International Journal of Computer Vision 38(2), 173–186, 2000

Contour Tracking in Clutter: A Subset Approach

DANIEL FREEDMAN AND MICHAEL S. BRANDSTEIN
Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02138, USA
freedman@hrl.harvard.edu
msb@hrl.harvard.edu

Abstract. A new method for tracking contours of moving objects in clutter is presented. For a given object, a model of its contours is learned from training data in the form of a subset of contour space. Greater complexity is added to the contour model by analyzing rigid and non-rigid transformations of contours separately. In the course of tracking, multiple contours may be observed due to the presence of extraneous edges in the form of clutter; the learned model guides the algorithm in picking out the correct one. The algorithm, which is posed as a solution to a minimization problem, is made efficient by the use of several iterative schemes. Results applying the proposed algorithm to the tracking of a flexing finger and to a conversing individual's lips are presented.

Keywords: contour tracking, low-level vision, visual clutter, subset learning, iterative minimization, Legendre polynomials, morphological filters

1. Problem Motivation

This work addresses the tracking of moving contours in a video-stream. Specifically, given a sequence of images in which a known object of interest is in motion, the goal is to track the object's silhouette across time-varying images. This problem has been treated quite extensively in the computer vision literature, and there are several different algorithms which have been developed to address it. In this paper, a new contour tracker, the "subset-tracker," is proposed, based on a rather different philosophy than those of the existing trackers.

Contour tracking has a number of useful applications. With regard to medicine, various imaging techniques, such as magnetic resonance imaging and computer tomographic scanning, require some means of tracking contours for the automated analysis of human organ performance. An application of this kind is contained in Ayache et al. (1992). In an entirely different setting, a common military and espionage application is surveillance, which is well-suited for the algorithms of contour tracking; see, for example, Sullivan (1992). A new application has to do with the tracking of human lips. It has been noted that observation of a talker's lips may be very useful for the recognition and/or enhancement speech which has been severely degraded by noise. While the acoustic signal, or pressure wave, is of poor quality, the visual signal, or lip-motion, may be pristine. It is known that both hearing and hearing-impaired people use the observation of lips to their advantage in order to better comprehend a talker in noisy environments. This observation suggests that exploiting lip contour information has the potential to aid both in denoising speech waveforms and in recognizing noisy speech. A necessary prerequisite for such an algorithm is the ability to track a talker's lips. Recent examples of lip-tracking, used for speech analysis, can be found in Dalton et al. (1995), Mak and Allen (1994), Luettin et al. (1996), Kaucic et al. (1996) and Bregler and Konig (1994). On the basis of these potential and existing applications, it may be concluded that contour tracking is an interesting and useful problem, worthy of further study.

Section 2 of this paper briefly reviews existing approaches to the problem. Section 3 presents an overview of the difficulties presented by contour tracking, and discusses issues of terminology. Section 4 presents the details of the proposed tracking algorithm. Section 5 presents experimental results, and some conclusions are discussed in section 6.

2. Existing Tracking Algorithms

2.1. Standard Approaches to Contour Tracking

There are several existing approaches to contour tracking. The deformable template approach (Yuille et al., 1992; Lipson et al., 1990) involves finding a model parameterization for the contours of the object to be tracked, and matching this representation with successive images in the video stream in order to detect the contours of interest. The goal is to find the deformable template which best matches the image at hand, i.e., to find the values of the parameters r which minimize an energy function. The energy is the sum of two terms, $E(r; I) = E_g(r) + E_m(r; I)$ where I is the image. $E_g(r)$, the "geometric energy," takes on small values for contours whose shape is more likely to be observed in reality; $E_m(r; I)$, the "matching energy," takes on small values when the contour described by r corresponds well to a contour found in the image. Thus, minimizing the total energy of the contour $E(r; I)$ involves a balance between finding a contour which matches the image well with finding a contour which is likely to be found in practice. The problem, as it has been posed above is a static one, which corresponds to finding an object contour in a single image. For tracking contours over time, this method may be adapted to a dynamic setting by simply repeating the procedure on a frame by frame basis. If the minimization technique requires an initial condition (e.g. gradient descent), the contour from the previous frame may be used.

The elastic snake approach, first presented in Kass et al. (1987) and subsequently elaborated in Amini et al. (1998) and Xu et al. (1993) also features an energy minimization problem. Here the energy is a functional of the contour itself (rather than parameters) and has three terms. The first two terms represent elastic and tensile energy, and ensure that the snake is smooth; the third term is an image-dependent term which pushes the snake towards the feature of interest. In an effort to find contours, an image-dependent term which depends on edge strength, such as $\|\nabla I\|^2$, may be desirable. By performing gradient descent in order to minimize the functional, changing image energy (which results from having multiple images) may be accounted for; that is, the algorithm is easily adapted to solving the tracking problem.

There are several papers which present Kalman trackers; the most notable are Blake et al. (1993, 1995) and Brockett and Blake (1994). The philosophy of the Kalman tracker is the same as that of the standard Kalman filter. In particular, it requires a learned linear stochastic dynamical model which describes the evolution of the contour to be tracked. Learning takes place prior to running. Assuming that the observation of the contour has been corrupted by Gaussian noise, the conditional density of the contour given all past observations may be found, and then used to estimate the contour position.

The condensation tracker, as outlined in Blake and Isard (1988), is similar in spirit to the Kalman tracker, in that it assumes a dynamical model describing contour motion is known, and that imprecise observations are made. However, it is more general: neither the dynamical system nor the observation process need be linear. In particular, the dynamical model may be given by a probability density $p(c_t \mid c_{t-1})$, and the observation process by another density $p(\tilde{c}_t \mid c_t)$. The conditional density $p(c_t \mid \tilde{c}_t, \ldots, \tilde{c}_1)$ may be propagated forward in time using the numerical technique known as the "condensation" method; this density may then be used for estimating the current contour, for example through maximum likelihood estimation. The advantage of this approach over the Kalman filter is that the observation process given by $p(\tilde{c}_t \mid c_t)$ may now model clutter. That is, the density may be multimodal, rather than the unimodal observations employed with the Kalman filter. As a result, multiple hypotheses may be simultaneously entertained, and robustness to clutter is expected.

2.2. Deficiencies in Existing Trackers

The deformable template approach has two principal problems. First, for any given application, it must be hand-crafted; that is, if it is desired to track the motion of lips, two specific energy functions that are appropriate for lips must be designed. This process can involve a good deal of trial and error, and is a clear drawback to a general contour-tracker. Second, while the method is useful for locating contours in a single image, it is very slow at tracking contours through multiple images. The snake method circumvents the first problem; snakes will in principal find the edges of any object whose outlines are a continuous closed curve. However, the method is *too* general in the sense that its modes of detection are not tuned to the motion of any particular object. In this way, it is not particularly efficient. Furthermore, like deformable templates, elastic snakes also suffer from problems of slowness in tracking contours through multiples images.

The Kalman tracker does not suffer from any of the problems mentioned above. It has the right level of generality in that it need not be hand-crafted for any specific application and a dynamical system for the relevant class of contours can be learned. Furthermore, the process of finding observations is fast, so that the whole algorithm is much quicker than either of the above techniques. However, the Kalman tracker ignores a basic problem in tracking: the presence of clutter. Clutter can come from detected edges which are either extraneous to the object, or intrinsic to it. In either case, there are often multiple edges, and there is no single observation; rather, there are a multiplicity of observed contours. This problem is dealt with in turn by the condensation tracker, which is designed specifically to deal with clutter, and indeed shows robustness to clutter in the experimental context. It seems, therefore, that the condensation tracker should be sufficient for most tracking applications. Why, then, propose an alternative?

The major difference between the condensation approach and the proposed approach is that the former requires a dynamical model of the object being tracked, whereas the latter does not. In particular, the subset tracker only uses the more basic information concerned with the *shape* or *geometry* of the object's silhouette. The theoretical reason for focusing on this more basic information is that there are many situations in which the available training curves, which are used for learning prior to the running the algorithm, may be sufficient for learning the space in which object "lives," but are insufficient for learning the dynamics of the object. (Of course, such a theoretical assertion has no teeth without experimental results to support it; these are provided in Section 5.) Despite this difference, the subset tracker possesses the main advantages presented by the condensation approach. Like the condensation tracker, the subset tracker maintains the right level of knowledge of the particular object which it is tracking, knowledge which is learned rather than constructed by hand. Further, it too is specifically designed to be resistant to clutter. But in ignoring dynamical information, an entirely new framework is required, one which is explored in the following sections.

3. Overview, Notation and Terminology

3.1. Overview

Before entering into a detailed discussion of notation, it is worthwhile understanding the basic approach to contour tracking that will be presented in this paper. In a given frame, the previous frame's contour is taken as a starting point for searching for the edges of the new contour. At each of the sites along which search is carried out, several edges may be detected; this is due to the fact that the object being tracked is not the only object present in the scene. That is, there is clutter, and this is what makes the problem difficult. For if there were only one edge detected per site, then the contour in the current frame is effectively detected without any further computation. It is worth noting that even if the background is fairly pristine, and the only object present is the one being tracked, spurious edges may still be detected due to the inaccuracy of the edge-detection algorithms, the noise present in the image (due to quantization, blooming, and so on), and due to the fact that the object may possess many interior edges if is not monochrome. All of these difficulties may be seen in Fig. 1, which shows the image and corresponding edge-map from a male speaker. The goal of the algorithm is to decide which of the edges detected are the correct ones, that is which correspond to the contour and which are due to clutter. More specifically, the goal is to find the contour which runs through at most one edge at each site and best describes the contour of the object being tracked. The phrase "at most" is used due to the fact that it may be possible that the correct edge may not be detected at all.

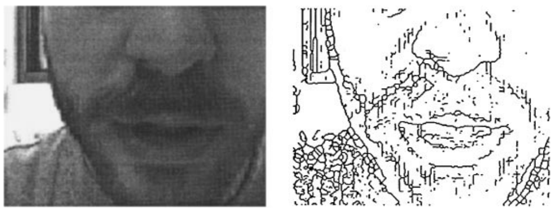
Figure 1. An image and its corresponding edge-map, illustrating clutter.

Contour tracking in clutter, then, is about deciding which of a possibly large number of potential contours discovered by edge-detection best describes the contour of the object being tracked. The proposed tracker makes this decision by finding the contour whose shape is closest to that of the object under study, where the possible shapes of this object have been learned beforehand. Before delving into the details of this decision-making procedure, it is necessary to discuss issues of notation. The two basic elements of the algorithm are edges and contours; terminology for each of these will be gone into in turn.

3.2. *Edge Notation*

The N equally-spaced points along the previous frame's contour in whose neighbourhood search is initiated are referred to as *sites*. At each site, several edges are detected; the *set* of edges detected at the nth site is denoted E_n, and an element of this set is denoted e_n. Note that E_n may be empty if no edges are detected at a particular site. An *edge-vector* is an N-vector of points one from each site $e = (e_1, \ldots, e_N)$. In fact, an edge-vector can either be seen as an N-vector if each element is taken to be a point, or a $2N$-vector if each element is taken to be a real number. Given an edge-vector, a contour may be interpolated through its constituent points; this contour is referred to as an *observed curve*. The particular method of interpolation which is used relies on the specific formalism used for describing curves, and is detailed in Appendix D. The generic observed curve is denoted ε, and the set of such objects is denoted $\mathcal{E}$.

3.3. *Contour Notation*

A contour c is a continuous function $c : [-1, 1] \to \Re^2$, which will often be written $c = (c_x, c_y)$. The space of all contours will be denoted $\mathcal{C}$. $\mathcal{C}_D$ is defined to be the space of contours which can be expressed as a pair of expansions in the first D Legendre polynomials. Of course, $\mathcal{C}_D \subset \mathcal{C}$. Recall that Legendre polynomials may be written as $P_i(s) = \sum_{j=1}^{D} G_{ij} s^{j-1}$ for $i = 1, \ldots, D$, where G is a $D \times D$ matrix whose properties are reviewed in Appendix A; they are defined on $s \in [-1, 1]$ (hence the curious choice of domain for contours). The Legendre polynomials are preferable to other sets of orthogonal functions, such as Bessel functions, due to their explicit expressions which make computations easily amenable to the basic operations of linear algebra. In addition, their orthogonality simplifies various aspects of the algorithms which would be made difficult if non-orthogonal functions, such as splines, were used. A contour in $\mathcal{C}_D$ may be identified entirely by $2D$ real numbers; namely, if

$$c_x(s) = \sum_{i=1}^{D} \gamma_{x,i} P_i(s) \quad \text{and} \quad c_y(s) = \sum_{i=1}^{D} \gamma_{y,i} P_i(s)$$

then $c(s) = (c_x(s), c_y(s))$ is completely specified by the two D-vectors γ_x and γ_y. A compact notation is $\gamma = (\gamma_x, \gamma_y)$, in which γ so specified is taken to be a $2D$ column vector. The relationship between c and γ may be written more concisely as $c(s) = (\gamma_x^T GZ(s), \gamma_y^T GZ(s))$ where $Z(s) = [s^0, s^1, \ldots, s^{D-1}]^T$. (Note: s is taken to be an affine function of arc-length; that is, if $\hat{s} \in [0, L]$ is the arc-length, then s is given by $s = 2\hat{s}/L - 1$.)

A typical object will give rise to a *class* of contours: most objects may both transform in a rigid fashion, and deform in a non-rigid fashion. Each such transformation will lead to a new contour. The goal is to find a model which will capture all of these contours. The idea which is put forward here is that the most concise way to do this is through the use of a subset C of $\mathcal{C}$, the contour space. This subset captures all of the possible contours which can arise from different deformations of the object under consideration. In fact, the subset of interest will be taken to be a subset of $\mathcal{C}_D$; in particular, the value of D is fixed at some value high enough to capture all of the detail in the contours of the object of interest. In this case, the problem of finding an appropriate subset of the space of contours has been reduced to finding a subset of $\Re^{2D}$. Such a subset may be broken down into two parts: one comprising similarity transformations, and a second comprising non-rigid deformations. It is worth making this distinction, as a well developed mathematical theory exists for similarity transformations, so these need not be learned from training contours. By contrast, non-rigid deformations vary from object to object, and so must be learned. Begin by discussing this learning procedure.

In order to learn a subset, from training contours $\{\gamma_k\}_{k=1}^K$, comprising non-rigid deformations, the Karhunen-Loeve transform (Cootes et al., 1994) may be employed. If the mean vector is $\bar{\gamma}$, and the orthogonal vectors are $\{p_i\}_{i=1}^q$ (where $q \leq 2D$ and is often much smaller than $2D$), then the subset of $\Re^{2D}$ may be taken to be the "shifted linear" space $\Gamma = \{\gamma : \gamma = \delta + \bar{\gamma},\ \delta \in \Delta\}$ where $\Delta = \text{span}\{p_1, \ldots, p_q\}$. The problem with this approach is that the space C is not compact. Compactness is a desirable property for the contour subset, as the subset should be bounded; otherwise, the subset will contain contour shapes which do not correspond to the object of interest. (This can be easily understood by examining the contour given by $\bar{\gamma} + \psi_1 p_1$ as $\psi_1 \to \infty$. Alternatively, it may be noted that all contours of relevance must be contained with the image, which has a fixed and bounded domain.) The Karhunen-Loeve procedure is thus modified to give a compact subset. Specifically, the subset Γ is emended to be

$$\Gamma = \left\{ \gamma : \gamma = \sum_{i=1}^{q} \psi_i p_i + \bar{\gamma}, \quad |\psi_i| \leq b_i \right\}$$

which is itself compact and induces a compact C. The vectors p_i and the value of q are found as before, and the bounds $\{b_i\}_{i=1}^{q}$ are found by $b_i = \max_{1\le k\le K} |\gamma_k^T p_i|$. While it might seem more natural, within the KL context, to take Γ to be an ellipsoid rather than a polytope, the latter models the contour subset sufficiently well in practice, and makes computations simpler. The following formalism usefully summarizes:

$$\begin{aligned} C &= \left\{c \in \mathcal{C}_D : c(\cdot) = \left(\gamma_x^T G Z(\cdot), \gamma_y^T G Z(\cdot)\right), \gamma \in \Gamma\right\} \\ \Gamma &= \{\gamma \in \Re^{2D} : \gamma = p\psi + \bar{\gamma}, \quad \psi \in \Psi\} \qquad (1) \\ \Psi &= \{\psi \in \Re^{q} : |\psi_i| \le b_i, \quad i = 1, \dots, q\} \end{aligned}$$

where p is the $2D \times q$ matrix whose columns are the p_i. The key insight gained through this formalism is that any contour of the object of interest is specified by a q-dimensional vector ψ which is a member of the compact set Ψ.

3.4. *Similarity Transformations*

A method for dealing with similarity transformations may now be discussed. Any euclidean similarity transformation may be represented as $\gamma \to W(\gamma)v$ where

$$W(\gamma) = \begin{bmatrix} \mu & \mathbf{0} & \gamma_x & -\gamma_y \\ \mathbf{0} & \mu & \gamma_y & \gamma_x \end{bmatrix}$$

and where $\mathbf{0} = [0, 0, \dots 0]^T$, μ is the first column of $(G^T)^{-1}$, and $v \in \Re^4$. (For a derivation of the above transformation, see Appendix B.) At any given frame t, $\mathcal{V}^t$, the set of possible similarity transformations, is $\Re^4$. However, suppose that the camera is fixed, and it is known that the object itself may only translate (including translation towards the camera) and rotate at a certain rate. Then $\mathcal{V}^t$ may be closely approximated by its superset $\tilde{\mathcal{V}}^t = \{v : \underline{v}_i^t \le v_i \le \bar{v}_i^t, i = 1, \dots, 4\}$ where the values of bounds, in terms of more basic quantities, are given in Appendix C. If there is not too much interframe similarity motion, then $\mathcal{V}^t$ will be very close to $\tilde{\mathcal{V}}^t$.

4. The Tracking Algorithm

Having finished with preliminary definitions and computations, the description of the actual tracking algorithm may be given in detail. As was alluded to in Section 3.1, the basic idea is as follows. Given the contour from the previous frame, the current frame's image is searched for edges at N equally-spaced points along the old contour. Search is in circular regions centered at each site. At each site, several edges are detected; the goal of the tracking algorithm is to sort out which of the edges correspond to the true contour. Framing the problem in this manner makes it particularly amenable to tracking in clutter.

4.1. *Basic Setup and Objective Function*

Following the edge search, data consists of the sets of edges detected at each site, $\{E_n\}_{n=1}^{N}$, and through the compounding of these, the set of observed curves, $\mathcal{E}$. The tracking algorithm is posed as the solution to a particular optimization problem, namely the minimization of the following objective function:

$$\min_{\varepsilon \in \mathcal{E}, c \in C, \omega \in \Omega} \|c - \omega\varepsilon\|$$

where Ω is the set of allowed euclidean similarity transformations, and ω is one such transformation. This requires some explanation. The idea is to find the edge-vector, that is, the list of observed edges, (at most) one from each site, whose interpolated contour is closest to the learned contour subset C. (The manner in which the these contours are interpolated is discussed in Appendix D.) Of course, as was discussed in Section 3.3, C is learned from training contours in a single orientation; thus, euclidean similarity transformations of the observed curves must be taken into account, which results in the presence of ω. Note that the norm $\|\cdot\|$ above is the L_2 norm, as has been the case all along. The particular part of the optimization solution which is of interest is the minimizing c; in particular, $\omega_{\min}^{-1} c_{\min}$ is taken to be the contour for the current frame.

This optimization problem is peculiar in several ways. First, it has both continuous and discrete elements. While the set $\mathcal{E}$ is fundamentally discrete, due to the fact that a finite number of edges are observed, both c and ω are continuous. Furthermore, $\mathcal{E}$ is generated by searching at N sites; since typically a few edges will be found at each site, $\mathcal{E}$ is thus $O(m^N)$ in size (where m is the geometric mean of the number of edges found at each site). As a result, exhaustive search over the discrete part of the problem is ruled out for any reasonable sized N. The goal of this section, therefore, will be to describe an efficient way to solve the optimization problem. The basic method will be as follows. First, an algorithm will be detailed (see Section 4.2) which enables a *local* minimum of the

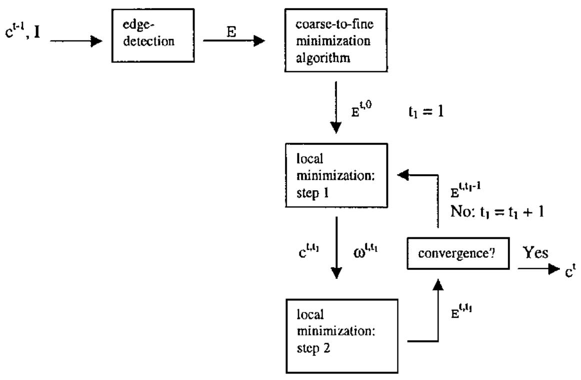

Figure 2. The contour tracking algorithm.

function to be found; this algorithm is iterative in nature, and has two distinct steps. Second, an algorithm for finding a good initial condition is explained; this initial condition is then fed into the local minimization routine, which then may result in the global minimum (and if not, something close to the global minimum). This latter procedure is referred to as the coarse-to-fine minimization algorithm, and its details are spelled out in Section 4.3. The block diagram in Fig. 2 may be useful in understanding the overall algorithm.

4.2. *The Local Minimization Algorithm*

The basic approach to minimizing the objective function is to use two separate algorithms. The first finds a local minimum; the second finds a "good" initial condition for the local minimization algorithm. In this section, the local minimization algorithm will be analyzed.

In order to find a local minimum, a gradient descent algorithm may not be employed; this is due to the partially discrete nature of the problem. In any event, gradient descent is often quite slow. Instead, an iterative procedure is proposed which is guaranteed to converge to the local minimum. In the procedure shown in Fig. 3, t_1 is the time-step of the iterations, as opposed to t which is the overall time (i.e., which frame the algorithm has reached).

```
Begin with $\varepsilon^{t,0}$ (as given by the coarse-to-fine algorithm, see section 4.3). Let $t_1 = 1$.
STEP 1: Given $\varepsilon^{t,t_1-1}$,   $(c^{t,t_1}, \omega^{t,t_1}) = \operatorname{argmin}_{c,\omega} \|c - \omega\varepsilon^{t,t_1-1}\|$
STEP 2: Given $c^{t,t_1}$ and $\omega^{t,t_1}$,   $\varepsilon^{t,t_1} = \operatorname{argmin}_{\varepsilon} \|c^{t,t_1} - \omega^{t,t_1}\varepsilon\|$
$t_1 \leftarrow t_1 + 1$. Go to STEP 1.
```

Figure 3. The local minimization algorithm.

Proof: By inspection, each step of the procedure reduces the value of the function $\|c - \omega\varepsilon\|$ from its previous value. Furthermore, the function $\|c - \omega\varepsilon\|$ is bounded from below by 0. Thus, this iterative procedure must lead to a local minimum. □

As the algorithm stands, not much has been gained, since it is not known how to actually implement step 1 or step 2. Note first that the problem may be written as

$$\min_{\varepsilon \in \mathcal{E}^t, \gamma \in \Gamma, v \in \tilde{\mathcal{V}}^t} \|\gamma - W(\varepsilon)v\|$$

using the results of Section 3.4, where now the norm $\|\cdot\|$ is the usual euclidean norm in $\Re^{2D}$. A method of attack for step 1 may now be elucidated.

4.2.1. Step 1. Step 1 may be solved itself via an iterative procedure. Again, this procedure will have its own time-scale t_2, which is subordinate to both t and t_1. However, for ease of notation, both superscripts t and t_1 will be suppressed. The algorithm can be expressed using pseudo-code, as is shown in Fig. 4.

The proof that this procedure does indeed lead to the required minimum is included in Appendix E. Two comments are in order. First, due to the fact that the iterative procedure leads to a global minimum, the initial conditions in γ and v are relevant only insofar as the speed of convergence is concerned. Second, it may now be clear why the use of orthogonal polynomials is quite helpful. Solving the problem $\min_{\gamma \in \Gamma, v \in \tilde{\mathcal{V}}^t} \|\gamma - W(\varepsilon^{t,t_1-1})v\|$ would be made much more complex if the constraints on ψ were not $|\psi_i| \le b_i$, but rather $\underline{b} \le A\psi \le \bar{b}$.

4.2.2. Step 2. Step 2 may now be discussed. As it stands, step 2 seems to require an exhaustive search, in that all interpolated edge-vectors must be tried out. This is an $O(m^N)$ operation, and is thus not reasonable. Instead, the following approach may be used:

$$\begin{aligned}
\|c - \omega\varepsilon\|^2 &= \int_{-1}^{1} \|c(s) - (\omega\varepsilon)(s)\|^2\, ds \\
&= \sum_{n=1}^{N} \int_{-1+2\frac{n-1}{N}}^{-1+2\frac{n}{N}} \|c(s) - (\omega\varepsilon)(s)\|^2\, ds \\
&\approx \frac{2}{N} \sum_{n=1}^{N} \|c(s_n) - \omega e_n\|^2
\end{aligned}$$

```
Given: $\varepsilon^0$, $\gamma^0$, $v^0$.

       $W = W(\varepsilon^0)$
       $\psi^0 = p^T(\gamma^0 - \bar{\gamma})$
       $U = p^T p$ ($U_i$ denotes the $i^{th}$ row of $U$)
Let:   $R = W^T W$ ($R_i$ denotes the $i^{th}$ row of $R$)
       Notation: $\langle x \rangle_a^b \equiv \begin{cases} x & \text{if } a \le x \le b \\ a & \text{if } x < a \\ b & \text{if } x > b \end{cases}$

$t_2 = 1$
do
    for $i$ = 1 to $q$
        $u = Wv^{t_2-1} - \bar{\gamma}$
        $\psi_i^{t_2} = \langle (u_i - U_i \psi^{t_2-1} + U_{ii}\psi_i^{t_2-1})/U_{ii} \rangle_{-b_i}^{b_i}$
        $t_2 \leftarrow t_2 + 1$
    end
    for $j$ = 1 to 4
        $r = W(p\psi^{t_2-1} + \bar{\gamma})$
        $v_j^{t_2} = \langle (r_j - R_j v^{t_2-1} + R_{jj} v_j^{t_2-1})/R_{jj} \rangle_{\underline{v}_j^t}^{\bar{v}_j^t}$
        $t_2 \leftarrow t_2 + 1$
    end
until $\psi$ and $v$ have converged
```

Figure 4. The step 1 algorithm.

where $s_n = -1 + 2(n - 1/2)/N$ and e_n is the nth point of the edge-vector $e = (e_1, \ldots, e_N)$ from which ε was interpolated. The approximation is valid as long as N is large, so that c does not vary much too much over the intervals of length $2/N$. If the approximation is valid, the problem

$$\min_{\varepsilon \in \mathcal{E}} \|c - \omega\varepsilon\|$$

is equivalent to

$$\min_{e_1 \in E_1, \ldots, e_N \in E_N} \frac{2}{N} \sum_{n=1}^{N} \|c(s_n) - \omega e_n\|^2$$
$$= \frac{2}{N} \sum_{n=1}^{N} \left[\min_{e_n \in E_n} \|c(s_n) - \omega e_n\|^2 \right]$$

in that both problems will give rise to the same edge-vector e and hence the same observed curve ε. The latter problem is an $O(N)$ problem: it is a simple matter to determine, at each site, which of the edges (properly oriented by ω) is the closest to the point $c(s_n)$. Of course all edges must be transformed by ω beforehand, but this is also just an $O(N)$ operation. Thus, it has been demonstrated that both steps of the iterative procedure for finding a local minimum can be implemented efficiently.

4.3. The Coarse-to-Fine Minimization Algorithm

Local minimization of the objective function is useful, but as is always the case in minimization problems, what is really desired is the global minimum. What is suggested is not a way to reach this global minimum with certainty, but rather, a way to cleverly pick an initial condition $\varepsilon^{t,0}$ for the local minimization routine. This initial condition picking is referred to as the coarse-to-fine (CTF) minimization algorithm.

The idea of the CTF algorithm is to initially pick edges at sites which are spaced far apart, and to subsequently fill in the sites which lie between the initial sites. In this sense, the algorithm is indeed coarse-to-fine: it begins by finding a reasonable guess for the contour on a coarse scale, and then refines this by filling in the finer scales. Suppose there are integers d and M such that $N = d^M + 1$. There are M consecutive stages of decisions. Let $\chi_{ij}^m = id^{M-m} + (j-1)d^{M-m+1} + 1$. In stage 1, a single decision is made, namely, which edges at the sites $\{\chi_{i1}^1\}_{i=0}^d$ should be selected. The edges are selected as $\Xi(\chi_{01}^1, \ldots, \chi_{d1}^1)$, where Ξ is defined by

$$\Xi(\pi_1, \ldots, \pi_r)$$
$$= \operatorname*{argmin}_{\{e_{\pi_i} \in E_{\pi_i}\}_{i=1}^r} \left(\min_{h \in H(\pi_1, \ldots, \pi_r)} \left\| [e_{\pi_1}; \ldots; e_{\pi_r}] - h \right\| \right)$$

In the above definition, $\pi_1, \ldots, \pi_r$ are any r sites, and $H(\pi_1, \ldots, \pi_r)$ is the set of all possible configurations of points at such sites, as can be derived from the learned subset C (see Appendix F for more details). That is, Ξ is the set of points $e_{\pi_1}, \ldots, e_{\pi_r}$ at sites $\pi_1, \ldots, \pi_r$ which, out of all of the observed points at the relevant sites, are closest to those which have been learned. Using knowledge of H, the inner minimization problem can be solved as is shown in Appendix F.

In the mth stage, $m = 2, \ldots, M$, there are d^{m-1} decisions made. The jth such decisions involves selecting edges at the sites $\{\chi_{ij}^m\}_{i=1}^{d-1}$, as $\hat{\Xi}(\chi_{1j}^m, \ldots, \chi_{d-1,j}^m$;

$\chi_{0j}^m, \chi_{dj}^m)$, where $\hat{\Xi}$ is defined by

$$\begin{aligned}&\hat{\Xi}(\pi_2, \ldots, \pi_{r-1}; \pi_1, \pi_r)\\ &\quad = \underset{\{e_{\pi_i} \in E_{\pi_i}\}_{i=2}^{r-1}}{\operatorname{argmin}} \left(\min_{h \in H(\pi_1, \ldots, \pi_r)} \left\| \left[e_{\pi_1}; \ldots; e_{\pi_r} \right] - h \right\| \right)\end{aligned}$$

That is, e_{π_1} and e_{π_r} are fixed, as they have been selected in a previous stage. Thus edges are selected in a coarse-to-fine manner, and there are a total of $\sum_{i=0}^{M-1} d^i = (d^M - 1)/(d-1) < N/(d-1)$ decisions. Since the time for each decision is $O(1)$, the algorithm's overall running time is $O(N)$.

A numerical example will make this more concrete. Suppose $N = 4^3 + 1 = 65$ sites; then there are 3 stages of decisions. In the first stage, a single decision is made: the choice of edges at sites 1, 17, 33, 49, and 65 simultaneously. They are chosen as $\Xi(1, 17, 33, 49, 65)$. The edges chosen at these sites are then fixed for the remainder of the procedure, i.e. for the final two stages. In the second stage, there are four decisions made. The first decision involves the choice of edges at sites 5, 9, and 13, holding the edges chosen at sites 1 and 17 (from the first stage) constant. These sites are chosen as $\hat{\Xi}(5, 9, 13; 1, 17)$. The second decision involves the choice of edges at sites 21, 25, 29, holding the edges chosen at sites 17 and 33 constant. The third decision involves the choice of edges at sites 37, 41, 45 with edges at sites 33 and 49 as constant; the fourth decision picks out edges at sites 53, 57, and 61 with edges at sites 49 and 65 constant. For the third stage, the edges picked out in the first two stages are held fixed. The third stage involves sixteen decisions. The first of these picks out edges at sites 2, 3, and 4 holding the edges at sites 1 and 5 as constant. The sixteenth picks out edges at site 62, 63, and 64 while holding the edges at sites 61 and 65 as constant. In this way, edges at all 65 sites are chosen in a coarse to fine manner.

4.4. Practical Considerations

The following practical issues arise in the implementation of the algorithm. Edge-detection takes place in two steps. First, $\nabla^2 G$ with threshholding is performed, yielding a binary image. Second, morphological thinning is performed on the binary image, leading to the edge-map. The use of the morphological operation makes the whole process quite invariant to the choice of threshhold. Many algorithms, such as Blake et al. (1993) and Blake and Isard (1998), use edge search which is normal to the old contour, in order to avoid the "aperture problem." However, normal search along these lines frequently results in missed edges, particularly in regions of high curvature, or if the frame rate is relatively low. In this work, the edge search is conducted in circular regions around N equally-spaced points of the old contour. The size of the search region varies at each point, and is learned from the training contours: the largest distance between the same site in any successive pair of training contours is taken to be the radius of the search region for that site. This value can be enlarged slightly to take into account the maximum allowed interframe euclidean similarity motion, but in practice this makes little difference.

A problem which arises in learning C is the following. It is desired that the object be in one specific orientation, so that no euclidean similarity transformations are captured by the learned subset (only non-rigid deformations are so captured). However, this may be difficult to ensure in the case of any given application. For example, in the application to be discussed in the following section, a speaker's lips are to be tracked. In order to acquire the training contours without extra motion due to euclidean similarity transformations, the speaker must keep her head still at all times; practically, this is quite difficult. However, it is not unreasonable for this to occur over a short time-period, say the first two seconds. Suppose, then, there are K training contours, of which the first $\bar{K}$ are in a fixed orientation. In order to transform the frames $k > \bar{K}$ so that they are in the same orientation as the first K, the following problem is solved:

$$\gamma_k^{tr} = \min_{v \in \Re^4, 1 \le k' \le \bar{K}} \|\gamma_{k'} - W(\gamma_k) v\| = \min_{1 \le k' \le \bar{K}} \|Q(\gamma_k)\gamma_{k'}\|$$

where $Q = I - W(W^T W)^{-1} W^T$.

Another consideration has to do with issues of speed in implementing the step 1 algorithm. Rather than finding v initially, the algorithm assumes that v is the same as it was last frame, and focuses on finding c. Once this has been done, c and v are found simultaneously. A similar operation is performed in the case of CTF minimization: the euclidean similarity parameters are taken to have their values from the previous frame.

An interesting issue may arise in the detection of edges. It is possible that at a given site, no edges are detected; this can be due to the fact that the threshhold for edge-detection may be set too high. In this case, there is no problem, since no edge will have to be chosen at that particular site. However, a problem can arise if the correct edge is undetected, but incorrect edges are. In this case, the optimal thing is to select *no* edge at that particular site. In step 2 of the local minimization routine, when is such a "null" edge selected? The criterion that is employed is the following: if interpolating between the edges chosen at sites $n-1$ and $n+1$ yields a choice which is closer to $c(s_n)$ than any of the observed edges in E_n do, then the edge at site n is chosen to be null. The difficulty is that now, edges at multiple sites must be selected simultaneously. This problem may be dealt with through the use of a dynamic programming algorithm.

5. Experimental Results

Three sets of results are presented to illustrate the effectiveness of the proposed tracker; a summary is given in Table 1. In the first experiment, the lips of a female speaker were tracked. The speaker was wearing lipstick in order to highlight the lips; in addition, the edge-map was generated from the green portion of the RGB colour image to further improve contrast. Nonetheless, the edges of lips are difficult to detect, as can be seen in Fig. 5. Clutter in this case is due to the fact that a low edge-detection threshhold must be employed, in order to actually find the lip-edges; in so doing, many extraneous (often spurious) edges are detected as well. Furthermore, over the relevant search range the lips interfere with each other. Nonetheless, the tracker succeeds, as can be seen in Fig. 5. Indeed, the tracker is correct 94% of the time, and tracks for a 10 second sequence. Particularly important is that the tracker is able to recover from errors; see Fig. 6. No results are available on the speed of the algorithm; this is because the algorithm has been implemented in MATLAB rather than C, and thus the execution is considerably slower than it would be otherwise (due to the fact that MATLAB is interpreted).

In the second experiment, the lips of a male speaker were tracked. Two aspects of this experiment differentiate it from the previous one. First, there is far more clutter: the speaker is not wearing lipstick, possesses a

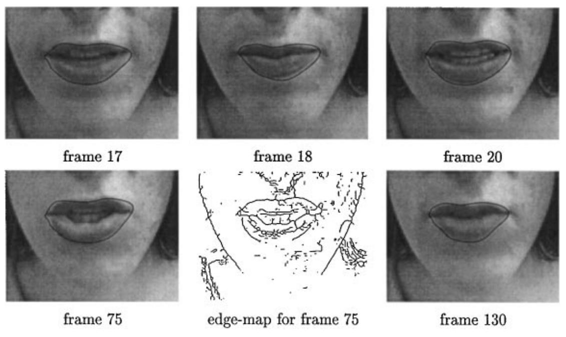

Figure 5. Tracking a speaker's lips.

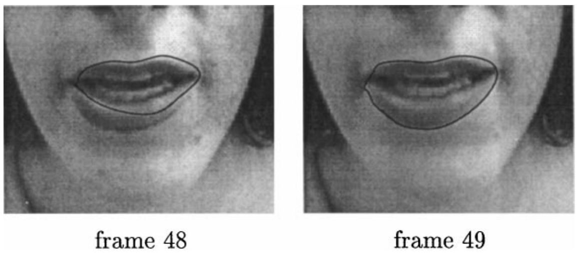

Figure 6. Recovering from mistakes.

beard and moustache, and is filmed in relatively poor lighting. Second, the training sequence from the first experiment is used to train the tracker. The reason for the latter is that a successful outcome would hold out the possibility that the tracker need not be retrained in the case of each particular speaker (for, say, audio-visual speech recognition), and would thus be more robust. Given these two differentiating factors, it would be expected that the tracker would not perform as well as in the first experiment, and indeed it does not. The contour estimates are not nearly as crisp, and the tracker makes more mistakes; see Fig. 7. Nonetheless, the tracker is able to follow the lips of the male speaker for the entire 6.6 second sequence.

In the third experiment, a moving finger is tracked. Clutter is in the form of both the background writing (much of which is small, and therefore leads to many extraneous edges) as well as the self-clutter of the doubled over finger. The motion of the finger illustrates two

Table 1. Summary of the experiments.

Experiment	N	D	Video rate	Resolution	Training sequence	Running sequence
Lips 1	80	20	13 Hz	320 by 240	200 frames = 15.4 s	130 frames = 10.0 s
Lips 2	80	20	13 Hz	320 by 240	(same as Lips 1)	86 frames = 6.6 s
Finger	80	20	30 Hz	320 by 240	81 frames = 2.7 s	202 frames = 6.7 s

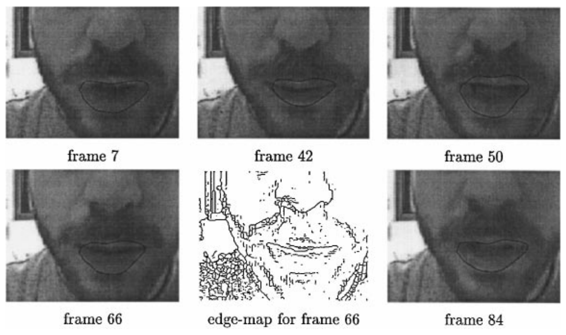

Figure 7. Tracking a more challenging speaker's lips.

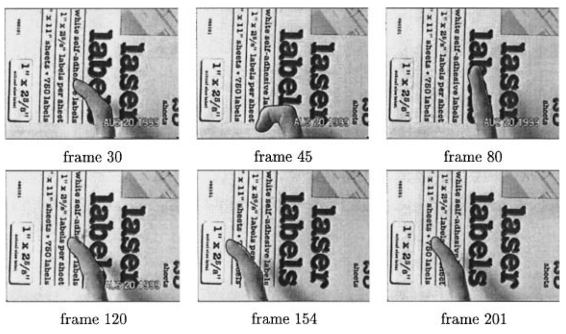

Figure 8. Tracking a flexing and translating finger.

different kinds of tracking: flexing, which is a highly nonrigid type of motion, and translation. Furthermore, the motion is relatively fast (flexing takes just over half a second). The tracker sucessfully follows the finger for 202 frames, or 6.7 seconds. Results are shown in Fig. 8.

For purposes of comparison, the condensation tracker was also run on part of the third sequence. In particular, the condensation tracker was trained on the same sequence as the subset tracker, and the reduced dimension derived from applying the Karhunen Loeve transform (in this case, ten dimensions) was used as the space to learn the dynamical model for flexing. Results of running the condensation algorithm on the flexing portion of the finger motion are shown in Fig. 9, with the corresponding frames of the results from the subset tracker shown in Fig. 10. Although both trackers succeed in tracking for the entire 24 frame (0.8 second) sequence, the contours from the subset tracker are demonstrably clearer and crisper than those from the condensation tracker. Further insight into a comparison of the two trackers is obtained by examining a 20 frame (0.7 second) sequence in which the finger is

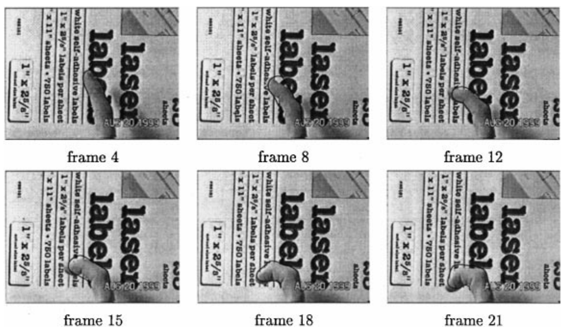

Figure 9. Condensation tracker results.

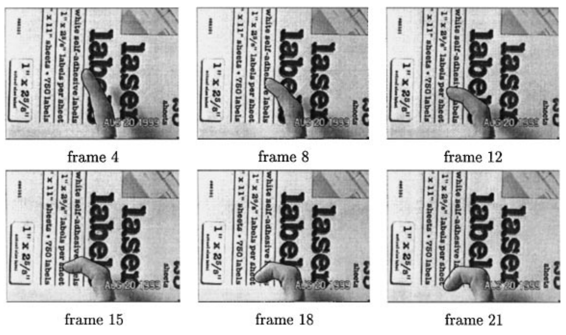

Figure 10. Subset tracker results.

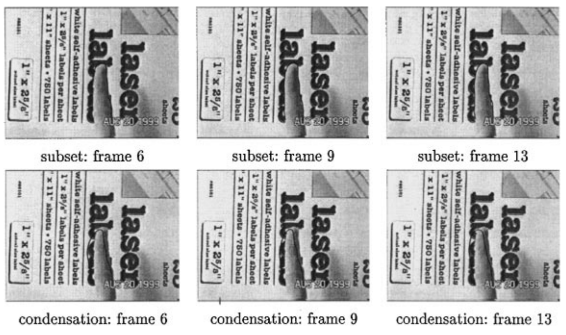

Figure 11. Comparing results on a still finger.

almost completely still. As can be seen in Fig. 11, the subset tracker is successful in finding the static finger; the condensation tracker, by contrast, is much less successful. In particular, while the condensation tracker never entirely loses lock, it gives results which do not correspond very closely to the finger's true silhouette. This is due to the fact that the dynamical model used in condensation is learned for a flexing motion, and no "pause" motion is included in the training sequence. While it may be argued that a dynamical model could include both types of motion (indeed, possibly such a switching model could even be learned from a larger training sequence), this misses the point. The advantage of an algorithm which makes no use of dynamical models is *precisely* that the exact type of motion that will be encountered in a given application often cannot be anticipated. If all modes of motion could be entirely anticipated, then the tracking problem would be a much simpler one. In this case, one very simple non-learned mode, namely no motion at all, was tracked; the condensation tracker is unable to deal with this, while the subset tracker, which does not rely on dynamics, is successful. (As was noted above, speed results are not available due to the use of MATLAB rather than C for implementation. In this case, the subset tracker was approximately three times faster per frame than the condensation tracker; however, it is not known whether this difference reflects any real speed considerations, or whether it is simply due to MATLAB artifacts.)

The objection may be raised that a dynamical model is actually built into the assumptions of the subset tracker, namely, that the interframe euclidean similarity motion is restricted to lie in a certain set $\tilde{\mathcal{V}}^t$ (see Section 3.4). However, this is simply a restriction of the form $\|\text{velocity}\| \leq$ some constant; this is a very mild, non-restrictive form of dynamical model. Any tracker must have some dynamical assumptions built in: the object is assumed not to move all that much between frames, and is assumed to stay within the image. These assumptions, however, do not come close to defining a full-fledged dynamical model of the type that is required for effective tracking using a condensation-type algorithm.

Full video sequences of the first and third experiments can be viewed at web-site http://himmel.hrl.harvard. edu/daniel/research.html.

6. Summary and Conclusions

A new approach to tracking the contours of a moving object has been presented. In this approach, a subset of curve space is learned beforehand, and it is this type of learning which differentiates the current tracker from other trackers. In particular, no dynamical knowledge is assumed. The tracking problem is posed as a minimization problem, which attempts to find the curve, constructed from edge-points in the image, which is closest to the learned set of curves. In this way, the algorithm chooses a curve in the image which is most like the object being tracked. Difficulties in solving this optimization problem arise due to the combined discrete (observed curve)—continuous (learned curve) nature of the problem, as well as the fact that the discrete set is huge. These problems are circumvented through the use of a fast, iterative, local minimization algorithm, as well as a coarse-to-fine algorithm for picking out a reasonable initial condition for the local minimization routine. Results demonstrate the efficacy of the tracker in several different experiments; further, advantages of not relying on a learned dynamical model are illustrated, in the superior performance of the subset tracker over a condensation tracker.

There are several directions for future research. First, a more general learning method, in which the learned subset of a curve space is represented as a generic, finite-dimensional manifold, could be developed. This would allow for more accurate shape representation than is possible with the current method (a polytope subset of a linear space derived from applying the KLT). Second, a global optimization algorithm might be developed, thereby obviating the need for the coarse-to-fine algorithm. The latter is a reasonable heuristic, but has no theoretical guarantees attached to it. A related topic for research would be development of complexity bounds on the efficiency of the algorithm. Finally, it might be useful to develop a method for automatically setting the threshhold level in the edge-detection routines, thereby allowing the tracker to adapt to changing lighting conditions.

Appendix A: Properties of Legendre Polynomials

The Legendre polynomials are denoted $P_i(s)$, $i = 1, 2, \ldots,$ and are defined on $s \in [-1, 1]$. Note that the convention that is used here is slightly different from the standard convention. Ordinarily, the polynomial index begins at 0; here the index is shifted up by one. In addition, the polynomials are taken to be normalized to unity norm, also counter to convention. The $D \times D$ matrix G contains all of the relevant information about the first D polynomials, as they are defined by

$$P_i(s) = \sum_{j=1}^{D} G_{ij} s^{j-1} \quad i = 1, \ldots, D$$

G is defined by the following set of equations: let F be a $D \times D$ matrix, with

$$F_{11} = 1$$
$$F_{ii} = a_i F_{i-1,i-1} \quad i > 1$$

$$F_{i,i-1} = 0 \quad i > 1$$
$$F_{i,j-2} = b_{ij} F_{ij} \quad j = i, i-1, \ldots, 3 \text{ or } 4$$
$$F_{ij} = 0 \quad j > i$$

where $a_i = \frac{2i-3}{i-1}$ and $b_{ij} = \frac{(j-2)(j-1)}{(j-3)(j-2)-(i-1)i}$. Then G is given by

$$G_{ij} = \sqrt{\frac{2i-1}{2}} F_{ij}$$

Appendix B: Derivation of the Euclidean Similarity Transformation for Legendre Polynomials

The euclidean similarity transformation for a point in $\Re^2$, $P = [x \;\; y]^T$, is given by

$$P \to W_p(P)v$$

where

$$W_p(P) = \begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$$

with $v \in \Re^4$; see, for example, Blake (1998a). Letting the point of interest be $P = [\gamma_x^T GZ(s) \gamma_y^T GZ(s)]$, then the x-coordinate transforms as

$$\hat{\gamma}^T GZ(s) = v_1 + \big(\gamma_x^T GZ(s)\big)v_3 + \big(\gamma_x^T GZ(s)\big)v_4$$

This gives that

$$\sum_{j=1}^{D} \left(\sum_{i=1}^{D} G_{ij} \hat{\gamma}_{x,i} \right) s^{j-1} = \sum_{j=1}^{D} \left(\sum_{i=1}^{D} G_{ij} \gamma_{x,i} v_3 - \sum_{i=1}^{D} G_{ij} \gamma_{y,i} v_4 + v_1 \delta_{j,1} \right) s^{j-1}$$

where δ_{kl} is the Kronecker delta function. Since the above equation must hold for all $s \in [-1, 1]$, it must hold true separately for each power of s, and thus with a little bit of manipulation,

$$G^T \hat{\gamma}_x = (G^T \gamma_x)v_3 - (G^T \gamma_y)v_4 + v_1[10 \ldots 0]^T$$

Therefore,

$$\hat{\gamma}_x = v_3 \gamma_x - v_4 \gamma_y + v_1 \mu$$

where μ is the first column of $(G^T)^{-1}$. A similar equation can be derived for $\hat{\gamma}_y$, thus yielding the transformation matrix

$$W(\gamma) = \begin{bmatrix} \mu & \mathbf{0} & \gamma_x & -\gamma_y \\ \mathbf{0} & \mu & \gamma_y & \gamma_x \end{bmatrix}$$

where $\mathbf{0} = [0, 0, \ldots 0]^T$.

Appendix C: Euclidean Similarity Bounds

The bounds are given by:

$$\begin{aligned}
\underline{v}_1^t &= (1 - \sigma^{t-1}(1 + \Delta\sigma)\bar{c}^{t-1})\bar{x}^{t-1} \\
&\quad + \sigma^{t-1}(1 + \Delta\sigma)^{-1}\underline{s}^{t-1}\bar{y}^{t-1} + \tau_x^{t-1} - \Delta\tau_x \\
\bar{v}_1^t &= (1 - \sigma^{t-1}(1 + \Delta\sigma)^{-1}\underline{c}^{t-1})\bar{x}^{t-1} \\
&\quad + \sigma^{t-1}(1 + \Delta\sigma)\bar{s}^{t-1}\bar{y}^{t-1} + \tau_x^{t-1} + \Delta\tau_x \\
\underline{v}_2^t &= -\sigma^{t-1}(1 + \Delta\sigma)\bar{s}^{t-1}\bar{x}^{t-1} \\
&\quad + (1 - \sigma^{t-1}(1 + \Delta\sigma)\bar{c}^{t-1})\bar{y}^{t-1} + \tau_y^{t-1} - \Delta\tau_y \\
\bar{v}_2^t &= -\sigma^{t-1}(1 + \Delta\sigma)^{-1}\underline{s}^{t-1}\bar{x}^{t-1} \\
&\quad + (1 - \sigma^{t-1}(1 + \Delta\sigma)^{-1}\underline{c}^{t-1})\bar{y}^{t-1} + \tau_y^{t-1} + \Delta\tau_y \\
\underline{v}_3^t &= \sigma^{t-1}(1 + \Delta\sigma)^{-1}\underline{c}^{t-1} \\
\bar{v}_3^t &= \sigma^{t-1}(1 + \Delta\sigma)\bar{c}^{t-1} \\
\underline{v}_4^t &= \sigma^{t-1}(1 + \Delta\sigma)^{-1}\underline{s}^{t-1} \\
\bar{v}_4^t &= \sigma^{t-1}(1 + \Delta\sigma)\bar{s}^{t-1}
\end{aligned}$$

where

$$\underline{c}^{t-1} = \min_{\theta \in \Theta^t} \cos\theta \quad \bar{c}^{t-1} = \max_{\theta \in \Theta^t} \cos\theta$$
$$\underline{s}^{t-1} = \min_{\theta \in \Theta^t} \sin\theta \quad \bar{s}^{t-1} = \max_{\theta \in \Theta^t} \sin\theta$$

and $\Theta^t = [\theta^{t-1} - \Delta\theta, \theta^{t-1} + \Delta\theta]$.

Appendix D: Interpolation of Edge-Vectors

The following method is used to interpolate edge-vectors into observed curves. Suppose that the contours in question are open; closed contours can be interpolated using a very slight modification. The edge-vector given is $e = (e_1, \ldots, e_N) = (x_1, y_1, \ldots, x_N, y_N)$. The interpolation problem is set up in the following way: let ε_p be the piecewise linear interpolation of the edge-vector e. Then the goal is to find the contour $\varepsilon \in \mathcal{C}_D$

which is closest to ε_p, i.e., which minimizes

$$\|\varepsilon - \varepsilon_p\| = \sqrt{\int_{-1}^{1} [(\varepsilon_x(s) - \varepsilon_{p,x}(s))^2 + (\varepsilon_y(s) - \varepsilon_{p,y}(s))^2]\, ds}$$

This is a relatively easy problem due to the fact that the Legendre polynomials form an orthonormal set. In particular, the following familiar result can be used: if $\varepsilon_x(s) = \sum_{i=1}^{D} \gamma_{x,i} P_i(s)$, then

$$\gamma_{x,i} = \langle \varepsilon_{p,x}, P_i \rangle = \int_{-1}^{1} \varepsilon_{p,x}(s) P_i(s)\, ds$$

and similarly for $\gamma_{y,i}$. Focus solely on the x-part of the contour; the y-part will follow analogously. Let $s_1 = 0$ and $s_n = s_{n-1} + \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2}$, $n = 2, \ldots, N$; this is the arc-length at each of the vertices of the piecewise linear interpolation. Thus, the piecewise linear interpolation may be written separately for $N - 1$ pieces ($n = 1, \ldots, N - 1$):

$$\varepsilon_{p,x}(s) = \alpha_{x,n} + \beta_{x,n} s \quad \text{for } s_n \le s \le s_{n+1}$$

where

$$\alpha_{x,n} = \frac{x_n s_{n+1} - x_{n+1} s_n}{s_{n+1} - s_n}, \qquad \beta_{x,n} = \frac{x_{n+1} - x_n}{s_{n+1} - s_n}$$

Let A and B be $D \times (N - 1)$ matrices given by

$$A_{ij} = \frac{(s_{j+1} - s_j)^i}{i}, \qquad B_{ij} = \frac{(s_{j+1} - s_j)^{i+1}}{i+1}$$

Further, let α_x be the column vector with $N - 1$ entries $\alpha_{x,1}, \ldots \alpha_{x,N-1}$ and similarly for β_x. Then the following relation may be shown to hold:

$$\gamma_x = G(A\alpha_x + B\beta_x)$$

γ_y may be found by exact analogy.

Appendix E: Proof of Iterative Algorithm for Step 1

Proof: First, show that $\tilde{\gamma}^{t,t_1} = \lim_{t_2 \to \infty} \gamma^{t,t_1,t_2}$ and $\tilde{v}^{t,t_1} = \lim_{t_2 \to \infty} v^{t,t_1,t_2}$ represent a local minimum of the function $J(\gamma, v) = \|\gamma - Wv\|$ subject to the constraints $\gamma \in \Gamma$, $v \in \tilde{\mathcal{V}}^t$. Note that:

1. J may be expressed as a function of $\psi = p^T(\gamma - \bar{\gamma})$:

$$\begin{aligned} J^2(\gamma, v) &= \|\gamma - Wv\|^2 \\ &= \|p\psi + \bar{\gamma} - Wv\|^2 \\ &= \psi^T U \psi - 2\psi^T u + u^T u \end{aligned}$$

Thus, holding v and ψ_k, $k \neq i$ constant, then

$$\operatorname*{argmin}_{|\psi_i| \le b_i} J(\gamma, v) = \left\langle \frac{\left(u_i - U_i \psi^{t_2 - 1} + U_{ii} \psi_i^{t_2 - 1}\right)}{U_{ii}} \right\rangle_{-b_i}^{b_i}$$

which may be shown by simple differentiation. That is, setting $\psi_i^{t_2}$ equal to the right-hand side of the above equation reduces the value of $J(\gamma, v)$, by taking its minimum with respect to ψ_i.

2. J may be expressed as a function of v:

$$\begin{aligned} J^2(\gamma, v) &= \|Wv - \gamma\|^2 \\ &= v^T R v - 2v^T r + r^T r \end{aligned}$$

Thus, holding ψ and v_k, $k \neq j$ constant, then

$$\operatorname*{argmin}_{\underline{v}_j^t \le v_j \le \bar{v}_j^t} J(\gamma, v) = \left\langle \frac{\left(r_j - R_j v^{t_2 - 1} + R_{jj} v_j^{t_2 - 1}\right)}{R_{jj}} \right\rangle_{\underline{v}_j^t}^{\bar{v}_j^t}$$

which may be shown by simple differentiation. That is, setting $v_j^{t_2}$ equal to the right-hand side of the above equation reduces the value of $J(\gamma, v)$, by taking its minimum with respect to v_i.

Finally, note that $J(\gamma, v)$ is bounded from below by 0; this completes the proof that $(\tilde{\gamma}^{t,t_1}, \tilde{v}^{t,t_1})$ constitutes a local minimum.

It now remains to show that this local minimum constitutes a global minimum. Since $J(\gamma, v)$ is positive and quadratic in both γ and v, the unconstrained problem $\min_{\gamma,v} J(\gamma, v)$ has a single minimum. By inspection, adding the constraints $\gamma \in \Gamma$ and $v \in \tilde{\mathcal{V}}^t$ will not affect the uniqueness of the minimum (unless two minima have exactly the same value, which does not affect the following statement). Thus, the local minimum is in fact a global minimum. □

Appendix F: CTF Algorithm: The Subset *H*

Let $s_{\pi_1}, \ldots, s_{\pi_r}$ be the arc-lengths associated with the sites $\pi_1, \ldots, \pi_r$. (The arc-length associated with the

nth of N sites is $s_n = -1 + 2(n-1)/(N-1)$.) Then the set of points at the site π_1 may be written

$$H(\pi_1) = \{h \in \Re^2 : h = [Z^T(s_{\pi_1})G^T\gamma_x;\, Z^T(s_{\pi_1})G^T\gamma_y],\ \gamma \in \Gamma\}$$

where as is common convention, the semicolon indicates that vectors (in this case scalars) should be stacked. Extending this to the case of multiple sites is straightforward:

$$H(\pi_1, \dots, \pi_r) = \{h \in \Re^{2r} : h = [S^T G^T \gamma_x;\, S^T G^T \gamma_y],\ \gamma \in \Gamma\}$$

where $S = S(\pi_1, \dots, \pi_r) = [Z(s_{\pi_1}), \dots, Z(s_{\pi_r})]$. Using knowledge of the set Γ and its implicit definition in terms of the set Ψ, the above expression for H can be rewritten as

$$H(\pi_1, \dots, \pi_r) = \{h \in \Re^{2r} : h = Q(\pi_1, \dots, \pi_r)\psi + q(\pi_1, \dots, \pi_r),\ \psi \in \Psi\}$$

where

$$Q(\pi_1, \dots, \pi_r) = \begin{bmatrix} S^T(\pi_1, \dots, \pi_r)G^T p_x \\ S^T(\pi_1, \dots, \pi_r)G^T p_y \end{bmatrix}$$

and

$$q(\pi_1, \dots, \pi_r) = \begin{bmatrix} S^T(\pi_1, \dots, \pi_r)G^T \bar{\gamma}_x \\ S^T(\pi_1, \dots, \pi_r)G^T \bar{\gamma}_y \end{bmatrix}$$

and p_x is the top half (first D columns) of p, p_y is the bottom half, and likewise for $\bar{\gamma}_x$ and $\bar{\gamma}_y$.

In order to solve

$$\min_{h \in H(\pi_1, \dots, \pi_r)} \left\| \left[e_{\pi_1}; \dots; e\pi_r\right] - h \right\|$$

the problem may be recast as

$$\min_{\psi \in \Psi} \| Q\psi + q - \zeta \|$$

where $\zeta = [e_{\pi_1}; \dots; e\pi_r]$ and the arguments of Q and q have been suppressed for convenience. (Note that $[e_{\pi_1}; \dots; e\pi_r]$ is shorthand for the vector $[x_{\pi_1}, \dots, x_{\pi_r}, y_{\pi_1}, \dots, y_{\pi_r}]^T$.) The above minimization can be solved for quickly in an iterative manner along the lines of step 1 of the local minimization problem. Specifically, the following algorithm may be used:

```
**Let:** $A = Q^T Q$ and $a = Q^T(\zeta - q)$.
$t = 1$
*do*
    *for* $i = 1$ *to* $q$
        $\psi_i^t = \left\langle \left(a_i - A_i \psi^{t-1} + A_{ii}\psi_i^{t-1}\right) / A_{ii} \right\rangle_{-b_i}^{b_i}$
        $t \leftarrow t + 1$
    *end*
*until* $\psi$ *has converged*
```

References

Amini, A., Tehrani, S., and Weymouth, T. 1988. Using dynamic programming for minimizing the enegy of active contours in the presence of hard constraints. In *Proc. 2nd Intern. Conf. Comput. Vis.*, pp. 95–99.

Ayache, N., Cohen, I., and Herlin, I. 1992. Medical image tracking. In *Active Vision*, A. Blake and A. Yuille (Eds.). Cambridge, MA: MIT Press, pp. 285–302.

Blake, A. Curwen, R., and Zisserman, A. 1993. A framework for spatio-temporal control in the tracking of visual contours. *Int. J. Comp. Vis.*, 11(2):127–145.

Blake, A. and Isard, M. 1998. CONDENSATION—conditional density propagation for visual tracking. *Int. J. Comp. Vis.*, 29(1):5–28.

Blake, A., Isard, M., and Reynard, D. 1995. Learning to track the visual motion of contours. *Artificial Intelligence* 78:101–134.

Bregler, C. and Konig, Y. 1994. Eigenlips for robust speech recognition. In *Proceedings IEEE ICASSP*, Vol. II, pp. 669–672.

Brockett, R. and Blake, A. 1994. Estimating the shape of a moving contour. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 3247–3252.

Cootes, T., Hill, A., Taylor, C., and Haslam, J. 1994. Use of active shape models for locating structures in medical images. *Image and Vis. Comp.*, 12(6):355–365.

Dalton, B., Kaucic, R., and Blake, A. 1995. Automatic speechreading using dynamic contours. In *Proceedings NATO ASI Conference on Speechreading by Man and Machine: Models, Systems, and Applications.*

Kass, M., Witkin, A., and Terzopoulos, D. 1987. Snakes: Active contour models. In *Proc. 1st Intern. Conf. Comput. Vis.*, London.

Kaucic, R., Dalton, B., and Blake, A. 1996. Real-time lip tracking for audio-visual speech recognition application. In *Proceedings ECCV*, pp. 376–387.

Lipson, P., Yuille, A., O'Keefe, D., Cavanaugh, J., Taafe, J., and Rosenthal, D. 1990. Deformable templates for feature extraction from medical images. In *Proc. 1st Europ. Conf. Comput. Vis.*

Luettin, J., Thacker, N., and Beet, S. 1996. Visual speech recognition using active shape models and hidden markov models. In *Proceedings IEEE ICASSP*, pp. 817–820.

Mak, M. and Allen, W. 1994. Lip-motion analysis for speech segmentation in noise. *Speech Communication* 14(3):279–296.

Sullivan, G. 1992. Visual interpretation of known objects in constrained scenes. *Phil. Trans. Roy. Soc. London B* 337:109–118.

Xu, G., Segawa, E., and Tsuji, S. 1993. Robust active contours with insensitive parameters. In *Proc. 4th Intern. Conf. Comput. Vis.*, Berlin.

Yuille, A., Hallinan, P., and Cohen, D. 1992. Feature extraction from faces using deformable templates. *Int. J. Comp. Vis.*, 8(2):99–112.