure 20: Plot of the maximum and minimum values assumed by the pixels after each iteration of the AMSS for a B/W image taken at random from the web, for different time steps.

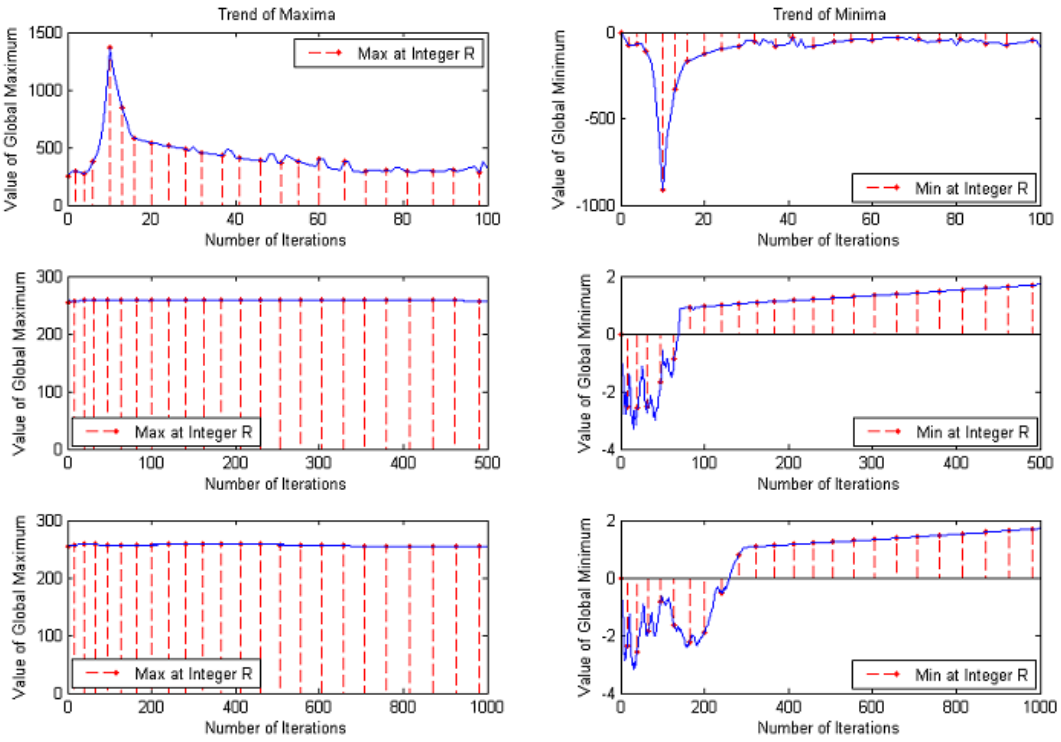

Figure 21: Plot of the maximum and minimum values assumed by the pixels after each iteration of the AMSS for a B/W image taken at random from the web, for different time steps.

Linearity and radiality.

Different gradient thresholds and different treatments of the pixels going out of the range produce identical output images. As before, the optimal choice for time step is $\Delta t = 0.1$ and, as can be seen in Figure 22, AMSS preserves linearity and radiality better than MCM.

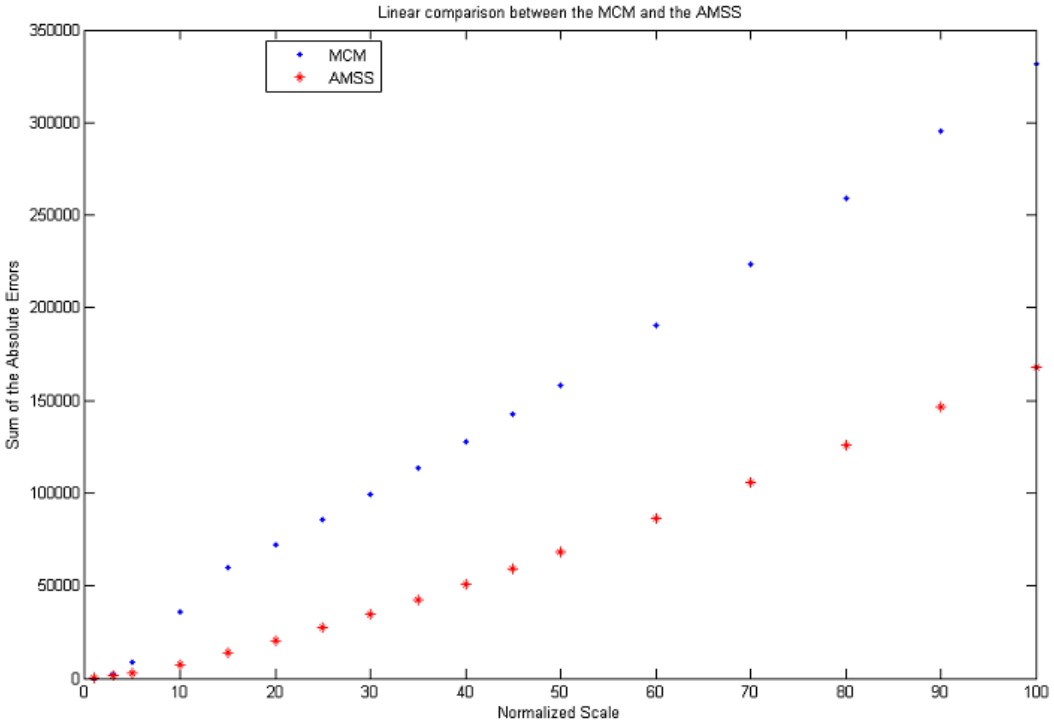

Figure 22: Linear comparison between MCM and AMSS.

Ellipticity.

Our aim is to verify that the evolved level lines (see Figure 23) are ellipses of constant axial ratio (equivalently, of constant eccentricity). In practice, due to quantization and pixelization isolevel sets are not ellipses but elliptical crowns. We are content with the numerical evaluation of the difference between the actual level lines and a real ellipse. To reduce the mirroring effect, for computations we considered only isolevel sets not too close to the boundaries of the image. Then we evaluated the major and minor axes, the focuses and the maximum and minimum sum of the distances between a generic point and the focuses. The difference between these two values is to be minimized.

As usual, different thresholds for the gradient and different treatments of the pixels out of the range produce the same output images. The best choice for the time step is $\Delta t = 0.05$, but the improvement in performances with respect to the case $\Delta t = 0.1$ is so small that makes us prefer $\Delta t = 0.1$ for timing considerations. The axial ratio decreases slowly but almost monotonically, passing from the original value of 2 to 1.25 just before the isolevel set 20 disappears. The error on the distance remains quite close to the initial value of 4: unfortunately even for the initial image the accuracy in the detection of ellipses is scarce.

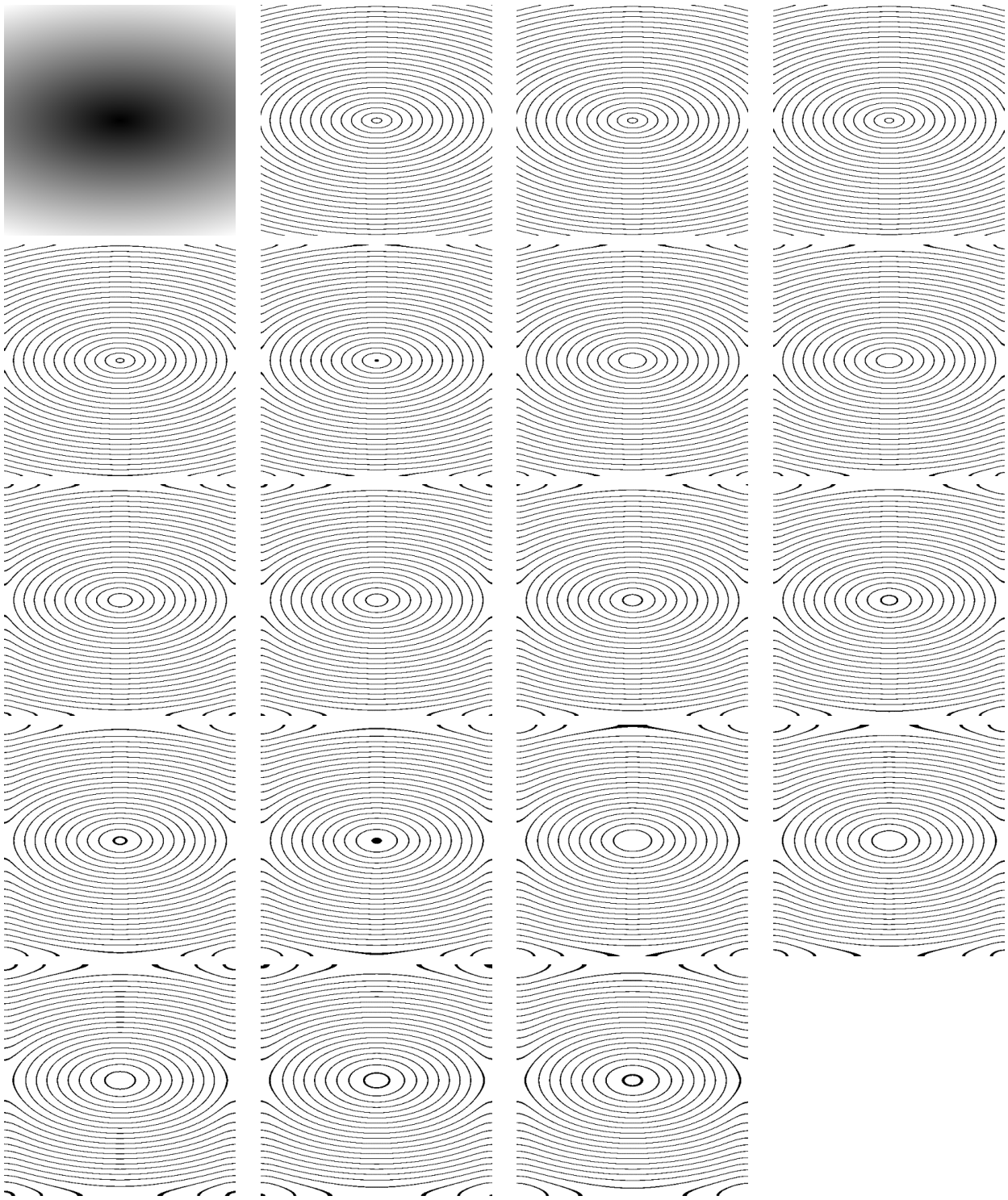

Figure 23: Top left, original image. Rest of images (from left to right and from top to bottom): AMSS evolution of some level lines.

2.5.4 Relationship between Normalized Scale and Number of Iterations

As for the MCM, we complete our analysis with a discussion about the number of iterations versus the normalized scale. More precisely, we are interested in the range of values for which the following theoretical formula is valid in numerical simulations

$$n_{iter} = \frac{3}{4\Delta t} R^{\frac{4}{3}}.$$

As can be seen in the following graphs (figures 24 to 27), the approximation is reasonable until a normalized scale of about 34. Afterwards, the experimental values get further and further from the theoretical ones. For R low the problems are probably due to pixelization and scarce similarity between digital and analogical circles. However the results seem to be worse than the ones obtained for the MCM: for R in the range $[10, 34]$ there is a percent error always higher than 5%, while for the MCM, if R is in the range $[14, 46]$, then the percent error does not exceed 1%.

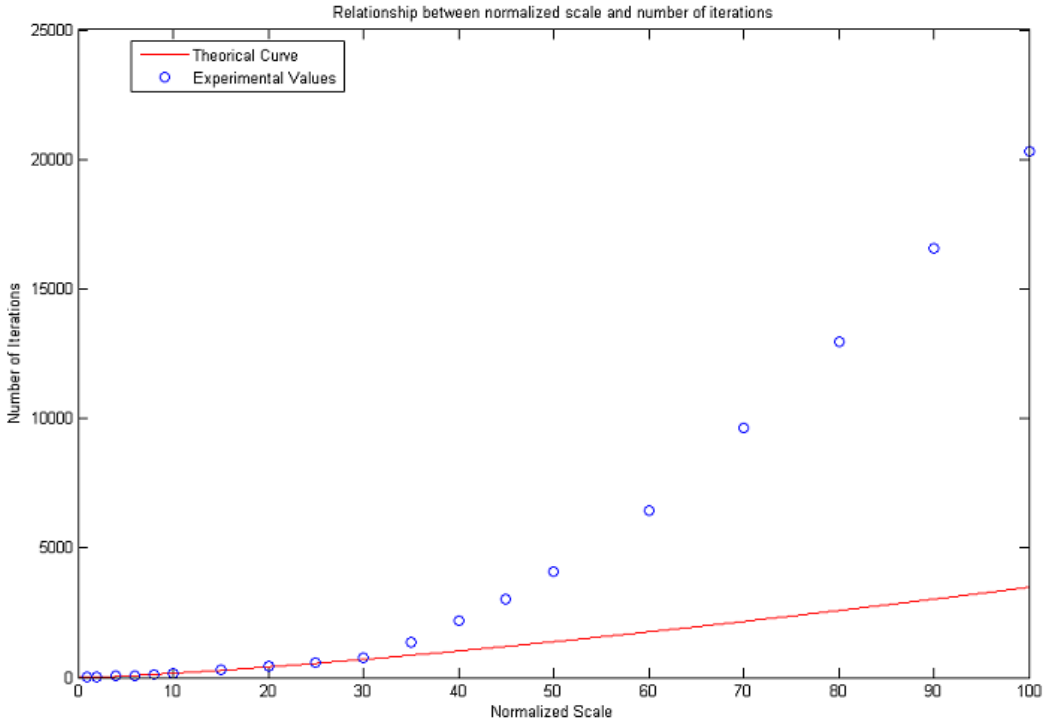

Figure 24: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 100]$

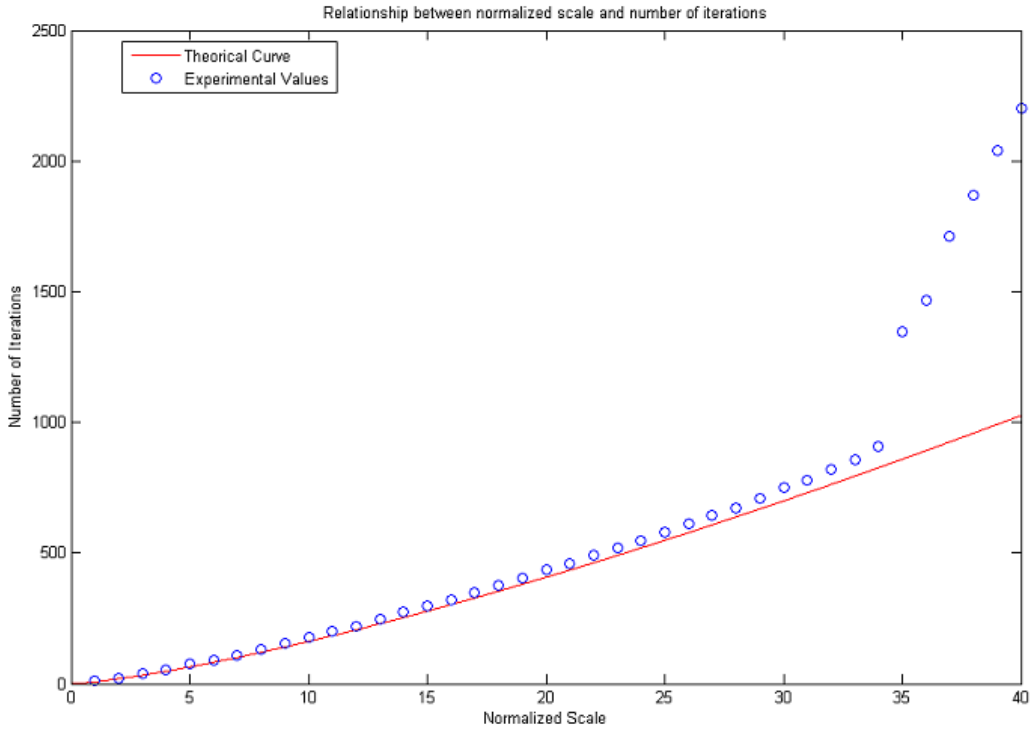

Figure 25: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 40]$

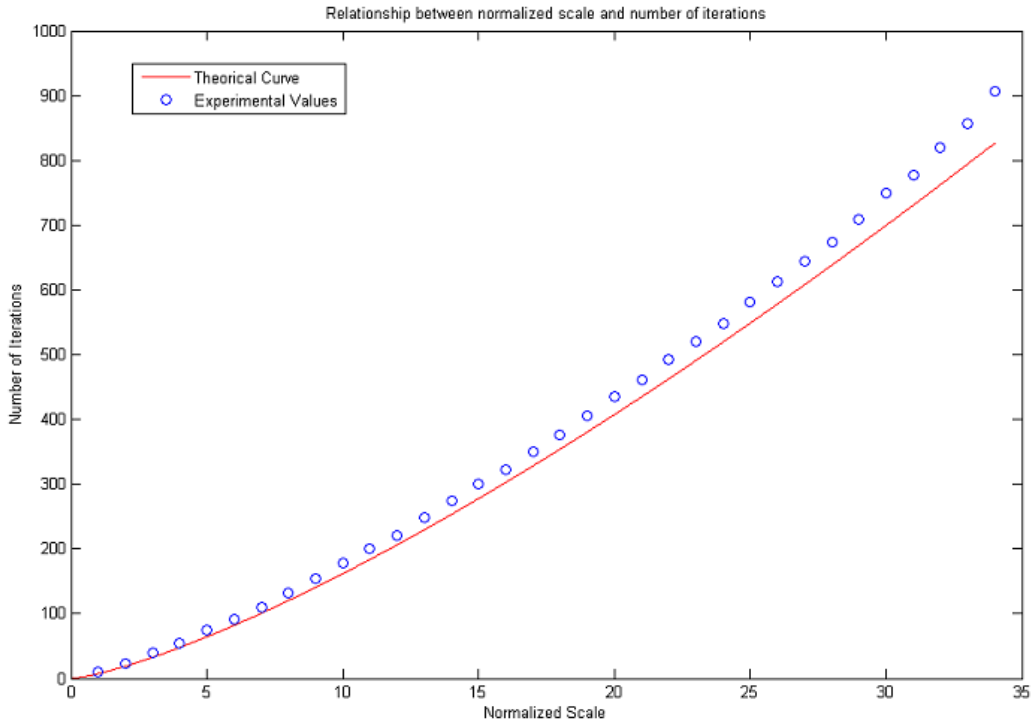

Figure 26: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 34]$

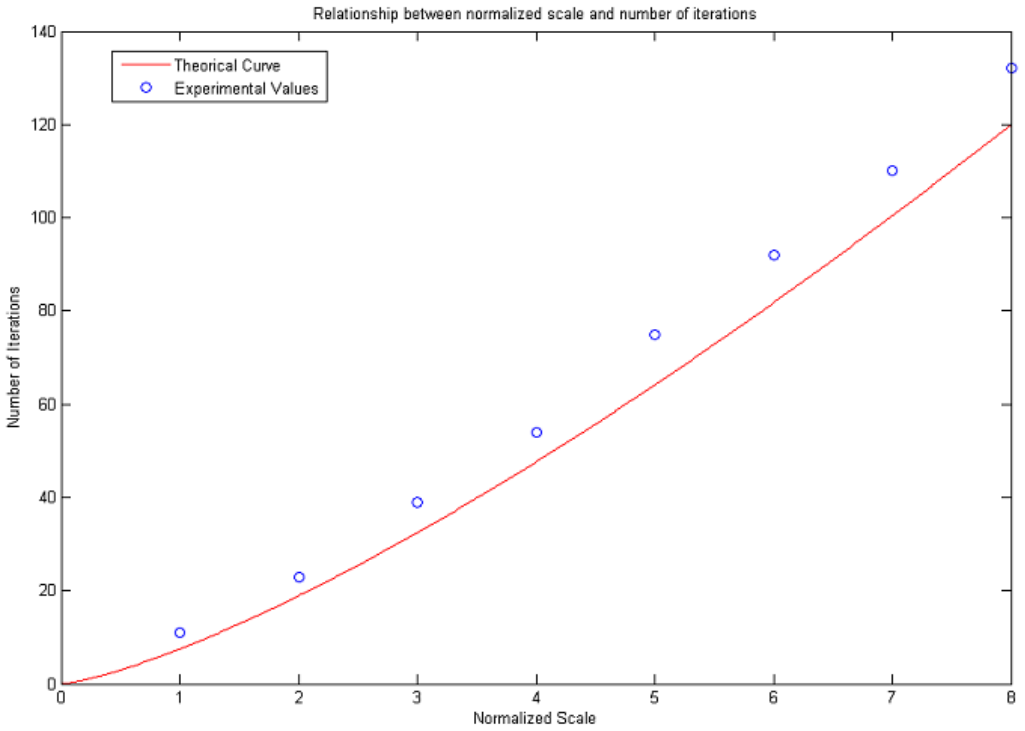

Figure 27: Relationship between normalized Scale and number of iterations for normalized scales in the range $[1, 8]$

3 Examples

Remark: the normalized scale R is always given with respect to the original image. To find out the normalized scale for the zoomed detail, the value of R must be multiplied by the zooming factor.

3.1 Geometrical Figures

Let's start our gallery with some examples taken from geometry (figures 28 and 29). The MCM is applied at increasing integer normalized scales up to a maximum value of 10.

As expected, the corners gradually shrink becoming round and a *fattening effect* occurs between joint level sets.

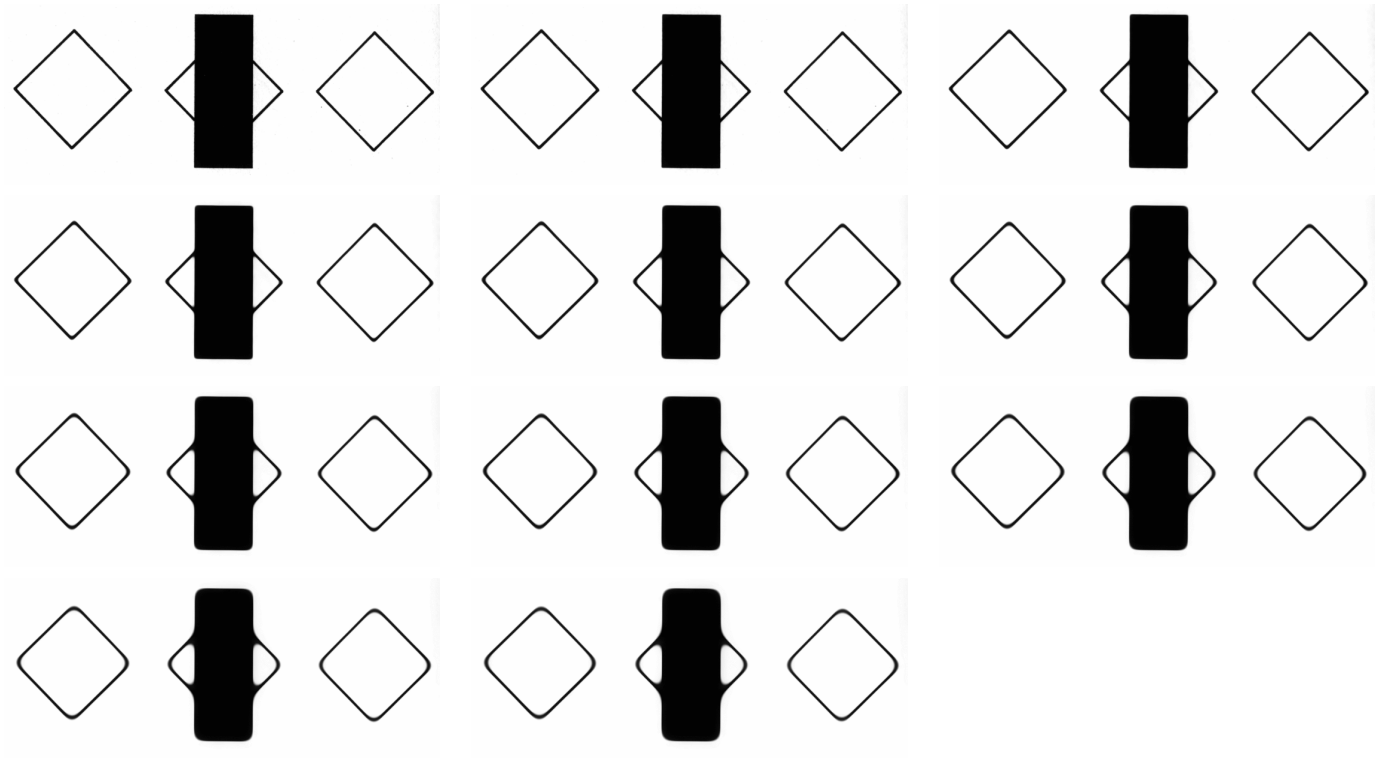

Figure 28: Top left, original image. Rest of images (from left to right and from top to bottom): MCM evolution of some level lines.

In Figure 30 we apply the AMSS to a chessboard image until the middle squares disappear ($R \approx 34$). This example also shows that the algorithm is **not contrast invariant**, which means that new gray levels can appear in the output image. The little gray shade then fattens conquering progressively almost all the space available.

On the other hand, curves with thin boundary break and eventually collapse. This is a major drawback of pixel-based algorithms, which are limited to the initial grid and are blind to the intrinsic geometry of the image. A subpixel vision is required for accurate evolutions[2].

Finite difference schemes are bad tools when dealing with figures such as the one depicted in Figure 31. In fact the thickness of the ellipse is not big enough to create a black ribbon that can be evolved by the algorithm according to the theory. Generally speaking, making weighted means with the values of the first neighbours is a good idea when those values are continuously varying, but turns out to be highly problematic when handling black stripes on white backgrounds: if the majority of the neighbours are white, also the curve is going to turn white.

[2]see for example *level lines shortening*(http://www.ipol.im/pub/algo/cmmm_image_curvature_microscope/).

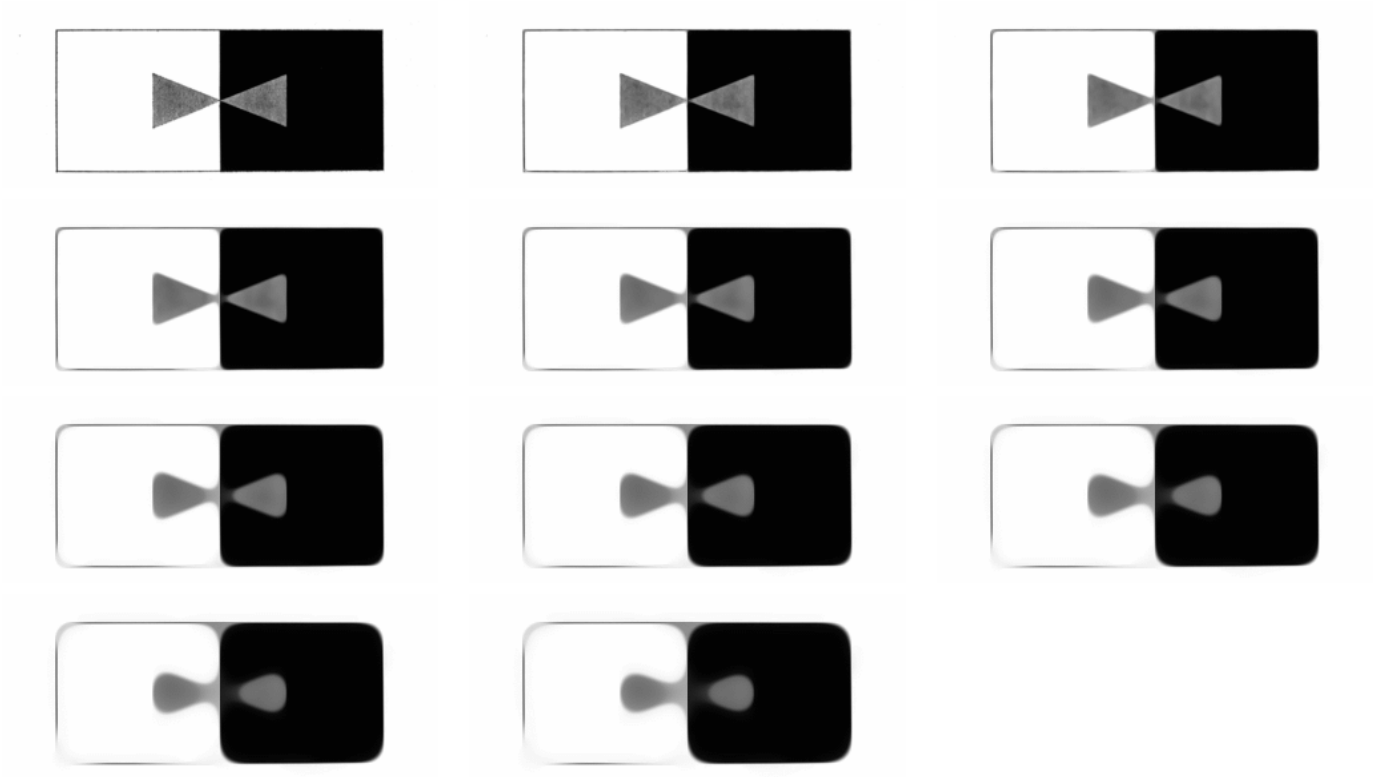

Figure 29: Top left, original image. Rest of images (from left to right and from top to bottom): MCM evolution of some level lines.

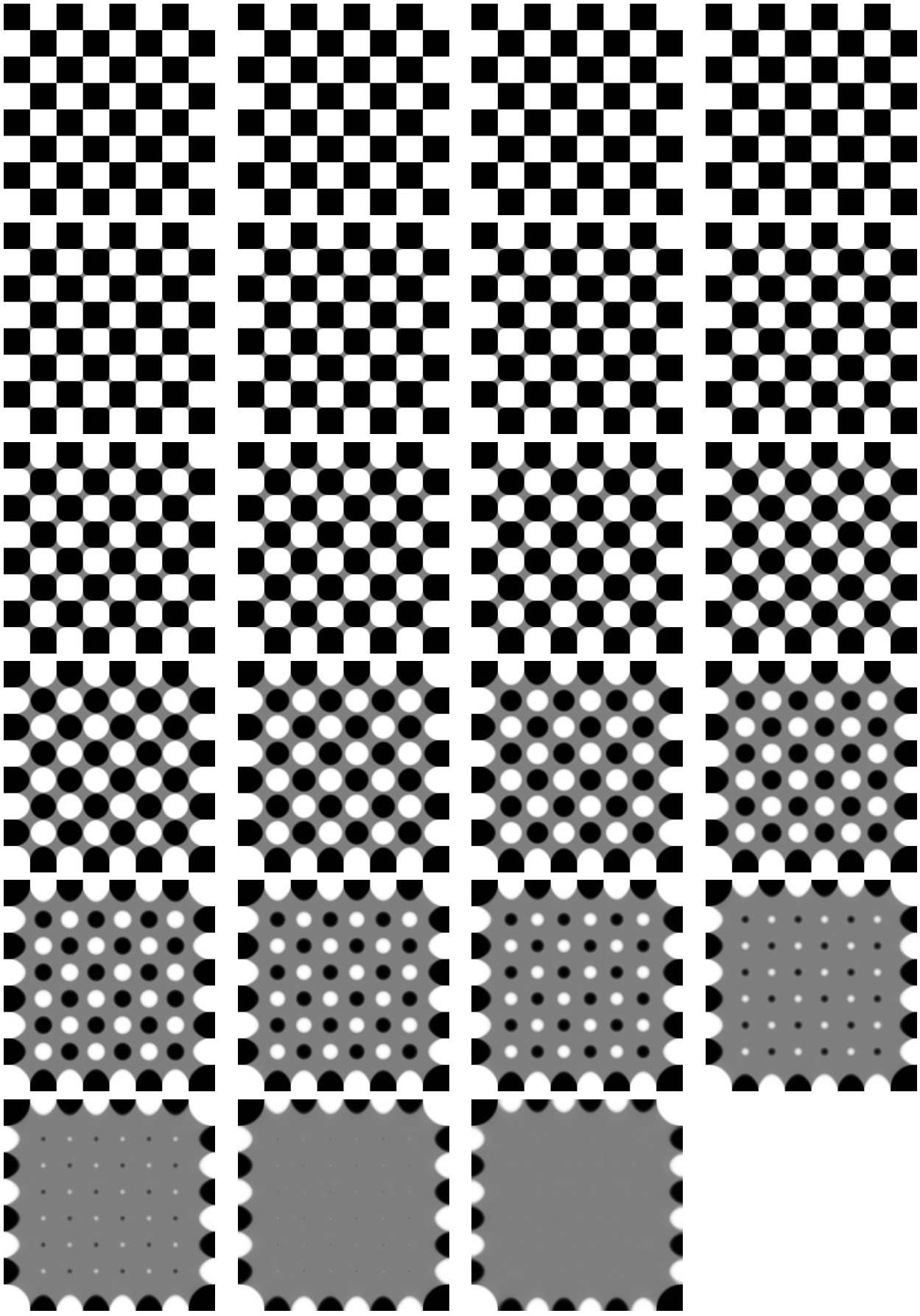

Figure 30: Top left, original image. Rest of images (from left to right and from top to bottom): AMSS evolution of some level lines.

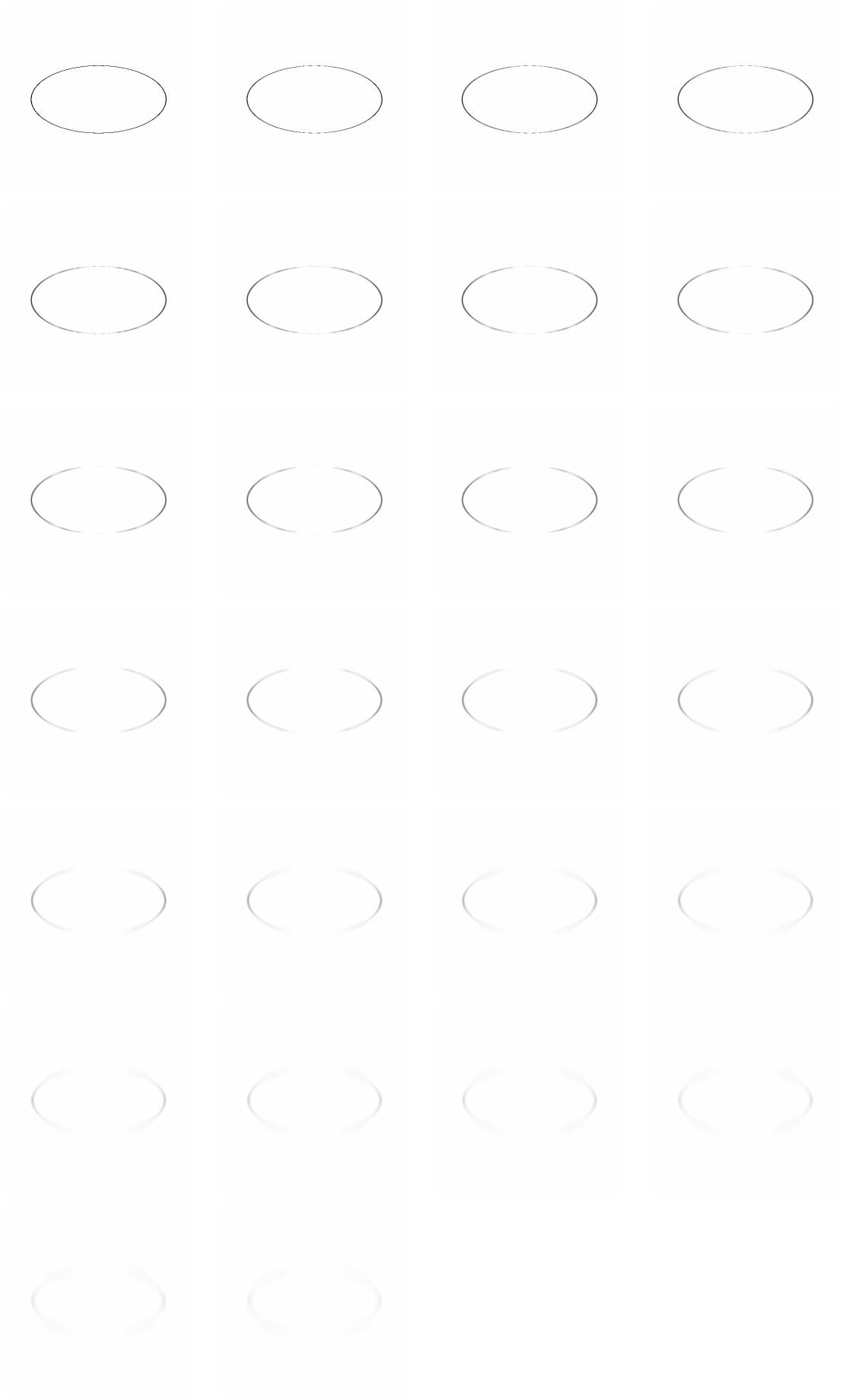

Figure 31: Top left, original image. Rest of images (from left to right and from top to bottom): AMSS evolution of some level lines.

3.2 Graphs

In Figure 32 we consider a 3× zoomed detail of one of the graphs analyzed before and we then apply the MCM and AMSS evolutions at normalized scale 1. Pixelization is eliminated and a visual improvement can be recognized, since the transition between colored parts and white background becomes smooth. The main drawback lies instead in the attenuation of shades, which makes the colors seem dull.

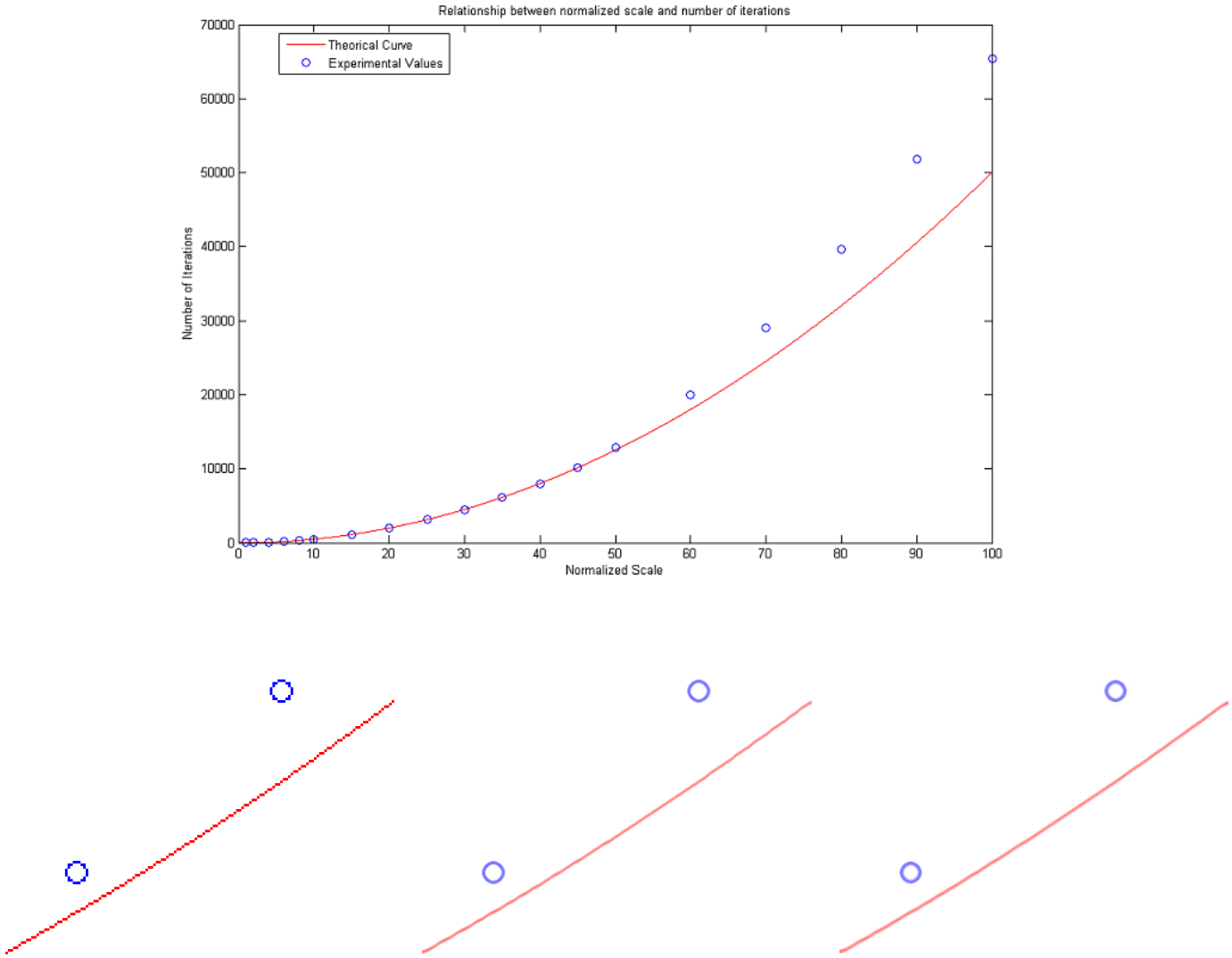

Figure 32: Top, original graph. Bottom: left, zoom by pixel duplication; center, MCM evolution at scale 4/3; right, AMSS evolution at scale 4/3.

3.3 Signs

In the following examples, we have a look at the geometry of road signs.

In Figure 33, from the original image we extract a detail and we apply the MCM and the AMSS for $R = 1$. MCM/AMSS improve the quality of the image, because the pixelization effects disappear (see the corners of the **P** and especially the sketch representing the bike). The normalized scale has

been chosen wisely because a higher value of R would have lead to the elimination of some particulars, such as the handle of the bicycle.

Figure 33: Top, original image. Bottom: left, 3× zoomed detail from original; center, MCM evolution at scale 1; right, AMSS evolution at scale 1.

For the original photo in Figure 34-top, we zoom at first by a factor of 2 considering the whole sign and then by a factor of 6, focusing only on the truck ban. The results of applying AMC and AMSS to these images are shown in the two last rows of Figure 34. In the middle row we see that the bigger writings are enhanced and we notice a real improvement, while the smaller ones in the lower part of the image appear attenuated. To obtain a better resolution one should zoom some more, discarding the rest of the sign. In the last row of the figure the reconstruction of the circular shape is good, but the lines that form the drawing of the bus are too thin to be satisfactorily managed by the algorithm.

Figure 34: Top, original image. Middle row: left, 2× zoomed detail from original; center, MCM evolution at scale 1; right, AMSS evolution at scale 1. Bottom: left, 6× zoomed detail from original image in previous row; center, MCM evolution at scale 2/3; right, AMSS evolution at scale 2/3.

In Figure 35 a new example is shown. Pixelization effects are removed, the image becomes smoother and the result seems more natural to the human eye. A final example is shown in Figure 36.

Figure 35: Top, original image. Bottom: left, 4× zoomed detail from original; center, MCM evolution at scale 1/2; right, AMSS evolution at scale 1/2.

Figure 36: Top, original image. Bottom: left, 4× zoomed detail from original; center, MCM evolution at scale 1/2; right, AMSS evolution at scale 1/2.

3.4 JPEG Artifacts in Text and Handwriting

The well known JPEG coding is a commonly used method to compress images. Differently from TIF and PNG formats which are lossless, JPEG is lossy in the sense that after dividing the image in 8×8 pixel blocks and applying a discrete cosine transform (DCT), we make a quantization operation. In this way we lose inevitably some information, which from a visual point of view means artifacts, such as Gibbs's oscillations, staircase noise or checkerboarding. The MCM and the AMSS can be used to reduce these unwanted effects and actually yield a visual improvement if we zoom a specified part of the text (see results in Figure 37).

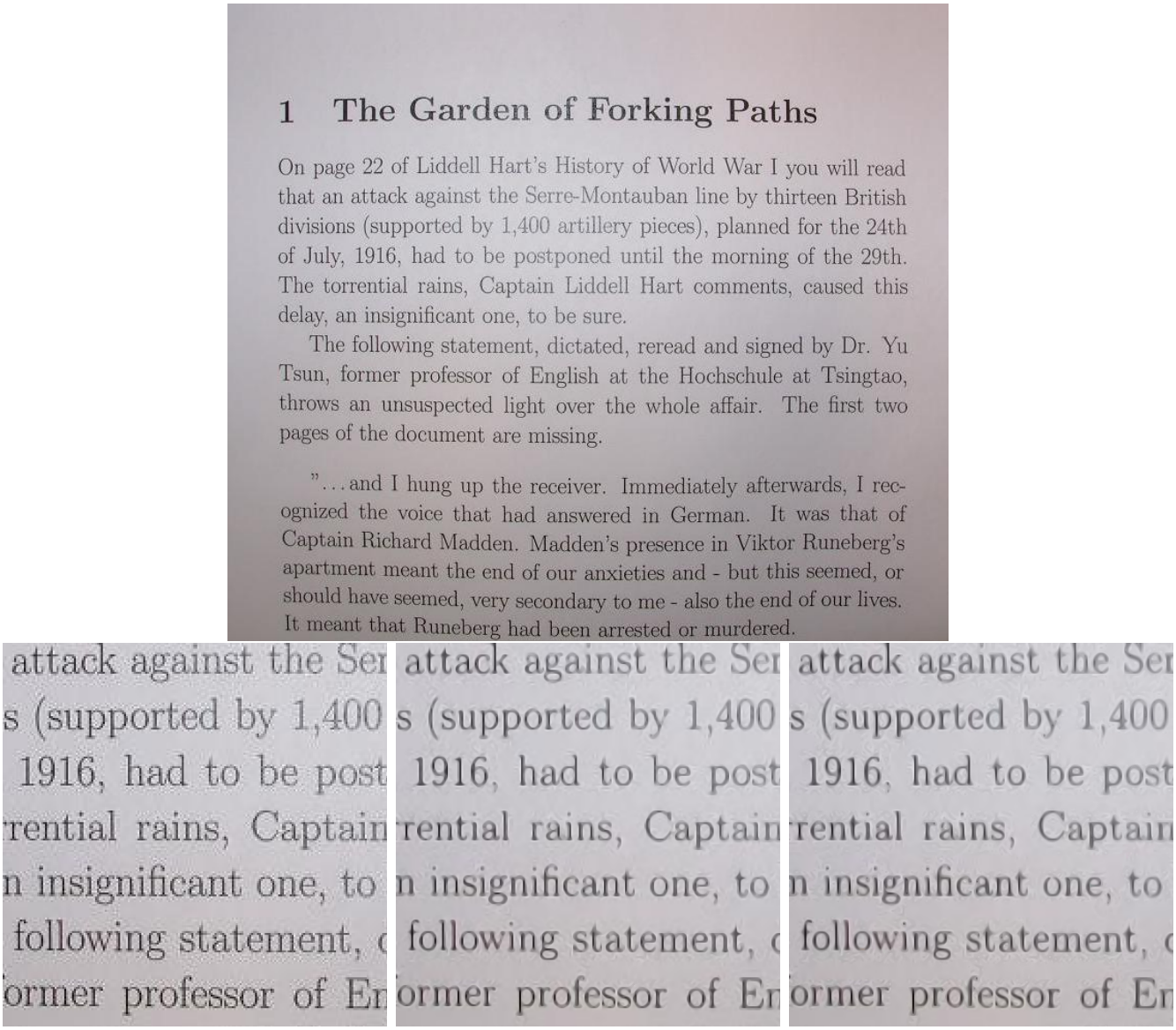

Figure 37: Top, original image. Bottom: left, 2× zoomed detail from original; center, MCM evolution at scale 1; right, AMSS evolution at scale 1.

The results are even better in the example shown in Figure 38, because the letters are spaced out enough. Nevertheless we pay the renewed smoothness of contours with a little bit of blurring.

In the example in Figure 39 the first three words can be read more easily in the evolved image, while the two **P** of **CAPPUCCINO** are too close and deform.

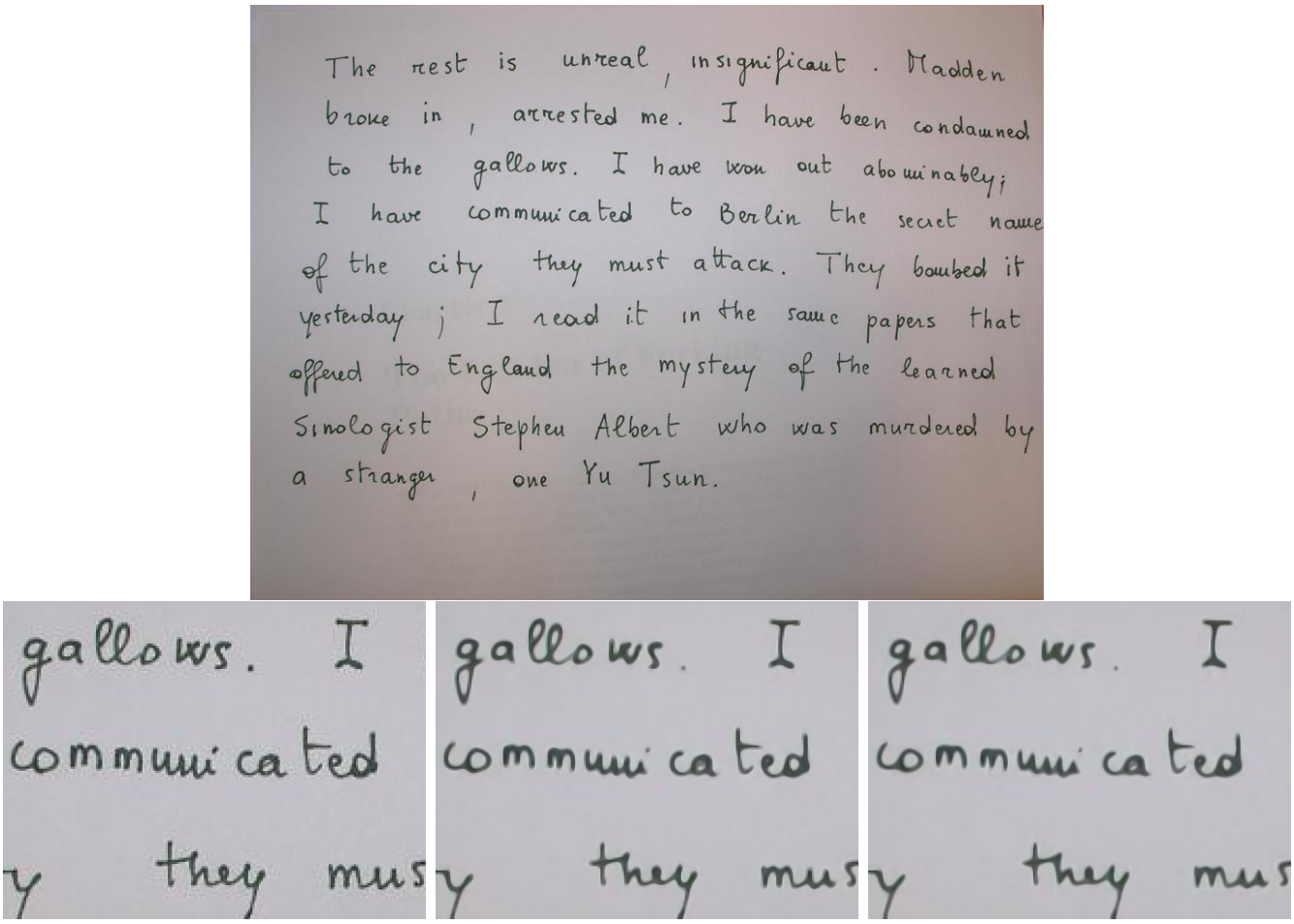

Figure 38: Top, original image. Bottom: left, 3× zoomed detail from original; center, MCM evolution at scale 4/3; right, AMSS evolution at scale 4/3.

Figure 39: Top, original image. Bottom: left, 4× zoomed detail from original; center, MCM evolution at scale 5/8; right, AMSS evolution at scale 5/8.

3.5 Numberplates

Registration numbers need often preprocessing before applying recognition algorithms. MCM/AMSS smoothing might be one of the methods, though it does not always work well.

In the example in Figure 40 the single letters do not mix up even if their borders seem to melt.

Figure 40: Top, original image. Bottom: left, 6× zoomed detail from original; center, MCM evolution at scale 2/3; right, AMSS evolution at scale 2/3.

Nevertheless some letters are more readable than others. For example the fourth typeface of the numberplate in Figure 41 is a Q. If a recognition is difficult in the first image, it is absolutely impossible to distinguish the Q from an O in the evolved pictures. In fact the short oblique stem that characterizes the letter shrinks after the application of the MCM/AMSS.

Figure 41: Top, original image. Bottom: left, 4× zoomed detail from original; center, MCM evolution at scale 3/4; right, AMSS evolution at scale 3/4.

3.6 Fingerprints

The usage of fingerprints for identification is more and more common nowadays. The application of the algorithm to the bad quality image in Figure 42 allows us to restore the original smoothness of the fingerprint ridges.

Figure 42: Top, original image. Bottom: left, 3× zoomed detail from original; center, MCM evolution at scale 4/3; right, AMSS evolution at scale 4/3.

In Figure 43 the quality of the image is a bit better, so we can try to apply a bigger zooming factor. The contours appear more regular but the diffusion is too strong, due to oscillatory level sets. The gray shades that blur the image can be eliminated applying a threshold with $\lambda = 127.5$ or adopting a **stack filter**. Both the operations will return as output a binary image.

Figure 43: Top, original image. Bottom: left, 5× zoomed detail from original; center, MCM evolution at scale 1; right, AMSS evolution at scale 1.

3.7 Lamps

In Figure 44 big particulars zoomed by a reasonable factor are easily handled by the algorithms, while, as the second example shows (Figure 45), dealing with small details in the background yields worse results. The blurring effect is considerable and the evolved image is closer to an impressionist painting than to a real photo.

Figure 44: Top, original image. Bottom: left, 3× zoomed detail from original; center, MCM evolution at scale 5/6; right, AMSS evolution at scale 5/6.

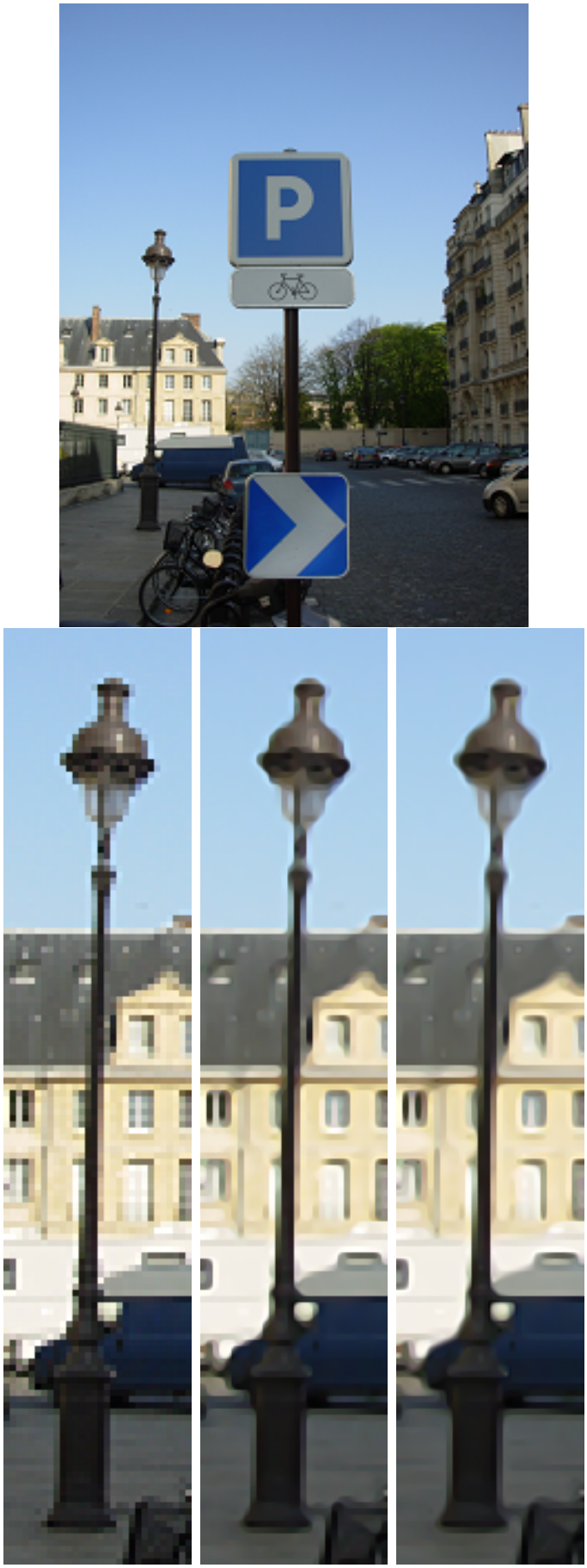

Figure 45: Top, original image. Bottom: left, 3× zoomed detail from original; center, MCM evolution at scale 2/3; right, AMSS evolution at scale 2/3.

3.8 Leafy Branches

In Figure 46, this particular of a tree, despite the small dimensions, contains a great amount of details and the calculation of curvatures is practically impossible. Diffusion prevails and the result is *impressionistic*.

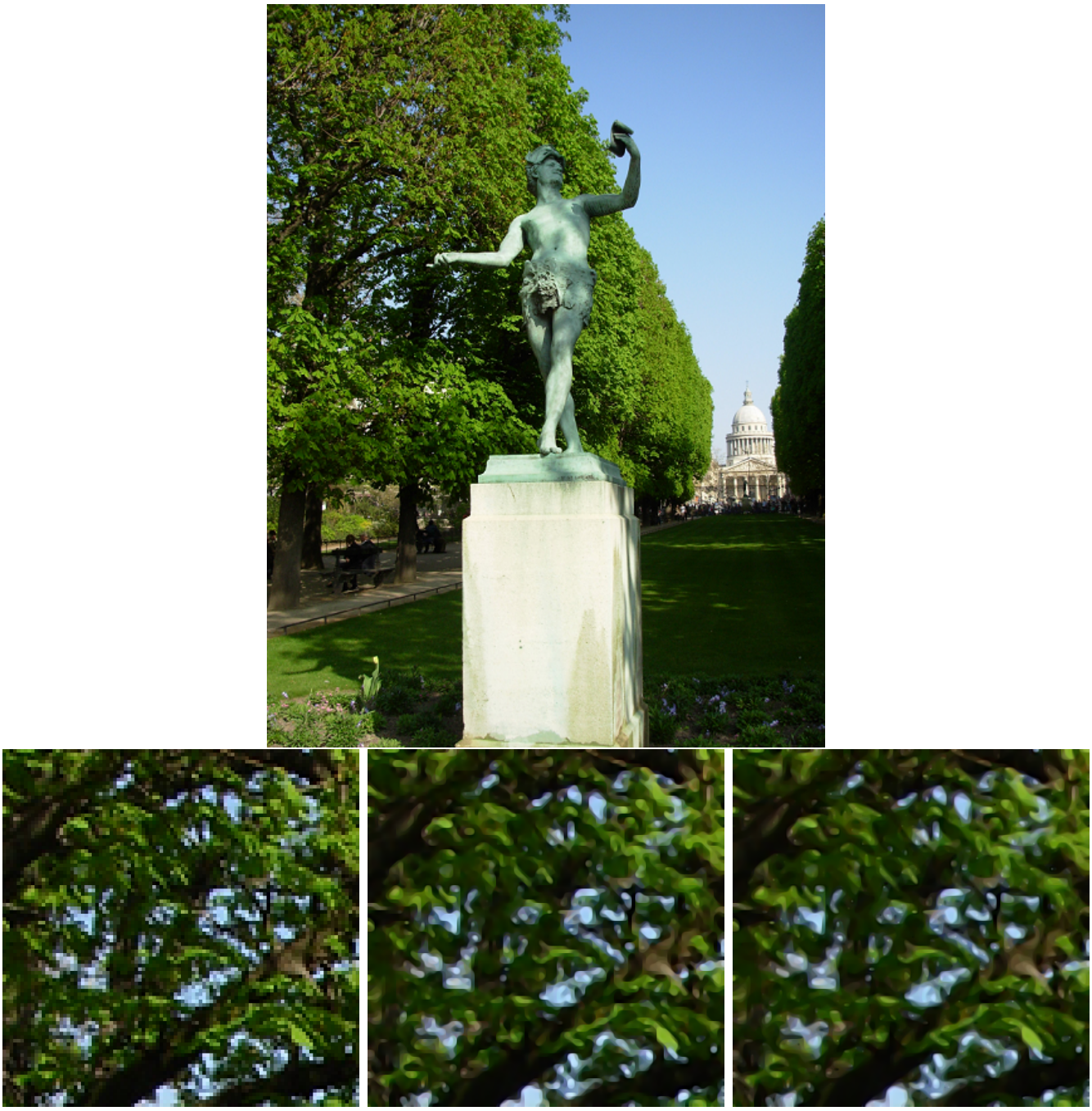

Figure 46: Top, original image. Bottom: left, 4× zoomed detail from original; center, MCM evolution at scale 3/4; right, AMSS evolution at scale 3/4.

However, as shown in Figure 47, if the zooming factor and the amount of details considered reduce, an improvement in the quality of the image is still possible for twigs and leaves.

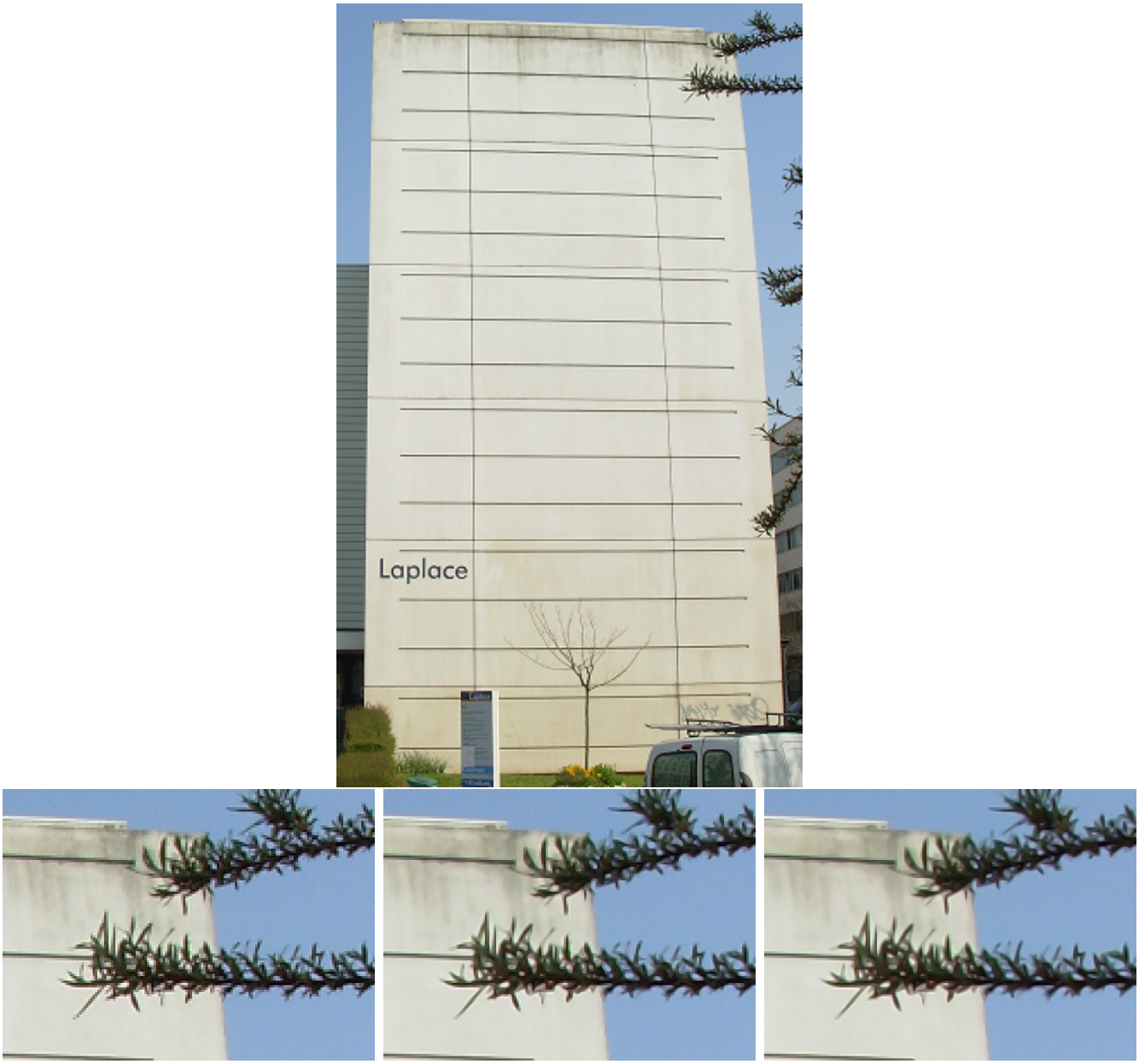

Figure 47: Top, original image. Bottom: left, 3× zoomed detail from original; center, MCM evolution at scale 2/3; right, AMSS evolution at scale 2/3.

3.9 Fences

Consider the original image in Figure 48-top and a detail obtained with different zoom factors. If with a 4× zoom we can still recognize acceptable output images, the successive 8× zoom is definitely too much for our finite difference schemes: the algorithms fail in identifying and enhancing the actual contours of the picture.

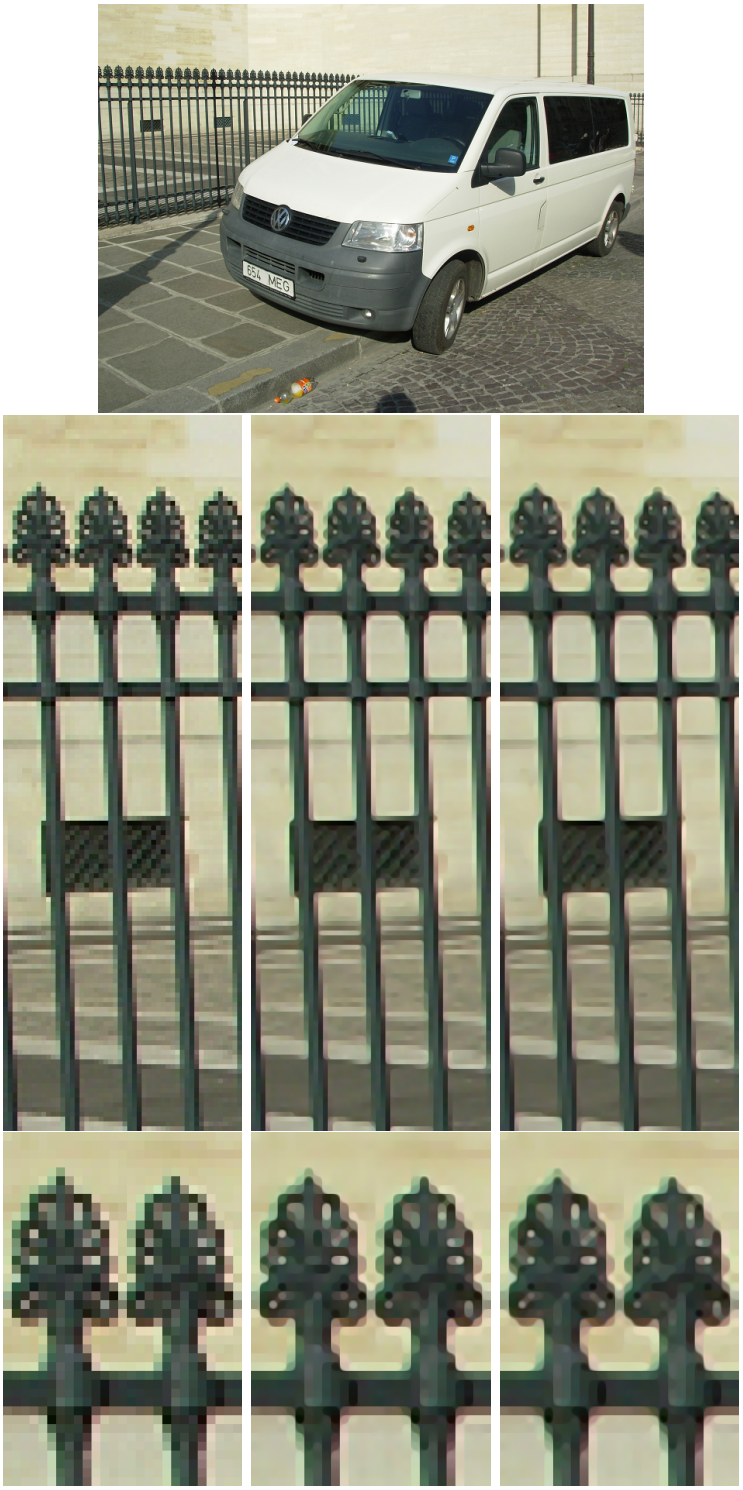

Figure 48: Top, original image. Middle row: left, 4× zoomed detail from original; center, MCM evolution at scale 1/2; right, AMSS evolution at scale 1/2. Middle row: left, 8× zoomed detail from original; center, MCM evolution at scale 3/8; right, AMSS evolution at scale 3/8.

3.10 Buildings

In Figure 48 it is shown that geometrical patterns representing contours can be restored by the MCM and the AMSS as long as the surrounding background does not have a complex structure.

Figure 49: Top, original image. Bottom: left, 6× zoomed detail from original; center, MCM evolution at scale 5/6; right, AMSS evolution at scale 5/6.

In Figure 50 we see that as the normalized scale increases the color of facades gains in homogeneity and the horizontal lines tend to shrink.

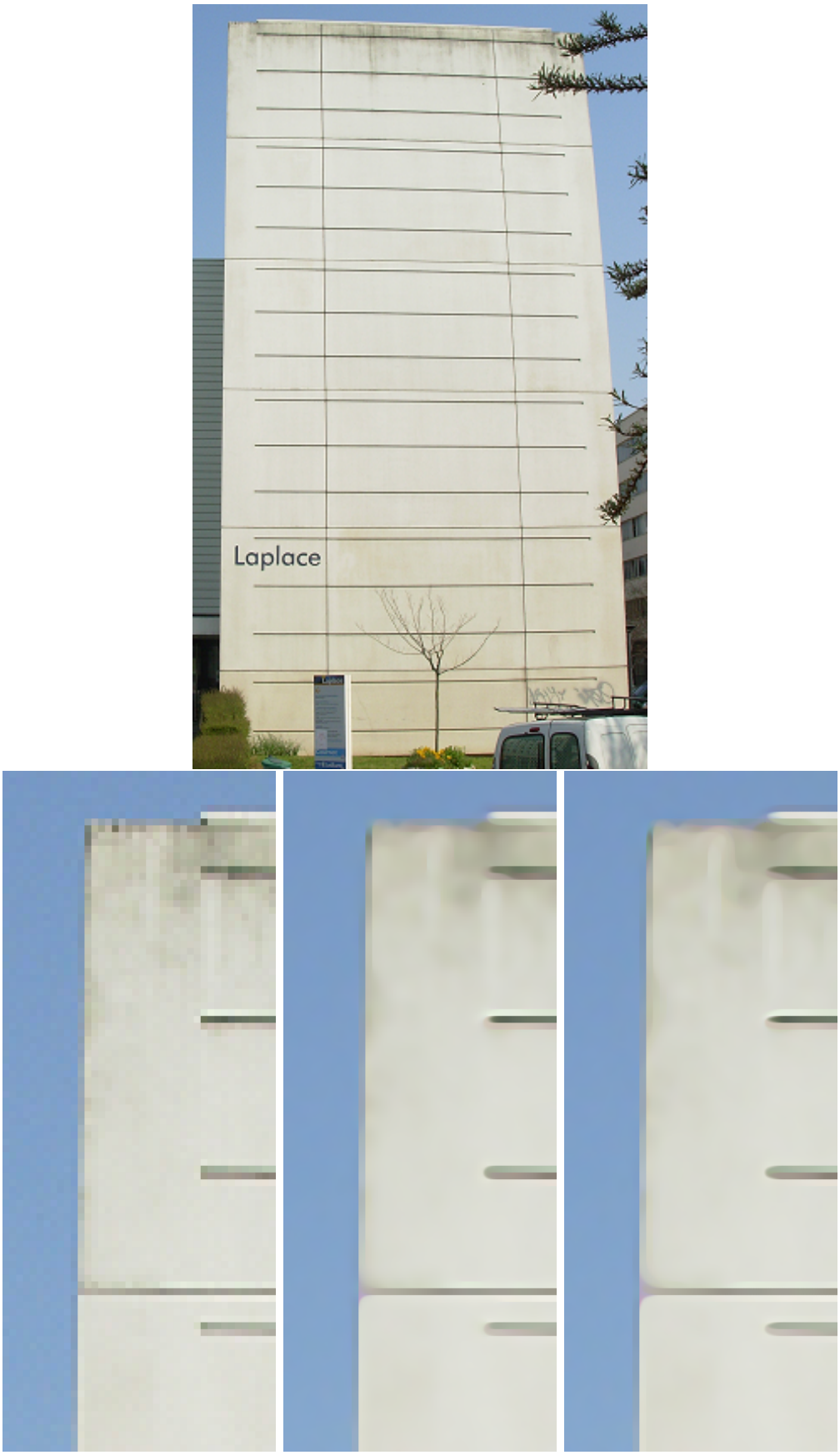

Figure 50: Top, original image. Bottom: left, 6× zoomed detail from original; center, MCM evolution at scale 1; right, AMSS evolution at scale 1.

Image Credits

All the images CC-BY Marco Mondelli.

References

[1] L. Alvarez, F. Guichard, P.-L. Lions, J.-M. Morel, "Axioms and fundamental equations of image processing", Archive for Rational Mechanics and Analysis, 123(3), pp. 199–257, 1993. http://dx.doi.org/10.1007/BF00375127

[2] L. Alvarez, J.M. Morel, "Formalization and computational aspects of image analysis", Acta Numerica, pp. 1–59, 1994. http://dx.doi.org/10.1017/S0962492900002415

[3] L. Moisan, "Modeling and Image Processing", course notes, 2005

[4] M.A. Grayson, "The heat equation shrinks embedded plane curves to round points", Journal of Differential Geometry, 26(2), pp. 285–314, 1987. http://projecteuclid.org/euclid.jdg/1214441371

[5] G. Sapiro, A. Tannenbaum, "On affine plane curve evolution", Journal of Functional Analysis, 119(1), pp. 79–120, 1994. http://dx.doi.org/10.1006/jfan.1994.1004

[6] L.C. Evans, J. Spruck, "Motion of level sets by mean curvature", Journal of Differential Geometry, 33(3), pp. 635–681, 1991. http://projecteuclid.org/euclid.jdg/1214446559

[7] Y.G. Chen, Y. Giga, S. Goto, "Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations", Proceedings of the Japan Academy, Series A, Mathematical Sciences, 65(7), pp. 207–210, 1989. http://dx.doi.org/10.3792/pjaa.65.207

[8] J. Froment, F. Guichard, L. Moisan, "Megawave code", 2001. http://megawave.cmla.ens-cachan.fr/

[9] A. Ciomaga, M. Mondelli, "On the Finite Difference Schemes for Curvature Motions", Proceedings of International Student Conference on Pure and Applied Mathematics, 2010.