Fast computation of a contrast-invariant image representation

Pascal Monasse, Frédéric Guichard

Abstract— **This article sets out a new representation of an image which is contrast independent. The image is decomposed into a tree of "shapes" based on connected components of level sets, which provides a full and non-redundant representation of the image. A fast algorithm to compute the tree, the Fast Level Lines Transform (FLLT), is explained in details. Some simple and direct applications of this representation are shown.**

Keywords— **Image representation, Image coding, Mathematical morphology, Contrast invariance.**

EDICS: 2-MRPH Morphological Processing

I. Introduction

IMAGE representations can be different depending on their purpose. For a deblurring, restoration, denoising purpose, the representations based on the Fourier transform are generally the best since they rely on the generation process of the image (Shannon theory), and/or on the frequency models of the degradation as for additive noise, or spurious convolution kernel. The wavelets theory [1], [2], achieves a localization of the frequencies. However, from the image analysis point of view, the preceding representations are not quite well adapted due to the fact that the wavelets are not translation invariant, the Fourier transform is non local and therefore very window-dependent, and both of them have quantized observation scales.

Scale-space and edge detection theories propose to represent the images by some "significant edges". The algorithms are generally in two parts (which can be merged), first the images are linearly or not smoothed [3], [4], and second an edge detector is applied on the smoothed images. The earliest scale-space based "edges" representation is the zero-crossing of the Laplacian across the Gaussian pyramid (see figure 1). According to David Marr, those zero crossings represent the "raw primal sketch" of the image, that is the information on which further vision algorithms should be based [5], [6]. Many new developments and improvements have been proposed for detecting "edges", as for example in [7].

In general, the "edges" extraction can also be formulated variationaly [8], [9]. The image is approximated by a function that stands in a class where "edges" are properly defined. (An example of simple class is the piecewise constant images having a bounded discontinuity length. With such a class, the boundaries of the approximated function are interpreted as the "edges"). Then, a balance between how close and how complex the approximation is (e.g. the complexity can be the length of the boundary), defines a scaled representation of the image. Despite the generality of the approach (somehow everything is variational), it suffers from the fact that there is no theory that says what should be the model. These representations by the "edges" suffer, according to us, from two major drawbacks that have been discussed (see [10], [11], [2]...) but not solved within the scale-space theory. First, the geometrical representation of the edges is incomplete: it does not allow a full reconstruction of the image. Second, the decomposition in scale yields a redundant representation.

Pascal Monasse is with CMLA, at the Ecole Normale Supérieure de Cachan, 61, Avenue du Président Wilson, 94235 Cachan Cedex. France. E-mail: monasse@cmla.ens-cachan.fr.

Frédéric Guichard is with Inrets/Dart. 2, Avenue du Général Malleret-Joinville, F-94114 Arcueil Cedex. France. E-mail: fguichar@ceremade.dauphine.fr and Cognitech, Inc. 225 Sth Lake Ave, Pasadena CA 91101.

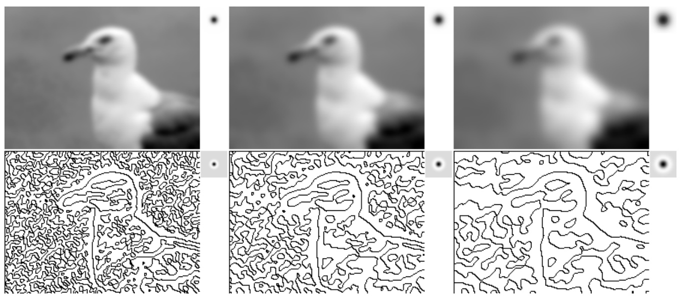

Fig. 1. Original scale space theory. To extract more global structure from an image, convolutions with Gaussian of variances which are powers of 2 are performed. One computes the Laplacian of the resulting smooth images and displays the lines along which the Laplacian changes sign: the so called "zero crossings of the Laplacian". Up: we display the results of the smoothing, and the respective kernels of scale 1, 2 then 4, from left to right. Down: we display the zero-crossing of the Laplacian, and the kernels that correspond to the Laplacian at the scales used upward.

Another problem is linked to the fact that image gray level is not an absolute data, since in many cases the contrast function is captor dependent and not known. E.g. for natural images, the contrast depends on the type of the camera, on the digitalization process (gray level quantization), brightness of the weather... Despite this instability, the perception of shape of the objects might "look" the same in many different screens, using different cameras... The invariance under change of contrast has been first stated as a Gestalt principle by Wertheimer [12]. Matheron and after him Serra proposed a "morphological" representation of images by their level sets [13], [14]. It yields a complete, contrast invariant representation of the images which does not depend on parameters. A recent variant of this representation is proposed in [15] by considering the boundary of these sets, that is the level lines or "topographic map".

In this paper, we discuss a decomposition of the images into the connected components (cc) of their level lines, where in addition the components can be structured into a tree representing their geometrical inclusions. We propose a fast algorithm to perform this decomposition. And, at last, we will see that it is well adapted to image manipulation such as image simplification, comparison...

II. From level sets to interior of level lines

We consider the following model for an image u. u is a function from a rectangle $\Omega = [0, W) \times [0, H)$ to $\mathbb{R}$ being constant on each "pixel" $[j, j+1) \times [i, i+1)$, of value in the set $0, 1, \ldots, U$ (usually, $U = 255$). It is convenient to extend the image on the plane $\mathbb{R}^2$ by setting $u = u_0$ outside Ω where u_0 is an arbitrary fixed real value. Other different models can also be considered (including the cases of finite support upper or lower semi-continuous functions).

A. Representations invariant under global contrast change

Given an image u, we call upper level set $\mathcal{X}_\lambda$ of value λ and lower level set $\mathcal{X}^\mu$ of value μ the subsets of $\mathbb{R}^2$:

$$\mathcal{X}_\lambda \quad = \quad \{x \in \mathbb{R}^2,\ u(x) \geq \lambda\} \qquad (1)$$

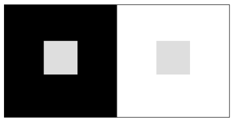

Fig. 2. Even if the two gray squares have exactly the same gray, they might appear different. It seems that our visual system is in difficulty to say if two separated points have the same intensity. However, within each square the intensity is perceived as uniform.

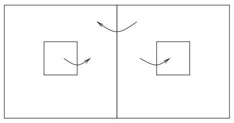

Fig. 3. Level lines description of the image of figure 2. (The arrows represent the comparisons "brighter than" that are stored but not represented in the image.) Note that this representation does not allow to compare the gray level of the small squares.

$$\mathcal{X}^\mu \quad = \quad \{x \in I\!R^2,\ u(x) \leq \mu\}. \tag{2}$$

The data of the family of the $\mathcal{X}_\lambda$ (or of the family of the $\mathcal{X}^\mu$), is sufficient to reconstruct the image [13], [16], [17]:

$$u(x) = \sup\ \{\lambda\ /\ x \in \mathcal{X}_\lambda\} = \inf\ \{\mu\ /\ x \in \mathcal{X}^\mu\}. \tag{3}$$

A second property is their global invariance by contrast change. We say that two functions u and v have globally the same level sets if for every λ there is μ such that $\mathcal{X}_\mu v = \mathcal{X}_\lambda u$, and conversely. If we apply to u a contrast change, that is, an increasing function g, then $v = g(u)$ and u have globally the same level sets. Conversely, assume that two functions u and v have globally the same level sets, then v differs from u only by a contrast change ($\exists g$, such that $v = g(u)$). Therefore, the set of the level sets is a contrast independent representation of the image. For example, the level set description is often used to design contrast invariant filters.

Note that the level sets are nested; the family of upper (resp. lower) level sets is decreasing (resp. increasing):

$$\forall \lambda \leq \mu,\ \mathcal{X}_\lambda \supset \mathcal{X}_\mu,\ \mathcal{X}^\lambda \subset \mathcal{X}^\mu. \tag{4}$$

B. Representations invariant under local change of contrast

We believe that the representation of an image by its set of level sets suffers from the same problem as the frequencies based representation: the basic "atoms" (here level sets) are global within the image. Regardless of their distance, the fact that two points have the same gray level is strongly coded in that representation. Such points are in the same "atom".

Now, it is not clear if our visual system can in general say whether or not two separated pixels have the same gray level (see figure 2). However, such a comparison seems to be performed successfully locally: each of the small squares of figure 2 appears to have an uniform intensity, in the sense that it is perceptually impossible to separate them into smaller pieces.

So, leaving now analogy with perception, our model is:

We assume our sensor is such that each pixel knows only if it is brighter, equal, or darker than its neighbor pixels, and that these comparisons can be propagated. (5)

What is then the remaining information left? The answer is roughly in the lines of the figure 3. These lines are the boundaries of the level sets of the image, that is the *level lines*. In addition to these lines, is left for each line whether or not the interior is brighter than the exterior.

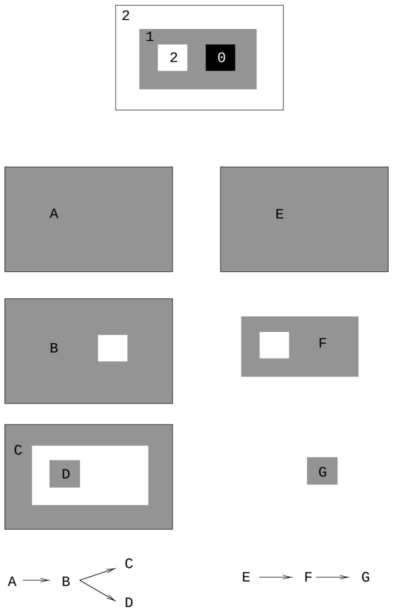

Fig. 4. The trees of connected components of upper and lower level sets of a simple image.

This representation does not differ too much with respect to the set of level sets, since we keep their boundaries. But now, all connected components of the level sets are de-coupled. A local and contrast invariant representation of the images is the set of the connected components of all the level sets (the locality is then driven by the connectedness). The idea of defining connected components of gray-level based regions in the image is proposed in [18], [19], [20] (see also in [20] the references of chapter 8) in order to define adaptive filtering and in [21] where it is explicitly aimed at stereo matching. However, the regions considered in these papers are not contrast invariant.

C. The inclusion tree

The relation (4) states that the level sets are nested. When going from the whole level sets to their connected components, these relations are of course still true. Now, a connected component can contain several connected components. These inclusions can be represented into a tree, as shown in figure 4. As we can see, the cc's of upper and lower level sets trees differ. In addition, we see that we end up with a non natural description of the inclusion. Naturally, in the example of figure 4, one would have expected to have the two small squares included into the gray rectangle, and included into the white background. But the inclusion is for these trees mostly driven by the gray level rather than by the geometrical inclusion. At last, we see that we need both trees if we want to have the two small squares represented, since each of them appears in one description, and not in the other.

The model (5) leads us to consider the level lines instead of the upper or lower level sets. This will give us one single inclusion tree describing the image, in which a white object on black background is represented in the same manner as a black object on a white background.

Consider a bounded connected component of the level set (upper or lower) $\mathcal{C}$ and its border $\partial\mathcal{C}$. This border is an union of Jordan curves.

$$\partial\mathcal{C} = \bigcup_i J_i(\mathcal{C}) \tag{6}$$

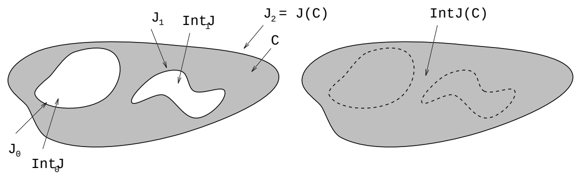

Fig. 5. An example of a connected component of a level set C not simply connected. Its border is composed of three closed Jordan curves, J_0, J_1 and J_2. The exterior Jordan curve is J_2. Int J_0 and Int J_1 are shown, so as Int $J_2 = C \cup$ Int $J_0 \cup$ Int J_1 at the right.

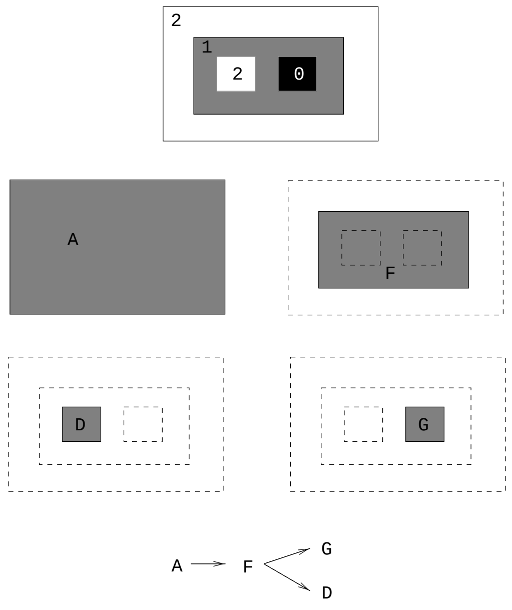

Fig. 6. The tree given by the FLLT corresponding to a simple image. Notice that D is a hole in F.

We know, thanks to Jordan theorem [22], that each closed Jordan curve J has an interior and an exterior (that is the complementary of J has two open connected components, one of them bounded and the other not; the interior of J, Int J, is by definition the bounded one). If $\mathcal{C}$ is furthermore simply connected, its border is composed of only one closed Jordan curve. Though there can be "holes" in the connected components, we know that there is only one of the $J_i(\mathcal{C})$ such that $\mathcal{C} \subset$ Int $J_i(\mathcal{C})$. We will denote this Jordan curve $J(\mathcal{C})$ (see figure 5). In the following, for any connected component of a level set $\mathcal{C}$, we will call **"shape" the interior of** $J(\mathcal{C})$. The shape corresponds to the connected component and its "holes".

The sorting of the shapes can then be made thanks to their geometrical inclusions. We can then create a tree structure as follows: each node corresponds to a shape; descendants are the shapes included into it, and the parent is the smallest shape that contains it (see figure 6).

D. A scale-space representation

All classical representations we have seen have a scale-space structure. That is, a structure that allows a separation between large "size" and small "size" behaviors. For example, for the frequency domain, the wavelength is the scale, in the wavelets representation it is the dyadic scale reduction...

The inclusion tree induces also a scale-space structure of the image. Indeed, due to inclusion, the shapes of the tree are sorted with respect to their sizes. A shape obviously contains only objects that have a smaller size. Therefore the scale is here directly the size of the shapes in term of number of pixels.

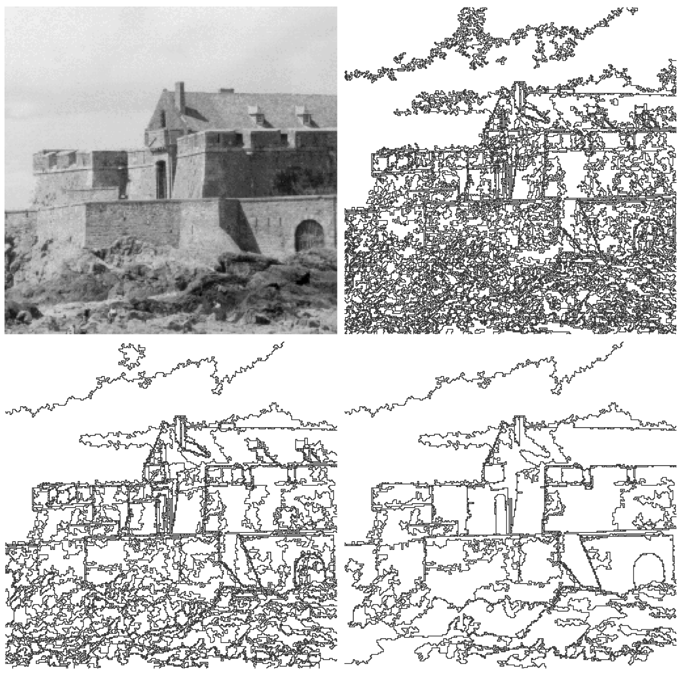

Fig. 7. We illustrate here the scale-space relation of the inclusion tree. Up-Left: original picture 256x256, and then Up-right, and Down, the boundaries of the shapes having an area larger than 10, 40, 800 pixels.

Large scale objects will be kept near the root of the tree, whereas small scale objects will be near the leaves. Figure 7 represents the level lines at scale 10, 40 and 800 pixels.

III. Principles of the Fast Level Lines Transform

Let us introduce the notations: from an image and its upper and lower level sets ($\mathcal{X}_k$ and $\mathcal{X}^l$), we note $cc(\mathcal{X}_k, x)$ the connected component of $\mathcal{X}_k$ containing a point x. The boundary of $cc(\mathcal{X}_k, x)$ will be called $J(\mathcal{X}_k, x)$ and the corresponding shape (its interior) Int $J(\mathcal{X}_k, x)$. Note that Int $J(\mathcal{X}_k, x)$ is just a set of pixels and the notation involving k and x is used only to distinguish the different shapes. Indeed there is only one shape containing x and based on the level set $\mathcal{X}_k$. The family of the shapes associated to an image will be noted $\mathcal{T}$.

A. A fast algorithm

The Fast Level Lines Transform (in short FLLT) is a decomposition of an image into shapes, together with an inherent structure of tree organizing them, which is *complete* information on the image, in the sense that it is non-redundant and sufficient information to reconstruct the image.

Of course, to extract a connected component of a level set $\mathcal{X}_k$, we could threshold the image at the gray level k and extract the components of the binary image we obtain. But there is a much smarter way to get it: the FLLT is a fast algorithm because it takes advantage of the tree structure of the interiors of level lines. It is a pyramidal algorithm, based on a region growing principle. Indeed, each connected component of $\mathcal{X}_k$ is made with connected components of $\mathcal{X}_{k+1}$ and connected components of the isolevel set $u^{-1}(k)$, so that starting from one component of $\mathcal{X}_{k+1}$, we let the region grow to pixels of value k (new pixels). If this region is in contact with other components of $\mathcal{X}_{k+1}$, we include them in the region and resume the growing. We get a component of $\mathcal{X}_k$ when the region cannot grow any more, that is all neighbor pixels of the region have a gray value less than k.

As was explained above, a component of level set can have holes within. A shape is constructed from it by "filling" the holes, which are components of other level sets.

B. The output of the FLLT

As output of the FLLT, we get

- The family of shapes ($\mathcal{T}$) ordered in a tree structure (so that we know what shape is contained in another).
- Within this tree structure, we store for each node (an element of $\mathcal{T}$) whether or not it is brighter than its parent.
- For each pixel the smallest shape of $\mathcal{T}$ containing it.

This is enough to reconstruct the image, of course up to a local contrast change. Indeed, given such information, we can define an image v as follows: We choose an arbitrary gray level for the root, and then we attribute to each node the gray level of its parent plus 1 (resp. minus 1) if it is brighter (resp. darker) than its parent. Finally, we attribute to each pixel in the image the gray level of the smallest shape containing it. We get an image that differs from the original one by a local change of contrast, as defined in [15], [23].

The minimal necessary and sufficient data structure for a shape is its position in the tree (a pointer to its parent, to one child and to the next child of its father; the children of a given shape are thus chained, this is a convenient data structure for a tree where the number of children is not known in advance) and a boolean indicating if it is brighter than its parent.

In the experiments presented in this article, we stored in addition the original gray level of the shape, so that we can reconstruct an image with comparable contrast for visual convenience. But this information has been used only for final display.

C. Sketch of the algorithm

The steps of the algorithm are the following:

1. Build the tree of connected components of lower level sets and the tree of connected components of upper level sets, taking into account the holes in each connected component.
2. Find for each hole in a connected component the connected component in the other tree corresponding to it.
3. Merge both trees, putting connected components corresponding to holes as descendants of the ones containing them.

IV. Details of the algorithm

A. Building the two trees

It is clear that each bounded connected component of level set of the image contains a local regional extremum (see [24] for definition), namely a local maximum for upper level sets and a local minimum for lower level sets. Thus the algorithm is as follows for connected components of lower level sets:

1. Scan the image, pixel by pixel, until you find a not tagged local minimum, x_0, let g be its gray-level.
2. Create a new region (a set of pixels), $\mathcal{R}$, initially void, a set of pixels to add, $\mathcal{A}$, initialized with x_0, a set of neighbor pixels, $\mathcal{N}$, with currently no pixel.

$$\mathcal{R} \leftarrow \emptyset \quad \mathcal{A} \leftarrow \{x_0\} \quad \mathcal{N} \leftarrow \emptyset.$$

3. Examine all neighbors of $\mathcal{A}$ not in $\mathcal{R}$, put them in the set of neighbors $\mathcal{N}$: $\mathcal{N} \leftarrow \mathcal{N} \cup (\{x \text{ neighbor of a pixel in } \mathcal{A}\} \backslash \mathcal{R})$, and let

$$g_{\mathcal{N}} \leftarrow \min_{x \in \mathcal{N}} u(x). \qquad (7)$$

Add the pixels of $\mathcal{A}$ to $\mathcal{R}$, tag the pixels of $\mathcal{A}$ and update the number of connected components of the border of $\mathcal{R}$.

$$\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{A}.$$

4. Three cases are dealt with differently.

(a) If $g < g_{\mathcal{N}}$, create a new connected component of lower level set. If the number of connected components of the border is more than 1, follow each border to find which is the exterior border and which are the holes. The exterior border is the one that contains the leftmost pixel. Keep one pixel belonging to each hole found.
Recompute $g_{\mathcal{N}}$ according to equation (7) and set $g \leftarrow g_{\mathcal{N}}$. Update the sets,

$$\mathcal{A} \leftarrow \{x \in \mathcal{N} \;/\; u(x) = g\} \; \mathcal{N} \leftarrow \mathcal{N} \backslash \{x \in \mathcal{N} \;/\; u(x) = g\}, \quad (8)$$

and return to step 3.

(b) If $g = g_{\mathcal{N}}$, the connected component is not complete. Update $\mathcal{A}$ and $\mathcal{N}$ according to (8) and return to step 3.

(c) If $g > g_{\mathcal{N}}$, set the gray-level of the pixels of $\mathcal{R}$ to g, and go to step 1.

In the algorithm, the neighboring pixel are defined by the 4 or 8 connectedness (the choice is discussed below).

The set $\mathcal{N}$ is an array of pixels sorted by their gray-level, so that it is easy to extract the pixels of given gray-level. The way we deal with holes is explained later.

B. Finding the shapes of the holes

At the preceding step, we know for each shape if it has holes, and, if it has, one pixel belonging to each hole. We have to find the shape of this hole. This can be easily done. Consider a point y belonging to one hole of $\text{cc}(\mathcal{X}_k, x)$. Then the hole corresponds to a shape of the type $\text{cc}(\mathcal{X}^l, y)$ with $l < k$, since the hole is a connected component of the complementary and it contains y. In order to find the shape corresponding to the hole, it suffices to take the smallest shape containing y in the tree of lower level sets, and to go up the tree while the current shape has a gray level less than k. For each y characterizing a hole in the shape $\text{cc}(\mathcal{X}_k, x)$,

- Set $S \leftarrow \text{cc}(\mathcal{X}^{u(y)}, y)$.
- If Gray-level(Parent(S)) $< \lambda$, $S \leftarrow$ Parent(S), else exit.

The shape of the hole is S at the end of this algorithm. Notice that this shape can itself have another hole inside, etc.

C. Merging the trees

The last step consists in merging both trees. In fact, if no shape has a hole in it, there is nothing to do, we simply put an universal ancestor (the root of the tree) corresponding to the whole image $\mathcal{X}_{-\infty} = \mathcal{X}^{+\infty}$ and put as its children all the shapes of both trees having no parent.

Now, if there are holes, it means that shapes from one tree must be moved to the other tree. Consider the following property:

If $\text{cc}(\mathcal{X}_k, x)$ has a hole $\text{cc}(\mathcal{X}^l, y)$, then we have the alternative:

- either one of the children of $\text{cc}(\mathcal{X}_k, x)$ has also a hole, which contains $\text{cc}(\mathcal{X}^l, y)$;
- or $\text{cc}(\mathcal{X}^l, y)$ has no parent (except the universal ancestor).

In order to merge the trees, we only have to find for each shape S with a hole H in it whether one of its children S' has a hole H' containing H. If it is the case, we do nothing. Otherwise, we put the shape of the hole H (and all its descendants) as child of the shape S.

D. Concluding the algorithm

Let us now explain how to extract the smallest shape containing a given pixel x. We have four situations.

- If $cc(\mathcal{X}_{u(x)}, x)$ and $cc(\mathcal{X}^{u(x)}, x)$ are not bounded, then there is no shape containing x.
- If $cc(\mathcal{X}_{u(x)}, x)$ is bounded and $cc(\mathcal{X}^{u(x)}, x)$ not, the shape is $cc(\mathcal{X}_{u(x)}, x)$.
- If $cc(\mathcal{X}_{u(x)}, x)$ is not bounded and $cc(\mathcal{X}^{u(x)}, x)$ is, the shape is $cc(\mathcal{X}^{u(x)}, x)$.
- If both $cc(\mathcal{X}_{u(x)}, x)$ and $cc(\mathcal{X}^{u(x)}, x)$ are bounded, then

$$\begin{aligned} \text{Int } J(\mathcal{X}_{u(x)}, x) \quad &\subset \quad \text{Int } J(\mathcal{X}^{u(x)}, x) \text{ or} \\ \text{Int } J(\mathcal{X}_{u(x)}, x) \quad &\supset \quad \text{Int } J(\mathcal{X}^{u(x)}, x) \end{aligned}$$

and then the smallest shape is the contained one.

E. Complexity of the algorithm

We scan each pixel once since it is included in exactly one smallest shape. To make a component of level set grow, we have to compare the gray levels of all the neighboring pixels. But how many pixels are neighbors of a component of level set? Each pixel has at most 8 neighbors (4 in 4-connectedness), so that it is a neighbor of at most 8 shapes, and there are $O(N)$ pixels (N is the number of pixels) that are neighbors of a component of level set. Each time, it is important to have the gray levels of the neighbors sorted, done in $O(N)$ if the image is quantized, by using queues, as in [25] and in $O(N \log N)$ otherwise. We need also to follow the boundaries of the shapes when they have holes, which can be done in $O(N)$.

Compare with the complexity of a direct implementation: for each gray level, the image is thresholded, and the connected components of the black pixels are extracted. The extraction of the components of a binary image takes linear time, $O(N)$. This is to be multiplied by the number of gray levels in the image (usually 256). The complexity if of order $O(N)$, as the FLLT. But now, if the image is not quantized, the number of different gray levels in the image can be N. So we get a complexity of order $NO(N) = O(N^2)$, compared to $O(N \log N)$ for the FLLT.

We implemented both algorithms on a Pentium 200 MMX, compiled with gcc under Linux operating system. With an 8-bit image of size 750×600 ($N = 450,000$ pixels), we got CPU times of 6.5 s for the FLLT, 88.3 s for the method by successive thresholds, and if the image was previously quantized by a factor 8, 5.1 s for the FLLT and 12.1 s for the direct method.

V. Technical parts of the algorithm

A. Which connectedness to choose?

In discrete images we have two notions of connectedness, 4- and 8-connectedness, according to the number of neighbors of the pixels. Which notion of discrete connectedness to use? The answer is: use 4-connectedness for lower level set and 8-connectedness for upper level set, or the inverse, but do not take the same connectedness for lower and upper level sets.

If we used 8-connectedness for both, Jordan theorem would not be verified, and we could get intersecting level lines (see figure 8). Neither can we use 4-connectedness for both (see figure 9): in this case, there is no reconstruction for the image.

Another possibility is to consider an hexagonal lattice. Shift one other two lines by half a pixel, so that a pixel has 6 neighbors and there is only one connectedness, the 6-connectedness. There is no dissymmetry between upper and lower level sets, but the choice of the direction of the shift was arbitrary.

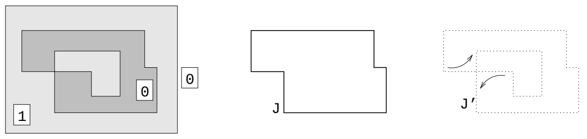

Fig. 8. Jordan theorem is not verified if we consider 8-connectedness for both lower and upper level sets. Here, $\mathcal{X}^0$ has one bounded connected component C. The exterior border of this set is J, whereas the interior border of the level set $\mathcal{X}_1$ is J'. The complementary of J has two connected components, but the exterior of J', which is included in the complementary, intersects both.

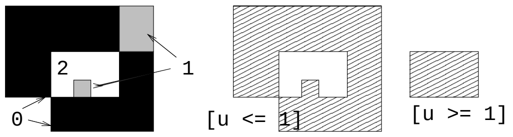

Fig. 9. If we consider 4-connectedness for both lower and upper level sets, we do not have a reconstruction. Left: original image. Middle and right: two shapes extracted from the image. The two shapes have one pixel of intersection, whereas none is included in the other.

B. Dealing with holes

B.1 The detection of holes

Remember that the construction of the connected components of level sets is done by region-growing. We start with one pixel, then add its neighbors if possible, then the neighbors of the neighbors, etc. The number of holes is the number of connected components of the *border* minus 1 (because of the exterior border). The idea here is that by adding a point to a connected set, we can update the number of connected components of the border using only the local configuration.

The border of a region is encoded by inter-pixel directions: each time two 4-neighbors belong one to the region and the other not, we put an inter-pixel direction between both, oriented so that it leaves the pixel in the region to its right by convention (see figure 10). The modification of the border when we add a pixel to a region is straightforward.

The goal was to find the change of the number of connected components of the border when we add one pixel to an existing

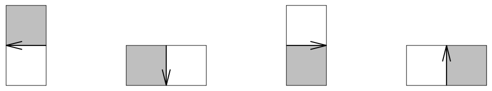

Fig. 10. The usual way the local configuration of the border of a region is coded. In each figure, the pixel inside the region is the dark one. The border is indicated by inter-pixel directions, always leaving the region to the right.

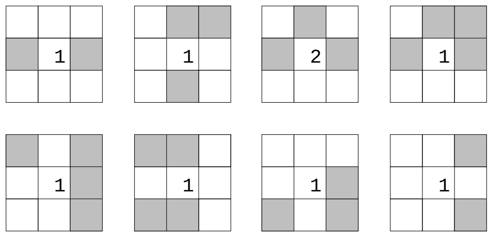

Fig. 11. The increment of the number of connected components of the border when we add a point to a connected region according to its neighborhood. Up: for a region in 4-connectedness (complementary in 8-connectedness). Down: for a region in 8-connectedness (complementary in 4-connectedness). Dark pixels: inside the region. White pixels: outside the region.

(connected) region. It is enough to know the configuration of the region in the 3×3 neighborhood of the pixel. The increment of the number of connected components of the border when we add one pixel is shown for a few configurations in figure 11.

B.2 The determination of the holes

Knowing the number of connected components of the boundary of a shape, the number of holes is simply this number minus 1. If there are holes, we must find one pixel in each. For that, we follow each connected component of the (oriented) border.

VI. Experiments

A. Simplifications of image

A first kind of applications is based on manipulation of the tree itself. Figure 12 represents an image at different scales 0, 10, 40, 800 pixels. At scale 800, all shapes having an area less than 800 pixels have been removed, which is a drastic simplification of the image! Note that this look-like segmentation is not based on the contrast. This scale dependent representation is an implementation of a filtering proposed by Masnou in [26].

This simplification of the image is also close to the one proposed by Vincent, in [24]. Indeed, he proposed to eliminate small connected components successively of the upper level sets, and then of the lower level sets. But, with respect to the scaled inclusion tree, first this method introduces a dissymmetry between black and white objects. And, second the area criterion to remove shapes does not count the area of some of its holes depending of the respective gray level (see figure 13).

If $\mathcal{B}$ is the set of connected sets containing the origin O, of area larger than a, and the operators $SI_{\mathcal{B}}$ and $IS_{\mathcal{B}}$ are

$$SI_{\mathcal{B}}\, u(x) = \sup_{B\in\mathcal{B}} \inf_{y\in x+B} u(y) \tag{9}$$

$$IS_{\mathcal{B}}\, u(x) = \inf_{B\in\mathcal{B}} \sup_{y\in x+B} u(y), \tag{10}$$

they do not commute: $SI_{\mathcal{B}} \circ IS_{\mathcal{B}} \neq IS_{\mathcal{B}} \circ SI_{\mathcal{B}}$. We can see that Vincent defines in [24] two slightly different filters.

For a connected set B, let us call its "filled interior" B plus its holes (that is, the smallest simply connected set containing B) and its interior area the area of its "filled interior". The variant proposed by Masnou defines instead $\mathcal{B}'$ as the set of connected sets of points whose "filled interior" contains the origin 0 and of *interior* area larger or equal to a. The operators $SI_{\mathcal{B}'}$ and $IS_{\mathcal{B}'}$ are defined as in (9) and (10), and this time the operators do commute: $SI_{\mathcal{B}'} \circ IS_{\mathcal{B}'} = IS_{\mathcal{B}'} \circ SI_{\mathcal{B}'}$, which avoids to introduce a dissymmetry between upper and lower level sets.

One can also select the shapes based on other criteria than their size. We can for example remove the irregular shapes according to a measure of the complexity of their boundaries, see figure 14. This allows some selective removals of the shapes based on their probability to have been generated by physical objects, or by the captor device (noise, dithering, etc).

If one is interested in a representation of the image that has a very low number of shapes (for compression purposes, to reduce the computation cost of the following manipulations, etc), one can also remove the shapes that have an area close to their parent and to one of their children, because the corresponding level lines must be nearly the same. Figure 15 shows this experiment applied to the image of figure 14. Notice that such a simplification, very easy with the inclusion tree, would have been intricate without.

B. Comparison of images

The inclusion tree allows also a fast comparison between images, in a manner independent from the contrast (see figure 16).

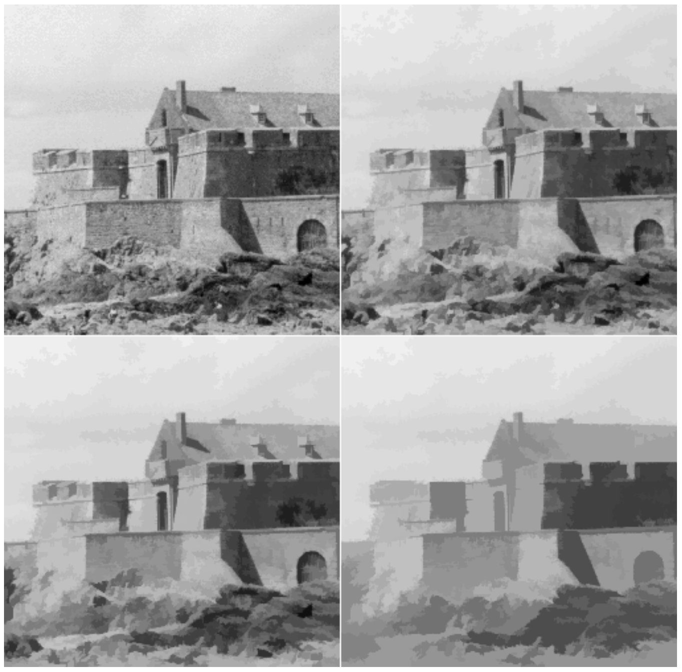
Fig. 12. Up-left: original image 256x256, Up-right: image at scale 10 (all shapes having area less than 10 pixels are removed), Down: image at scale 40, and 800 (sum of the CPU time for the 3 processed image: 0.38 s on a P200MMX).

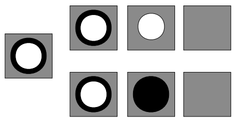
Fig. 13. Left: a simple image. Up: Successive removals of the connected components of the upper and then lower level sets with an increasing area threshold. The black ring disappears before the white circle, since the (lower) level set in which it is embedded has a smaller area than the (upper) one of the white circle. Down: representations of the image across the scales of the inclusion tree: the circles disappear according to their interior size.

As in [27], we associate to each shape some basic characteristics, as its size, position, inertia... They are then used to compare shapes. A shape of an image 1, will be said to match in the image 2 if there exists a shape of the image 2 that has approximatively the same parameters at the same location [28]. Notice that using the pyramidality is very advantageous to compute the characteristics. Since the moments are additive characteristics, when constructing a shape, we have already computed the moments of its descendants, so it suffices to add them to the moments associated to the new pixels. In this way, we count exactly one time each pixel, which would not be the case if the shapes were extracted by successive thresholds.

Figure 16 is made as follows: we reconstruct the image from the subset of the shapes in left image that have no matching shape in the right image. A comparison with a simple pixel by pixel difference shows the importance of being contrast invariant if we want to identify images of the same scene. Notice also that this reconstruction is not symmetrical, which allows to identify the shapes from one image that are not present in the other one, which is impossible with a pixel by pixel difference.

Following the technique described in [27], one can also reconstruct an image from the set of the shapes that have matched. Given two images u_1 and u_2, we define $u_1 \cap u_2$ as the set of the

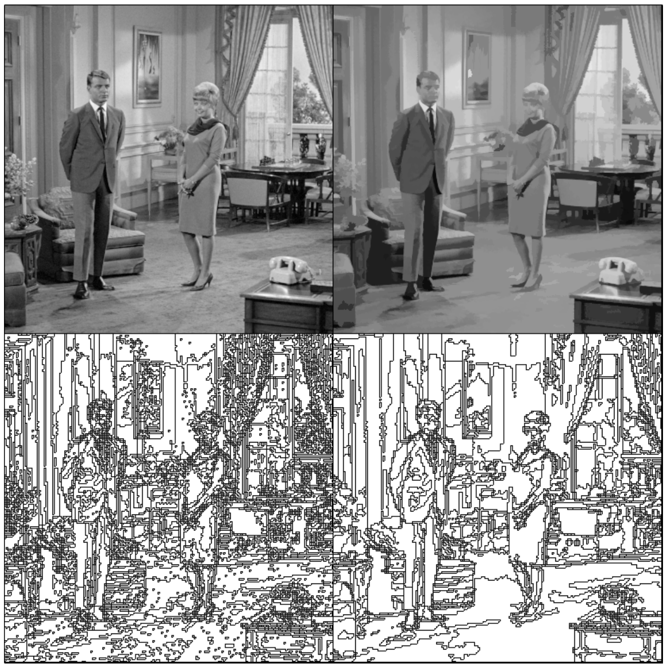

Fig. 14. Left: original image (512×512), composed of 89379 shapes. Right: simplified image, where level sets of area too small (< 20 pixels) or of too irregular boundary are removed from the tree. There are $11,371$ (13%) shapes of sufficient area and there remains only 4174 (5%) sufficiently regular shapes. Notice that nothing important seems to be lost. Bottom: level lines of the images.

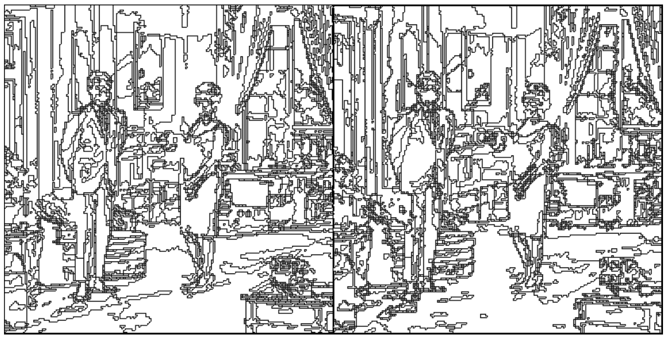

Fig. 15. Left: The 4174 regular level lines of the simplified image of the last figure. Right: The remaining 1753 regular level lines after removal of the shapes that have an area similar to their parent and to one of their children. Notice how close the images look, in spite of the high number of eliminated shapes.

shapes of u_1 that match in u_2. That is, from the inclusion tree of u_1: $\mathcal{T}_1$, we remove all the shapes for which we cannot find a shape in the inclusion tree of u_2 that is similar. This yields a tree $\mathcal{T}_1'$, from which we can reconstruct an image: $u_1 \cap u_2$.

This definition of the intersection differs slightly from the ones defined in [27]. Indeed, the authors define the "shapes" as the connected components of the bi-level sets (that is, connected part of the pixels that have a gray level between two values λ and μ, $\lambda \leq \mu$). By this, they obtain for $u_1 \cap u_2$ a multi-valued image, where each point can take the values between a lower and an upper intersection image. We believe that this is not fully satisfactory since first, the so defined intersection is multi-valued, second, the complexity is high due to the two thresholdings that define the bi-level sets: for 8-bit images, this yields $32,640$ cuts of the image! Their lower (resp. upper) intersections are somehow related to a lower (resp. a upper)

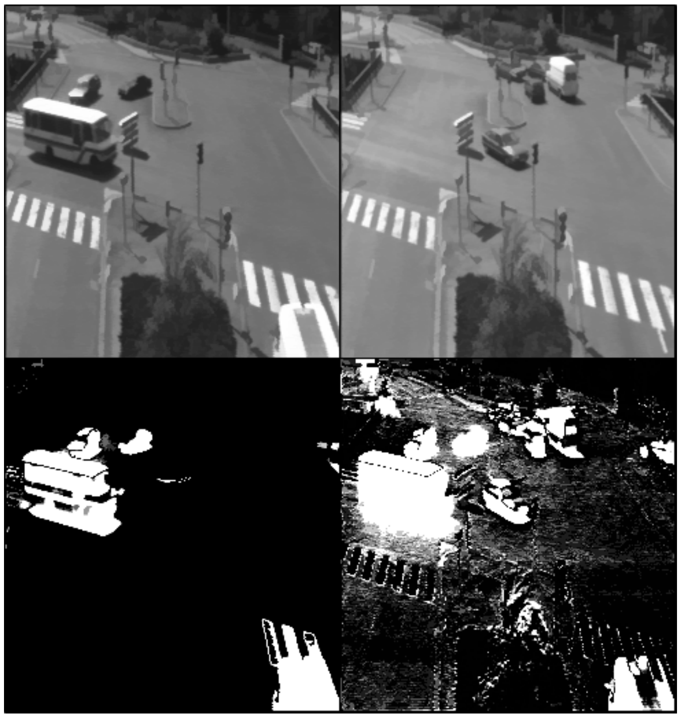

Fig. 16. Comparisons of images. Up: two original images. Down: The quantized image (left) reconstructed from the shapes of image 1 having no similar shape in image 2 and a quantized pixel by pixel difference of the images (right).

intersection of images obtained by considering the connected components of the lower (resp. upper) level sets that match. Our definition of the intersection based on the inclusion tree yields one single image!

Note also that $u_2 \cap u_1$ differs from $u_1 \cap u_2$. $u_1 \cap u_2$ corresponds roughly to image u_1 minus the objects not present in u_2. Of course, the remaining objects and their positions did not change. Figure 17 displays the intersections between two images (in the middle row) and the removed objects (in the last row).

VII. Conclusion

The inclusion tree is a non-redundant and full representation of an image, invariant to local changes of contrast. The basic objects are the interiors of the connected pieces of the level lines, that we call "shapes". The organization of the shapes of an image can be represented in a tree built on their geometrical inclusion. This representation inherits a scale-space property, where the scale is the area. It does not involve any smoothing of the image [29], therefore at any scale, the image does not appear blurred, on the contrary to the linear scale-space.

At some scale (minimum number of pixels), shapes have a certain stability that allows some contrast independent comparisons between images as shown in the experiments.

The locality chosen in this representation is driven by the connectedness. The boundary of a shape, which is a connected component of the level lines, is then taken as a whole. However, in presence of partial occlusions for example, only pieces of the shapes can be compared. Two strategies can address this problem. The simplest would be to define partial comparison between shapes. Another would be to aggregate pixel information into "shapes" based on another criterion than connectedness...

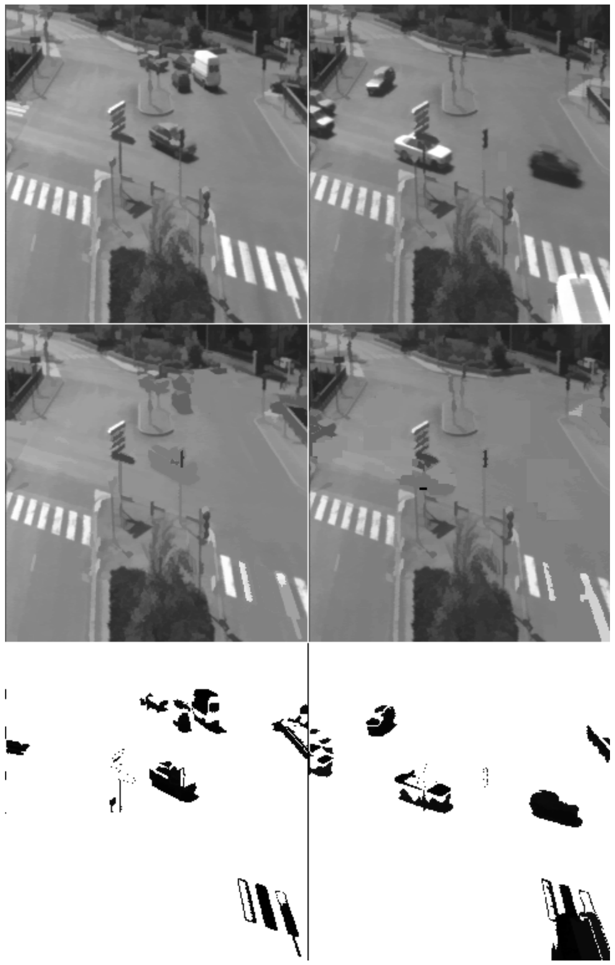

Fig. 17. Up: two images 256x256 of a crossing (few minutes differences). Middle : left (resp. right) objects of the up-left (resp. up-right) image that are entirely in the right (resp. left). Down: Objects that are not matching entirely (sum of the CPU time for the two intersections: 5.1 s on a P200MMX).

Acknowledgments

The authors thank Jean-Michel Morel, Coloma Ballester and Manolo González for fruitful discussions about the subject. Part of this work has been done using the MegaWave2 image processing environment (`http://www.cmla.ens-cachan.fr`), Cognitech, Inc. image processing facilities (`http://www.cognitech.com`), and Inrets facilities.

References

[1] Y. Meyer, *Wavelets: Algorithms and Applications*, SIAM, Philadelphia, 1993.

[2] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.

[3] L. Alvarez, F. Guichard, P.L. Lions, and J.M. Morel, "Axioms and fundamental equations of image processing," *Arch. Rational Mechanics and Anal.*, vol. 16, no. 9, pp. 200–257, 1993.

[4] B.M. ter Haar Romeny, Ed., *Geometry-Driven Diffusion in Computer Vision*, Kluwer Academic Publishers, 1994.

[5] D. Marr, *Vision*, Freeman and Co., 1982.

[6] R. Hummel, "Representations based on zero-crossing in scale-space," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition.* IEEE, 1986, pp. 204–209.

[7] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[8] M. Nitzberg and D. Mumford, "The 2.1 sketch," in *Proceedings of the* 3^d *International Conference on Computer Vision*, Osaka, Japan, 1990.

[9] J.M. Morel and S. Solimini, *Variational Methods in Image Processing*, Birkhäuser, 1994.

[10] J.J. Koenderink, "The structure of images," *Biological Cybernetics*, vol. 50, pp. 363–370, 1984.

[11] Witkin, "Scale-space filtering," *Proc. of IJCAI, Karlsruhe*, pp. 1019–1021, 1983.

[12] M. Wertheimer, "Untersuchungen zur lehre der gestalt, ii," *Psychologische Forschung*, , no. 4, pp. 301–350, 1923.

[13] G. Matheron, *Random Sets and Integral Geometry*, John Wiley, N.Y., 1975.

[14] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, 1982.

[15] V. Caselles, B. Coll, and J.M. Morel, "Topographic maps," preprint CEREMADE, 1997.

[16] F. Guichard and J.M. Morel, "Partial differential equations and image iterative filtering," in *Tutorial International Conference of Image Processing*, Washington D.C., 1995.

[17] F. Guichard and J.M. Morel, "Partial differential equations and image iterative filtering," *State of the Art in Numerical Analysis*, 1996.

[18] A. Rosenfeld, "On connectivity properties of grayscale pictures," *Pattern Recognition*, vol. 16, pp. 47–50, 1983.

[19] Y. Wang and Bhattacharya, "On parameter-dependent connected components of gray images," *Pattern Recognition*, vol. 29, no. 8, pp. 1359–1368, 1996.

[20] L. Yaroslavsky and M. Eden, *Fundamentals of digital optics*, Birkhäuser, 1996.

[21] Y. Wang and P. Bhattacharya, "Hierarchical stereo correspondence using features of gray connected components," in *Proceedings of the International Conference on Image Processing*, Santa Barbara, 1997, IEEE, pp. 264–267.

[22] M.H.A. Newman, *Elements of the Topology of Plane Sets of Points*, Cambridge University Press, 1951.

[23] V. Caselles, J.L. Lisani, J.M. Morel, and G. Sapiro, "Shape preserving local histogram modification," in *Proceedings of the International Conference of Image Processing*, Lausanne, Switzerland, 1996, IEEE.

[24] L. Vincent, "Grayscale area openings and closings, their efficient implementation and applications," in *Proceedings of the* 1^{st} *Workshop on Mathematical Morphology and its Applications to Signal Processing*, J. Serra and Ph. Salembrier, Eds., Barcelona, Spain, 1993, pp. 22–27.

[25] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[26] S. Masnou and J.M. Morel, "Image restoration involving connectedness," in *Proceedings of the* 6^{th} *International Workshop on Digital Image Processing and Computer Graphics*, Vienna, Austria, 1998, SPIE, vol. 3346.

[27] C. Ballester, E. Cubero-Castan, M. González, and J.M. Morel, "Contrast invariant image intersection," preprint CEREMADE, 1998.

[28] P. Monasse, "Contrast invariant image registration," in *Proceedings of Int. Conf. on Accoustics, Speech and Signal Processing*, Phoenix, Arizona, 1999, vol. 6, pp. 3221–3224.

[29] P. Monasse and F. Guichard, "Scale-space from a level lines tree," to appear in Proc. of 2^{nd} Workshop on Scale-Space Theories in Computer Vision, 1999.

Pascal Monasse, born in 1971, received his engineer diploma from the french Ecole Nationale des Ponts et Chaussées in math and computer science in 1995. His previous research, at Laboratoire de Météorologie Dynamique, Ecole Normale Supérieure (Ulm), France, concerned the construction of wavelet bases used in new schemes of resolution of partial differential equations. He is currently a PhD student (adviser: Jean-Michel Morel) at the Ecole Normale Supérieure de Cachan (France). His research fields are mathematical morphology applied to image representation, image registration, comparison of images and motion estimation.

Frédéric Guichard is a former student of the Ecole Normale Supérieure (Ulm) of Paris (France), in math. He received the PhD level in 1994 at the University Paris Dauphine (France). Jean-Michel Morel was his adviser. The PhD proposed a characterisation of the scale-spaces (theory of smoothing) by partial differential equations. He received the Cisi ingénierie award in 1993, and the Science and Defence award (French Defence Ministry) in 1996 for joined researches. During 1995-1996, he developed in Cognitech, Inc. (Pasadena, CA USA) an algorithm for motion estimation and images fusion (patent pending). He is now working for the French Ministry of Equipment at Inrets (Paris), on developing video processing tools for roads surveillance, car/people detections or counting, and anti-collision systems for vehicles.