A MultiScale Particle Filter Framework for Contour Detection

Nicolas Widynski and Max Mignotte

Abstract—We investigate the contour detection task in complex natural images. We propose a novel contour detection algorithm which jointly tracks at two scales small pieces of edges called edgelets. This multiscale edgelet structure naturally embeds semi-local information and is the basic element of the proposed recursive Bayesian modeling. Prior and transition distributions are learned offline using a shape database. Likelihood functions are learned online, thus are adaptive to an image, and integrate color and gradient information via local, textural, oriented, and profile gradient-based features. The underlying model is estimated using a sequential Monte Carlo approach, and the final soft contour detection map is retrieved from the approximated trajectory distribution. We also propose to extend the model to the interactive cut-out task. Experiments conducted on the Berkeley Segmentation data sets show that the proposed MultiScale Particle Filter Contour Detector method performs well compared to competing state-of-the-art methods.

Index Terms—Particle filtering, sequential Monte Carlo methods, statistical model, multiscale contour detection, BSDS

1 Introduction

Detecting contours is an ubiquitous task in image processing, as it is often the basis of higher level applications, such as segmentation, recognition, tracking, etc. The intrinsic variability of natural images makes this task a proper challenge. In this paper, we define a contour as a visually salient, well-defined chain of connected pixels. This definition may be interpreted in terms of the Gestalt Theory, which underlines the importance of perceptual grouping and continuation properties for human visual perception. Also, saliency is contextual, suggesting that it conforms the Helmhotz principle, which confers more importance to rare geometric patterns. Properties of good continuation and saliency shall serve as motivations of the proposed contour detector.

A connected pixel set is the atomic element of the proposed method, and is called an *edgelet*. This term shall not be mistaken with the edgelet transform (an image representation method), however, our definition is similar to the ones in [1], [2]. The structure of an edgelet is learned offline using a shape database. This results in the modeling of an empirical prior distribution. This choice differs from contour detection approaches integrating a prior information by imposing a potentially restrictive mathematical model. The contextual visual saliency is learned online using tail distributions, notably employed in the *a contrario* framework proposed by Desolneux et al. [3]. The associated likelihoods integrate image feature statistics to be adaptive to the image and hence get high responses only on perceptually significant contours. We also want to express the bounds between the edgelets, reflecting the continuity principle of the Gestalt Theory. This can be done in a Markovian modeling fashion by defining a spatial transition model between consecutive edgelets. Prior, transition, and likelihoods models are the basic ingredients of Sequential Monte Carlo methods. Among those, it turns out that particle filtering techniques are particularly well-suited for estimating these kinds of recursive distributions.

This is not the first attempt to use a particle filtering technique to extract contours. In 2001, Pérez et al. [4] proposed the *JetStream*, a well-known algorithm that retrieves one contour curve from an image by tracking points locally at a fixed step length. The authors proposed a semi-automatic routine to extract complex contours by allowing the user to constrain the contour path. This approach is useful for the interactive cut-out task, but by nature, can hardly be applied to the challenging problem of automatic contour detection. Other particle filtering techniques have been used in the context of vessels and arteries detection in 3D CT data [5], [6]. Like the *JetStream* algorithm, these techniques have been mainly dedicated for semi-automatic and/or single detection tasks. Contrary to the aforementioned methods, our particle filtering framework is fully automatic, semi-local, and contextually-dependent. Moreover, compared to our preliminary model [7], the edgelets are defined at two scales, meaning that the algorithm locally tracks the edgelets along contours by sequentially operating the computations on each scale. This yields to our new MultiScale Particle Filter Contour Detector (MS-PFCD).

This paper is organized as follows. Section 2 makes a study of previous works proposed in the contour detection literature. In Section 3, we propose to learn the distributions that handle the multiscale edgelets and define our Bayesian model. The tracking algorithm for contour detection based on a particle filtering technique is then described in Section 4. We show extensive experimental results on the BSDS300 and the BSDS500 in Section 5. Section 6 proposes several tools to carry out interactive contour detections with the proposed model. We finally conclude in Section 7.

• *The authors are with the Department of Computer Science and Operations Research (DIRO), University of Montreal, Montreal, Canada. E-mail: {widynski, mignotte}@iro.umontreal.ca.*

Manuscript received 20 Aug. 2012; revised 6 Jan. 2014; accepted 14 Feb. 2014. Date of publication 23 Feb. 2014; date of current version 10 Sept. 2014. Recommended for acceptance by C. Rother. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPAMI.2014.2307856

0162-8828 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

2 Related Work

Since the appearance of human-annotated image database, a noticeable effort has been made to propose novel, sophisticated, and efficient contour detection methods. In particular, Martin et al. [8] have introduced the Berkeley Segmentation data set (BSDS300) in which 300 natural color images have been manually segmented by several contributors. The data set is divided into a training set of 200 images and a test set of 100 images. In [9], Arbelaez et al. have extended the original Berkeley Segmentation data set by proposing the BSDS500 in which the BSDS300 images are used for training and validation, and 200 new images are used for testing. There exist plenty of contour detection methods in the literature. Among them, we gave in this section priority to the ones evaluated on the Berkeley Segmentation data sets. As they are not technically contour detection methods, we excluded approaches that start from an edge detection map and then try to improve the detections. This concerns segmentation and grouping algorithms.

2.1 Learning-Based Approaches

2.1.1 Probability-of-Boundary (Pb)-Based Approaches

Authors of the BSDS300 introduced a key contour detector [10], the Probability-of-Boundary detector, that will be used and improved a considerable number of times afterwards. This Pb detector computes four oriented features: the brightness, the color, the texture gradients, and the oriented energy. The feature responses are independently optimized through a learning procedure on the BSDS300 training data set and are then combined using a logistic regression classifier in order to define a probability that a pixel belongs to a contour. Ren [11] proposed a multiscale version of the Pb detector, that we call here the MS-Pb detector, and which, besides computing the Pb on six scales, integrates two cues that handle multiscale consistency: the localization, which penalizes pixels too far from the expected contours; and the relative contrast, which normalizes the Pb value of a pixel using the values of its neighbors so as to support the most locally relevant pixels. Maire et al. [12] proposed another multiscale version of the Pb detector, presented as well in [9]. Their gPb detector is a linear combination of the Pb features computed on three scales and of a globalization term. This latter term integrates a global information using the Normalized Cuts criterion [13], which encodes the similarity between pixels by eigenvector analysis. The underlying similarity matrix is computed from the multiscale Pb. The globalization term is then defined as a linear combination of oriented gradient filters of eigenvector matrices. Kokkinos proposed to use the Pb features with an Anyboost classifier [14]. The optimization is done using a criterion based on an approximation of the F-measure. Contributions also include the use of the whole set of feature-label pairs during the training stage. Finally, the author remarked that introducing dense discriminative features extracted from appearance descriptors improved the results.

2.1.2 Dictionary-Based Learning

Dollar et al. [15] proposed the BEL algorithm, that stands for Boosted Edge Learning. They learn a Probabilistic Boosting Tree classifier using thousands of simple features. Each of these is computed on a large-sized image patch, which enables the integration of high level information. For example, they can integrate information related to the Gestalt law, e.g., the completion and the parallelism. They can also consider the image background implicitly or detect roads in a satellite view. Mairal et al. [16] used a discriminative sparse image representation to combine classifiers defined at different scales. By training the classifiers using two dictionnaries (good and bad edges), this provides a confidence value that a pixel belongs to a contour. An advantage is that this approach learn specific dictionnaries to detect class-specific contours. Recently, Xen and Bo [17] replaced the Pb features from the gPb pipeline by sparse code gradients (SCG), i.e., automatically learned local patches that are represented using sparse coding. Like the gPb, patches are computed at different scales and orientations, but unlike it, they are then classified using a linear SVM. This approach can also be adapted to RGB + depth cameras.

2.2 Statistical-Based Approaches

Felzenszwalb and McAllester [18] considered a statistical edge model, whose elementary variables are two-pixel-long segments. Features are computed on these segments. They are able to define a prior distribution relying on the number of contours in an image and that encourages smooth contours. The maximum a posteriori (MAP) estimation of the resulting joint distribution of on- and off-contour pixels is transformed on a weighted min-cover problem. This problem is NP-hard and the solution is hence approximated using a greedy heuristic algorithm. Konishi et al. [19] proposed a statistical multiscale data-driven edge detector. The statistical inference is expressed as a log-likelihood ratio test between the on- and off-edge non-parametric learned distributions of the filter responses. Besides, they give the possibility to model a posterior distribution that embeds spatial edge grouping. Destrempes and Mignotte [20] modeled the image as a graph, whose vertices are the pixels and the edges are the pixels adjacence relationships. They define four parametric likelihoods, conditioned by the element label, on- or off-contour, and the element type, vertices or edge. The prior is a Gibbs distribution, and includes a smoothing parameter. These parameters are learned online using an Iterative Conditional Estimation procedure. The edge detection task is then viewed as finding the MAP of the associated filtering distribution. This model can also be used for localization of shapes [21] and semi-automatic extraction of contours. Recently, Payet and Todorovic [22] proposed to estimate a probability map of edge saliences that they call tPb. Edges are computed considering texture variations at different scales and randomly organized adjacent pairs of windows.

3 Learning the Bayesian Model

Before giving the motivations of the proposed section and further, the proposed sequential Bayesian framework, we introduce some notations. The basic element of the framework is a two-scales set of connected points. Let $\mathrm{E} = (\mathrm{E}^1, \ldots, \mathrm{E}^{M_\mathrm{E}}) \in \Gamma_\mathrm{E} \subset \Omega_\mathrm{E}^{M_\mathrm{E}}$ be a set of M_E four-connected points defined at the coarse scale. Each point E^i is defined

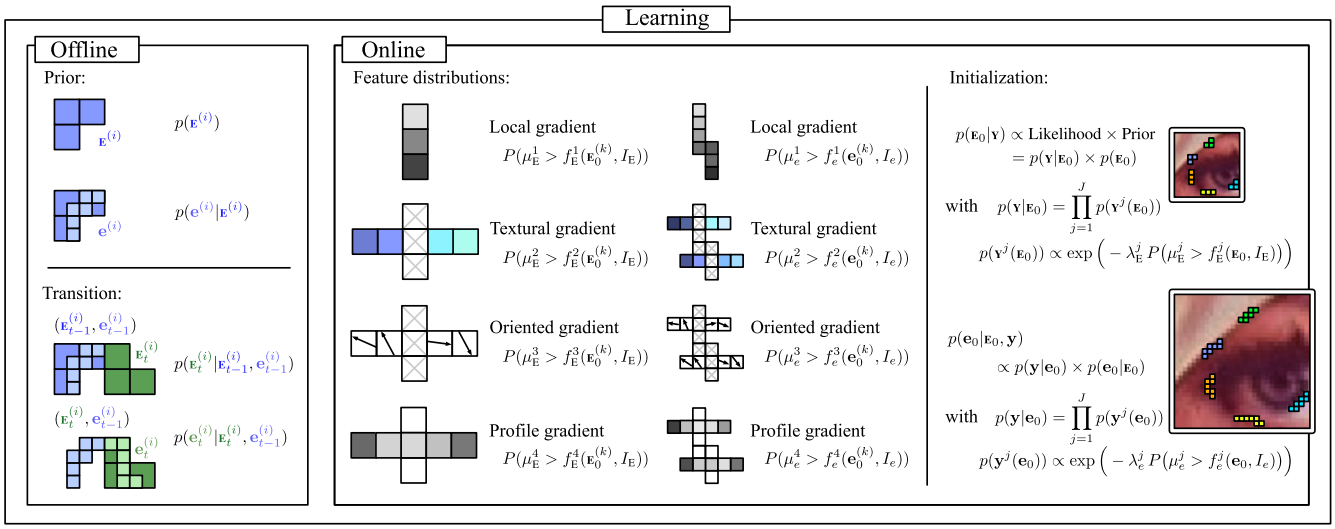

Fig. 1. The learning procedure is divided into two steps. The offline step estimates the prior and the transition distributions, which are used to generate samples in the contour tracking procedure. The online step is performed on the image to be tracked, and aims at learning: the feature distributions, in order to recognize the meaningful contours in the image; and the initialization distribution, in order to (re-)initialize the tracking in the contour detection procedure. (i) denotes a sample in the offline procedure, (k) denotes a sample in the online procedure.

in the image domain Ω_{E} of the coarse scale. The value of M_{E} is fixed and is a parameter of the method. For example, in our experiments we set $M_{\mathrm{E}} = 3$ to balance computational cost and detection robustness. The vector E is henceforth referred to as an *edgelet* defined at the *coarse* scale.

Let $\mathbf{e} = (\mathbf{e}^1, \ldots, \mathbf{e}^{M(\mathbf{e})}) \in \Gamma(\mathbf{e}) \subset \Omega_e^{M(\mathbf{e})}$ be a set of $M(\mathbf{e})$ four-connected points defined at the reference scale. Each point $\mathbf{e}^i$ is defined in the image domain Ω_e of the reference scale (full resolution). Unlike the coarse scale, the value of $M(\mathbf{e})$ is variable, its range depends on M_{E}. The vector $\mathbf{e}$ is henceforth referred to as an *edgelet* defined at the *reference* scale. The coarse scale is defined as a factor of 2 of the reference scale.

The proposed contour detection method is based on a spatial tracking approach. This means that we want to define an edgelet at a certain step, or time, of the tracking procedure. Then E and $\mathbf{e}$ are indexed by time, t, and can be modeled as stochastic processes, $\{\mathrm{E}_t\}_{t\in\mathbb{N}}$ and $\{\mathbf{e}_t\}_{t\in\mathbb{N}}$, respectively. This time modeling is only artificial, as the temporal process refers actually to a spatial process. We also need to introduce $\mathrm{Y} \in \mathcal{Y}_{\mathrm{E}}$ and $\mathbf{y} \in \mathcal{Y}_e$, the observations extracted from the image features. Unlike the states E_t and $\mathbf{e}_t$, observations $\mathbf{y}$ and $\mathbf{y}$ are not time-stamped, as they could be gathered at once, e.g., at the beginning of the algorithm. Our Bayesian tracking procedure requires the definition of height distributions: the prior $p(\mathrm{E})$ and $p(\mathbf{e}|\mathrm{E})$, the transitions $p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1})$ and $p(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1})$, the likelihoods $p(\mathrm{Y}|\mathrm{E}_t)$ and $p(\mathbf{y}|\mathbf{e}_t)$, and the initializations $p(\mathrm{E}_0|\mathrm{Y})$ and $p(\mathbf{e}_0|\mathrm{E}_0, \mathbf{y})$. The prior and transition distributions are carried out offline whereas the likelihood function and initialization distribution are carried out online since they depend on the image. The initialization distribution is used to circumvent a limitation of any particle filtering technique that assumes known the starting conditions. Also, we present four features that compose the likelihood: the local gradient, the textural gradient, the oriented gradient, and the profile gradient. All these distributions are illustrated on Fig. 1 and are the subject of the following sections.

3.1 Offline Learning: Prior Models

The vectors E and $\mathbf{e}$ are small pieces of a contour, integrating more information than a classical pixel-wise formulation. Nevertheless, they remain semi-local, in order to be applied generally to most of the contours. By learning their prior distributions, we avoid imposing mathematical constraints that may decrease the detection efficiency of an algorithm, since it is impractical to define a mathematical model that captures every possible contour singularity.

We want to learn the prior distribution of the coarse scale $p(\mathrm{E}^{2:M}|\mathrm{E}^1)$, with $\mathrm{E}^{i:j} \triangleq (\mathrm{E}^i, \ldots, \mathrm{E}^j)$ and $i < j$, which is centered with respect to its first point E^1. This way, the distribution only captures the configuration of the edgelet. We also want to learn the prior distribution of the reference scale $p(\mathbf{e}|\mathrm{E})$. Algorithm 1 presents the procedure to collect the sets of S_{p} multiscale edgelets $\overline{B}_{\mathrm{E}} = \{\overline{\mathrm{E}}^{(1)}, \ldots, \overline{\mathrm{E}}^{(S_{\mathrm{p}})}\}$ and $\overline{B}_{\mathbf{e}} = \{\overline{\mathbf{e}}^{(1)}, \ldots, \overline{\mathbf{e}}^{(S_{\mathrm{p}})}\}$, with $\overline{\mathrm{E}}^{(s)}$ the sth realization of the set at the coarse scale and $\overline{\mathbf{e}}^{(s)}$ its corresponding realization at the reference scale. The data have been collected using the BSDS300 training data set.

Algorithm 1: Collecting the multiscale edgelets

Input: A shape database
Output: Couple of edgelet sets $(\bar{B}_{\mathrm{E}}, \bar{B}_e)$
begin
  for $s = 1, \ldots, S_{\mathrm{p}}$ **do**
    - Get an image I and a segmentation H at random
    - Extract a multiscale edgelet $(\bar{\mathrm{E}}^{(s)}, \bar{\mathbf{e}}^{(s)})$ from (I, H) at random
    - Center $\bar{\mathrm{E}}^{(s)}$ with respect to $\bar{\mathrm{E}}^{1,(s)}$
return $(\{\bar{\mathrm{E}}^{(1)}, \ldots, \bar{\mathrm{E}}^{(S_{\mathrm{p}})}\}, \{\bar{\mathbf{e}}^{(1)}, \ldots, \bar{\mathbf{e}}^{(S_{\mathrm{p}})}\})$

Let $B_{\mathrm{E}} = \{\mathrm{E}^{(1)}, \ldots, \mathrm{E}^{(S'_{\mathrm{p}})}\}$ be the subset of the distinct elements of $\overline{B}_{\mathrm{E}}$, and $o_{\mathrm{E}}^{(s)}$ be the occurrence number of the sth element of B_{E} in $\overline{B}_{\mathrm{E}}$, then the prior distribution approximation of $p(\mathrm{E}^{2:M}|\mathrm{E}^1)$ is given by:

$$p(\mathrm{E}^{2:M}|\mathrm{E}^1) \approx \sum_{s=1}^{S'_\mathrm{p}} \tilde{o}_\mathrm{E}^{(s)} \delta_{\mathrm{E}^{2:M,(s)}}^{\mathrm{E}^{2:M}}, \quad (1)$$

with $\tilde{o}_\mathrm{E}^{(s)} = o_\mathrm{E}^{(s)}/S_\mathrm{p}$ the empirical frequency and δ_a^b the Kronecker function, i.e., $\delta_a^b = 1$ if $a = b$, and 0 otherwise.

Let $B_e^{\mathrm{E}^{(s)}} = \{\mathbf{e}^{(1,s)}, \ldots, \mathbf{e}^{(S_\mathrm{p}^{(s)},s)}\}$ be the set of the distinct elements of $\overline{B}_e$ that have been extracted from $\mathrm{E}^{(s)}$, and $o_e^{(r,s)}$ be the occurrence number of the rth element of $B_e^{\mathrm{E}^{(s)}}$ in $\overline{B}_e$ and whose corresponding coarse edgelet is $\mathrm{E}^{(s)}$, then the prior distribution approximation of $p(\mathbf{e}|\mathrm{E})$ is given by[1]:

$$p(\mathbf{e}|\mathrm{E}) \approx \sum_{s=1}^{S'_\mathrm{p}} \sum_{r=1}^{S_\mathrm{p}^{(s)}} \tilde{o}_e^{(r,s)} \delta_{\mathrm{E}^{(s)}}^{\mathrm{E}} \delta_{\mathbf{e}^{(r,s)}}^{\mathbf{e}}, \quad (2)$$

with $\tilde{o}_e^{(r,s)} = o_e^{(r,s)}/S_\mathrm{p}^{(s)}$.

3.2 Offline Learning: Transition Models

We defined in the previous section a way to initialize edgelets. Next, to randomly extract full contours with our tracking algorithm, we need to generate a candidate multiscale edgelet at a certain time t given the previous one at $t-1$. This is what the transition distributions $p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1})$ and $p(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1})$ are designed to do. Both distributions are conditioned by $\mathbf{e}_{t-1}$ in order to guarantee connexity from one time to another. Finally, in order to learn the distribution $p(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1})$, we define $B_e^{-\mathrm{E}^{(s)}}$ the set of the distinct elements of $\overline{B}_e$ that are potential predecessors of $\mathrm{E}^{(s)}$. Using the same shape data set as in Section 3.1, the approximation procedures of these two distributions are given in Algorithms 2 and 3.

Algorithm 2: Approximation of the transition distribution of an edgelet at the coarse scale

Input: Prior distributions, a shape database
Output: Approximation of $p(\mathbf{E}_t|\mathbf{E}_{t-1}, \mathbf{e}_{t-1})$
begin
  foreach $(\mathbf{E}^{(s)}, \mathbf{e}^{(r,s)}) \in B_\mathrm{E} \times B_e^{\mathbf{E}^{(s)}}$ **do**
    repeat
      - Get an image I and a segmentation H at random
      - Extract a multiscale edgelet from (I, H) such that $(\mathbf{E}_{t-1}, \mathbf{e}_{t-1}) = (\mathbf{E}^{(s)}, \mathbf{e}^{(r,s)})$ and let $\mathbf{E}_t = \mathbf{E}^{(u)} \in B_\mathrm{E}$ be its successor at the coarse scale
      - Increment by $1/S_\mathrm{t}$ the probability: $p(\mathbf{E}_t = \mathbf{E}^{(u)}|\mathbf{E}_{t-1} = \mathbf{E}^{(s)}, \mathbf{e}_{t-1} = \mathbf{e}^{(r,s)})$
    until S_t *times*
return $\{p(\mathbf{E}_t^{(u)}|\mathbf{E}_{t-1}^{(s)}, \mathbf{e}_{t-1}^{(r,s)}), \forall(u,s,r)\}$

3.3 Online Learning: Observation Model

In this section, we define the observation model $p(\mathrm{Y}, \mathbf{y}|\mathrm{E}_t, \mathbf{e}_t)$, which measures the adequation between the data $(\mathrm{Y}, \mathbf{y})$, and a multiscale edgelet at a time t, $(\mathrm{E}_t, \mathbf{e}_t)$:

$$p(\mathrm{Y}, \mathbf{y}|\mathrm{E}_t, \mathbf{e}_t) = p(\mathrm{Y}|\mathrm{E}_t)\, p(\mathbf{y}|\mathbf{e}_t), \quad (3)$$

where independence hypotheses have been assumed to simplify the estimation. Also, we consider a particular case in which the likelihoods $p(\mathrm{Y}|\mathrm{E}_t)$ and $p(\mathbf{y}|\mathbf{e}_t)$ are similarly defined. Then, in order to lighten the content of the following section, we will only define the likelihood $p(\mathbf{y}|\mathbf{e}_t)$ and its associated features. The definitions at the coarse scale can simply be obtained by swapping $\mathbf{e}_t$ for E_t, the subscripts of e for E, and $\mathbf{y}$ for Y.

Algorithm 3: Approximation of the transition distribution of an edgelet at the reference scale

Input: Prior distributions, a shape database
Output: Approximation of $p(\mathbf{e}_t|\mathbf{E}_t, \mathbf{e}_{t-1})$
begin
  foreach $(\mathbf{E}^{(s)}, \mathbf{e}^{-(r,s)}) \in B_\mathrm{E} \times B_e^{-\mathbf{E}^{(s)}}$ **do**
    repeat
      - Get an image I and a segmentation H at random
      - Extract two edgelets from (I, H) such that $\mathbf{E}_t = \mathbf{E}^{(s)}$ and $\mathbf{e}_{t-1} = \mathbf{e}^{-(r,s)}$ and let $\mathbf{e}_t = \mathbf{e}^{(u,s)} \in B_e^{\mathbf{E}^{(s)}}$ be the successor at the reference scale
      - Increment by $1/S_\mathrm{t}$ the probability: $p(\mathbf{e}_t = \mathbf{e}^{(u,s)}|\mathbf{E}_t = \mathbf{E}^{(s)}, \mathbf{e}_{t-1} = \mathbf{e}^{-(r,s)})$
    until S_t *times*
return $\{p(\mathbf{e}_t^{(u,s)}|\mathbf{E}_t^{(s)}, \mathbf{e}_{t-1}^{-(r,s)}), \forall(u,s,r)\}$

To make our detector robust, we consider several observations, i.e., $\mathbf{y} = (\mathbf{y}^1, \ldots, \mathbf{y}^J)$. The component $\mathbf{y}^j : \Gamma(\mathbf{e}) \to \mathcal{Y}_e^j$ is a vector observation related to the jth feature, and designates observations from the whole set of edgelets included in an image $I_e : \Omega_e \to [0,1]$. Let $\mathbf{y}^j(\mathbf{e})$ be the observation at the edgelet $\mathbf{e}$. The joint likelihood $p(\mathbf{y}|\mathbf{e}_t)$ reads:

$$p(\mathbf{y}|\mathbf{e}_t) = \prod_{j=1}^{J} p(\mathbf{y}^j(\mathbf{e}_t)), \quad (4)$$

where we assumed a conditional independence of the observations given the edgelet state. This hypothesis is reasonable when considering competing marginal likelihoods $\{p(\mathbf{y}^j(\mathbf{e}_t)), \forall j\}$, which impose that their values are relatively high when estimated on true contours. This expression can be associated to the $p_\mathrm{on}/p_\mathrm{off}$ likelihood ratio, used in particular in the *JetStream* algorithm [4], in which we defined $p_\mathrm{on} = p(\mathbf{y}^j(\cdot))$ and $p_\mathrm{off} = \mathcal{U}(\cdot)$. This simplification discards a term dependent on the whole set of observations, $\mathbf{y}^j$, and rather focuses on the observation at the considered edgelet, $\mathbf{y}^j(\mathbf{e}_t)$. As we will see further in this section, modeling p_on is enough to express a notion of context.

Before defining these features, we propose a general formulation of the densities $\{p(\mathbf{y}^j(\cdot)), \forall j\}$. It is clear that the quantity of information of each feature depends on the image itself. For example, a local gradient norm feature typically overdetects in textured images whereas its sensitivity drops in blurry ones. Thus, the interpretation of the feature responses should be different in these two cases, meaning that a candidate should be considered more relevant when

1. Actually, the prior distribution at the reference scale is $p(\mathbf{e}, \mathbf{m}|\mathrm{E}) = p(\mathbf{e}|\mathrm{E}, \mathbf{m})\, p(\mathbf{m}|\mathrm{E})$, with $\mathbf{m} \in \mathbb{N}$ the r.v. which denotes the length of the edgelet at the reference scale, $p(\mathbf{e}|\mathrm{E}, \mathbf{m}) \triangleq p(\mathbf{e} = (\mathbf{e}^1, \ldots, \mathbf{e}^\mathbf{m})|\mathrm{E})$, and $p(\mathbf{m}|\mathrm{E})$ a probability proportional to the number of edgelets at the reference scale of length $\mathbf{m}$ included in E. By defining (2) as $p(\mathbf{e}|\mathrm{E}) \triangleq p(\mathbf{e}|\mathrm{E}, \mathbf{m})$ with $\mathbf{e}$ belonging to the set of all the possible edgelets of different lengths, the probability $p(\mathbf{m}|\mathrm{E})$ is implicitly computed, and $\mathbf{m}$ can thus be omitted.

it obtains a singular high feature response value in the image. In other terms, *context* is important. In visual perception, context is related to the Helmhotz principle, which states that relevant geometric structures have a very low probability of occurring in a random context. This idea has been used in an *a contrario* framework [2], [3], in which the pioneer authors defined a notion of *meaningfulness* of an event. Here, an event is an edgelet and we want to compute how *meaningful* the feature response of this edgelet is on the image. Observations are thus defined as the tail distributions of the feature responses:

$$\mathbf{y}^j(\mathbf{e}_t) = P\big(\mu_e^j > f_e^j(\mathbf{e}_t, I_e)\big), \tag{5}$$

with μ_e^j the random variable associated to the feature $f_e^j : \Gamma(\mathbf{e}) \times \Omega_e \to \mathbb{R}$. This distribution can be viewed as a distribution of false alarms. Hence, the lower the probability $P(\mu_j > f)$, the more meaningful this event, i.e., the less likely the event corresponds to a false alarm. The approximation of the tail distributions is given in Algorithm 4 and results in:

$$P\big(\mu_e^j > f_e^j(\mathbf{e}_t, I_e)\big) \approx \frac{1}{S_f} \sum_{s=1}^{S_f} \mathbb{1}_{\left[-\infty, f_e^j\left(\mathbf{e}_0^{(s)}, I_e\right)\right]}\big(f_e^j(\mathbf{e}_t, I_e)\big),$$

with $\mathbb{1}_{\mathcal{A}}(x)$ the indicator function. We define the likelihoods in a way to support low values of false alarms probabilities:

$$p(\mathbf{y}^j(\mathbf{e}_t)) \propto \exp\big(-\lambda_e^j P\big(\mu_e^j > f_e^j(\mathbf{e}_t, I_e)\big)\big), \tag{6}$$

with $\lambda_e^j \in \mathbb{R}^+$ a learned constant value which aims at providing a good localization of high likelihoods and at impacting the influence of the feature f_e^j on the tracking procedure.

Algorithm 4: Approximation of the feature distributions

Input: Prior distributions, an image defined at two scales: $I_{\mathrm{E}} : \Omega_{\mathrm{E}} \to [0,1]$ and $I_e : \Omega_e \to [0,1]$

Output: Approximation of $P\big(\mu_{\mathrm{E}}^j > f_{\mathrm{E}}^j(\mathbf{e}_0, I_{\mathrm{E}})\big)$ and $P\big(\mu_e^j > f_e^j(\mathbf{e}_0, I_e)\big)$

begin

  for $s = 1, \ldots, S_{\mathrm{f}}$ **do**

    - Generate a starting point $\mathbf{e}_0^{1,(s)} \sim \mathcal{U}[\Omega_{\mathrm{E}}]$
    - Generate the edgelet shape $\mathbf{e}_0^{2:M,(s)}$ according to its prior distribution: $\mathbf{e}_0^{2:M,(s)} | \mathbf{e}_0^{1,(s)} \sim p(\mathbf{e}^{2:M} | \mathbf{e}^1)$
    - Generate the edgelet shape $\mathbf{e}_0^{(s)}$ according to its prior distribution: $\mathbf{e}_0^{(s)} \sim p(\mathbf{e} | \mathbf{e}_0^{(s)})$
    - Compute and store $f_{\mathrm{E}}^j(\mathbf{e}_0^{(s)}, I_{\mathrm{E}})$
    - Compute and store $f_e^j(\mathbf{e}_0^{(s)}, I_e)$

return *Ordered sets* $\{f_{\mathrm{E}}^j(\mathbf{e}_0^{(s)}, I_{\mathrm{E}}), \forall s\}$, $\{f_e^j(\mathbf{e}_0^{(s)}, I_e), \forall s\}$

We now describe our four paired features: f_{E}^1 and f_e^1 the local gradients, f_{E}^2 and f_e^2 the textural gradients, f_{E}^3 and f_e^3 the oriented gradients, and f_{E}^4 and f_e^4 the profile gradients. The main novelty about the proposed features comes from the edgelet modeling. In particular, we are expecting to provide robust feature responses, since they are semi-local and thus less dependent on noise. Again, the features being identical at the coarse and reference scales, their definitions are given using an edgelet $\mathbf{e}$ defined at the reference scale and of length M (actually $M(\mathbf{e})$, but we omitted the reference to $\mathbf{e}$ for more clarity). Definitions of the coarse scale features can simply be obtained using the corresponding notation system.

3.3.1 Local Gradient

This classical feature uses the 2×2 gradient norm $|\nabla I_e|$. The gradient feature f_e^1 is computed along the edgelets $\mathbf{e}_t$:

$$f_e^1(\mathbf{e}_t, I_e) = \Phi\big(\big(|\nabla I_e(\mathbf{e}_t^i)|\big)_{1 \le i \le M}\big).$$

The flexibility comes from the fusion operator Φ. One can set $\Phi = \min$, $\Phi = \max$, or a weighted mean $\Phi(v^1, \ldots, v^M) = \sum_{i=1}^M W(i)\, v^i$, with $W : \{1, \ldots, M\} \to [0,1]$ a weighting function. Note that since the image I_e is multidimensional, we take on each point the maximum gradient value among the different channels.

3.3.2 Textural Gradient

The textural gradient feature aims at getting low response values on texture locations, while getting high ones on object contours. For a point $\mathbf{e}_t^i$ of an edgelet $\mathbf{e}_t$, we consider its normal segment. The two sides of the normal segment of three consecutive points $(\mathbf{e}_t^{i-1}, \mathbf{e}_t^i, \mathbf{e}_t^{i+1})$ are noted $\overleftarrow{n}(\mathbf{e}_t^i)$ and $\overrightarrow{n}(\mathbf{e}_t^i)$. In a texture, the intuition is that pixel values along the first segment should not really differ from the ones of the second segment. Let $h_{\mathrm{T}}[a] = \{h_{\mathrm{T}}^r[a]\}_{r=1}^R$ be the histogram of a set of pixels a, where r is the bin index of a histogram of length R. In the case of color images, the length R equals $R_{\mathrm{T}} \times R_{\mathrm{T}} \times R_{\mathrm{T}}$, with R_{T} the number of bins by channel. Distances between pairs of histograms along normals of the curve are combined to form the textural gradient feature:

$$f_e^2(\mathbf{e}_t, I_e) = \Psi\big(\big(d_{\mathrm{B}}\big(h_{\mathrm{T}}[\overleftarrow{n}(\mathbf{e}_t^i)], h_{\mathrm{T}}[\overrightarrow{n}(\mathbf{e}_t^i)]\big)\big)_{2 \le i \le M-1}\big),$$

with Ψ the fusion operator, and d_{B} the Bhattacharyya distance between two histograms, i.e., $d_{\mathrm{B}}(h[a], h[b])$ is the square root of $1 - \sum_{r=1}^R \sqrt{h^r[a]\, h^r[b]}$. In order to reduce the spread of the histogram values, the widths of the bins are defined by the R_{T}-quantiles of each independent channel.

3.3.3 Oriented Gradient

The use of a histogram of oriented gradients [23] has been proposed several times in the contour detection and segmentation literature [9], [11], [12]. We keep the same notations introduced in the previous textural gradient feature section except that here, the histogram $h_{\mathrm{O}}[a] = \{h_{\mathrm{O}}^r[a]\}_{r=1}^R$ contains $R = 4 \times R_{\mathrm{m}}$ bins, with 4 the number of considered orientations (vertical, horizontal, and two diagonals) and R_{m} the number of magnitude bins. Hence for each point belonging to the normal of an edgelet point $\mathbf{e}_t^i$, we compute and store into the histogram its four oriented gradient values. Then, distances between pairs of histograms along normals of the curve are combined to form the oriented gradient feature:

$$f_e^3(\mathbf{e}_t, I_e) = \Upsilon\big(\big(d_{\mathrm{B}}\big(h_{\mathrm{O}}\big[\overleftarrow{n}\big(\mathbf{e}_t^i\big)\big], h_{\mathrm{O}}\big[\overrightarrow{n}\big(\mathbf{e}_t^i\big)\big]\big)\big)_{2\leq i\leq M-1}\big),$$

with Υ the fusion operator, and d_{B} the Bhattacharyya distance between two histograms. In the manner of the textural gradient, the widths of the channel bins are defined by the R_{m}-quantiles.

3.3.4 Profile Gradient

Sun et al. [24] proposed to analyze the gradient norm profile for image enhancement and super-resolution. The idea is to use the symmetry and the monotonicity of the gradient norm profile rather than the norm value. This profile is learned and represented by a parametric gradient profile model. In our application, we compare this learned profile to the ones extracted from the image. We expect that in presence of clutter, the shape of the gradient norm profile should be more reliable than the gradient norm values themselves. With $n(\mathbf{e}_t^i)$ the normal segment of three consecutive points $(\mathbf{e}_t^{i-1}, \mathbf{e}_t^i, \mathbf{e}_t^{i+1})$, the profile gradient feature is defined as:

$$f_e^4(\mathbf{e}_t, I_e) = \Xi\big(\big(-d_{\mathrm{KL}}\big(\big|\nabla\tilde{I}\big[n\big(\mathbf{e}_t^i\big)\big]\big|, G\big(n\big(\mathbf{e}_t^i\big); \mathbf{e}_t^i, \sigma_e, \kappa_e\big)\big)\big)_{2\leq i\leq M-1}\big)$$

with $|\nabla\tilde{I}[n(a)]|$ a normalized vector of absolute gradient values computed along the normal $n(a)$; $G(n(a); a, \sigma, \kappa)$ a normalized vector of generalized exponential density values computed along the normal $n(a)$ with respect to the center point a and of shape parameters σ and κ; and d_{KL} the Kullback-Leibler divergence between two discrete distributions of L elements, i.e., $d_{\mathrm{KL}}(p, q) = \sum_{l=1}^{L} p^l \log p^l/q^l$, with p^l and q^l the probabilities at point l. By denoting by $n(a)^k$ the kth point of the normal, and by $\|\cdot\|_2$ the L^2 norm, its generalized exponential density value is $g(\|n(a)^k - a\|_2; \sigma, \kappa)$, with the density $g(x; \sigma, \kappa)$ defined as:

$$g(x; \sigma, \kappa) = \frac{\kappa\sigma(\kappa)}{2\sigma\Gamma\left(\frac{1}{\kappa}\right)} \exp\Big(-\Big(\sigma(\kappa)\Big|\frac{x}{\sigma}\Big|\Big)^{\kappa}\Big),$$

with $\Gamma(.)$ the gamma function and $\sigma(\kappa) = \sqrt{\Gamma(\frac{3}{\kappa})/\Gamma(\frac{1}{\kappa})}$ the scaling factor that makes the second moment of the distribution g equal to σ^2.

3.4 Online Learning: Initialization Model

We finally estimate the initialization distributions $p(\mathrm{E}_0|\mathrm{Y})$ and $p(\mathbf{e}_0|\mathrm{E}_0, \mathbf{y})$. These distributions find their use in 1. the generation of samples in the tracking initialization step and 2. The reinitialization of samples in the tracking procedure. This latter action occurs, for example, when the tracking of a contour is over, in which case the initialization distribution starts a tracking procedure on a novel contour. Although it is possible to carry out these operations using the prior distributions $p(\mathrm{E}_0)$ and $p(\mathbf{e}_0|\mathrm{E}_0)$, the observations are more likely to provide samples located on true contours. The approximation procedures are described in Algorithms 5 and 6.

4 Contour Detection by Tracking Based on Particle Filter

We defined in Section 3 several distributions that manipulate the edgelets E_t and $\mathbf{e}_t$. In this section, we define the framework that handles these distributions by integrating them in a sequential Monte Carlo approach. Our goal is to estimate the distribution of the joint edgelets $\mathbf{x}_{0:t} = (\mathrm{E}_{0:t}, \mathbf{e}_{0:t})$ conditioned by a set of joint observations $\mathbf{z} = (\mathrm{Y}, \mathbf{y})$. Hence, at a time t, $\mathbf{e}_{0:t}$ defines a contour map at the reference scale of $t+1$ edgelet elements. The estimation of the so-called *trajectory distribution* $p(\mathbf{x}_{0:t}|\mathbf{z})$ is thus completed using a particle filtering technique that we present now.

Algorithm 5: Approximation of the initialization distribution at the coarse scale

Input: Prior and feature distributions, an image I_{E}
Output: Approximation of $p(\mathrm{E}_0|\mathrm{Y})$
begin
  for $s = 1, \ldots, S_{\mathrm{i_E}}$ **do**
    - Generate a starting point $\mathbf{E}_0^{1,(s)} \sim \mathcal{U}[\Omega_{\mathrm{E}}]$
    - Generate the edgelet shape $\mathbf{E}_0^{2:M,(s)}$ according to its prior distribution: $\mathbf{E}_0^{2:M,(s)}|\mathbf{E}_0^{1,(s)} \sim p(\mathbf{E}^{2:M}|\mathbf{E}^1)$
    - Compute the joint likelihood $w(\mathbf{E}_0^{(s)})$: $w(\mathbf{E}_0^{(s)}) \propto \prod_{j=1}^{J} \exp(\lambda_{\mathrm{E}}^j \; P\big(\mu_{\mathrm{E}}^j > f_{\mathrm{E}}^j(\mathbf{E}_0^{(s)}, I_{\mathrm{E}})\big))$ s.t. $\sum_{r=1}^{S_{\mathrm{i_E}}} w(\mathbf{E}_0^{(r)}) = 1$
return $\frac{1}{S_{\mathrm{i_E}}} \sum_{s=1}^{S_{\mathrm{i_E}}} w(\mathbf{E}_0^{(s)})\, \delta_{\mathbf{E}_0^{(s)}}^{\mathbf{E}_0}$

Algorithm 6: Approximation of the initialization distribution at the reference scale

Input: Prior, feature and coarse scale initialization distributions, an image I_e
Output: Approximation of $p(\mathbf{e}_0|\mathrm{E}_0, \mathbf{y})$
begin
  for $u = 1, \ldots, S_{\mathrm{i_E}}$ **do**
    for $s = 1, \ldots, S_{\mathrm{i}_e}$ **do**
      - Generate the edgelet shape $\mathbf{e}_0^{(s,u)}$ according to its prior distribution: $\mathbf{e}_0^{(s,u)} \sim p(\mathbf{e}|\mathbf{E}_0^{(u)})$
      - Compute the joint likelihood $w(\mathbf{e}_0^{(s,u)})$: $w(\mathbf{e}_0^{(s,u)}) \propto \prod_{j=1}^{J} \exp(\lambda_e^j \; P\big(\mu_e^j > f_e^j(\mathbf{e}_0^{(s,u)}, I_e)\big))$ s.t. $\sum_{r=1}^{S_{\mathrm{i}_e}} w(\mathbf{e}_0^{(r,u)}) = 1$
return $\{\frac{1}{S_{\mathrm{i}_e}} \sum_{s=1}^{S_{\mathrm{i}_e}} w(\mathbf{e}_0^{(s,u)})\, \delta_{\mathbf{e}_0^{(s,u)}}^{\mathbf{e}_0}, \forall(u, s)\}$

4.1 Estimating the Multi-Object Trajectory Distribution

In this section, we consider a special case of the Bayesian recursion, in which the observations are not indexed by time.

4.1.1 Estimating the Trajectory Distribution

Let $\mathbf{x}_t \in \mathcal{X}$ be the hidden state of a stochastic process at time t and $\mathbf{z} \in \mathcal{Z}$ be the measurement state. Under the Markovian hypothesis of the hidden states and the conditional independence hypothesis of the observations given the states, the trajectory distribution $p(\mathbf{x}_{0:t}|\mathbf{z})$ is given by [4]:

$$p(\mathbf{x}_{0:t}|\mathbf{z}) \propto p(\mathbf{x}_{0:t-1}|\mathbf{z})\, p(\mathbf{z}|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{7}$$

whose normalizing term is independent from $\mathbf{x}_{0:t}$. As there is no closed form expression of this distribution, we propose

to estimate it in a recursive way using a sequential simulation-based method, the *particle filter*. The method consists in computing the empirical distribution [25]:

$$P_N(d\mathbf{x}_{0:t}|\mathbf{z}) = \sum_{n=1}^{N} w_{0:t}^{(n)} \delta_{\mathbf{x}_{0:t}^{(n)}}(d\mathbf{x}_{0:t}), \quad (8)$$

where $\delta_{\mathbf{x}_{0:t}^{(n)}}(\cdot)$ is a Dirac mass centered on a hypothetic state realization $\mathbf{x}_{0:t}^{(n)}$ of the state $\mathbf{x}_{0:t}$, also called particle, $w_{0:t}^{(n)}$ is its weight, $d\mathbf{x}_{0:t}$ is an event of infinitesimal support, and N is the number of particles. The recursion estimation of the trajectory distribution in (7) can be carried out by three steps:

1) The diffusion step propagates the particle swarm $\{\mathbf{x}_{0:t-1}^{(n)}, w_{t-1}^{(n)}\}_{n=1}^{N}$ using an importance function $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(n)}, \mathbf{z})$.
2) The update step then computes new particle weights using observation $\mathbf{z}$, as:

$$w_t^{(n)} \propto w_{t-1}^{(n)} \frac{p\big(\mathbf{z}|\mathbf{x}_t^{(n)}\big)p\big(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}\big)}{q\big(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(n)}, \mathbf{z}\big)}, \text{ s.t. } \sum_{n=1}^{N} w_t^{(n)} = 1.$$

3) If necessary, the resampling step avoids a particle degeneracy problems by repopulating the particle set [25].

4.1.2 Partitioned Sampling

Estimating the posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{z})$ with the multiscale nature of the state $\mathbf{x}_{0:t} = (\mathrm{E}_{0:t}, \mathbf{e}_{0:t})$ may raise a problem. In fact, as described in Section 4.1.1, the particle filter makes use of an importance sampling procedure which involves the simulation and the weighting of the particles. This procedure suffers from a curse of the dimensionality: it has been shown in [26] that N^2 particles are necessary to achieve the same level of estimation performance as when estimating a single scale with N particles. To alleviate this problem, MacCormick et al. proposed the Partitioned Sampling algorithm which decomposes the vector state by partitioning the state space, and then by handling one component, i.e., scale, at a time [26], [27]. We present here a basic version of this algorithm, adapted to our purpose.

First, we introduce the weighted resampling procedure. This transforms a particle set $\{\mathbf{x}_{0:t}^{(n)}, w_t^{(n)}\}_{n=1}^{N}$ into another one $\{\tilde{\mathbf{x}}_{0:t}^{(n)}, w_t^{(n)}/g(\mathbf{x}_{0:t}^{(n)})\}_{n=1}^{N}$ while keeping the distribution intact [27]. The weighting function g is strictly positive and defined such that $\sum_{n=1}^{N} g(\mathbf{x}_{0:t}^{(n)}) = 1$. The new particle set is obtained by simulating according to the empirical distribution defined by the weights $\{g(\mathbf{x}_{0:t}^{(n)})\}_{n=1}^{N}$.

We describe now the Partitioned Sampling algorithm. From Section 3, we know that the edgelet propagations can be performed sequentially and that the likelihood factorization enables to deal with each scale independently, hence each vector observation is related to one edgelet scale. By considering the state E_t first, the Partitioned Sampling algorithm consists in

1) Propagating the particles using the marginal importance function of E_t;
2) Computing the weighting function such that $g(\mathbf{x}_{0:t}^{(n)}) \propto p(\mathrm{Y}|\mathrm{E}_t^{(n)})p(\mathrm{E}_t^{(n)}|\mathrm{E}_{t-1}^{(n)}, \mathbf{e}_{t-1}^{(n)})/q(\mathrm{E}_t^{(n)}|\mathrm{E}_{0:t-1}^{(n)}, \mathbf{e}_{t-1}^{(n)}, \mathrm{Y})$;
3) Performing a weighted resampling procedure on the particle set $\{(\mathrm{E}_{0:t}^{(n)}, \mathbf{e}_{0:t-1}^{(n)}), w_t^{(n)}\}_{n=1}^{N}$;
4) Propagating the particles using the marginal importance function of $\mathbf{e}_t$; and
5) Computing the particle weights $w_t^{(n)} \propto p(\mathbf{y}|\mathbf{e}_t^{(n)})\, p(\mathbf{e}_t^{(n)}|\mathrm{E}_t^{(n)}, \mathbf{e}_{t-1}^{(n)})/q(\mathbf{e}_t^{(n)}|\mathrm{E}_t^{(n)}, \mathbf{e}_{t-1}^{(n)}, \mathbf{y})$.

Defining the weighting function g in that way enables the generation of more samples for higher values of the marginal likelihood of E_t and greatly simplifies the computation in the last step of the Partitioned Sampling procedure since the g term is present both at the numerator and denominator of the final particle weight computation.

4.2 Particle Filter Contour Detection Algorithm

In this section, we describe our particle filter method dedicated to the contour detection task. We introduce $\mathbf{c}_t \in \{0, 1\}$ a binary random variable of jump: if $\mathbf{c}_t = 0$, the tracking of contour at time t goes on, otherwise, i.e., if $\mathbf{c}_t = 1$, the edgelet is initialized to a new contour. This is useful when the tracking of the current contour is lost or finished. The hidden state $\mathbf{x}_t$ is then composed of an edgelet E_t at the coarse scale, an edgelet $\mathbf{e}_t$ at the reference scale, and a jump variable $\mathbf{c}_t$, yielding to $\mathbf{x}_t = (\mathrm{E}_t, \mathbf{e}_t, \mathbf{c}_t)$. A particle filter requires the definition of four distributions: a prior $p(\mathbf{x}_0)$, to initialize particles; an importance function $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z})$, to predict a particle at time t given the past states and observations; a trajectory prediction $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, to define the prior evolution of a particle at time t given the past states; and a likelihood $p(\mathbf{z}|\mathbf{x}_t)$, to weight the particles using the last known measure. While the prior and the likelihood are learned in Section 3, the importance function and the transition need to be defined. The tracking procedure is summarized in Fig. 2.

4.2.1 Transition

First, we define the trajectory transition $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ such that the edgelet distribution depends on the jump variable. Also, we consider the edgelet transition at the coarse scale first, and make the edgelet transition at the reference scale dependent from the coarse scale edgelet, as proposed in Section 3:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1}, \mathbf{c}_t)\, p(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1}, \mathbf{c}_t)\, p(\mathbf{c}_t). \quad (9)$$

The jump variable $\mathbf{c}_t$ is assumed independent from $\mathbf{c}_{t-1}$ by simplicity. This way, it is related to the length of a contour. Let $p(\mathbf{c}_t = 1) = \beta$ be the probability of jump. The edgelet transition of the coarse scale is a mixture of the prior and the transition distributions learned in Sections 3.1 and 3.2, respectively. It depends on the value of the switching variable $\mathbf{c}_t$ in such a way that if it designates a jump, then the prior distribution is considered. Otherwise, it consists in a transition:

$$\begin{aligned} p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1}, \mathbf{c}_t) = {} & \mathbf{c}_t\, p\big(\mathrm{E}_t^{2:M}|\mathrm{E}_t^1\big)\, p\big(\mathrm{E}_t^1\big) \\ & + (1 - \mathbf{c}_t)\, p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1}), \end{aligned} \quad (10)$$

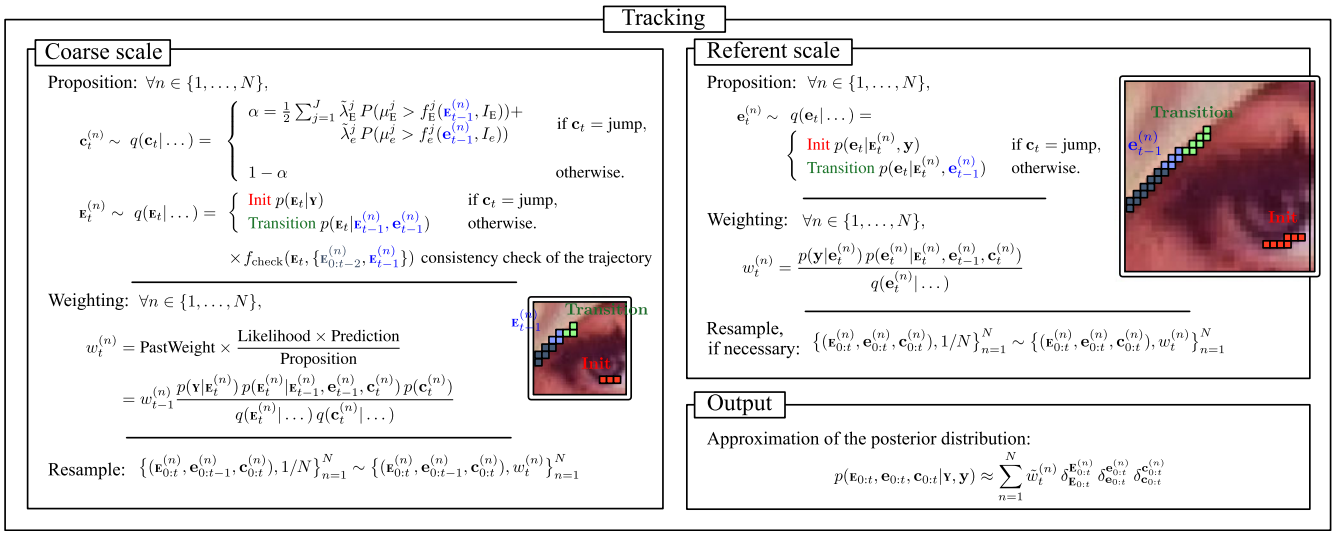

Fig. 2. The particle filter method approximates the trajectory distribution by a finite discrete set of samples called particles. In a contour tracking application, we propose to model the state of a particle with two components: $(\mathrm{E}_t, \mathrm{E}_t)$ an edgelet at two scales and $\mathbf{c}_t$ a jump variable, the latter being useful when the tracking of the current contour is finished. Each scale is proceeded sequentially. The tracking procedure at a particular scale is divided into two steps. The proposition step propagates the particles from time $t-1$ to time t, using the transition, the initialization, and the feature distributions defined in Section 3. The second step weights the propagated particles proportionately to their likelihood, giving more importance to relevant edgelets. These two steps approximate the trajectory distribution.

with the prior $p(\mathrm{E}_t^{2:M}|\mathrm{E}_t^1)$ learned in Section 3.1, the starting point defined using a uniform distribution over Ω_{E}, i.e., $p(\mathrm{E}_t^1) = \mathcal{U}[\Omega_e]$, and the transition $p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1})$ learned in Section 3.2. Following the example of the coarse scale, the edgelet transition of the reference scale is defined as:

$$p(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1}, \mathbf{c}_t) = \mathbf{c}_t\, p(\mathbf{e}_t|\mathrm{E}_t) + (1-\mathbf{c}_t)\, p(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1}), \quad (11)$$

with the prior $p(\mathbf{e}_t|\mathrm{E}_t)$ learned in Section 3.1 and the transition $p(\mathbf{e}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1})$ learned in Section 3.2.

4.2.2 Importance Function

We now consider the importance function. Its role is to generate the particles. It is possible to set $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ in order to propagate the particles using the previous transition. However, a more sophisticated design can drastically improve the estimation efficiency by reducing the variance of the particle weights [25], [28]. Thus, a good importance function should integrate both the transition and the likelihood, in order to get a support that includes the one of the posterior distribution. We propose to define the probability of jump as a function of the meaningfulness of an edgelet at $t-1$. The importance function also uses the edgelet trajectory to constrain the particle to move on unvisited pixels. Like the transition defined in the previous section, the jump propagation is considered first, then the edgelet at the coarse scale, and finally the one at the reference scale:

$$\begin{aligned} q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}) = {} & q(\mathrm{E}_t|\mathrm{E}_{0:t-1}, \mathbf{e}_{t-1}, \mathbf{c}_t, \mathrm{Y}) \\ & \times q(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1}, \mathbf{c}_t, \mathbf{y}) \\ & \times q(\mathbf{c}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1}, \mathbf{c}_{t-1}, \mathrm{Y}, \mathbf{y}). \end{aligned} \quad (12)$$

The distribution $q(\mathbf{c}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1}, \mathbf{c}_{t-1}, \mathrm{Y}, \mathbf{y})$ is set to $(1-\mathbf{c}_t)(1-\alpha) + \mathbf{c}_t\,\alpha$. The value α is the probability to jump to an unexplored contour, and depends on how meaningful the past edgelets were according to the J feature distributions:

$$\alpha = \frac{1}{2}\sum_{j=1}^{J} \tilde{\lambda}_{\mathrm{E}}^j\, P\big(\mu_{\mathrm{E}}^j > f_{\mathrm{E}}^j(\mathrm{E}_{t-1}, I_{\mathrm{E}})\big) + \tilde{\lambda}_e^j\, P\big(\mu_e^j > f_e^j(\mathbf{e}_{t-1}, I_e)\big), \quad (13)$$

with $\tilde{\lambda}_{\mathrm{E}}^j$ and $\tilde{\lambda}_e^j$ the normalized values of λ_{E}^j and λ_e^j of (6), respectively. The edgelet importance function at the coarse scale includes a trajectory constraint f_{check} as well as the transition and initialization distributions respectively learned in Sections 3.2 and 3.4:

$$\begin{aligned} q(\mathrm{E}_t|\mathrm{E}_{0:t-1}, \mathbf{e}_{t-1}, \mathbf{c}_t, \mathrm{Y}) = {} & n_{\mathrm{q}}^{-1}\; f_{\mathrm{check}}(\mathrm{E}_t, \mathrm{E}_{0:t-1}) \\ & \times [\mathbf{c}_t\, p(\mathrm{E}_t|\mathrm{Y}) + (1-\mathbf{c}_t)\, p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1})]. \end{aligned} \quad (14)$$

To avoid an echo detection effect, the trajectory constraint f_{check} is set to 0 if any point in the edgelet E_t is closer than a Manhattan distance of 1 with any point of the current contour (with exception of the edgelet at $t-1$) or is closer than a Manhattan distance of 2 with any point of the past contours. Otherwise $f_{\mathrm{check}}(\cdot)$ is set to 1. Generating particles according to this distribution can be done using a rejection sampling: a sample $\mathrm{E}_t^{(n)}$ is generated according to $\mathbf{c}_t^{(n)}\, p(\mathrm{E}_t|\mathrm{Y}) + (1-\mathbf{c}_t^{(n)})\, p(\mathrm{E}_t|\mathrm{E}_{t-1}^{(n)}, \mathbf{e}_{t-1}^{(n)})$ and accepted with a probability of $f_{\mathrm{check}}(\mathrm{E}_t^{(n)}, \mathrm{E}_{0:t-1}^{(n)})$. The normalizing constant n_q is approximated using an importance sampling method:

$$n_{\mathrm{q}} \approx \frac{1}{N_{\mathrm{q}}}\sum_{m=1}^{N_{\mathrm{q}}} f_{\mathrm{check}}\big(\mathrm{E}_t^{(m)}, \mathrm{E}_{0:t-1}\big), \quad (15)$$

with $\mathrm{E}_t^{(m)} \sim \mathbf{c}_t\, p(\mathrm{E}_t|\mathrm{Y}) + (1-\mathbf{c}_t)\, p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1})$ and N_{q} a small number of samples. Once the state of the coarse scale edgelet has been generated, it remains to propagate that of the reference scale edgelet. The edgelet importance function at the reference scale is then simply a mixture of an initialization distribution learned in Section 3.4 and the transition distribution learned in Section 3.2:

$$q(\mathbf{e}_t|\mathrm{E}_t,\mathbf{e}_{t-1},\mathbf{c}_t,\mathbf{y}) = \mathbf{c}_t\, p(\mathbf{e}_t|\mathrm{E}_t,\mathbf{y}) + (1-\mathbf{c}_t)\, p(\mathbf{e}_t|\mathrm{E}_t,\mathbf{e}_{t-1}). \quad (16)$$

4.2.3 Stopping Criterion

Although it is popular to perform tracking tasks, the particle filter does not embed a natural stopping procedure. In most of the cases, defining it is clearly not obvious. Here, we can take benefit of the feature tail distributions, which provide robust statistics of the data to be retrieved. We assume that the algorithm first tracks the most meaningful contours. This is due to the resampling technique used in the particle filters, which duplicates the good particles and discards the bad ones. Thus, because of the trajectory constraint f_{check} that compels the tracking to visit new contours, the detections become less and less meaningful. So, one could stop the algorithm after t steps, and in this case the stopping criterion would be related to the number of contour points in an image. However, besides not being adaptive to the image, this criterion would depend on the image dimensions. Therefore, we define the stopping criterion such that it depends on the meaningfulness of the last extracted contours. To this end, we defined in (13) the probability of jump using the meaningfulness of an edgelet, and this probability grows with time. Hence, we stop the detection when the proportion of jumps reaches a fixed threshold:

$$\frac{1}{KN}\sum_{k=1}^{K}\sum_{n=1}^{N}\mathbf{c}_{t-k+1}^{(n)} \geq \gamma. \quad (17)$$

Since it relies on the number of jump operated in the K last steps, this criterion also depends on the length of the retrieved contours, meaning that the repetition of small contour detections likely indicates that the detection is over.

4.2.4 Diversity

The resampling technique does not alter the posterior distribution but impacts on the diversity of the particles, especially for the past states. In practice, this means that most of the particles share the same trajectory, which may degrade the quality of the estimator. To alleviate this effect, we propose to divide the N particles into L independent particle filters, leading to the following final posterior distribution:

$$p(\mathbf{x}_{0:t}|\mathrm{Y},\mathbf{y}) = \frac{1}{L}\sum_{l=1}^{L} p(\mathbf{x}_{0:t,l}|\mathrm{Y},\mathbf{y}). \quad (18)$$

Each particle filter approximates the trajectory distribution using $N_{\mathrm{L}} = N/L$ particles. More elaborated techniques [29] aim at reducing the particle impoverishment effect, and at the cost of an increase of the algorithm computational complexity, might also be applied to our model.

4.2.5 Contour Detector

We consider here the contour detector at the reference scale, but the definition at the coarse scale can be similarly obtained. The soft contour detector is an image $O : \Omega_e \rightarrow [0,1]$, with $O(z)$ the confidence value that the pixel z belongs to a contour. This is computed by an average of the estimations given by the L particle filters:

$$\forall z \in \Omega_e, O(z) = \frac{1}{L}\sum_{l=1}^{L}\max_n w_{t_l,l}^{(n)}\, \mathbb{1}_{\mathbf{e}_{0:t_l,l}^{(n)}}(z), \quad (19)$$

with t_l the last step performed by the lth particle filter. An optional non-maximum suppression step may then be employed to produce thin contours [9], [30]. The final tracking procedure is given in Algorithm 7.

Algorithm 7: MS-PFCD Algorithm

Output: Particle filter contour detector map O

begin

Initialization *($t = 0$)*

for $l = 1,\ldots,L$ **do**

for $n = 1,\ldots,N_{\mathrm{L}}$ **do**

+ Generate $\mathrm{E}_{0,l}^{(n)} \sim p(\mathrm{E}_0|\mathrm{Y})$
+ Generate $\mathbf{e}_{0,l}^{(n)} \sim p(\mathbf{e}_0|\mathrm{E}_{0,l}^{(n)},\mathbf{y})$
+ Set $\mathbf{c}_{0,l}^{(n)} = 0$ and $w_{0,l}^{(n)} = 1/N_{\mathrm{L}}$

Tracking *(increment t)*

for $l = 1,\ldots,L$ **do**

- Coarse scale step: **for** $n = 1,\ldots,N_{\mathrm{L}}$ **do**

+ Generate $\mathbf{c}_{t,l}^{(n)} \sim q(\mathbf{c}_t|\mathbf{x}_{t-1,l}^{(n)},\mathrm{Y},\mathbf{y})$
+ Generate $\mathrm{E}_{t,l}^{(n)} \sim q(\mathrm{E}_t|\mathrm{E}_{0:t-1,l}^{(n)},\mathbf{e}_{t-1,l}^{(n)},\mathbf{c}_{t,l}^{(n)},\mathrm{Y})$
+ Compute and normalize the particle weights:

$$w_{t,l}^{(n)} \propto w_{t-1,l}^{(n)}\frac{p(\mathrm{Y}|\mathrm{E}_{t,l}^{(n)})}{q(\mathrm{E}_{t,l}^{(n)}|\mathrm{E}_{0:t-1,l}^{(n)},\mathbf{e}_{t-1,l}^{(n)},\mathbf{c}_{t,l}^{(n)},\mathrm{Y})}\times \frac{p(\mathrm{E}_{t,l}^{(n)}|\mathrm{E}_{t-1,l}^{(n)},\mathbf{e}_{t-1,l}^{(n)},\mathbf{c}_{t,l}^{(n)})\, p(\mathbf{c}_{t,l}^{(n)})}{q(\mathbf{c}_{t,l}^{(n)}|\mathrm{E}_{t-1,l}^{(n)},\mathbf{e}_{t-1,l}^{(n)},\mathbf{c}_{t-1,l}^{(n)},\mathrm{Y},\mathbf{y})}$$

- Resample $\left\{(\mathrm{E}_{0:t,l}^{(n)},\mathbf{e}_{0:t-1,l}^{(n)},\mathbf{c}_{0:t,l}^{(n)}), w_{t,l}^{(n)}\right\}_{n=1}^{N_{\mathrm{L}}}$
- Reference scale step: **for** $n = 1,\ldots,N_{\mathrm{L}}$ **do**

+ Generate $\mathbf{e}_{t,l}^{(n)} \sim q(\mathbf{e}_t|\mathrm{E}_{t,l}^{(n)},\mathbf{e}_{t-1,l}^{(n)},\mathbf{c}_{t,l}^{(n)},\mathbf{y})$
+ Compute and normalize the particle weights:

$$w_{t,l}^{(n)} \propto \frac{p(\mathbf{y}|\mathbf{e}_{t,l}^{(n)})\, p(\mathbf{e}_{t,l}^{(n)}|\mathrm{E}_{t,l}^{(n)},\mathbf{e}_{t-1,l}^{(n)},\mathbf{c}_{t,l}^{(n)})}{q(\mathbf{e}_{t,l}^{(n)}|\mathrm{E}_{t,l}^{(n)},\mathbf{e}_{t-1,l}^{(n)},\mathbf{c}_{t,l}^{(n)},\mathbf{y})}$$

- If needed, resample $\left\{(\mathrm{E}_{0:t,l}^{(n)},\mathbf{e}_{0:t,l}^{(n)},\mathbf{c}_{0:t,l}^{(n)}), w_{t,l}^{(n)}\right\}_{n=1}^{N_{\mathrm{L}}}$

Global Stopping Criterion

If all the particle filters have reached the stopping criterion γ, stop the algorithm, otherwise, continue the tracking for the unfinished filters

return $\forall z \in \Omega_e, O(z) = \frac{1}{L}\sum_{l=1}^{L}\max_n w_{t_l,l}^{(n)}\, \mathbb{1}_{\mathbf{e}_{0:t_l,l}^{(n)}}(z)$

5 Experiments

Experiments on contour detection have been conducted on the BSDS300 and the BSDS500 to compare the proposed approach to state-of-the-art contour detection methods [9], [10], [11], [12], [14], [15], [16], [17], [18], [22]. As stated in Section 2, only contour detection methods are presented here. Thus, we considered the contour detector gPb and not the segmentation method from the gPb, namely the gPb-owt-ucm [9]. In the same manner, we considered the edge detector tPb proposed in [22], instead of the boundary detector,

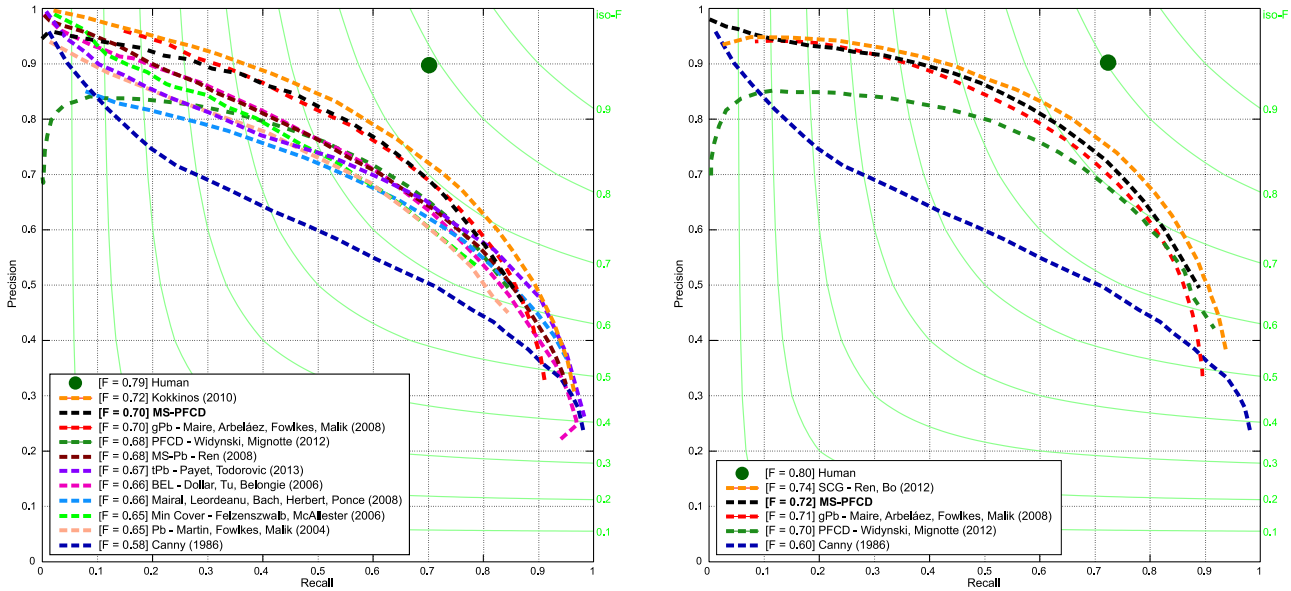

Fig. 3. Scores obtained by state-of-the-art methods and our proposed MultiScale Particle Filter Contour Detector model on (left) the BSDS300 and (right) the BSDS500. The evaluation is based on the F-Measure, which is a combination of the precision measure and the recall one: it supports a high number of true detections while it penalizes over-segmentation and missed detections. The final score is the optimal threshold computed among the 100 and 200 test images of the BSDS300 and BSDS500, respectively.

the tPb-SLEDGE, which achieved greater results by using a graph representation with a formulation of the Helmholtz principle to recover the missing edges.

5.1 Parameter Discussion

All the parameters have been learned using a gradient ascent on the F-measure on the training data set of the BSDS300.

The length of an edgelet at the coarse scale has been found optimal at $M_{\mathrm{E}} = 3$ (with a 4-connexity neighborhood). The number of samples in the learning procedures must be large enough to obtain a good approximation of the respective distributions, depending on the length of an edgelet. We set $S_p = 2 \times 10^6$ for the prior, $S_t = 10^5$ for the transition, $S_f = 10^6$ for the features and $S_{i_{\mathrm{E}}} = 1.5 \times 10^5$ and $S_{i_e} = 15$ for the initialization distributions. Results obtained with edgelets of greater lengths, i.e., $M_{\mathrm{E}} = 4$, and $M_{\mathrm{E}} = 5$, showed a slight decline in the F-measure scores, while requiring considerably more samples (S_p and S_t) to be estimated. This can be attributed to the increase of the dimensionality that imposes more particles to estimate the trajectory distribution.

For the observation model, we compute the textural gradient using a histogram of $R = 5 \times 5 \times 5 = 125$ bins. For this feature, the image is defined on the CIE Lab colorspace. The number of bins by orientation R_{m} for the oriented gradient feature is 10. In order to consider enough points to create the histograms, the length of each side of the normal segment is set to 11 pixels, with a line width of 5. For the profile gradient, the parameters $\sigma_{\mathrm{E}} = \sigma_e$ and $\kappa_{\mathrm{E}} = \kappa_e$ are respectively set to 0.7 and 1.6 and the profile is computed on a 5-pixel-long vector normal segment [24]. We use a mean for the fusion operators Ψ, Υ, and Ξ, and a min operator for Φ. The values of the feature multiplicative constants are set to $\lambda_{\mathrm{E}}^1 = 5$, $\lambda_{\mathrm{E}}^2 = 6$, $\lambda_{\mathrm{E}}^3 = 2$, and $\lambda_{\mathrm{E}}^4 = 16$ at the coarse scale, and set to $\lambda_e^1 = 5$, $\lambda_e^2 = 6$, $\lambda_e^3 = 3$, and $\lambda_e^4 = 16$ at the reference scale. The prior probability of jump β is set to 0.00015. The parameter $K = 200$ ensures the monotonical increase of the stopping criterion. Finally, a particle filter is stopped whenever its number of steps is greater than 150 and the proportion of its jumps γ reach 0.137.

We fix the total number of particles N to 5,625 to provide a good tradeoff between detection performance and computational cost. We set the number of particle filters L to 75 in order to smooth the results. Thus the number of particles by filter N_{L} is $5{,}625/75 = 75$, which is enough to obtain a satisfying accuracy of each particle filter. Note that N_{L} may impact on the stopping criteria: using a larger number of particles results in a slight reduction of γ, although it does not compensate for the additional computational cost. We approximate n_q using a small number of samples $N_q = 50$.

5.2 Results

As we can see in the precision-recall curves illustrated in Fig. 3, our MS-PFCD method performs well, with a F-Measure score at 0.70 (recall: 0.70, precision: 0.69) on the 100 test images of the BSDS300, and a F-Measure score at 0.72 (r: 0.73, p: 0.71) on the 200 test images of the BSDS500. Due to the stochastic nature of the algorithm, we performed the experiment 15 times and obtained a variance of 2.96×10^{-7}.

Full results are reported in Table 1, which presents three measures. Only the methods providing the three measures on either data set are reported. The first and most popular measure is the optimal data set scale (ODS) F-Measure score. It is obtained using the global optimal threshold on the data set. When no additional information is provided, this metric is simply referred to as the *F-Measure score*. The optimal image scale (OIS) is the F-Measure score obtained using the optimal threshold on each image. The last measure is the average precision (AP) and corresponds to the area under the precision-recall curves of Fig. 3. Both the

TABLE 1
Comparison of Results Obtained on the BSDS300 and the BSDS500

	BSDS300			BSDS500		
	ODS	OIS	AP	ODS	OIS	AP
Human	0.79	0.79	–	0.80	0.80	–
MS-PFCD	0.70	0.71	0.71	0.72	0.74	0.74
SCG [17]	0.71	–	–	0.74	0.76	0.77
gPb [9]	0.70	0.72	0.66	0.71	0.74	0.65
PFCD [7]	0.68	0.69	0.67	0.70	0.72	0.69
Canny [30]	0.58	0.62	0.58	0.60	0.63	0.58

ODS is the optimal scale on the data set, OIS the optimal scale on each image, and AP the average precision on the recall range.

TABLE 2
Single Comparisons of Contour Detection Methods on the BSDS300

	BSDS300							
	MS-PFCD	gPb	PFCD	MS-Pb	BEL	Min-. cover	Pb	All
MS-PFCD	–	47 (5)	56 (4)	53 (4)	60 (1)	60 (1)	60 (2)	49
gPb [9]	48 (5)	–	69 (6)	70 (13)	80 (8)	87 (7)	87 (2)	32
PFCD [7]	40 (4)	25 (6)	–	50 (4)	57 (6)	58 (5)	57 (7)	20
MS-Pb [11]	43 (4)	17 (13)	46 (4)	–	54 (9)	66 (20)	66 (11)	6
BEL [15]	39 (1)	12 (8)	37 (6)	37 (9)	–	57 (9)	55 (11)	9
Min-co. [18]	39 (1)	6 (7)	37 (5)	14 (20)	34 (9)	–	37 (24)	3
Pb [10]	38 (2)	11 (2)	36 (7)	23 (11)	34 (11)	39 (24)	–	5

Scores are compared based on the OIS criteria. Each row indicates the number of images that obtain better scores than the method in the corresponding column. Number of equalities are pointed out in parentheses. The left part of the table presents one-to-one comparisons whereas the column on the right presents a one-against-all comparison.

precision-recall curves and the quantitative table results show that our MS-PFCD performs well compared to state-of-the-art contour detection methods, while it compares favorably to the reference method, i.e., the gPb, on the BSDS500. The difference is especially visible for the AP measure in both data sets.

We also propose to compare the methods on single images based on the OIS scores in Table 2. Again, only the methods providing these data are reported. The upper part of the table compares the methods in a pairwise manner. We can observe that although the OIS score of the MS-PFCD method is slightly below the one of the gPb on the BSDS300 (Table 1), the two methods are almost even when compared side by side. This could suggest that these methods outperform each other for very different images. This interpretation seems consistent with the one-against-all comparison reported on the right side of Table 2 in which the MS-PFCD method obtains the best OIS result for nearly half the test data set. Fig. 4 illustrates a few contour detection results obtained by the proposed algorithm using the output at the reference scale defined in (19). From these, we can see that the method successfully ignores very textured regions of the image. This is especially visible in the socks pattern of the young lady and in the leopard marks. This ability may be partly attributed to the use of the tail distributions of the features, that tend to highlight only the most significant contours. This is also due to the problem formulation as a sequential model that gives more detection probability to long contours. However, these two particularities may also explain the bad result observed with the fish image. The contours are not well-defined and salient enough to be deemed more meaningful than the texture of the scales.

In order to evaluate the contribution of each feature, and of the multiscale framework, an ablation study has been conducted on the BSDS300. F-measure scores have been computed using re-optimized model parameters. Outputs without the local gradient f^1, the textural gradient f^2, the oriented gradient f^3 and the profile gradient f^4, obtained F-measure scores of 0.67, 0.68, 0.69, and 0.66, respectively. These results are consistent with the optimal likelihood parameters obtained using the four features, which particularly demonstrate the importance of the profile gradient. Moreover, differences between the PFCD and the MS-PFCD algorithms are that the former is a single scale algorithm which does not use the oriented gradient. So, by comparing results obtained by the PFCD and by the MS-PFCD without the oriented gradient feature, this indicates that considering two scales instead of one improved the F-measure scores from 0.68 to 0.69.

On average, our C++ code runs in 3 minutes and 30 seconds on a single-thread Intel Core i7-980X CPU for a single image whose dimensions at the reference scale are 481-by-321 pixels. This approximatively corresponds to 450 iterations of the algorithm. Also, our algorithm can easily be parallelized since both the samples in the online learning step and the particle filters during the tracking one are independent. To this end, a Cuda implementation is also available and processes an image in 0.75 second on a Nvidia GTX 670 GPU card. Codes are available at http://www.iro.umontreal.ca/~mignotte/mspfcd/.

6 Interactive Cut-Out

Interactive cut-out aims at allowing the user to extract a region of interest from an image. As mentioned in Section 1, Pérez et al. [4] proposed a Bayesian probabilistic approach, the *JetStream* algorithm, to sequentially track control points spaced by a fixed distance. The authors suggested to constrain the tracking using manually defined regions through which the contour algorithm must not go. Formulating interactive cut-out task as a tracking approach is not specific to particle filtering techniques, as it has in particular been proposed before using dynamic programming and shortest-path algorithms, see, e.g., [31]. However, compared to deterministic approaches, a probabilistic formulation often offer more flexibility to be able to handle various contour configurations. In this section, we propose to adapt our contour detection algorithm to this semi-supervised case. We also extend the human interaction possibilities from the *JetStream* algorithm by defining two new tools. The first one compels the algorithm to go through manually-defined regions. The other tool enables the extraction of a complete contour by defining a rough contour of fixed width manually. Interactive segmentation results using these three tools are illustrated in Fig. 5.

Adapting our MS-PFCD to interactive cut-out is straightforward. In such an application, we assume the initialization point or region given, from which edgelets $(\mathrm{E}_0, \mathbf{e}_0)$ are generated. Plus, the goal is to extract only one curve at a time. Thus, we remove the jump variable $\mathbf{c}_t$ from the state $\mathbf{x}_t$. This means that, at the coarse scale, particles are propagated using the importance function:

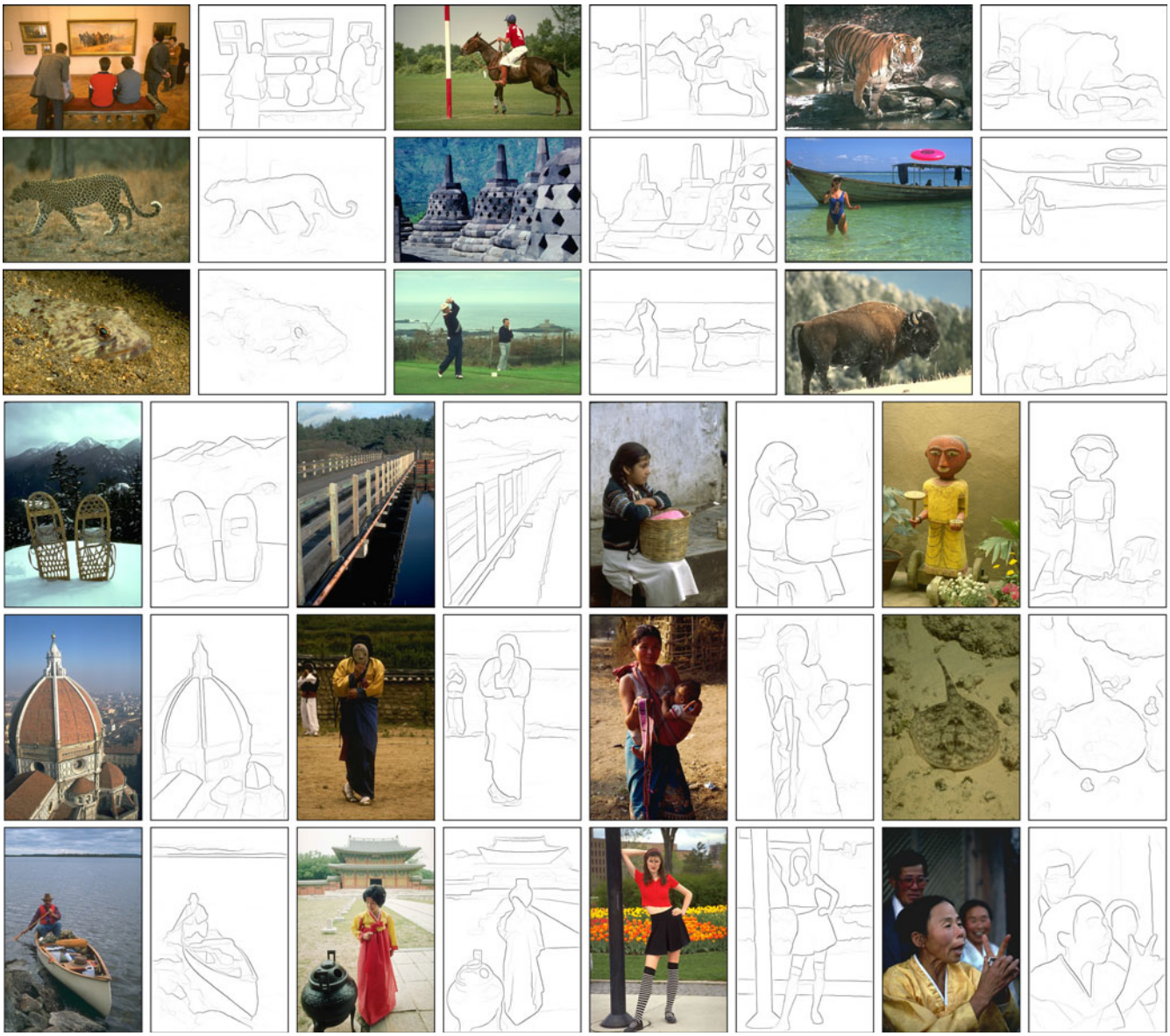

Fig. 4. Examples of contour detections obtained by the proposed MS-PFCD algorithm at the reference scale.

Fig. 5. Interactive cut-out experiments. Starting points are colored in purple. (Left) Forbidden points region tool. The contour detector runs without any interaction, unless the algorithm goes wrong, in which case forbidden points regions are drawn (in red). (Middle) Control points region tool. The procedure is iterative, each green dot defines a control points region, meaning that the algorithm must go through the next one. The number of interactions illustrated here is not necessarily minimal. (Right) Rough contour tool. The user roughly draws a contour. Then, the tracking algorithm extracts the contour by locally following the ordered path.

$$q(\mathrm{E}_t|\mathrm{E}_{0:t-1}, \mathbf{e}_{t-1}, \mathbf{y}) = n_{\mathrm{q}}^{-1}\, f_{\mathrm{check}}(\mathrm{E}_t, \mathrm{E}_{0:t-1})\, p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1}).$$

Transitions $p(\mathrm{E}_t|\mathrm{E}_{t-1}, \mathbf{e}_{t-1})$ and $p(\mathbf{e}_t|\mathrm{E}_t, \mathbf{e}_{t-1})$ now simply correspond to those learned in Section 3.2. This also implies that the initialization distribution and the stopping criterion become useless. Besides, we use only one particle filter with $N = 200$ particles. Since the initialization distribution is removed from this application, the number of samples used for the feature distribution learning may be drastically reduced, e.g., $S_f = 5{,}000$. We get a practical interactive contour detector, with an efficient learning stage, and a detection stage computational time that depends on the complexity of the constraint. The displayed contour corresponds to the particle that obtained the maximum weight before resampling.

6.1 Forbidden Points Region

This tool has been proposed with the *JetStream* algorithm [4]. The methodology consists in multiplying the likelihood by a very low constant value to particles going through a forbidden points region. It is preferable to multiply by a low value (e.g., 10^{-7}) instead of 0 to allow the algorithm to escape from a forbidden region in the unfortunate case where all the particles have been propagated in there. This tool is well adapted to particle filters due to the stochastic nature of the algorithm which tends to propose alternative paths in order to avoid a forbidden points region. The extraction is automatically stopped and contours are closed when current edgelets are found very close to the initial point.

6.2 Control Points Region

Besides the forbidden points region tool, it would also be useful to define a "must go through" tool. Such a zone is called a control points region. In [4], it is suggested, but not experimentally validated, to multiply the likelihood by a very high constant value when particles go through the control points region. This technique might be inefficient since it does not force the algorithm to go through the region of interest. Moreover, when it does, a stuck side effect may happen. Hence, we propose to proceed differently. At a certain time t of the contour extraction algorithm, if the user sets a new control points region, the algorithm generates trajectories using several particle filters until one of them reaches the region of interest. The maximal length of the generated paths is related to the Manhattan distance from the edgelet at t to the closest control point (e.g., twice this distance). The success of this technique depends on the control points region localization and size: if it is far from the edgelet at t or if it is small, it may take some time before a particle filter reaches the region area. For this tool, as well as the next one, closing the contours is handled explicitly by the user, as we can see in Fig. 5.

6.3 Rough Contour

We finally propose one last tool that is a combination of the two previous ones. The user manually defines a rough contour and the algorithm aims at extracting an accurate contour path from it. First, all the pixels that do not belong to the rough contour are considered forbidden. Then, taking advantage of the user interaction, we consider that the rough contour path is ordered and follows the true contour. Thus, intermediate control points regions are automatically spread at regular intervals within the rough contour.

Using the rough contour tool has several advantages. First, the extraction process is efficient since the space problem is constrained. Second, it is certainly more convenient for the user since, unlike the first two tools that often result in a trial-and-error procedure, this one only needs one simple fast interaction. This is especially obvious when comparing the number of user interactions in Fig. 5. Finally, by adjusting the rough contour pencil size, the user can balance between the burden of defining a quite precise contour and the accuracy of the contour extraction. However, drawing a rough contour may be fastidious when dealing with large images (e.g., satellite images), in which case one of the other tools, or the combination of the two, may be more appropriate.

7 Conclusion

We proposed a multiscale particle filter approach to track contours in complex natural images. The basic element of our model is a pair of edgelets, i.e., sets of connected pixels defined at two scales, that naturally embeds semi-local information. The underlying Bayesian model involves multiscale prior and transition distributions, which are learned on a shape database, and a multiscale likelihood component, which is adaptive to an image in order to retrieve only the most relevant contours. Experiments have been conducted on the Berkeley data sets and the proposed approach obtained competitive results with the state-of-the-art. We also extended our model for the interactive cut-out task. Qualitative results and reduced computational cost make of this method a practical tool.

Possible improvements of this work include: more elaborated features, in agreement with other state-of-the-art methods; multiscale parametric edgelet distributions, that would possess the advantage of being independent from the image dimensions; and finally a more sophisticated strategy to maintain an acceptable particle cloud diversity, in this case recent advances in the particle filtering literature might help.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments that greatly contributed to improve this paper. The first author is also grateful to Michäel Cadilhac and David Lesage for preliminary discussions on this work.

References

[1] B. Wu and R. Nevatia, "Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors," *Proc. IEEE 10th Int'l Conf. Computer Vision (ICCV)*, vol. 1, pp. 90-97, 2005.

[2] N. Widynski and M. Mignotte, "A Contrario Edge Detection with Edgelets," *Proc. IEEE Int'l Conf. Signal and Image Processing Applications (ICSIPA)*, pp. 421-426, 2011.

[3] A. Desolneux, L. Moisan, and J. Morel, "Edge Detection by Helmholtz Principle," *J. Math. Imaging and Vision*, vol. 14, no. 3, pp. 271-284, 2001.

[4] P. Pérez, A. Blake, and M. Gangnet, "Jetstream: Probabilistic Contour Extraction with Particles," *Proc. IEEE Eighth Int'l Conf. Computer Vision (ICCV)*, vol. 2, pp. 524-531, 2001.

[5] C. Florin, N. Paragios, and J. Williams, "Globally Optimal Active Contours, Sequential Monte Carlo and On-Line Learning for Vessel Segmentation," *Proc. Ninth European Conf. Computer Vision (ECCV)*, pp. 476-489, 2006.
[6] D. Lesage, E.D. Angelini, I. Bloch, and G. Funka-Lea, "Medial-Based Bayesian Tracking for Vascular Segmentation: Application to Coronary Arteries in 3D CT Angiography," *Proc. IEEE Fifth Int'l Symp. Biomedical Imaging: From Nano to Micro (ISBI)*, pp. 268-271, 2008.
[7] N. Widynski and M. Mignotte, "A Particle Filter Framework for Contour Detection," *Proc. 12th European Conf. Computer Vision (ECCV)*, pp. 780-794, 2012.
[8] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," *Proc. IEEE Eighth Int'l Conf. Computer Vision (ICCV)*, vol. 2, pp. 416-423, July 2001.
[9] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898-916, May 2011.
[10] D. Martin, C. Fowlkes, and J. Malik, "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530-549, May 2004.
[11] X. Ren, "Multi-Scale Improves Boundary Detection in Natural Images," *Proc. 10th European Conf. Computer Vision (ECCV)*, pp. 533-545, Jan. 2008.
[12] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik, "Using Contours to Detect and Localize Junctions in Natural Images," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1-8, 2008.
[13] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888-905, Aug. 2000.
[14] I. Kokkinos, "Boundary Detection Using F-Measure-, Filter-And Feature-(F3) Boost," *Proc. 11th European Conf. Computer Vision (ECCV)*, pp. 650-663, 2010.
[15] P. Dollar, Z. Tu, and S. Belongie, "Supervised Learning of Edges and Object Boundaries," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 1964-1971, 2006.
[16] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce, "Discriminative Sparse Image Models for Class-Specific Edge Detection and Image Interpretation," *Proc. 10th European Conf. Computer Vision (ECCV)*, pp. 43-56, 2008.
[17] X. Ren and L. Bo, "Discriminatively Trained Sparse Code Gradients for Contour Detection," *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 593-601, 2012.
[18] P. Felzenszwalb and D. McAllester, "A Min-Cover Approach for Finding Salient Curves," *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop (CVPRW)*, pp. 185-185, 2006.
[19] S. Konishi, A. Yuille, J. Coughlan, and S. Zhu, "Statistical Edge Detection: Learning and Evaluating Edge Cues," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 57-74, Jan. 2003.
[20] F. Destrempes and M. Mignotte, "A Statistical Model for Contours in Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 626-638, May 2004.
[21] F. Destrempes, M. Mignotte, and J. Angers, "Localization of Shapes Using Statistical Models and Stochastic Optimization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1603-1615, Sept. 2007.
[22] N. Payet and S. Todorovic, "SLEDGE: Sequential Labeling of Image Edges for Boundary Detection," *Int'l J. Computer Vision*, vol. 104, no. 1, pp. 15-37, 2013.
[23] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886-893, 2005.
[24] J. Sun, Z. Xu, and H.-Y. Shum, "Gradient Profile Prior and Its Applications in Image Super-Resolution and Enhancement," *IEEE Trans. Image Processing*, vol. 20, no. 6, pp. 1529-1542, June 2011.
[25] *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. De Freitas, and N. Gordon, eds., Springer, 2001.
[26] J. MacCormick and M. Isard, "Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking," *Proc. Sixth European Conf. Computer Vision (ECCV)*, pp. 3-19, 2000.
[27] J. MacCormick and A. Blake, "A Probabilistic Exclusion Principle for Tracking Multiple Objects," *Int'l J. Computer Vision*, vol. 39, no. 1, pp. 57-71, 2000.
[28] Z. Chen, "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond," technical report, McMaster Univ., 2003..
[29] A. Doucet, M. Briers, and S. Sénécal, "Efficient Block Sampling Strategies for Sequential Monte Carlo Methods," *J. Computational and Graphical Statistics*, vol. 15, no. 3, pp. 693-711, 2006.
[30] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.
[31] E.N. Mortensen and W.A. Barrett, "Intelligent Scissors for Image Composition," *Proc. ACM SIGGRAPH*, pp. 191-198, 1995.

Nicolas Widynski received the engineering degree from the Ecole pour l'Informatique et les Techniques Avancées and the image processing master's degree from the University Pierre and Marie Curie, both in Paris, France, in 2007. He received the PhD degree from Télécom ParisTech and Laboratoire d'Informatique de Paris 6 in 2010. He was a postdoctoral fellow at the Département d'Informatique et de Recherche Opérationnelle (DIRO), University of Montreal, Quebec, Canada, from 2011 to 2013. He is currently a postdoctoral fellow at the Centre de Recherche du Centre Hospitalier de l'Université de Montréal (CRCHUM), Quebec, Canada.

Max Mignotte received the DEA (postgraduate degree) in digital signal, image, and speech processing from the Institut National Polytechnique de Grenoble, France, Grenoble, in 1993 and the PhD degree in electronics and computer engineering from the University of Bretagne Occidentale, Brest, France, and the Digital Signal Laboratory (GTS), French Naval Academy, France, in 1998. He was a postdoctoral fellow in the Département d'Informatique et de Recherche Opérationnelle (DIRO), University of Montreal, Quebec, Canada, from 1998 to 1999, where he is currently with the Computer Vision and Geometric Modeling Laboratory as an associate professor. His current research interests include statistical methods, Bayesian inference, and hierarchical models for high-dimensional inverse problems.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.