A Particle Filter Framework for Contour Detection

Nicolas Widynski and Max Mignotte

Department of Computer Science and Operations Research (DIRO), University of Montreal,
C.P. 6128, succ. Centre-Ville, , Montreal (Quebec), H3C 3J7, Canada
{widynski,mignotte}@iro.umontreal.ca

Abstract. We investigate the contour detection task in complex natural images. We propose a novel contour detection algorithm which locally tracks small pieces of edges called edgelets. The combination of the Bayesian modeling and the edgelets enables the use of semi-local prior information and image-dependent likelihoods. We use a mixed offline and online learning strategy to detect the most relevant edgelets. The detection problem is then modeled as a sequential Bayesian tracking task, estimated using a particle filtering technique. Experiments on the Berkeley Segmentation Datasets show that the proposed Particle Filter Contour Detector method performs well compared to competing state-of-the-art methods.

1 Introduction

The contour detection task is an important issue in the field of image processing. Besides the resolution and the presence of noise and clutter, the intrinsic variability of natural images makes the detection a proper challenge. In the last decade, Martin et al. [1] have introduced the Berkeley Segmentation Dataset (BSDS300) in which 300 natural color images have been manually segmented by several contributors. For each image of the BSDS300, a set of handmade benchmark contour detection images is available and is used to quantify the reliability of the algorithm. The dataset is divided into a training set of 200 images and a test set of 100 images. This enables the standardization of the evaluation of contour detection techniques. The authors of the dataset proposed in [2] to combine local brightness, color, and texture cues in a learning logistic regression classifier procedure. Ren [3] first improved this approach by handling these local features at different scales. Maire et al. [4] and Arbeláez et al. [5] proposed to couple the use of these local features with global information obtained from spectral partitioning. This significantly improved the results such that they obtain the best ones to date. Dollar et al. [6] used a boosted edge learning algorithm to combine a large number of features accross different scales to learn a discriminative edge classifier. In [7], Mairal et al. applied a multiscale framework based on learned sparse representations to the edge detection of class-specific objects. Ren et al. [8] proposed a curvilinear continuity stochastic approach by modeling piecewise linear approximations of contours and employed a constrained Delaunay triangulation which tends to fill the gaps between the detections. Felzenszwalb et al. [9] tracked salient smooth curves by approximating the weighted min-cover problem. All these methods were evaluated on the BSDS300 and the results are pictured in Fig. 1.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part I, LNCS 7572, pp. 780–793, 2012.

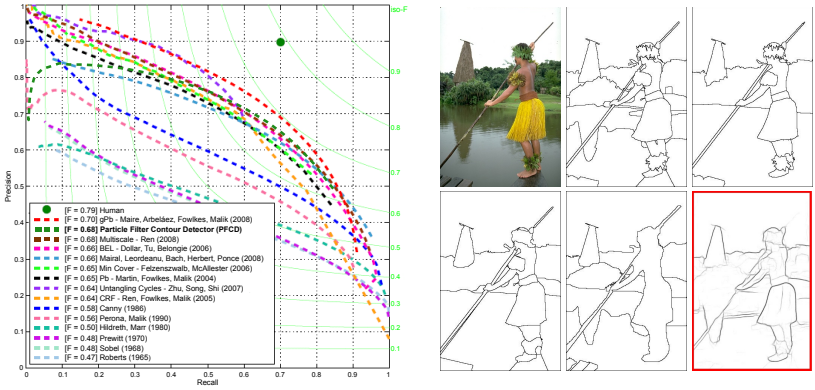

Fig. 1. (Left). Scores obtained by state-of-the art methods and our proposed Particle Filter Contour Detector (PFCD) model (illustration adapted from [5]). The evaluation is based on the F-Measure, which is a combination of the precision measure and the recall one: it supports a high number of true detections while it penalizes over-segmentation and missed detections. The final score is the optimal threshold computed among the 100 test images. (Right). Image from the BSDS300 and ground truth boundaries. The image outlined in red is obtained by our PFCD.

In this paper, we propose the use of a particle filtering technique to detect contours in natural images. The use of a particle filter to detect contours has been first proposed by Pérez et al. [10] with their well-known *JetStream* algorithm. This method aims at extracting only one curve from an image by locally tracking points at a fixed step length. The likelihood is also pixel-wise. The authors proposed a semi-automatic routine to extract complex contours by constraining the contour path using manually set forbidden regions. The power of this approach in interactive segmentation applications is undeniable. However, the method can hardly be applied to the challenging problem of automatic contour detection in the BSDS300. Similar methods have then been proposed [11–13], but are based on the *JetStream* framework, and hence dedicated to single extraction tasks.

At this point, a fair question would be: why should we use a particle filter to detect contours? To answer this question, we need to consider the motivations of the present work. The basic idea of our work is to adopt a semi-local strategy to detect small sets of connected edgels which we call edgelets. Their shapes are learned as a prior distribution using the BSDS300 learning dataset. The distribution automatically embeds every contour variation, without imposing any mathematical model. For example, with an edgelet length of 7, vertical and horizontal segments represent 18% of the shape database. In the detection algorithm, we need to evaluate the saliency of each possible edgelet contained in the image. In a probabilistic modeling, this may be done by considering a likelihood distribution. We propose to combine three features: the local gradient, its profile, and a texture gradient. These features are semi-local since they are computed on edgelets. We could stop our detection algorithm here, by estimating a Bayesian posterior distribution.

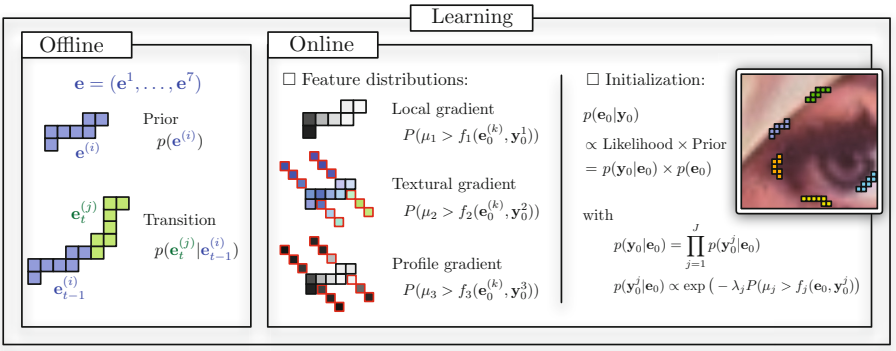

Fig. 2. The learning procedure is divided in two steps. The offline step estimates the prior and the transition distributions, which are used to generate samples in the contour tracking procedure. The online step is performed on the image to be tracked, and aims at learning: the feature distributions, in order to recognize the meaningful contours in the image; and the initialization distribution, in order to (re-)initialize the tracking process in the contour detection procedure.

However, this would generate detection artefacts due to a lack of consistency between detected edgelets. Thus we propose to define an artificial temporal bound between the edgelets, similar to what is done in a classical Markovian relaxation modeling. Once again, we use the BSDS300 to learn the dynamics between two consecutive edgelets. Hence, due to its random and evolutive nature, we define an edgelet as a stochastic process. It turns out that estimating this sequential Bayesian model may be efficiently done using a particle filtering technique, leading to our Particle Filter Contour Detector (PFCD) method.

This paper is organized as follows. In Sect. 2, we propose to learn the distributions that define our sequential Bayesian model. The tracking algorithm for contour detection based on a particle filtering technique is then described in Sect. 3. We show experimental results on the BSDS300 and BSDS500 [5] in Sect. 4, before concluding in Sect. 5.

2 Learning the Bayesian Model

Let $\mathbf{e} = (\mathbf{e}^1, \ldots, \mathbf{e}^M) \in \Omega^M$ be a set of M 4-connected points. Each point $\mathbf{e}^i$ is defined in the image domain Ω. The number M is fixed and is a parameter of the method. For exemple, in our experiments we set $M = 7$ to balance the computational cost and the detection robustness. The vector $\mathbf{e}$ is henceforth refered to an *edgelet*. The proposed contour detector is a tracking based on approach. This means that we want to define an edgelet at a certain step, or time, of the tracking procedure. Then $\mathbf{e}$ is indexed by time, t, and can be defined as a stochastic process, $\{\mathbf{e}_t\}_{t\in\mathbb{N}}$. We also need to introduce $\mathbf{y}_t \in \mathcal{Y}$, the measurement state. Our Bayesian tracking procedure requires the definition of four distributions: the prior $p(\mathbf{e})$, the transition $p(\mathbf{e}_t|\mathbf{e}_{t-1})$, the likelihood $p(\mathbf{y}_t|\mathbf{e}_t)$, and the initialization $p(\mathbf{e}_0|\mathbf{y}_0)$. They are illustrated on Fig. 2 and are the subject of the following sections.

Algorithm 1. Approximation of the prior distribution of an edgelet

Input: A shape database
Output: Approximation of the prior distribution $p(\mathbf{e}^{2:M}|\mathbf{e}^1)$
begin
  for $s = 1, \ldots, S_\mathrm{p}$ **do**
    Select an image I and a segmentation H at random
    Extract an edgelet $\mathbf{e}^{(s)}$ from (I, H) at random
    Center it with respect to $\mathbf{e}^{1(s)}$
return $\frac{1}{S_\mathrm{p}} \sum_{s=1}^{S_\mathrm{p}} \delta_{\mathbf{e}^{(s)}}^{\mathbf{e}}$, *where* $\delta_a^b = 1$ *if* $a = b$, *and* 0 *otherwise*

Algorithm 2. Approximation of the transition distribution of an edgelet

Input: Distribution $p(\mathbf{e})$, a shape database
Output: Approximation of the transition distribution $p(\mathbf{e}_t|\mathbf{e}_{t-1})$
begin
  foreach *distinct element* $\mathbf{e}'^{(u)} \in \{\mathbf{e}'^{(1)}, \ldots, \mathbf{e}'^{(S'_\mathrm{p})}\} \subset \{\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(S_\mathrm{p})}\}$ **do**
    repeat
      Select an image I and a segmentation H at random
      Extract two consecutives edgelets from (I, H) such that $\mathbf{e}_{t-1} = \mathbf{e}'^{(u)}$ and
        $\mathbf{e}_t = \mathbf{e}'^{(v)} \in \{\mathbf{e}'^{(1)}, \ldots, \mathbf{e}'^{(S'_\mathrm{p})}\}$ its successor
      Increment by $1/S_\mathrm{t}$ the probability $p(\mathbf{e}_t = \mathbf{e}'^{(v)}|\mathbf{e}_{t-1} = \mathbf{e}'^{(u)})$
    until S_t *times*

2.1 Offline Learning: Prior Model

The vector $\mathbf{e}$ is a small piece of a contour, integrating more information than a classical pixel-wise formulation. Nevertheless, it remains semi-local, in order to be applied generally to most of the contours. By learning its prior distribution, we avoid imposing mathematical constraints that may decrease the power of detection of an algorithm, since it is impractical to define a mathematical model that captures every possible contour singularity. Algorithm 1 presents the offline approximation procedure of the prior distribution $p(\mathbf{e})$. The parameter S_p denotes the number of samples and $\mathbf{e}^{(s)}$ is the s-th realization of the approximation set. Each sample $\mathbf{e}^{(s)}$ is centered with respect to its first point $\mathbf{e}^{1(s)}$. Using the BSDS300 training dataset, we learn the distribution $p(\mathbf{e}^{2:M}|\mathbf{e}^1)$ to capture only the shapes of the edgelets.

2.2 Offline Learning: Transition Model

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 possible edgelet shape at a certain time t given the previous one at $t-1$. This is what the transition distribution $p(\mathbf{e}_t|\mathbf{e}_{t-1})$ is designed to. It can be approximated using Algorithm 2. We use the same shape dataset as in Sect. 2.1.

2.3 Online Learning: Observation Model

In this section, we define a density $p(\mathbf{y}_t|\mathbf{e}_t)$ which measures the adequation between data known at a time t, $\mathbf{y}_t$, and an edgelet $\mathbf{e}_t$. To make our detector robust, we first consider several observations, i.e. $\mathbf{y}_t = (\mathbf{y}_t^1, \dots, \mathbf{y}_t^J)$, with $\mathbf{y}_t^j \in \mathcal{Y}_j$, each of these being related to a special feature. The joint likelihood $p(\mathbf{y}_t|\mathbf{e}_t)$ is defined considering a conditional independence hypothesis of the $\mathbf{y}_t^j, 1 \leq j \leq J$, given $\mathbf{e}_t$:

$$p(\mathbf{y}_t|\mathbf{e}_t) = \prod_{j=1}^{J} p(\mathbf{y}_t^j|\mathbf{e}_t) \ . \tag{1}$$

This simplifies the estimation since we just need to define the marginal likelihoods $p(\mathbf{y}_t^j|\mathbf{e}_t)$. The observation $\mathbf{y}_t^j$ is related to a feature $f_j \colon \Omega^M \times \mathcal{Y}_j \to \mathbb{R}$. The features $\{f_j\}_{j=1}^J$ are computed along an edgelet $\mathbf{e}_t$ and integrate color and gradient information to precisely localize contours.

Before defining these features, we propose a general formulation of the densities $p(\mathbf{y}_t^j|\mathbf{e}_t)$. It is clear that the quantity of information of each feature depends on the image itself. For example, a classical gradient feature typically overdetects in textured images whereas its sensitivity drops in blurry ones. The interpretation of the feature responses should be different in these two cases, meaning that a candidate should be more relevant when it obtains a singular high feature response value in the image. This idea has been used in an *a contrario* framework [14], and is related to the notion of *meaningfulness* of an event. Here, an event is an edgelet $\mathbf{e}_t$ and we want to compute how *meaningful* the feature response on $\mathbf{e}_t$ is on the image. This implies to learn a distribution $P\big(\mu_j > f_j(\mathbf{e}_t, \mathbf{y}_t^j)\big)$, with μ_j a random variable associated to the feature f_j, in order to consider the shape of the feature response distribution into the likelihood. This distribution can be assimilated as a distribution of false alarms. Hence, the lower the probability $P\big(\mu_j > f_j(\mathbf{e}_t, \mathbf{y}_t^j)\big)$, the more meaningful this event, i.e., the less likely the event $\mathbf{e}_t$ corresponds to a false alarm. The approximation of the feature distribution is given in Algorithm 3. Finally, we define the likelihood in a way to support low values of $P\big(\mu_j > f_j(\mathbf{e}_t, \mathbf{y}_t^j)\big)$:

$$p(\mathbf{y}_t^j|\mathbf{e}_t) \propto \exp\Big(-\lambda_j \, P\big(\mu_j > f_j(\mathbf{e}_t, \mathbf{y}_t^j)\big)\Big) \ , \tag{2}$$

with $\lambda_j \in \mathbb{R}^+$ a learned multiplicative constant value which both aims at providing a good localization of high likelihood and impacts the relevance of the feature j on the tracking procedure.

We now describe our three features: f_1 the local gradient, f_2 the textural gradient, and f_3 the profile gradient. 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 less dependent on noise.

Local Gradient. This classical feature uses the 2×2 gradient norm $|\nabla I|$ of the image I. The gradient feature f_1 is computed along an edgelet $\mathbf{e}_t$:

$$f_1(\mathbf{e}_t, \mathbf{y}_t^1) = \Phi\Big(\Big(\big|\nabla I \left(\mathbf{e}_t^i\right)\big| \Big)_{1\leq i\leq M}\Big) \ . \tag{3}$$

Algorithm 3. Approximation of the feature distribution

Input: Distribution $p(\mathbf{e})$, an image $I\colon \Omega \to \mathbb{R}$
Output: Approximation of the feature distribution $P\big(\mu_j > f_j(\mathbf{e}_0, \mathbf{y}_0^j)\big)$
begin
  for $s = 1, \ldots, S_\mathrm{f}$ **do**
    Select a starting point $\mathbf{e}_0^{1(s)}$ at random: $\mathbf{e}_0^{1(s)} \sim \mathcal{U}[\Omega]$
    Generate the edgelet shape $\mathbf{e}_0^{2:M(s)}$ according to the prior distribution:
      $\mathbf{e}_0^{2:M(s)} | \mathbf{e}_0^{1(s)} \sim p(\mathbf{e}^{2:M} | \mathbf{e}^1)$
    Compute and store $f_j(\mathbf{e}_0^{(s)}, \mathbf{y}_0^j)$
return $\frac{1}{S_\mathrm{f}} \sum_{s=1}^{S_\mathrm{f}} \mathbb{1}_{\left[-\infty, f_j(\mathbf{e}_0^{(s)}, \mathbf{y}_0^j)\right]}\big(f_j(\mathbf{e}_0, \mathbf{y}_0^j)\big)$

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 $W\colon \{1, \ldots, M\} \to [0, 1]$ a weighting function. In our experiments, we set $W(i) = 1/M, \forall i$. Note that when the image I is multidimensionnal, we take on each point the maximum gradient value among the different channels.

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 color pixel values along the first segment should not really differ from the ones of the second segment. Let $h[a] = \{h^r[a]\}_{r=1}^{R}$ be the histogram of a set of pixels a, where r is the bin index of an histogram of length R. Distances between pairs of histograms along normals of the curve are combined to form the texture feature:

$$f_2(\mathbf{e}_t, \mathbf{y}_t^2) = \Psi\Big(\Big(d_\mathrm{B}\big(h[\overleftarrow{n}(\mathbf{e}_t^i)], h[\overrightarrow{n}(\mathbf{e}_t^i)]\big)\Big)_{2 \le i \le M-1}\Big) \ , \tag{4}$$

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 $1 - \sum_{r=1}^{R} \sqrt{h^r[a]\, h^r[b]}$.

Profile Gradient. The last feature comes from a recent idea from Sun et al. [15]. They propose to analyse the gradient profile for image enhancement and super-resolution. This profile is learned and represented by a parametric gradient profile model. This model does not use the gradient value of a contour, but the symmetry and the monotonicity of its profile. Thus, in our application, we expect that this feature detects sharp contours as well as smooth ones. It can also be useful when there are instabilities in the gradient norm, due to noise, while the symmetry and the monotony of its profile should remain. 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:

$$f_3(\mathbf{e}_t, \mathbf{y}_t^3) = \Xi\Big(\Big(-d_\mathrm{KL}\big(\big|\nabla \tilde{I}\big[n(\mathbf{e}_t^i)\big]\big|, g_\mathrm{p}(n(\mathbf{e}_t^i) | \mathbf{e}_t^i, \sigma_\mathrm{p}, \lambda_\mathrm{p})\big)\Big)_{2 \le i \le M-1}\Big) \ , \tag{5}$$

Algorithm 4. Approximation of the initialization distribution

Input: Distributions $P\big(\mu_j > f_j(\mathbf{e}_0, \mathbf{y}_0^j)\big)$, $p(\mathbf{e}_0)$, an image I
Output: Approximation of the initialization distribution $p(\mathbf{e}_0|\mathbf{y}_0)$
begin
  for $s = 1, \ldots, S_\mathrm{i}$ **do**
    Select a starting point $\mathbf{e}_0^{1(s)}$ at random: $\mathbf{e}_0^{1(s)} \sim \mathcal{U}[\Omega]$
    Generate the edgelet shape $\mathbf{e}_0^{2:M(s)}$ according to the prior distribution:
      $\mathbf{e}_0^{2:M(s)}|\mathbf{e}_0^{1(s)} \sim p(\mathbf{e}_0^{2:M}|\mathbf{e}_0^1)$
    Compute the joint likelihood $w(\mathbf{e}_0^{(s)})$:
      $w(\mathbf{e}_0^{(s)}) \propto \prod_{j=1}^{J} \exp(\lambda_j\, P\big(\mu_j > f_j(\mathbf{e}_0^{(s)}, \mathbf{y}_0^j)\big))$ s.t. $\sum_{u=1}^{S_\mathrm{i}} w(\mathbf{e}_0^{(u)}) = 1$
return $\frac{1}{S_\mathrm{i}} \sum_{s=1}^{S_\mathrm{i}} w(\mathbf{e}_0^{(s)})\, \delta_{\mathbf{e}_0^{(s)}}^{\mathbf{e}_0}$

with $\big|\nabla \tilde{I}\big[n(\mathbf{e}_t^i)\big]\big|$ a vector of normalized absolute gradient values computed along the normal $n(\mathbf{e}_t^i)$, and $g_\mathrm{p}(n(\mathbf{e}_t^i)|\mathbf{e}_t^i, \sigma_\mathrm{p}, \lambda_\mathrm{p})$ a vector of generalized exponential distribution values computed along the normal $n(\mathbf{e}_t^i)$ with respect to the center point $\mathbf{e}_t^i$. The parameters σ_p and λ_p control the shape of the distribution g_p. We use the Kullback-Leibler divergence d_KL to measure the difference between two discrete distributions of L elements, $d_\mathrm{KL}(p, q) = \sum_{l=1}^{L} p^l \log p^l/q^l$, with p^l and q^l the probabilities at point l.

2.4 Online Learning: Initialization Model

We finally estimate an initialization distribution $p(\mathbf{e}_0|\mathbf{y}_0)$. This distribution finds its 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 process on a novel contour.

Although it is possible to carry out these operations using the prior distribution $p(\mathbf{e}_0)$, the observations $\mathbf{y}_0$ are more likely to provide samples located on true contours. The approximation procedure is described in Algorithm 4.

3 Contour Detection by Tracking Based on Particle Filter

We defined in Sect. 2 several distributions that manipulate a piece of contour, namely an edgelet $\mathbf{e}_t$, of length M. In this section, we define the framework that handle these distributions by integrating them in a sequential Monte Carlo approach. Our goal is to estimate the distribution of the edgelets $\mathbf{e}_{0:t}$ conditioned by a set of observations $\mathbf{y}_{0:t}$. Hence, at a time t, $\mathbf{e}_{0:t}$ defines a contour map of $t+1$ edgelet elements. The estimation of this posterior distribution is recursively done by approximating the *filtering distribution*, $p(\mathbf{e}_t|\mathbf{y}_{0:t})$. This is the subject of the following section.

3.1 Estimating the Filtering Distribution

Let $\mathbf{x}_t \in \mathcal{X}$ be the hidden state of a stochastic process at time t and $\mathbf{y}_t \in \mathcal{Y}$ be the measurement state. Under the Markovian hypothesis of the hidden states and the

conditional independence hypothesis of the observations given the states, the classical filtering problem aims at estimating the posterior distribution $p(\mathbf{x}_t|\mathbf{y}_{0:t})$:

$$p(\mathbf{x}_t|\mathbf{y}_{0:t-1}) = \int_{\mathcal{X}} p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1})\, d\mathbf{x}_{t-1} \ , \tag{6}$$

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{y}_{0:t-1})}{\int_{\mathcal{X}} p(\mathbf{y}_t|\mathbf{x}'_t)\, p(\mathbf{x}'_t|\mathbf{y}_{0:t-1})\, d\mathbf{x}'_t} \ . \tag{7}$$

The distribution $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$ corresponds to the prediction step, and $p(\mathbf{x}_t|\mathbf{y}_{0:t})$ to the update step. If the system relating the hidden and observable variables were linear and Gaussian, the computation of the filtering distribution would be a straightforward use of the Kalman filter. Unfortunately, this is not the case here since we learned the transition and the likelihood in Sect. 2 in a non-parametrical way. On the other hand, as the hidden state $\mathbf{x}_t$ is discrete and finite, an exact numerical solution can be computed using a grid state-space method. However, except for very small edgelet lengths, the cardinality of the state space is too big to make this computation reliable in practice. Hence, we propose to use a sequential simulation-based method, the *particle filter*, to approximate the filtering distribution [16]. The method consists in computing the empiric density

$$P_N(d\mathbf{x}_t|\mathbf{y}_{0:t}) = \sum_{n=1}^{N} w_t^{(n)} \delta_{\mathbf{x}_t^{(n)}}(d\mathbf{x}_t) \ , \tag{8}$$

where $\delta_{\mathbf{x}_t^{(n)}}(\cdot)$ is a Dirac mass centered on a hypothetic state realization $\mathbf{x}_t^{(n)}$ of the state $\mathbf{x}_t$, also called particle, $w_t^{(n)}$ its weight, and $d\mathbf{x}_t$ an event of infinitesimal support. Particles are generated using an importance function $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(n)}, \mathbf{y}_{0:t})$ and are then weighted proportionately to (7).

3.2 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 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 $\mathbf{e}_t$ and a jump variable $\mathbf{c}_t$, i.e. $\mathbf{x}_t = (\mathbf{e}_t, \mathbf{c}_t)$. A particle filter requires to define four distributions: a prior $p(\mathbf{x}_0)$, to initialize particles; an importance function $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t})$, 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 $p(\mathbf{y}_t|\mathbf{x}_t)$, to weight the particles using the last known measure $\mathbf{y}_t$. While the prior and the likelihood are respectively learned in Sect. 2.1 and 2.3, the importance function and the transition need to be defined. As we will see in the present section, they are closely related to those learned in Sect. 2. The tracking procedure is summarized on Fig. 3 and detailed in Algorithm 5.

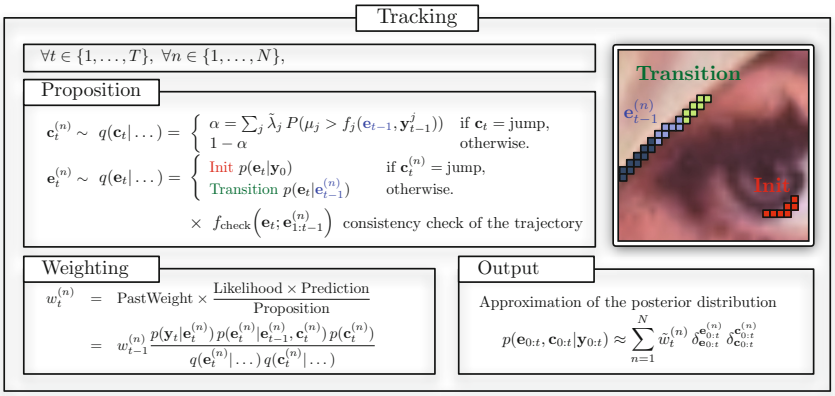

Fig. 3. The particle filter method approximates the filtering distribution by a finite discrete set of samples called particles. In a contour tracking application, we propose to model the state of a particle with two components: $\mathbf{e}_t$ an edgelet and $\mathbf{c}_t$ a jump variable, the latter being useful when the tracking of the current contour is finished. The tracking procedure 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 Sect. 2. The second step weights the propagated particles proportionately to their likelihood, giving more importance to relevant edgelets. These two steps approximate the filtering distribution.

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. For simplicity, we consider that the jump variable $\mathbf{c}_t$ is independent from $\mathbf{c}_{t-1}$. Hence,

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

with $p(\mathbf{c}_t = 1) = \beta$ the probability of jump. The edgelet transition $p(\mathbf{e}_t|\mathbf{e}_{t-1}, \mathbf{c}_t) = \mathbf{c}_t\, p(\mathbf{e}_t^{2:M}|\mathbf{e}_t^1)\, p(\mathbf{e}_t^1) + (1-\mathbf{c}_t)\, p(\mathbf{e}_t|\mathbf{e}_{t-1})$ includes the prior $p(\mathbf{e}_t^{2:M}|\mathbf{e}_t^1)$ learned in Sect. 2.1, the starting point $p(\mathbf{e}_t^1) = \mathcal{U}[\Omega]$, and the transition $p(\mathbf{e}_t|\mathbf{e}_{t-1})$ learned in Sect. 2.2.

Importance Function. It is possible to set $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ in order to propagate the particles using the trajectory transition. However, a more sophisticated design can drastically improve the estimation efficiency by reducing the variance of the particle weights [16, 17]. 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. Furthermore, edgelets are very likely to visit already visited contours, slowing down the algorithm. Here, we propose to define an importance function that uses the past observation to make a particle jump when it was not meaningful at $t-1$. It also uses the edgelet trajectory to constrain the particle to move on unvisited pixels:

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t}) = q(\mathbf{e}_t|\mathbf{e}_{0:t-1}, \mathbf{c}_t, \mathbf{y}_0)\, q(\mathbf{c}_t|\mathbf{e}_{t-1}, \mathbf{c}_{t-1}, \mathbf{y}_{t-1}) \ . \tag{10}$$

The distribution $q(\mathbf{c}_t|\mathbf{e}_{t-1}, \mathbf{c}_{t-1}, \mathbf{y}_{t-1})$ 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 much meaningful the past edgelet $\mathbf{e}_{t-1}$ was according to the J feature distributions:

$$\alpha = \sum_{j=1}^{J} \tilde{\lambda}_j \, P(\mu_j > f_j(\mathbf{e}_{t-1}, \mathbf{y}_{t-1}^j)) \ , \tag{11}$$

with $\tilde{\lambda}_j$ the normalized value of λ_j of (2). The edgelet importance function includes a trajectory constraint f_{check} as well as the transition and initialization distributions respectively learned in Sect. 2.2 and 2.4:

$$q(\mathbf{e}_t|\mathbf{e}_{0:t-1}, \mathbf{c}_t, \mathbf{y}_0) = n_{\text{q}} \, f_{\text{check}}(\mathbf{e}_t, \mathbf{e}_{t-1}, \mathbf{e}_{0:t-2}) \big[\mathbf{c}_t \, p(\mathbf{e}_t|\mathbf{y}_0) + (1 - \mathbf{c}_t) \, p(\mathbf{e}_t|\mathbf{e}_{t-1})\big] \ . \tag{12}$$

The trajectory constraint $f_{\text{check}}(\cdot)$ is set to 0 if any point in the edgelet $\mathbf{e}_t$ go through any point of $\mathbf{e}_{t-1}$ or is closer than a Manhattan distance of 2 with any point of the past trajectory $\mathbf{e}_{0:t-2}$. This is done to prevent an echo detection effect. Otherwise we set $f_{\text{check}}(\cdot) = 1$. Generating particles according to (12) can be done using a rejection sampling: a sample $\mathbf{e}_t^{(n)}$ is generated according to $\mathbf{c}_t^{(n)} \, p(\mathbf{e}_t|\mathbf{y}_0) + (1 - \mathbf{c}_t^{(n)}) \, p(\mathbf{e}_t|\mathbf{e}_{t-1}^{(n)})$ and accepted with a probability of $f_{\text{check}}(\mathbf{e}_t^{(n)}, \mathbf{e}_{t-1}^{(n)}, \mathbf{e}_{0:t-2}^{(n)})$. Finally, the normalizing constant n_q is approximated using an importance sampling method, i.e. $n_{\text{q}} \approx \frac{1}{N_{\text{q}}} \sum_{m=1}^{N_{\text{q}}} f_{\text{check}}(\mathbf{e}_t^{(m)}, \mathbf{e}_{t-1}^{(n)}, \mathbf{e}_{0:t-2}^{(n)})$, with $\mathbf{e}_t^{(m)} \sim \mathbf{c}_t \, p(\mathbf{e}_t|\mathbf{y}_0) + (1 - \mathbf{c}_t) \, p(\mathbf{e}_t|\mathbf{e}_{t-1}^{(n)})$.

Stopping Criterion. We discussed how to init and iterate the tracking algorithm, but the question of its stopping remains. The easiest way is based on the fact 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. Then, according to the feature distributions, detected contours become less and less meaningful, and one may stop the algorithm when the meaningfulness reaches a given threshold. In particular, we defined in (11) the probability of jump using the meaningfulness of an edgelet, and this probability grows with time. Then, we stop a particle filter when the proportion of jumps reaches a fixed threshold:

$$\frac{1}{K+1} \sum_{k=0}^{K} \sum_{n=1}^{N} w_t^{(n)} \mathbf{c}_{t-k}^{(n)} \geq \gamma \ . \tag{13}$$

Diversity. A resampling technique is used to avoid a degeneracy problem (all the particles but one converge to zero after a few steps), inherent to the particle filtering technique [16, 17]. This 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_{L} particles into L independent particle filters, leading to the following final posterior distribution:

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \frac{1}{L} \sum_{l=1}^{L} p(\mathbf{x}_{0:t}^l|\mathbf{y}_{0:t}) \ . \tag{14}$$

Each particle filter approximates the filtering distribution using $N = N_{\text{L}}/L$ particles.

Algorithm 5. PFCD Algorithm

Output: Particle filter contour detector map O

begin

for $l = 1, \dots, L$ **do** **Initialization:** $t = 0$

for $n = 1, \dots, N$ **do**

Generate $\mathbf{e}_0^{l,(n)} \sim p(\mathbf{e}_0|\mathbf{y}_0)$

Set $\mathbf{c}_0^{l,(n)} = 0$

Set $w_0^{l,(n)} = 1/N$

for $l = 1, \dots, L$ **do** **Tracking:** increment t

for $n = 1, \dots, N$ **do**

Generate $\mathbf{c}_t^{l,(n)} \sim q(\mathbf{c}_t|\mathbf{e}_{t-1}^{l,(n)}, \mathbf{c}_{t-1}^{l,(n)}, \mathbf{y}_{t-1})$

Generate $\mathbf{e}_t^{l,(n)} \sim q(\mathbf{e}_t|\mathbf{e}_{0:t-1}^{l,(n)}, \mathbf{c}_t^{l,(n)}, \mathbf{y}_0)$

Compute the particle weights:

$$w_t^{l,(n)} \propto w_{t-1}^{l,(n)} \frac{p(\mathbf{y}_t|\mathbf{e}_t^{l,(n)}) p(\mathbf{e}_t^{l,(n)}, \mathbf{c}_t^{l,(n)}|\mathbf{e}_{t-1}^{l,(n)})}{q(\mathbf{e}_t^{l,(n)}, \mathbf{c}_t^{l,(n)}|\mathbf{e}_{0:t-1}^{l,(n)}, \mathbf{c}_{t-1}^{l,(n)}, \mathbf{y}_{0:t-1})} \text{ s.t. } \sum_m w_t^{l,(m)} = 1$$

Resample the particle set $\{(\mathbf{e}_{0:t}^{l,(n)}, \mathbf{c}_{0:t}^{l,(n)}), w_t^{l,(n)}\}_{n=1}^N$ if necessary [16]

If $\frac{1}{K+1} \sum_{k=0}^{K} \sum_{n=1}^{N} w_t^{l,(n)} \mathbf{c}_{t-k}^{l,(n)} \geq \gamma$, stop the filter l

If all the particle filters have reached γ, stop the contour detection algorithm

Otherwise, go to step 2 for the unfinished filters

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

Contour Detector. The soft contour detector is an image $O\colon \Omega \to [0,1]$, with $O(z)$ the confidence value that the pixel z belongs to a contour. This is computed by a mean of the estimations given by the L particle filters:

$$\forall z \in \Omega, O(z) = \frac{1}{L} \sum_{l=1}^{L} \max_n w_{t_l}^{l,(n)} \mathbb{1}_{\mathbf{e}_{0:t_l}^{l,(n)}}(z) \ , \tag{15}$$

with t_l the last step performed by the l-th particle filter. An optional non-maximum suppression step may then be employed to produce thin contours [5]. The final tracking procedure is given in Algorithm 5.

4 Experiments

To evaluate our algorithm, we replicate the scenario used in the evaluation of state-of-the-art contour detection methods [2–9], using the BSDS300 and the BSDS500. Most of the parameters discussed in this following section are set with sense and according to the literature. The model is consistent with these parameters, hence reasonable parameter settings will give the expected results.

4.1 Parameter Discussion

In our experiments, we set the length of an edgelet $M = 7$, 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 $S_\mathrm{p} = 2 \times 10^6$ for the prior, $S_\mathrm{t} = 10^5$ for the transition, and $S_\mathrm{f} = S_\mathrm{i} = 10^6$ for the feature and the initialization distributions. The prior probability of jump β is set to 0.005.

For the textural gradient feature, we use an histogram of $R = R_\mathrm{h}^3 = 125$ bins. The image is defined on the CIE Lab colorspace. The widths of the channel bins are defined by the R_h-quantiles, hence each channel bin contains $1/R_\mathrm{h}^{\mathrm{th}}$ of the channel distribution. In order to consider enough points to create relevant 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 σ_p and λ_p are respectively set to 0.7 and 1.6 [15]. We use a mean for the fusion operators Ψ and Ξ and a $\min$ for the operator Φ. The values of the feature multiplicative constants are learned using a trial and error procedure on the training dataset. We found $\lambda_1 = 4$, $\lambda_2 = 6$, and $\lambda_3 = 15$.

We fix the total number of particles N_L to 1500, to provide a good compromise between detection performance and computational cost. We set the number of particle filters L to 50 in order to smooth the results. Then the number of particles by filter N is $1500/50 = 30$, which is enough to obtain a satisfying accuracy of each particle filter. Note that N may impact on the stopping criteria: using a larger number of particles allows reducing slightly γ, although it does not compensate for the additionnal computational cost. We approximate n_q using a small number of samples $N_\mathrm{q} = 50$. Finally, the parameter $K = 200$ ensures the monotonical increase of the stopping criterion. The threshold $\gamma = 0.13$ is learned using a trial and error procedure.

4.2 BSDS Experiments

As we can see in Fig. 1, our PFCD method performs well, with a F-Measure score at 0.68 (recall: 0.71, precision: 0.65) on the 100 test images of the BSDS300. Due to the stochastic nature of the algorithm, we performed the experiment 15 times and obtained a variance of 4.8×10^{-7}. We also completed the experiment on the BSDS500 [5] and obtained a F-Measure score of 0.70 (recall: 0.72, precision: 0.68). On average, our non-optimized code runs in 4 minutes and 30 seconds, which is comparable to the gPb detector [5, 18]. Also, the algorithm can easily be parallelized since both the samples in the online learning step and the particle filters during the tracking one are independent.

Fig. 4 illustrates a few contour detection results obtained by the proposed algorithm. The soft detection map is obtained using (15). By integrating the feature distribution into the likelihood, the extracted information is adaptative to the image and enables to highlight the most meaningful contours. Consequently, this removes noisy responses and offers cleaner results. In particular, the skin texture is not detected in the image of a leopard, since this information is not meaningful in the image. This behavior may also explain bad results observable in the first image (moray eel): the point of view is too close, and the low amount of real contours are not salient enough to be relevant, this is why the data found in the texture is deemed meaningful. We can also observe that the rendering is smooth: this is firstly caused by the prior and transition models, that capture local contour information well, and then by the mixture of particle filters, which circumvents discontinuity effects. This makes the rough contours softer as it is especially visible in images depicting trees (see for example the bison image). We

Fig. 4. Examples of contour detections obtained by the proposed PFCD algorithm. Images are from the BSDS300. The top two images outlined in red get two of the worses F-Measure scores.

finally notice that the contours are well located and around a three pixel width, which is very close to the truth. This is partly due to the profile gradient feature, which precisely detects contours, even if they are not very salient.

5 Conclusion

We demonstrated throughout this paper the ability of the particle filter to track contours in complex natural images. Its flexibility and genericity enable to embed semi-local

prior and transition distributions learned on a shape database as well as adaptive likelihoods that detect contextual relevant pieces of edge. Our contribution covers both the learning stage and the tracking modeling. We tested our algorithm on the competing BSDS300 [1] and BSDS500 [5] datasets, and obtained very promising results with F-Measure scores of 0.68 and 0.70, respectively. This might further be improved by integrating multiscale features and global information.

References

1. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In: ICCV, vol. 2, pp. 416–423 (2001)
2. Martin, D., Fowlkes, C., Malik, J.: Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. PAMI 26, 530–549 (2004)
3. Ren, X.: Multi-Scale Improves Boundary Detection in Natural Images. In: ICCV, pp. 533–545 (2008)
4. Maire, M., Arbeláez, P., Fowlkes, C., Malik, J.: Using Contours to Detect and Localize Junctions in Natural Images. In: CVPR, pp. 1–8 (2008)
5. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour Detection and Hierarchical Image Segmentation. PAMI 33, 898–916 (2011)
6. Dollar, P., Tu, Z., Belongie, S.: Supervised Learning of Edges and Object Boundaries. In: CVPR, vol. 2, pp. 1964–1971 (2006)
7. Mairal, J., Leordeanu, M., Bach, F., Hebert, M., Ponce, J.: Discriminative Sparse Image Models for Class-Specific Edge Detection and Image Interpretation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 43–56. Springer, Heidelberg (2008)
8. Ren, X., Fowlkes, C., Malik, J.: Scale-Invariant Contour Completion using Conditional Random Fields. In: ICCV, vol. 2, pp. 1214–1221 (2005)
9. Felzenszwalb, P., McAllester, D.: A Min-Cover Approach for Finding Salient Curves. In: CVPRW, pp. 185–185 (2006)
10. Pérez, P., Blake, A., Gangnet, M.: Jetstream: Probabilistic contour extraction with particles. In: ICCV, vol. 2, pp. 524–531 (2001)
11. Lu, C., Latecki, L.J., Zhu, G.: Contour Extraction Using Particle Filters. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part II. LNCS, vol. 5359, pp. 192–201. Springer, Heidelberg (2008)
12. Korah, T., Rasmussen, C.: Probabilistic Contour Extraction with Model-Switching for Vehicle Localization. In: IEEE Intelligent Vehicles Symposium, pp. 710–715 (2004)
13. Pitié, F., Kokaram, A., Dahyot, R.: Oriented Particle Spray: Probabilistic Contour Tracing with Directional Information. In: Irish Machine Vision and Image Processing (2004)
14. Desolneux, A., Moisan, L., Morel, J.: Edge detection by helmholtz principle. Journal of Mathematical Imaging and Vision 14, 271–284 (2001)
15. Sun, J., Xu, Z., Shum, H.Y.: Gradient Profile Prior and Its Applications in Image Super-Resolution and Enhancement. IP 20, 1529–1542 (2011)
16. Doucet, A., De Freitas, N., Gordon, N. (eds.): Sequential Monte Carlo methods in practice. Springer (2001)
17. Chen, Z.: Bayesian filtering: From Kalman filters to particle filters, and beyond. Technical report, McMaster University (2003)
18. Catanzaro, B., Su, B., Sundaram, N., Lee, Y., Murphy, M., Keutzer, K.: Efficient, high-quality image contour detection. In: CVPR, pp. 2381–2388 (2009)