JetStream: Probabilistic Contour Extraction with Particles

Patrick Pérez, Andrew Blake, and Michel Gangnet
Microsoft Research
St George House, 1 Guildhall Street, Cambridge, CB2 3NH, UK
http://research.microsoft.com/vision

Abstract

The problem of extracting continuous structures from noisy or cluttered images is a difficult one. Successful extraction depends critically on the ability to balance prior constraints on continuity and smoothness against evidence garnered from image analysis. Exact, deterministic optimisation algorithms, based on discretized functionals, suffer from severe limitations on the form of prior constraint that can be imposed tractably. This paper proposes a sequential Monte-Carlo technique, termed JetStream, that enables constraints on curvature, corners, and contour parallelism to be mobilized, all of which are infeasible under exact optimization. The power of JetStream is demonstrated in two contexts: (1) interactive cut-out in photo-editing applications, and (2) the recovery of roads in aerial photographs.

Figure 1. Probabilistic extraction of contours with JetStream. Even in presence of clutter, JetStream enables the extraction of (Left) silhouettes for photo-editing purpose, and (Right) roads in aerial photographs.

1. Introduction

The automatic or semi-automatic extraction of contours in images is an important generic problem in early vision and image processing. Its domain of applications ranges from the generic tasks of segmenting images with closed contours to the extraction of linear structures of particular interest such as roads in satellite and aerial images.

Despite the variety of applications, most approaches to contour extraction turn out to rely on some minimal cost principle. In a continuous deterministic setting, this can be cast as the minimization over a proper set of plane curves $\boldsymbol{r}$ of a functional

$$E(\boldsymbol{r}; y) = \int_{\boldsymbol{r}} g(\kappa(s), y(\boldsymbol{r}(s)))ds \qquad (1)$$

κ is the curvature, s is the arc-length, and $y(\boldsymbol{r}(s))$ is some scalar or vector derived at location $\boldsymbol{r}(s)$ from the raw image data I, e.g., often $y(\boldsymbol{r}(s))$ is the gradient norm $|\nabla I(\boldsymbol{r}(s))|$. This functional captures some kind of regularity on candidate curves, while rewarding, by a lower cost, the presence along the curve of contour cues such as large gradients or edgels detected in a previous stage.

We are interested in the particular case where a starting point p can be picked (either manually or automatically). If local cost function g in (1) turns out not to depend on κ, the optimal curve is a geodesic which can be recovered, at least in the form of a chain of pixels, using dynamic programming-type techniques [2, 4, 13, 14, 15, 16]. Unfortunately, unless optimality is abandoned, and huge storage resources are available, [13], there are tight restrictions on the form of g. As a consequence, only a prior on the total length of curves can then be captured, which is insufficient in situations where a strong smoothing prior is needed. This type of approach also relies on pixel-based discretization of the paths, ruling out any capability for on-the-fly sub-pixel interpolation.

A second approach, which does not suffer from these limitations, consists in growing a contour from the seed point according to cost function E. Given the current con-

tour, a new segment is appended to it according both to a shape prior (mainly smoothness) and to the evidence provided by the data in the location under concern. This approach has been investigated in a deterministic way, as a tool to complete discontinuous contours provided by edge detectors [11]. A good early survey on this so-called edge linking (or grouping) problem can be found in [1].

A recent advance on edge-linking has been obtained by taking a *probabilistic* point of view: the contours are seen as the paths of a stochastic process driven by both an inner stochastic dynamics, and a statistic data model. This is however a difficult tracking problem. Indeed, the data likelihood, as a function of the state, typically exhibits many narrow spikes. As a consequence, the posterior densities are badly behaved, preventing the use of most standard tracking tools based on the Kalman filter and its variants. In our context, multi-modality is related to the clutter of contours that most images exhibit. Ideally we seek a tracking method that is able: (i) to avoid spurious distracting contours, (ii) to track, at least momentarily, the multiple off-springs starting at branching contours, and finally (iii) to interpolate over transient evidence "gaps". Toward that goal, two very different approaches have been proposed in the literature.

In [3], Cox *et al.* adapt multiple hypothesis tracking (MHT) techniques from the tracking literature. The resulting tracker can handle the intrinsic multi-modality of the problem, as well as evolve multiple tracks (contours) simultaneously. The technique is however restricted to a special type of data: because all possible data associations are enumerated at each step, it is more adapted to sparse data. Also these data must be of the same nature (e.g., position and possibly orientation) as the hidden state since multiple Kalman trackers requiring a linear measurement model are run along the branches of the multiple hypothesis tree. Due to these limitations, the technique is only applied on the sparse output of a contour detector, thus performing edge linking.

In [6], Geman and Jedinak introduce a very effective road tracking technique based on the so-called "active testing" approach. Both their dynamics and their data model have nevertheless to be discrete. They indeed make use of a "decision tree" (containing all possible sequences of measurements), and of a "representation tree" of all possible paths. As for the dynamics, they even limit it to three different moves (no change in the direction, and change of $\pm 5^{\,o}$). Active testing, as well as pruning, are then conducted on the resulting ternary representation tree, using entropic tools.

We propose to tackle this tracking problem with particle filtering [5, 8, 10]. This Monte Carlo technique, based on sequential importance sampling/resampling, provides a sound statistical framework for propagating sample-based approximations of posterior distributions, with almost no restriction on the ingredients of the model. Since samples from the posterior path distribution are maintained at each step, different decision criteria can be worked out to decide which is the final estimated contour, including the MAP used by Geman and Jedinak, and the expectation used by Cox *et al.* Particle filtering will offer the same features as the two previous methods (maintaining multiple hypothesis, and on-the-fly pruning), but within a more versatile and consistent, yet simpler, framework.

The power of the proposed technique, termed JetStream, will be demonstrated in two different contexts: (1) the interactive delineation of image regions for photo-editing purpose, and (2) the extraction of roads in aerial images (see two result samples in Fig. 1). Besides the common core described in Sections 2 and 3, specific ingredients are introduced for each of the two applications: the incorporation of user interaction in the cut-out application (Section 4), and an explicit use of road width as part of the dynamical system for road tracking (Section 5).

2 Probabilistic contour tracking

Tracking contours in still images is a rather unconventional tracking problem because of the absence of a real notion of time: the "time" is only associated with the progressive growing of the estimated contour in the image plane. Contrary to standard tracking problems where data arrive one bit after another as time passes by, the whole set of data y is standing there at once in our case.

This absence of a natural time will have some importance in the design of both the prior and the data model. As for the definition of data likelihood, the transposition of standard techniques from tracking literature requires to conduct a rather artificial sequential ordering of the data as the tracking proceeds [3]: at step i, data set is constituted of data "visible" from current location. We prefer to consider the data as a whole, getting its ordering as a natural by-product of the sequential inference.

Another consequence of this absence of natural time, is that there is no straightforward way of tuning the "speed", or equivalently the length of successive moves. Whatever the speed at which the points travel, only their trajectories matter. The combination of prior dynamics and data model should simply make sure that the contour has reasons to grow. If the dynamics permits slowing down, then the tracker risks getting stuck at locations with high likelihood, resulting in a cessation of the growing process.

2.1 Tracking framework

Let us now introduce the basics of our probabilistic contour tracker. We consider random points x_i in the plane $\Lambda = \mathbb{R}^2$. Any ordered sequence $x_{0:n} \equiv (x_0 \cdots x_n) \in \Lambda^{n+1}$ uniquely defines a curve in some standard way, e.g., the x_i's

are the vertices of a polyline in our experiments. The aim is to make grow such a sequence based on a *prior dynamics* $p(x_{i+1}|x_{0:i})$ that retains expected properties of the contours to be extracted, and on a *data model* $p(y|x_{0:n})$ that provides evidence about whether a measurement is, or is not, in the vicinity of the "true" contour.

Assuming a homogeneous second-order[1] dynamics with kernel q:

$$p(x_{i+1}|x_{0:i}) = q(x_{i+1}; x_{i-1:i}), \forall i \geq 2,$$

the *a priori* density on Λ^{n+1} is

$$p(x_{0:n}) = p(x_{0:1}) \prod_{i=2}^{n} q(x_i; x_{i-2:i-1}). \quad (2)$$

We also approximate measurements conditioned on $x_{0:n}$ as an independent spatial process

$$p(y|x_{0:n}) = \prod_{u \in \Omega} p(y(u)|x_{0:n}) \quad (3)$$

where $\Omega \subset \Lambda$ is a discrete set of measurement locations in the image plane, including the x_i's locations. Each individual likelihood in product (3) is either p_{on} if u belongs to $x_{0:n}$, or p_{off} if not:

$$\begin{aligned} p(y|x_{0:n}) &= \prod_{u \in \Omega \setminus x_{0:n}} p_{\text{off}}(y(u)) \prod_{i=0}^{n} p_{\text{on}}(y(x_i)|x_{0:n}) \\ &= \prod_{u \in \Omega} p_{\text{off}}(y(u)) \prod_{i=0}^{n} \frac{p_{\text{on}}(y(x_i)|x_{0:n})}{p_{\text{off}}(y(x_i))}. \quad (4) \end{aligned}$$

This likelihood is of the same form as the one derived by Geman and Jedinak in their active testing framework [6] The posterior density on Λ^{n+1} is derived, up to a multiplicative factor independent from $x_{0:n}$:

$$p_n(x_{0:n}|y) \propto p(x_{0:1}) \prod_{i=2}^{n} q(x_i|x_{i-2:i-1}) \prod_{i=0}^{n} \ell(y(x_i)) \quad (5)$$

where $\ell \equiv \frac{p_{\text{on}}}{p_{\text{off}}}$ denotes the point-wise likelihood ratio.[2] Density $p(x_{0:1})$ is a Dirac mass centered at locations picked by the user. The choice of the transition probability q and of the likelihood ratio ℓ will be discussed in next section.

The function $E_n(x_{0:n}; y) \equiv -\log p_n(x_{0:n}|y)$ can be seen as the n-sample discretization of a functional of type (1). Expressed as the minimization of this functional, the contour extraction problem then amounts to seeking the maximum a posteriori (MAP) estimate in our probabilistic setting.

[1] If a higher order dynamics seems more appropriate to the contours under concern, it can be used.

[2] Note that since the likelihood $p(y|x_{0:n})$ expresses the probability of the data given that $x_{0:n}$ defines *the only contour* in the image, not simply a part of the contour set, the posterior densities p_n's are not related through marginalization: $\int_{x_n} p_n(x_{0:n}|y)dx_n \neq p_{n-1}(x_{0:n-1}|y)$.

2.2 Iterative computation of posterior

The tracking philosophy relies on computing recursively posterior densities of interest. From (5), it comes the following recursion:

$$p_{i+1}(x_{0:i+1}|y) \propto p_i(x_{0:i}|y) q(x_{i+1}|x_{i-1:i}) \ell(y(x_{i+1})). \quad (6)$$

Although we have analytical expressions for ℓ and q, this recursion cannot be computed analytically: there is no closed form expression of the posterior distributions p_i's. The recursion can however be used within a sequential Monte Carlo framework where posterior p_i is approximated by a finite set $(x^m_{0:i})_{m=1\cdots M}$ of M sample paths (the "particles"). The generation of samples from p_{i+1} is then obtained in two steps.

In a first *prediction* (or proposal) step, each path $x^m_{0:i}$ is grown of one step $\tilde{x}^m_{i+1}$ by sampling from a proposal density function $f(x_{i+1}; x^m_{0:i}, y)$ over Λ, whose choice will be discussed shortly. If the paths $(x^m_{0:i})_m$ are fair samples from distribution p_i over Λ^{n+1}, then the extended paths $(x^m_{0:i}, \tilde{x}^m_{i+1})_m$ are fair samples from distribution fp_i over Λ^{n+2}. Since we are seeking samples from distribution p_{i+1} instead, we resort to *importance sampling*: these sample paths are *weighted* according to ratio p_{i+1}/fp_i (normalized over the M samples). The resulting weighted path set now provides an approximation of the target distribution p_{i+1}. This discrete approximating distribution is used in the second step of *selection*, where M paths are drawn with replacement from the previous weighted set. The new set of paths is then distributed according to p_{i+1}. The paths with smallest weights are likely to get discarded by this selection process, whereas the ones with large weights are likely to get duplicated.

Using the expression (6) of p_{i+1}, ratio p_{i+1}/fp_i boils down to $q\ell/f$. The weights thus read:

$$\pi^m_{i+1} \propto \frac{q(\tilde{x}^m_{i+1}; x^m_{i-1:i}) \ell(y(\tilde{x}^m_{i+1}))}{f(\tilde{x}^m_{i+1}; x^m_{0:i}, y)} \quad (7)$$

with $\sum_m \pi^m_{i+1} = 1$. It can be shown that the optimal proposal pdf is $f = q\ell / \int_{x_{i+1}} q\ell$ [5], whose denominator cannot be computed analytically in our case. The chosen proposal pdf must then be sufficiently "close" to the optimal one such that the weights do not degenerate (i.e., become extremely small) in the re-weighting process.

Based on the discrete approximation of the posterior p_i, different estimates of the "best" path at step i can be devised. An approximation of the MAP estimate is provided by the path of maximum weight (before resampling). A more stable estimate is provided by the mean path $\frac{1}{M} \sum_{m=1}^{M} x^m_{0:i}$, which is a Monte Carlo approximation of the posterior expectation $\mathbb{E}(x_{0:i}|y)$.

3 Model ingredients

3.1 Likelihood ratio ℓ

Most data terms for contour extraction are based on the spatial gradient in intensity or RGB space, and/or on edgels detected by standard means (e.g., with Canny detector). More sophisticated cues can be incorporated, such as color/intensity consistency on each side of the contour, texture, or blur, but their relevance varies obviously from one image to another.

Although simple, the norm of the luminance (or color) gradient remains a robust cue. To use it as part of our measurements, we must capture its marginal distributions both off contours (p_{off}) and on contours (p_{on}).

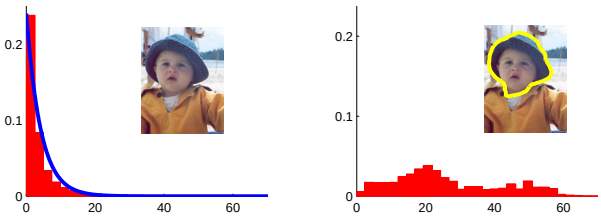


Figure 2. Gradient norm statistics. Normalized histograms of gradient norm on a baby photograph with cluttered background: (Left) over the whole image, along with the fitted exponential distribution; (Right) on the outline of the face and hat only.

The first marginal p_{off} can be empirically captured by the distribution of the norm of the gradient on the whole image (Fig. 2). In our experiments, this empirical distribution was always well approximated by an exponential distribution with parameter λ (amounting to the average norm over the image), which we take as p_{off}. As for p_{on} it is difficult to learn it *a priori*. The empirical distribution over an outline of interest appears as a complex mixture filling the whole range of values from 0 to a large value of gradient norm (Fig. 2). In the absence of an appropriate statistical device to capture adaptively this highly variable behavior, it seems better to keep the data likelihood p_{on} as less informative as possible. We simply use a uniform distribution.

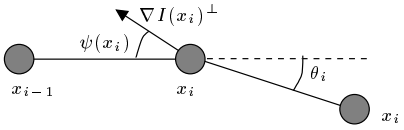


Figure 3. Position and angle notations.

Observing the angle $\psi(x_i) \in [-\pi/2, \pi/2]$, shown in Fig. 3, between the gradient normal $\nabla I(x_i)^\perp$ and the segment (x_{i-1}, x_i), indicates that the direction of the gradient also retains precious information that a data model based only on gradient norm neglects: the distribution of ψ is symmetric, and it becomes tighter as the norm of the gradient increases. More precisely, we found empirically that the distribution of $|\nabla I|^{0.5}\psi$ exhibits a normal shape $N(0, \sigma_\psi^2)$ (Fig. 4).

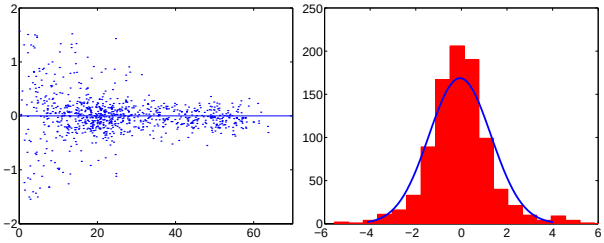


Figure 4. Gradient statistics. (Left) Plot of the angle ψ against the gradient norm for the face contour of Fig.2; (Right) histogram of $|\nabla I|^{0.5}\psi$ with its normal fit ($\sigma_\psi = 1.36$).

There is however an important exception to the validity of the image-gradient distribution above. At *corners*, the norm of the gradient is usually large but its direction cannot be accurately measured. Using a standard corner detector [7], each pixel u is associated with a label $c(u) = 1$ if a corner is detected, and 0 otherwise. Where a corner has been detected, it is then appropriate to accept a wide range of image gradient directions, but to continue to favour high gradient magnitude. We thus assume distribution $p(\psi(x_i)|x_{i-1:i}, |\nabla I(x_i)|, c(x_i) = 1)$ is uniform. We also assume that the probability of corner apparition is the same on relevant contours and on background clutter, and that the distribution of gradient direction off contours is uniform (the latter being supported by experimental evidence).

Finally the complete data model is defined on $y = (|\nabla I|, c)$ by the two likelihoods

$$p_{\text{on}}(\nabla I(x_i), c(x_i)|x_{i-1:i}) \propto \frac{c(x_i)}{\pi} + (1 - c(x_i))N\left(\psi(x_i); 0, \frac{\sigma_\psi^2}{|\nabla I(x_i)|}\right) \quad (8)$$

$$p_{\text{off}}(\nabla I(x_i), c(x_i)) \propto \exp -\frac{|\nabla I(x_i)|}{\lambda} \quad (9)$$

from which ratio $\ell = \frac{p_{\text{on}}}{p_{\text{off}}}$ is deduced up to a multiplicative constant.

3.2 Dynamics q

Because of the absence of natural time, it is better to consider a dynamics with *fixed step length* d. The definition of

the second-order dynamics $q(x_{i+1}|x_{i-1:i})$ then amounts to specifying an *a priori* probability distribution on direction change $\theta_i \in (-\pi, \pi]$ shown in Fig. 3.

The smoothness of the curve can be simply controlled by choosing this distribution as Gaussian with variance σ_θ^2 per length unit. For steps with length d, the resulting angular variance is $d\sigma_\theta^2$. However, under such a dynamics with typical standard deviation ranging from $0.05\sqrt{d}$ to $0.1\sqrt{d}$, substancial changes of direction are most unlikely. In order to allow for abrupt direction changes at the few locations where corners have been detected, we mix the normal distribution with a small proportion ν of uniform distribution over $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The dynamics finally reads:

$$\begin{aligned} &x_{i+1} = x_i + R(\theta_i)(x_i - x_{i-1}), \\ &\text{with } q(\theta_i) = \frac{\nu}{\pi} + (1-\nu)N(\theta_i; 0, d\sigma_\theta^2) \end{aligned} \tag{10}$$

where $R(\theta_i)$ is the rotation with angle θ_i, and, with a slight abuse of notation, we now use q to denote the prior angular density on direction change.

3.3 Proposal sampling function f

Now that both the dynamics and the data model are chosen, it remains to devise a proposal sampling function f which is as much related as possible to $q\ell$ under the constraint that it can be sampled from. Since the mixture dynamics (10) can be easily sampled, it is a natural candidate. In this case, $\tilde{x}_{i+1}^m$ is predicted from $x_{i-1:i}^m$ by sampling from (10) and the weights in (7) boil down to likelihood ratios $\ell(y(\tilde{x}_{i+1}^m))$ normalized over the M predicted positions.

With this standard choice $f = q$, corners will be mostly ignored since the expected number of particles undertaking drastic direction changes is νM, where typically $\nu = 0.01$ and $M = 100$. This can be circumvented by devising a proposal function that depends also on the output of the corner detector. We thus define the prediction step as:

$$\begin{aligned} &x_{i+1} = x_i + R(\theta_i)(x_i - x_{i-1}), \\ &\text{with } p(\theta_i) = \frac{c(x_i)}{\pi} + (1 - c(x_i))N(\theta_i; 0, d\sigma_\theta^2). \end{aligned} \tag{11}$$

At locations where no corners are detected, the proposal density is the normal component of the dynamics (10). If x_i lies on a detected corner, the next proposed location is obtained by turning of an angle picked uniformly between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. The impact of the corner-based component in the proposal in shown in Fig. 5.

Figure 5. Using corner detection in JetStream. (Left) With standard proposal function $f = q$ the particle stream overshoots the corners. (Right) Including a corner-based component in the proposal is sufficient to accommodate corners automatically.

The complete JetStream iteration is finally summarized in Procedure 1.

Procedure 1 JetStream Iteration

- current particle set: $(x_{0:i}^m)_{m=1\cdots M}$
- *Prediction*: for $m = 1 \cdots M$
  - if $c(x_i^m) = 1$ (corner), draw θ_i from uniform distribution on $(-\frac{\pi}{2}, \frac{\pi}{2})$
  - if $c(x_i^m) = 0$ (no corner), draw θ_i from normal distribution $N(0, \sigma_\theta^2)$
  - $\tilde{x}_{i+1}^m = x_i^m + R(\theta_i)(x_i^m - x_{i-1}^m)$
- *Weighting*: compute for $m = 1 \cdots M$

$$\pi_{i+1}^m = \frac{Kq(\theta_i)\ell(\nabla I(\tilde{x}_{i+1}^m), c(\tilde{x}_{i+1}^m))}{\frac{c(x_i^m)}{\pi} + (1 - c(x_i^m))N(\theta_i; 0, d\sigma_\theta^2)} \tag{12}$$

  with K such that $\sum_{k=1}^M \pi_{i+1}^k = 1$

- *Selection*: for $m = 1 \cdots M$, sample index $a(m)$ from discrete probability $\{\pi_{i+1}^k\}_k$ over $\{1 \cdots M\}$, and set

$$x_{0:i+1}^m = (x_{0:i}^{a(m)}, \tilde{x}_{i+1}^{a(m)}) \tag{13}$$

In all experiments, the step length was fixed to $d = 1$, and the mixture proportion in the dynamics was fixed to $\nu = 0.01$. The standard deviation σ_θ in the normal component of the dynamics was manually tuned within range (0.05,0.1). As for data model, parameter μ is the average gradient norm in the image under consideration, and standard deviation σ_ψ was set 1. Finally $M = 100$ particles were used.

4 Interactive cut-out

The extraction of a region of interest from one image to be pasted into another image is a fundamental photo-editing capability. Achieving high quality cut-out is difficult in practice because the foreground boundary needs to be located accurately, to within a pixel or better. Marking individual pixels by hand can be accurate but is laborious. Semi-automatic methods, such as Mortensen and Barret's *LiveWire* [15, 16][3] , speed up the process, but have to rely on assumptions about the shape properties of the bounding contour, whose validity is somewhat specific to a particular contour. A robust approach to the problem, therefore, is to provide a box of tools selected by the user, on a problem by problem basis. To turn JetStream in such a cut-out tool, we must give the user means of interacting with the flow of particles.

In practice, JetStream is run for a fixed number n of steps (100 in our experiments) from initial conditions $x_{0:1}$ chosen by the user. If the result is satisfactory, n more steps are undertaken. If not, a restart region within the particle flow, and an associated restart direction, can be chosen by the user.

In many situations however, a softer and simpler alternative type of guidance can be used. It is based on a probabilistic interactive device to supply additional information, deemed to be probabilistically independent from information derived automatically from the image. The further information is embedded in a *user likelihood ratio* $\ell^{\mathrm{u}}(x)$, and the assumed independence leads to it being applied multiplicatively in (5). Where $\ell^{\mathrm{u}}(x) < 1$, particle flow is discouraged, causing a *constriction* that repels the boundary contour, serving a function analogous to the "volcano" used with snakes [9]. Alternatively, a region in which $\ell^{\mathrm{u}}(x) > 1$ serves as a *channel* that draws in the particle stream. There are various possibilities for the use of this mechanism. We have found that, in practice, it suffices to provide the user with the facility to place one or more *dams*, defined as regions $R_k \subset \Omega$, such that

$$\ell^{\mathrm{u}}(x) = \begin{cases} \varepsilon & \text{if } x \in R_k, \text{ for some } k, \\ 1 & \text{otherwise} \end{cases}$$

with ε set to some small value such as $\varepsilon = 0.00001$. This is preferable to setting $\varepsilon = 0$ because, in the event that all particles are accidentally encased in dams, the particles can tunnel their way out. The practical operation of a dam is illustrated in Fig. 6.

[3]LiveWire is an interactive cut-out tool based on Dijkstra's algorithm. Given a starting point p, this dynamic programming technique provides the connected pixel chain that minimizes a cost function similar to our $-\log p(y|x_{0:n})$ in between p and any arbitrary point q, with n not fixed.

Figure 6. User interaction with dams. (Left) The desired contour (rim of baby's hat) is lost after the branching, but visualizing pixels "visited" by JetStream (shaded) shows that this contour was momentarily tracked by part of the stream; (Right) the insertion of a dam (dark rectangle) by the user is then sufficient to correct the flow.

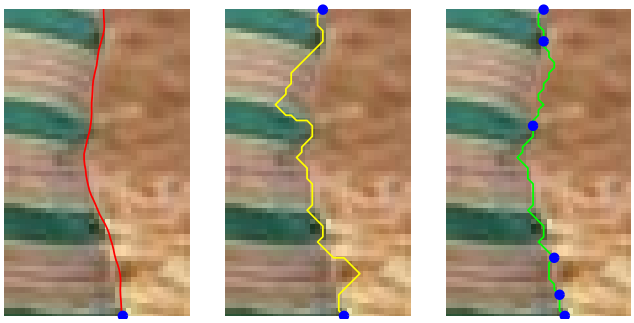
We illustrate the power of the interactive cut-out tool thus obtained on a strongly textured and cluttered image. Such images pose a particular challenge for any automatic procedure for cut-out, because any feature detector designed to respond to boundary is liable to respond also to the texture and clutter. The smoothness prior of JetStream combats this distracting effect, as the example of Fig. 7 shows. LiveWire also combats it but less strongly, given that its prior curve model penalizes length but not curvature. As a result, and given more user interaction, LiveWire nonetheless generates a less accurate boundary contour (Fig. 7.)

LiveWire, 38 interactions JetStream, 17 interactions

Figure 7. Cut-out with strong texture and clutter. LiveWire boundary is distorted, whereas JetStream behaves better with less user interaction.

The texture problem is further illustrated by the close-up in Fig.8. In this example the stripes on the skirt generate additional image features, competing with those generated by the boundary. Inclusion of the second-order dynamic

model with a relatively tightly set parameter $\sigma_\theta = 0.05\,\mathrm{rad}$, imposes enough smoothness for JetStream to capture the boundary of this example, entirely automatically, given just a starting point and direction.



JetStream LiveWire LiveWire

Figure 8. Automatic texture rejection. A fragment of a woman's skirt is detected here in several alternative ways: (Left) JetStream, given just the starting point (dark dot) and direction; (Middle) LiveWire, given two fixed endpoints (with parameters set to default values [15]); (Right) LiveWire with an additional four fixed points.

5 Road extraction

The problem of extracting, automatically or semi-automatically, roads from aerial and satellite images has received a great deal of attention, e.g., [6, 12, 17]. JetStream provides a new tool to address this problem.

As we saw in the previous section the contextual information JetStream conveys is already sufficient to improve a lot on the minimum cost pixel chain provided dynamic programming (DP) from the same starting point: JetStream output is less jaggy, and, in many cases, the DP path snaps to undesirable surrounding contours, including remote ones.

However, in the specific context of road extraction, one is in fact interested in recovering "ribbons". Using JetStream as defined previously results in paths jumping from one side of the ribbon to the other as shown in Fig. 9.

The flexibility of JetStream enables easy incorporation of this new geometric element to the model. The state-space is extended to include a width variable m_i, which indicates the distance at step i between the two sides of the ribbon (Fig. 10). These two sides being standard contours, the data likelihood is an immediate extension of the one presented so far. It expresses that x_i^+ and x_i^- defined as $x_i \pm m_i \frac{(x_i - x_{i-1})^\perp}{|x_i - x_{i-1}|}$ are on a contour while x_i is not:

$$p(y|x_{0:n}, m_{0:n}) = p_{\mathrm{off}}(y) \prod_i \frac{p_{\mathrm{on}}(y(x_i^+))p_{\mathrm{on}}(y(x_i^-))}{p_{\mathrm{off}}(y(x_i^+))p_{\mathrm{off}}(y(x_i^-))}. \quad (14)$$

Figure 9. Extracting roads with ribbon model. Using a ribbon-like extension of contour model enables a stable extraction of roads, even with fixed width in this example.

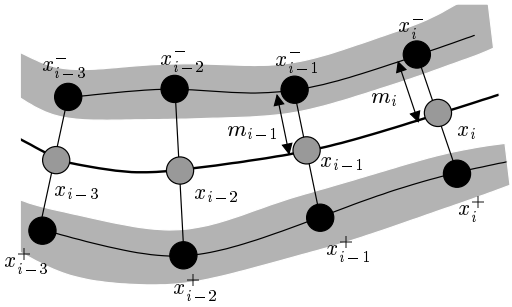


Figure 10. The ribbon-process geometry. Two contours are now tracked simultaneously at a varying distance m from the middle path.

We have chosen a simple first-order dynamics for m_i:

$$m_{i+1} = m_i + v_i,\ v_i \overset{\text{i.i.d.}}{\sim} N(0, d\sigma_m^2). \quad (15)$$

The parameter σ_m^2 captures (invariantly to the magnitude of d) the statistical rate of growth of the ribbon width, in units of width variance per unit road length. Setting this parameter to zero (i.e., using a fixed width ribbon) already provides interesting results as shown in Fig. 9.

For a better tracking, this parameter can be estimated from digitized road maps at the right scale. For the 1:25000 aerial photographs we used for instance, we found $\sigma_m = 0.005$. Note that the variance parameter of the dynamical model can be learnt the same way. For 1:25000 aerial photographs we found $\sigma_\theta = 0.05$rad.

As shown in Fig. 11, this simple width dynamics already proves effective to track the changes in road width. The right of Fig. 1 provides an other result sample of ribbon tracking, including one drastic change of direction (at the crossing).

Figure 11. Tracking of road with width variations. Using simple first-order dynamics on the width, sampled simultaneously with the position dynamics, enables roads with varying width to be tracked.

6 Conclusion

We have demonstrated the efficacy of a sequential Monte-Carlo approach to tracking continuous structures in cluttered images. The flexibility of the Monte Carlo approach made it possible to incorporate important elements of the prior model, including smoothness, corners, and ribbon-structure, which cannot tractably be used in deterministic approaches. The generic system can be easily specialized. As demonstrated, it can thus provide a valuable addition to the box of tools available for interactive cut-out in photo-editing applications, as well as an appealing way of extracting semi-automatically roads from aerial photographs.

A number of further issues are raised by this work. One is to investigate raising the order of the prior dynamics above 2, in order to capture a wider range of curve properties. For example $s = 4$ should suffice to capture oscillations, to help with segmenting corrugated boundaries. Furthermore, the coefficients of such a dynamic prior could possibly be learned, either off-line for each member of some gallery of standard curve types, or adaptively, as boundary construction progresses. Another area of investigation, is the possibility of explicit handling of branches, for example at T-junctions, so that the boundary splits automatically, with both branches continuing to grow.

Acknowledgements

The second author thanks Michael Isard for preliminary discussions on this work. All aerial photographs are courtesy of The GeoInformation Group.

References

[1] D. Ballard and C. Brown. *Computer vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] L. Cohen and R. Kimmel. Global minimum for active contour models: A minimal path approach. *Int. J. Computer Vision*, 24(1):57–78, 1997.

[3] I. Cox, J. Rehg, and S. Hingorani. A Bayesian multiple hypothesis approach to edge grouping and contour segmentation. *Int. J. Computer Vision*, 11(1):5–24, 1993.

[4] T. Deschamps and L. Cohen. Minimum paths in 3d images and application to virtual endoscopy. In *Proc. Europ. Conf. Computer Vision*, volume 2, Dublin, Ireland, June 2000.

[5] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[6] D. Geman and B. Jedynak. An active testing model for tracking roads in satelitte images. *IEEE Trans. Pattern Anal. Machine Intell.*, 18(1):1–14, 1996.

[7] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey88*, pages 147–152, 1988.

[8] M. Isard and A. Blake. Condensation–conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29(1):5–28, 1998.

[9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes : active contour models. *Int. J. Computer Vision*, 1(4):321–331, 1988.

[10] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *J. Computational and Graphical Stat.*, 5:1–25, 1996.

[11] A. Martelli. An application of heuristic search methods to edge and contour detection. *Commun. ACM*, 19(2):72–83, 1986.

[12] N. Merlet and J. Zerubia. New prospects in line detection by dynamic programming. *IEEE Trans. Pattern Anal. Machine Intell.*, 18(4):426–431, 1996.

[13] N. Merlet and J. Zerubia. Integration of global information for roads detection in satellite images. Technical Report 3239, INRIA, 1997.

[14] U. Montanari. On the optimal detection of curves in noisy pictures. *Commun. ACM*, 15(5):335–345, 1972.

[15] E. Mortensen and W. Barrett. Intellingent scissors for image composition. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, pages 191–198, Los Angeles, CA, August 1995.

[16] E. Mortensen and W. Barrett. Interactive segmentation with intelligent scissors. *Graph. Mod. Image Proc.*, 60(5):349–384, 1998.

[17] A. Yuille and J. Coughlan. Fundamental limits of Bayesian inference: order parameters and phase transition for road tracking. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(2):160–173, 2000.