Active Contours for Tracking Distributions

Daniel Freedman
Computer Science Department, Rensselaer Polytechnic Institute

August 26, 2002

Abstract

A new approach to tracking using geometric active contours is presented. The class of objects to be tracked is assumed to be characterized by a probability distribution over some variable, such as intensity, colour, or texture. The goal of the algorithm is to find the region within the current image, such that the sample distribution of the interior of the region most closely matches the model distribution. Several criteria for matching distributions are examined, and the curve evolution equations are derived in each case. A particular flow is shown to perform well in two experiments.

1 Introduction

This paper deals with the problem of tracking an object as it moves through a video-stream, based on photometric rather than geometric considerations. Throughout, the term "photometric variable" will be used loosely to mean a quantity such as intensity, colour, or texture; photometric variables are distinguished from geometric variables, such as edges. With this in mind, the algorithm may be explained in a straightforward fashion. The class of objects to be tracked is assumed to be characterized by a probability distribution over some photometric variable. In each frame of the video, the algorithm tries to find a region of the image whose interior generates a sample distribution over the relevant variable which most closely matches the model distribution. The goal is to cast this problem into the framework of geometric active contours, and to derive curve flows which optimize the relevant matching criteria. The idea behind the algorithm is illustrated in figure 1.

Posing the tracking problem in this way has the advantage of dealing directly with two difficulties that often confound such algorithms. First, the tracker does not rely on edges. Many trackers use edge information exclusively; examples include [8, 2, 7]. The problems associated with using edges are well-known. For instance, edge-detectors may be inaccurate, leading to the detection of spurious edges; edges may not be detected when contrast between adjacent surfaces fades due to illumination changes; and so on. However, even in the case of ideal edge-detection, such algorithms would err in their approach, simply by failing to take into account the rich amounts of information which are available in the photometric variables of the images. For example, much headway may be made in the design of a lip-tracker by noticing that human lips tend to come in a small number of colours. Indeed, colour-based methods are often used in special-purpose trackers (e.g [20, 12] present colour-based face-trackers), mostly to excellent effect. The second problem which is directly addressed by this tracker is the difficulty of tracking successfully through cluttered scenes.

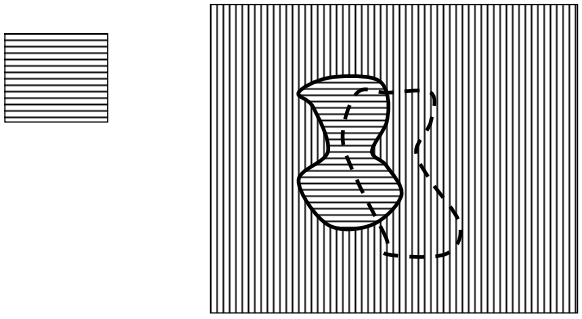

Figure 1: An illustration of the algorithm. On the left is a schematic representation of the model distribution, here taken to be a texture of horizontal lines. On the right is an image. The dashed line indicates the initial position of the region; within this region, the empirical distribution only partly matches the model distribution, as some of the background (a texture of vertical lines) is contained within the region. Thus, the curve which is the region's boundary will flow to the solid line; the resulting region maximizes the match between empirical and model distributions.

By posing the problem as one of matching distributions, the tracker has robustness built in from the start, which should allow for a reasonable chance of navigation through clutter.

A reasonable objection may be raised: why not incorporate both photometric and geometric considerations? This, of course, is the eventual goal of the research programme, the first step of which is presented in this paper. However, before attempting to include geometric concerns, it is instructive to see how well a pure photometric tracker can do. A recent general-purpose photometric tracker [6], to be discussed at greater length shortly, demonstrates the ability of such trackers to succeed.

The active contour literature is vast, so no attempt will be made to review it comprehensively. The field originated with the snake formulation of Kass, Witkin, and Terzopoulos [8], and many papers in a similar vein followed [1, 17, 19]. The recent trend has been towards geometric curve evolution [3, 9, 10, 4], and this paper will follow in that tradition. Many of these recent papers have focused on the novel level-set approach to implementing geometric curve flows [14], which allows for a stable numerical scheme, as well as for changes in topology to be handled without difficulty.

There are several recent papers in the active contour literature which bear closer relation to the current paper. Chan and Vese [5] solve a restricted form of the Mumford-Shah segmentation problem [11], assuming only two regions whose segments are piecewise constant. Yezzi *et al.* [22, 21] and Tsai *et al.* [18] also solve a number of segmentation problems, including the full-blown Mumford-Shah segmentation. Noteworthy as well is the paper of Paragios and Deriche [15], which solves a segmentation problem using both boundary and region information. While the approach taken in this paper may appear superficially similar to the current approach, a key difference is the fact that the authors assume that one has access not only to information about the object to be tracked, but about the background as well. In this sense, the algorithm developed in [15] is truly a segmentation algorithm; it is generally impractical, within the context of all-purpose trackers, to assume knowledge of the background, as the tracker must be able to work in as many situations as possible. Paragios and Deriche have extended this work to the context of motion estimation in [16]. All of the above mentioned papers are similar in spirit to the current paper, in that they use information contained in the interior of the contours within the geometric curve evolution

framework. However, the types of information used are quite different, as are the techniques used in deriving the flows.

Finally, it is worth mentioning the work of Comaniciu *et al.* [6], which is in a sense most in keeping with the present work. This algorithm attempts to follow a distribution by maximizing the Bhattacharyya measure between a model distribution and an empirical distribution from the current frame. However, this approach is not based on active contours, and one of its major drawbacks is precisely related to this fact: the shape of the object is assumed to be an ellipse. Of course, many objects are not even approximately elliptical (*cf* the flexing finger in section 4). The way in which the ellipse translates from frame to frame is the focus of the paper, and is given by so-called "mean-shift analysis"; however, the way in which the ellipse shrinks or enlarges is incorporated in an *ad hoc* fashion. Nonetheless, this tracker performs very well experimentally, and does so in real time.

2 Theory

2.1 Notation

Let z be the photometric variable of interest. For example, z could be an intensity, colour vector, or texture vector. The variable z lives in the space $\mathcal{Z}$, which is assumed to be a Euclidean space of dimension n for some $n \geq 1$. Thus, for intensities, $n = 1$; for colours $n = 3$; and for textures n is the dimension of the output of the relevant filter bank. It is assumed that the class of objects to be tracked is characterized by a model probability density over the variable z, specified by $q(z)$.

The goal is to try to match a sample probability density within a region of the image to the model density. Let $x \in \mathbb{R}^2$ specify the coordinates in the image plane, and let $Z : \mathbb{R}^2 \to \mathcal{Z}$ be a mapping from the image plane to the space of the photometric variable. Thus, if $\mathcal{Z}$ is the space of intensities, then $Z(x)$ is just a grayscale image; if $\mathcal{Z}$ is the space of colours, then $Z(x)$ is a colour image. Denote a region of the image plane by $\omega \subset \mathbb{R}^2$; let $\mathbf{c} = \partial\omega$ be its boundary. We wish to specify $p(z;\omega)$, the sample probability density within the region ω. Let $\theta(z)$ be the n-dimensional Heaviside function, i.e.

$$\theta(z) = \begin{cases} 1 & z_1, \ldots, z_n \geq 0, \\ 0 & \text{otherwise}. \end{cases}$$

Then we may write the cumulative distribution function defined inside the region ω as

$$F(z;\omega) = \frac{\int_\omega \theta(z - Z(x))dx}{\int_\omega dx}$$

Thus, the probability density $p(z;\omega)$ is given by

$$p(z;\omega) = \frac{\partial^n F(z;\omega)}{\partial z_1 \ldots \partial z_n} = \frac{\int_\omega \delta(z - Z(x))dx}{\int_\omega dx} \equiv \frac{N(z;\omega)}{A(\omega)}$$

where $\delta(\cdot)$ is the usual n-dimensional delta-function. Note that $A(\omega)$ is the area of the region ω.

2.2 Density Matching Criteria

The goal is to find the region ω in the image plane such that the sample density $p(z;\omega)$ most closely matches the model density $q(z)$. There are a variety of criteria which can be used to compare the

two densities. The first is the Kullback-Leibler distance:

$$K(\omega) \equiv K(q(\cdot), p(\cdot;\omega)) = \int q(z) \log\left(\frac{q(z)}{p(z;\omega)}\right) dz$$

The smaller the distance, the closer the two distributions are. (Note that the Kullback-Leibler distance is not truly a metric, as it is not symmetric in its argument.) A second choice is the Bhattacharyya measure:

$$B(\omega) \equiv B(p(\cdot;\omega), q(\cdot)) = \int \sqrt{p(z;\omega)q(z)}dz$$

This measure varies between 0 and 1, where 0 indicates complete mismatch, and 1 indicates a complete match. This is the matching measure used in [6].

A third criterion, called the "simple criterion" is now introduced. The reason for this is not any analytical superiority over the measures already discussed; in fact, it is an inferior measure. Rather, the issue is that although the flows for all three measures will be derived, the Kullback-Leibler and Bhattacharyya flows are both quite time-consuming in terms of implementation, while the simple flow is much speedier; this fact will be elaborated upon in section 3.

It would be nice if the criterion $\int p(z)q(z)dz$ could be used. This criterion is formally similar to the Bhattacharyya criterion, which makes it seem a plausible density-matching measure; it differs from the Bhattacharyya criterion in the fact that it is missing the square root, which makes it easier to the manipulate. However, the problem with using this measure is that solving

$$\max_{p(\cdot)} \int p(z)q(z)dz \quad \text{subject to} \quad \int p(z)dz = 1$$

does *not* yield $p^*(z) = q(z)$, in contradistinction to the solutions of analogous optimizations of the Kullback-Leibler and Bhattacharyya measures. Instead, it yields a solution $p^*(z) = \delta(z - z_0)$ where $z_0 = \text{argmax}_{z\in\mathcal{Z}}\, q(z)$. In this sense, it is not a true density-matching measure. Similarly, solution of $\max_\omega \int p(z;\omega)q(z)dz$ will lead to ω collapsing to a tiny region around a pixel x with value $Z(x) = z_1$, where $z_1 = \text{argmax}_{x\in\text{image}}\, q(Z(x))$. This is an unreasonable situation; however, noting that $p(z;\omega) = N(z;\omega)/A(\omega)$, the following related problem is proposed instead:

$$\max_{\omega} \int N(z;\omega)q(z)dz \quad \text{subject to} \quad A(\omega) \leq K$$

By thus "decoupling" the action of N and A, we again achieve the objective of non-degenerate distributions.

Note that without the constraint, the solution would be to have a region ω which encompasses the entire image, since as ω grows in area, $N(z;\omega)$ is non-decreasing for all z. By adding in the constraint, this trivial solution is eliminated. Writing the Lagrangian for this problem yields the "simple" matching criterion:

$$H(\omega) = \int N(z;\omega)q(z)dz - \lambda A(\omega)$$

where a minus sign has been inserted in front of the Lagrange multiplier for future convenience. It will turn out (see section 3) that the precise value of K will be irrelevant, and that λ can be set directly.

2.3 A Proposition Concerning Variational Derivatives

The goal of this section is to establish the validity of the following proposition, which will be useful in later computations. The proposition is given for simply-connected regions, but can be generalized.

Proposition: Let ω be an elementary region of $\mathbb{R}^2$, let $\mathbf{c} = \partial\omega$ be its boundary, and let $\Gamma(\omega) = \int_\omega \mu(x)dx$, where μ is C^1. Additionally, let $\frac{\delta\Gamma}{\delta\mathbf{c}}$ be a 2-vector whose i^{th} component is the variational derivative $\frac{\delta\Gamma}{\delta\mathbf{c}_i}$, assuming a particular parameterization for $\mathbf{c}$. Then there exists a parameterization of $\mathbf{c}$ for which

$$\frac{\delta\Gamma}{\delta\mathbf{c}} \propto \mu(\mathbf{c})\mathbf{n}$$

where $\mathbf{n}$ is the normal to $\mathbf{c}$.

Proof of Proposition: First, let us convert the expression for Γ from an area integral to a line integral using Green's Theorem. Recall that Green's Theorem states that if $P_1 : \omega \to \mathbb{R}$ and $P_2 : \omega \to \mathbb{R}$ are C^1, then

$$\int_\omega \left(\frac{\partial P_1}{\partial x_1} - \frac{\partial P_2}{\partial x_2}\right) dx_1 dx_2 = \int_{\partial\omega} P_2 dx_1 + P_1 dx_2$$

Now, let

$$P_1(x) = \frac{1}{2}\int_{-\infty}^{x_1} \mu(u_1, x_2)du_1$$

and

$$P_2(x) = -\frac{1}{2}\int_{-\infty}^{x_2} \mu(x_1, u_2)du_2$$

Since μ is C^1, so are P_1 and P_2. It is easy to verify that

$$\Gamma = \int_\omega \left(\frac{\partial P_1}{\partial x_1} - \frac{\partial P_2}{\partial x_2}\right) dx_1 dx_2$$

Thus, Green's theorem gives that

$$\begin{aligned}
\Gamma &= \int_{\partial\omega} (P_1 dx_2 + P_2 dx_1) \\
&= \int_0^1 [P_1(\mathbf{c}(s))\mathbf{c}_2'(s) + P_2(\mathbf{c}(s))\mathbf{c}_1'(s)]ds \\
&= \int_0^1 \Lambda(\mathbf{c}(s), \mathbf{c}'(s))ds
\end{aligned}$$

where in the second line, the parameterization of $\mathbf{c} = \partial\omega$ has been substituted.

In order to calculate $\frac{\delta\Gamma}{\delta\mathbf{c}}$, we may use results from the calculus of variations. In particular, it can be shown [13] that

$$\frac{\delta\Gamma}{\delta\mathbf{c}} = \frac{\partial\Lambda}{\partial\mathbf{c}} - \frac{d}{ds}\left[\frac{\partial\Lambda}{\partial\mathbf{c}'}\right]$$

Now,

$$\begin{aligned}
&\frac{\partial \Lambda}{\partial \mathbf{c}_1} - \frac{d}{ds}\left[\frac{\partial \Lambda}{\partial \mathbf{c}_1'}\right] \\
&= \frac{\partial P_1}{\partial \mathbf{c}_1}\mathbf{c}_2' + \frac{\partial P_2}{\partial \mathbf{c}_1}\mathbf{c}_1' - \frac{d}{ds}P_2 \\
&= \frac{\partial P_1}{\partial \mathbf{c}_1}\mathbf{c}_2' + \frac{\partial P_2}{\partial \mathbf{c}_1}\mathbf{c}_1' - \left[\frac{\partial P_2}{\partial \mathbf{c}_1}\mathbf{c}_1' + \frac{\partial P_2}{\partial \mathbf{c}_2}\mathbf{c}_2'\right] \\
&= \left(\frac{\partial P_1}{\partial \mathbf{c}_1} - \frac{\partial P_2}{\partial \mathbf{c}_2}\right)\mathbf{c}_2'
\end{aligned}$$

But

$$\frac{\partial P_1}{\partial \mathbf{c}_1} = \frac{1}{2}\mu(\mathbf{c}(s)), \quad \frac{\partial P_2}{\partial \mathbf{c}_2} = -\frac{1}{2}\mu(\mathbf{c}(s))$$

so that

$$\frac{\partial \Lambda}{\partial \mathbf{c}_1} - \frac{d}{ds}\left[\frac{\partial \Lambda}{\partial \mathbf{c}_1'}\right] = \mu(\mathbf{c}(s))\mathbf{c}_2'(s)$$

Similarly, it may be shown that

$$\frac{\partial \Lambda}{\partial \mathbf{c}_2} - \frac{d}{ds}\left[\frac{\partial \Lambda}{\partial \mathbf{c}_2'}\right] = -\mu(\mathbf{c}(s))\mathbf{c}_1'(s)$$

which finally leads to

$$\begin{bmatrix}\delta\Gamma/\delta\mathbf{c}_1 \\ \delta\Gamma/\delta\mathbf{c}_2\end{bmatrix} = \mu(\mathbf{c})\begin{bmatrix}\mathbf{c}_2' \\ -\mathbf{c}_1'\end{bmatrix}$$

Now, if the parameterization is in terms of normalized arc-length (i.e. arc-length divided by total arc-length), then $[\mathbf{c}_2',\ -\mathbf{c}_1]^T$ is proportional to the normal $\mathbf{n}$, so that

$$\frac{\delta\Gamma}{\delta\mathbf{c}} \propto \mu(\mathbf{c})\mathbf{n} \qquad \square$$

In the next three sections, we will use this result. In each case, we will use a gradient ascent or descent approach to find an optimum of Γ in terms of $\mathbf{c}$, i.e. a curve for which $\delta\Gamma/\delta\mathbf{c}$ is 0. As a result, we can safely ignore the positive constant of proportionality.

2.4 The Simple Flow

As we wish to maximize $H(\omega)$, we may use gradient *ascent*:

$$\frac{\partial \mathbf{c}}{\partial t} = \frac{\delta H}{\delta \mathbf{c}}$$

However,

$$\begin{aligned} H(\omega) &= \int_{\mathcal{Z}} N(z;\omega)q(z)dz - \lambda A(\omega) \\ &= \int_{\mathcal{Z}} \left[\int_{\omega} \delta(z - Z(x))dx\right] q(z)dz - \lambda \int_{\omega} dx \\ &= \int_{\omega} \left[\int_{\mathcal{Z}} \delta(z - Z(x))q(z)dz\right] dx - \lambda \int_{\omega} dx \\ &= \int_{\omega} (q(Z(x)) - \lambda)dx \end{aligned}$$

Using the proposition, we have that

$$\frac{\delta H}{\delta \mathbf{c}} = (q(Z(\mathbf{c})) - \lambda)\mathbf{n}$$

Thus, the simple flow is

$$\frac{\partial \mathbf{c}}{\partial t} = (q(Z(\mathbf{c})) - \lambda)\mathbf{n} \tag{1}$$

This equation is easy to understand. If the model density, evaluated at a particular pixel on the boundary, is larger than the threshold λ, then the curve expands to take in this pixel. In the context of a lip-tracker, we may think of the curve as expanding to include reddish pixels (for which the model density will be high), and contracting away from skin-coloured pixels in the neighbourhood of the lips (for which the model density will be low).

2.5 The Kullback-Leibler Flow

Here we wish to minimize $K(\omega)$, so that gradient *descent* is appropriate:

$$\frac{\partial \mathbf{c}}{\partial t} = -\frac{\delta K}{\delta \mathbf{c}}$$

Now,

$$\begin{aligned} K(\omega) &= \int_{\mathcal{Z}} q(z) \log \frac{q(z)}{p(z;\omega)} dz \\ &= \eta - \int_{\mathcal{Z}} q(z) \log p(z;\omega) dz \end{aligned}$$

where η is the negative differential entropy of the model distribution, and can be ignored as it does not depend of ω. Now, since $p(z;\omega) = N(z;\omega)/A(\omega)$, we may write

$$K(\omega) = \log(A(\omega)) - \int_{\mathcal{Z}} q(z) \log(N(z;\omega)) dz$$

so that

$$\frac{\delta K}{\delta \mathbf{c}} = \frac{1}{A}\frac{\delta A}{\delta \mathbf{c}} - \int_{\mathcal{Z}} q(z) \left[\frac{1}{N(z;\omega)}\frac{\delta N(z)}{\delta \mathbf{c}}\right] dz$$

Now, using the proposition,

$$A(\omega) = \int_\omega dx \;\Rightarrow\; \frac{\delta A}{\delta \mathbf{c}} = \mathbf{n}$$

Similarly,

$$N(z;\omega) = \int_\omega \delta(z - Z(x))dx \;\Rightarrow\; \frac{\delta N(z)}{\delta \mathbf{c}} = \delta(z - Z(\mathbf{c}))\mathbf{n}$$

(Note: the proposition required that μ be C^1, which $\delta(z - Z(x))$ is clearly not. To get around this problem, the following procedure may be used. For $\delta(\cdot)$, substitute a C^∞ approximation $\delta_\epsilon(\cdot)$, such that $\lim_{\epsilon\to 0}\delta_\epsilon = \delta$. For example, one could use a unit-normalized Gaussian with variance $\epsilon\mathbf{I}$, where $\mathbf{I}$ is the n-dimensional identity matrix. After the calculation is completed, the limit may be taken. Umder these assumptions, the results to be derived will not change.) Thus,

$$\begin{aligned}
\frac{\delta K}{\delta \mathbf{c}} &= \frac{\mathbf{n}}{A} - \int_{\mathcal{Z}} q(z)\left[\frac{\delta(z - Z(\mathbf{c}))\mathbf{n}}{\int_\omega \delta(z - Z(x))dx}\right]dz \\
&= \frac{\mathbf{n}}{A} - \int_{\mathcal{Z}}\left[\frac{q(z)}{N(z)}\delta(z - Z(\mathbf{c}))dz\right]\mathbf{n} \\
&= \frac{\mathbf{n}}{A} - \frac{q(Z(\mathbf{c}))}{N(Z(\mathbf{c}))}\mathbf{n} \\
&= \frac{p(Z(\mathbf{c})) - q(Z(\mathbf{c}))}{N(Z(\mathbf{c}))}\mathbf{n}
\end{aligned}$$

Thus, the Kullback-Leibler flow is given by

$$\frac{\partial \mathbf{c}}{\partial t} = \frac{q(Z(\mathbf{c})) - p(Z(\mathbf{c}))}{N(Z(\mathbf{c}))}\mathbf{n} \tag{2}$$

The intuitive meaning of this equation is clear. If the sample density, evaluated at a particular pixel on the boundary, is smaller than the model density, then the curve expands to take in this pixel. This makes sense: by taking in the pixel, the sample density for that value of z will increase, which leads to a better match between the sample density and the model density. Put another way: a lip-tracker based on the Kullback flow will expand to include a reddish pixel, and will contract away from non-reddish pixels.

This equation seems more sophisticated than the simple flow, in that it explicitly takes into account the sample density p. However, as we will see the simple flow is much more practical in terms of implementation, as it represents a pure differential equation; by contrast, the Kullback flow represents an integro-differential equation, which is much slower to implement.

2.6 The Bhattacharyya Flow

As in the case of the simple flow, we wish to maximize the Bhattacharyya measure, and thus we use gradient *ascent*:

$$\frac{\partial \mathbf{c}}{\partial t} = \frac{\delta B}{\delta \mathbf{c}}$$

The Bhattacharyya measure is given by

$$\begin{aligned} B(\omega) &= \int_{\mathcal{Z}} \sqrt{p(z;\omega)q(z)}dz \\ &= \int_{\mathcal{Z}} q^{1/2}(z)\frac{N^{1/2}(z;\omega)}{A^{1/2}(\omega)}dz \end{aligned}$$

so that

$$\begin{aligned} \frac{\delta B}{\delta \mathbf{c}} &= \int_{\mathcal{Z}} \frac{q^{1/2}(z)}{A(\omega)} \left[A^{1/2}(\omega)\frac{1}{2}N^{-1/2}(z;\omega)\frac{\delta N}{\delta \mathbf{c}} \right. \\ &\quad \left. -N^{1/2}(z;\omega)\frac{1}{2}A^{-1/2}(\omega)\frac{\delta A}{\delta \mathbf{c}} \right] dz \\ &= \frac{\mathbf{n}}{2A(\omega)} \left[A^{1/2}(\omega) \int_{\mathcal{Z}} q^{1/2}(z)N^{-1/2}(z;\omega)\delta(z - Z(\mathbf{c}))dz \right. \\ &\quad \left. -A^{-1/2}(\omega) \int_{\mathcal{Z}} q^{1/2}(z)N^{1/2}(z;\omega)dz \right] \\ &= \frac{\mathbf{n}}{2A(\omega)} \left[A^{1/2}(\omega)q^{1/2}(Z(\mathbf{c}))N^{-1/2}(Z(\mathbf{c})) \right. \\ &\quad \left. - \int_{\mathcal{Z}} q^{1/2}(z)p^{1/2}(z;\omega)dz \right] \\ &= \frac{\mathbf{n}}{2A(\omega)} \left[\frac{q^{1/2}(Z(\mathbf{c}))}{p^{1/2}(Z(\mathbf{c}))} - B(p,q) \right] \end{aligned}$$

In the foregoing, many of the arguments used in deriving the Kullback-Leibler flow have been recycled. We finally have that the Bhattacharyya flow is given by

$$\frac{\partial \mathbf{c}}{\partial t} = \frac{1}{2A(\omega)} \left[\frac{q^{1/2}(Z(\mathbf{c}))}{p^{1/2}(Z(\mathbf{c}))} - B(p,q) \right] \mathbf{n} \tag{3}$$

This equation has a similar intuitive understanding as the Kullback flow, except insofar as it is somewhat more aggressive in expanding (since $B(p,q)$ is less than 1, as long as p and q are not equal). Like the Kullback flow, the Bhattacharyya flow is also an integro-differential equation.

3 Implementation

In both of the experiments of section 4, the simple flow is implemented. As has already been alluded to, the reasons for this choice are that the simple flow can operate much faster than either the Kullback-Leibler flow or the Bhattacharyya flow. To see this, reexamine these two latter flows (see equations (2) and (3)). Note that in both cases, the right side of the equation contains integral quantities: p and N in the case of the Kullback-Leibler flow, p, B, and A in the case of the Bhattacharyya flow. These quantities may only be evaluated by performing various integrations over the entire region ω. Thus, technically both of these flows actually represent integro-differential equations. In any numerical scheme, the evaluation of these integrals at each time-step will be costly. By

contrast, the simple flow (see equation (1)) is such that the right side of the equation depends only on the position of the curve; it is a pure differential equation.

There are a variety of issues which arise in the implementation of the simple flow. First of all, it is natural to do this implementation using the level-set method [14]. Aside from all of the well-known advantages possessed by this framework, such as the ability to handle cusps, corners, and changes in topology, this particular evolution equation is naturally a flow in the normal direction, a necessary prerequisite for the application of this framework.

A second issue arises from the fact that images are actually discrete-valued, rather than continuous-valued; there is therefore the question of how to best approximate the densities. As in [6], we use histograms. In particular, we can learn the model density q in a straightforward way by taking an image (or multiple images) with an object (objects) drawn from the class of interest, and then finding the histogram within the object. Empirically, q has small support, and is 0 in most ($\approx 99\%$) of bins.

A third issue concerns the choice of the single parameter associated with the simple flow, namely λ. There is a simple way to choose λ, based on the previous observation that the model density q has small support. In particular, if we choose

$$\lambda = \frac{1}{2} \min_{z \in Z : q(z) > 0} q(z)$$

then any point p on the curve such that $q(Z(p)) > 0$ will flow outwards, and any point such that $q(Z(p)) = 0$ will flow inwards. That is, in propagating the simple flow, we will try to maintain the support of the model density.

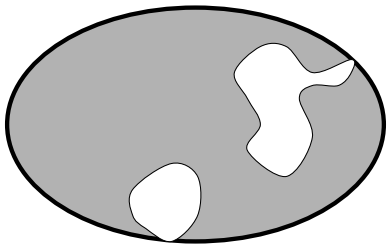

Figure 2: The simple flow can lead to highly concave regions. The black outline represents the boundary of the true location of the object; the gray region indicates the subset of this region whose z-values are in the support of the model density. Without the addition of the min-curvature flow, the gray region would be picked out by the simple flow.

In order to make the tracker run sufficiently quickly, we would like to choose the time-step Δt to be as large as possible. Since Δt must satisfy a CFL condition, it is actually useful to make the speed function more uniform, namely $F(\mathbf{c}) = q(Z(\mathbf{c})) - \lambda$ is replaced by

$$F(\mathbf{c}) = \text{sign}(q(Z(\mathbf{c})) - \lambda)$$

Finally, the following problem occurs. Due to inevitable changes in illumination, as time proceeds the density of the tracked object p diverges from the model density q. That is to say, the region containing the tracked object ceases to be constituted entirely by pixels whose z-value is in the support of the model density q. As a result, the tendency is for the simple flow to lead to highly

Figure 3: Tracking a walker. Left to right, and then down: (a) frame 5; (b) frame 38; (c) frame 75; (d) frame 105; (e) frame 133; (f) frame 160.

concave regions; this is illustrated in figure 2. In order to overcome this problem, an extra curvature term is added to eliminated some of the concavity, namely:

$$F(\mathbf{c}) = \text{sign}(q(Z(\mathbf{c})) - \lambda) - \epsilon \min\{\kappa(\mathbf{c}), 0\}$$

This min-curvature flow has the effect of popping out highly concave regions, at the cost of introducing a second parameter ϵ. However, this parameter has a useful physical interpretation. The conditions under which a concave part of the curve is popped out are given by

$$F(\mathbf{c}) > 0 \Leftrightarrow -1 - \epsilon\kappa(\mathbf{c}) > 0 \Leftrightarrow R(\mathbf{c}) < \epsilon$$

where $R(\mathbf{c})$ is the radius of curvature at the point $\mathbf{c}$ on the curve. That is, all points with negative curvature and with feature size (i.e. radius of curvature) of smaller than ϵ pixels will be removed. It is useful to have this physical interpretation, as many objects do possess some concavity, but only up to a certain scale; any concave region which is smaller than this scale must be an artifact of the flow itself (rather than a property of the object), and is therefore removed. Thus, as long as the rough scale of the allowed concavity is known, then the algorithm will work reasonably.

4 Results and Conclusions

The density tracker was run on two sequences: a walking individual, and a flexing and translating finger. The scene of the walker presents a more cluttered background, while the finger tests the ability of the tracker to follow the motion of a non-rigid object. Before examining the results of the experiments, it is worth discussing the finger sequence in the context of other tracking algorithms.

The condensation algorithm has been shown to have difficulty in following the motion of the finger in this sequence without a great deal of explicit modelling (e.g. modelling the finger as a

Figure 4: Tracking a flexing finger. Left to right, and then down: (a) frame 1; (b) frame 6; (c) frame 12; (d) frame 17; (e) frame 22; (f) frame 48.

robotic arm); for images, see [7]. The mean-shift tracker of Comaniciu *et al.* [6] also cannot track the finger, as it is not designed to deal with flexible shapes; rather, the tracking area is always forced to be an ellipse. It could possibly track an elliptical subsection of the finger; however, the goal of tracking is generally to follow an *entire* object as closely as possible. Finally, it is worth mentioning the tracker of Paragios and Deriche [15]. As was discussed in section 1, no explicit comparison can be made between this tracker and the current tracker, as the algorithm of Paragios and Deriche requires information about the background in order to segment or track. We assume that no such information is available, an assumption we feel is justified by the fact that the goal is to design a *general-purpose* tracker; such a tracker must be able to accomodate arbitrary backgrounds, without any prior knowledge.

The following are the salient characteristics of the walker experiment.

- The sequence was of length 160 frames (= 5.3 seconds at 30 Hz).
- The photometric variable z was taken to be colour, specified in HSV coordinates, normalized to run from 0 to 255.
- The model density q was built as a histogram out of the walker's face taken from a single image (which was not part of the running sequence). The bins were taken to be $8 \times 8 \times 8$, leading to $(256/8)^3$ = 32,768 bins.
- The concavity parameter ϵ was set at 20.
- Using a Pentium III machine operating at 933 MHz and an uncompiled MATLAB implementation, each frame required approximately 1 minute of running time.

Figure 5: The density spills over. Left to right: (a) frame 97; (b) frame 98; (c) frame 101, in which the density has ceased to spill over.

The results are shown in figure 3, which demonstrates the algorithm's proficiency in following the walker. One interesting artifact of the algorithm, more easily seen in a video than in still frames, is jitter: despite finding the walker's head correctly in all all frames, there is tendency for the precise shape of the contour to change on a frame by frame basis. This is due to the fact that there are no dynamical considerations incorporated into this algorithm. A second artifact may be labelled "density spill-over," and is illustrated in figure 5. Certain colours outside of the object of interest may match those inside the object of interest. This occurs in the case of the walker, as he walks by light coloured bookshelves and books, which match the colours of his face. This occurs in 3 frames of the sequence (frames 96-98), and has no long-term effect, as is illustrated in the final frame of figure 5.

The following are the salient characteristics of the finger experiment.

- The sequence was of length 270 frames (= 9.0 seconds at 30 Hz).
- The photometric variable z was taken to be colour, specified in RGB coordinates, normalized to run from 0 to 255.
- The model density q was built as a histogram out of an example finger taken from a single image (which was not part of the running sequence). The bins were taken to be $8 \times 8 \times 8$, leading to $(256/8)^3$ = 32,768 bins.
- The concavity parameter ϵ was set at 10.
- Using a Pentium III machine operating at 933 MHz and an uncompiled MATLAB implementation, each frame required approximately 2 minutes of running time.

The results are shown for flexing in figure 4, and translation in figure 6. Note that despite using RGB colour coordinates (instead of the generally preferred HSV coordinates which were used in the walker experiment), the algorithm performs quite well. Jitter is less evident in this sequence, and no density spill-over occurs, as the background colours are clearly quite different from the finger's colour.

There are several directions for future research. First, we will investigate using the Kullback and Bhattacharyya flows for tracking. In order to use these two flows, which are integro-differential equations, methods for speeding up their implementations will have to be discovered. Second, more complex measures of z, the photometric variable, will be used. Possibilities include texture, or

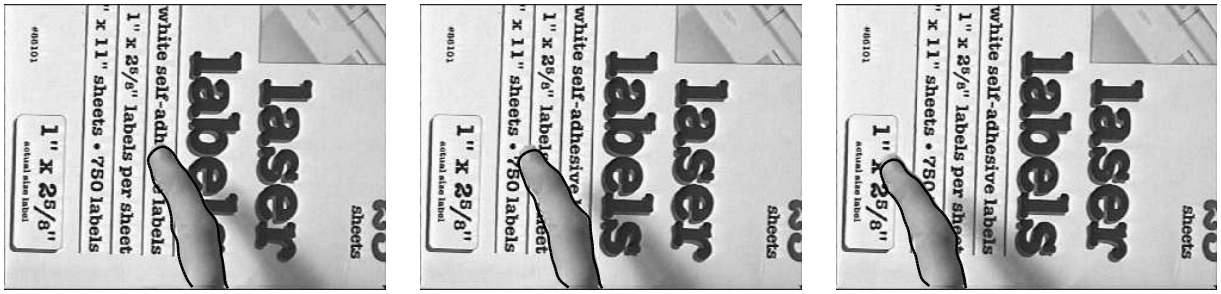

Figure 6: Tracking a translating finger. Left to right: (a) frame 227; (b) frame 237; (c) frame 257.

possibly a neighbourhood of texture vectors, in order to effectively capture a Markov Random Field type of structure. Finally, and most importantly, attempts will be made to incorporate geometric considerations into the algorithm. The experiments presented in this paper demonstrate the fact that photometric variables, by themselves, can sometimes be enough to guide a tracker; however, in order to increase both robustness and speed, the use of geometric variables are vital. The challenge is to find a way of incorporating geometry within the framework presented in this paper.

References

[1] A. Amini, S. Tehrani, and T. Weymouth. Using dynamic programming for minimizing the enegy of active contours in the presence of hard constraints. In *Proc. Int. Conf. Computer Vision*, pages 95–99, 1988.

[2] A. Blake and M. Isard. Condensation - conditional density propagation for visual tracking. *Int. J. of Comput. Vis.*, 29(1):5–28, 1998.

[3] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numer. Math.*, 66:1–31, 1993.

[4] V. Caselles, R. Kimmel, and G. Sapiro. On geodesic active contours. *Int. J. Comput. Vis.*, 22(1):61–79, 1997.

[5] T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Trans. Image Proc.*, 10(2):266–277, 2001.

[6] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, volume 2, pages 142–149, 2000.

[7] D. Freedman and M. Brandstein. Provably fast algorithms for contour tracking. In *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, volume 1, pages 139–144, 2000.

[8] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. In *Proc. Int. Conf. Computer Vision*, London, June 1987.

[9] S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum. Gradient flows and geometric active contour models. In *Proc. Int. Conf. Computer Vision*, pages 810–815, Cambridge, 1995.

[10] R. Malladi, J.A. Sethian, and B.C. Vemuri. Shape modelling with front propagation: A level set approach. *IEEE Trans. Pattern Anal. Machine Intell.*, 17:158–175, 1995.

[11] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. Pure Appl. Math*, 42:577–685, 1989.

[12] N. Oliver, A.P. Pentland, and F. Berard. Lafter: Lips and face real time tracker. In *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, pages 123–129, 1997.

[13] P. Olver. *Applications of Lie Groups to Differential Equations*. Springer, New York, 1993.

[14] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulation. *J. Comput. Phys.*, 79:12–49, 1988.

[15] N. Paragios and R. Deriche. Geodesic active regions for supervised texture segmentation. In *Proc. Int. Conf. Computer Vision*, volume 2, pages 926–932, 1999.

[16] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(3):266–280, 2000.

[17] D. Terzopoulos and R. Szeliski. Tracking with kalman snakes. In A. Blake and A. Yuille, editors, *Active Vision*, pages 3–20. MIT Press, Cambridge, MA, 1992.

[18] A. Tsai, A. Yezzi, and A.S. Willsky. A curve evolution approach to smoothing and segmentation using the mumford-shah functional. In *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, volume 1, pages 119–124, 2000.

[19] G. Xu, E. Segawa, and S. Tsuji. Robust active contours with insensitive parameters. In *Proc. Int. Conf. Computer Vision*, Berlin, May 1993.

[20] J. Yang and A. Waibel. A real-time face tracer. In *Proc. IEEE* 3^{rd} *WACV*, pages 142–147, Sarasota, 1996.

[21] A. Yezzi, A. Tsai, and A.S. Willsky. Binary and ternary flows for image segmentation. In *Proc. Int. Conf. Image Processing*, volume 2, pages 1–5, 1999.

[22] A. Yezzi, A. Tsai, and A.S. Willsky. A statistical approach to snakes for bimodal and trimodal imagery. In *Proc. Int. Conf. Computer Vision*, volume 2, pages 898–903, 1999.