(appeared in *International Journal of Computer Vision*, 38(2), pp. 99-127, July 2000)

Optical Flow Constraints on Deformable Models with Applications to Face Tracking

Douglas DeCarlo
Department of Computer Science
and Center for Cognitive Science
Rutgers University
Piscataway, NJ 08854-8019
decarlo@cs.rutgers.edu

Dimitris Metaxas
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389
dnm@central.cis.upenn.edu

Abstract

Optical flow provides a constraint on the motion of a deformable model. We derive and solve a dynamic system incorporating flow as a hard constraint, producing a model-based least-squares optical flow solution. Our solution also ensures the constraint remains satisfied when combined with edge information, which helps combat tracking error accumulation. Constraint enforcement can be relaxed using a Kalman filter, which permits controlled constraint violations based on the noise present in the optical flow information, and enables optical flow and edge information to be combined more robustly and efficiently. We apply this framework to the estimation of face shape and motion using a 3D deformable face model. This model uses a small number of parameters to describe a rich variety of face shapes and facial expressions. We present experiments in extracting the shape and motion of a face from image sequences which validate the accuracy of the method. They also demonstrate that our treatment of optical flow as a hard constraint, as well as our use of a Kalman filter to reconcile these constraints with the uncertainty in the optical flow, are vital for improving the performance of our system.

1 Introduction

The apparent motion of brightness in an image—the optical flow—constrains but does not necessarily determine the three-dimensional motion of an observed object. In reconstructing three-dimensional motion using optical flow and other ambiguous cues, combining separate solutions derived from different cues can only compromise among limited guesses. In contrast, adding optical flow constraints to disambiguate a problem of motion estimation can derive a good guess consistent with all the data. This paper describes such a constraint approach to optical flow within a deformable model framework [33, 38, 48] for shape and motion estimation. We show that this approach can greatly improve the ability to maintain accurate track of a moving object. For the applications here, we will be specifically investigating the tracking of human faces.

Image cues provide *constraints* which the estimated model should satisfy as much as possible. Typically, constraints from multiple cues are reconciled by statistically combining constraint solutions, so as to weight each source of information according to its reliability. This formulation treats data constraints as *soft*, in that the formulation biases the system towards satisfying the constraint, but does not enforce the constraint after combination. One of the distinguishing features of our approach is that we treat optical flow as a *hard* constraint on the extracted

motion of the model, which guarantees enforcement. By placing constraints from one cue onto the model *during* estimation, we limit the choices of parameter combinations available for solutions using other cues (which are included as soft constraints), thereby eliminating a portion of the search space. We claim that this simplifies the estimation process and leads to improved robustness by not only producing a lower dimensional estimation problem, but also by avoiding local minima.

The optical flow constraints are based on noisy data, which can lead to problems when using hard constraints (since the desired portion of the search space could be discarded due to noise). Previous approaches which used soft constraints did not encounter this problem, since a noisy constraint would simply be violated (and hence ignored) when combined with other information. But it is precisely this property which prevents soft constraints from limiting the search space in the first place (and hence loses the benefits of efficiency and robustness). Instead, we use an iterated extended Kalman filter to relax the optical flow constraint to allow for constraint violations which increase as uncertainty in the flow increases. In Section 4, we will be more precise by what is meant by relaxing hard constraints, as well as how the constraint relaxation takes place. Basically, the Kalman filter finds a middle-ground between the hard and soft constraint solutions that is in harmony with the level of uncertainty in the hard constraint. This retains the beneficial property of limiting the search space while being robust to noisy constraints.

Our approach can be summarized as follows. Within a deformable model framework, we start with a model-based version of the optical flow constraint equation, which constrains the velocities of the motion parameters of the model. In the theory of dynamic systems [41], velocity constraints such as these are called non-holonomic. The velocities of the motion parameters are already accounted for as resulting from the application of edge-based forces; finding the equilibrium resulting from these forces amounts to a straightforward optimization problem. With the addition of the optical flow constraints, a constrained optimization problem results, which is solved using Lagrange multipliers. The constrained solution contains two kinds of forces. One provides the standard linear least-squares model-based solution to the optical flow [28]. The second is a constraint enforcement term which ensures the optical flow constraint remains satisfied when combined with edge forces. The presence of the constraint enforcement term yields a profitable combination of the optical flow solution with the edge forces. We use a Kalman filter to realize this combination in a way that accounts for the uncertainty in the flow. Problems with tracking error accumulation are alleviated using the edge forces, which now keep the model aligned with the image without a statistically relevant violation of the optical flow constraint.

The applications we address here concentrate on the problem of estimating the shape and motion of a human face. This problem has been widely addressed in recent research, having applications in human-machine interaction. Face tracking is a particularly natural testbed for our research for two reasons. The actual shape and motion of faces makes edge and optical flow information easy to use and advantageous to combine; and the abundance of data describing human face shape [16] facilitates the development of three-dimensional models of faces.

We have constructed a model of the human face which captures the relevant aspects concerning their shape, motion and appearance. By using data from face anthropometry studies [16], the range of shapes produced capture the variabilities seen in the shape and appearance of faces across the human population. The design of the facial motion model employs aspects of the Facial Action Coding System (FACS) [13], which is a manual coding method for describing facial movements in terms of "action units". Our model separately encodes information

regarding an individual's appearance from their facial motions and expression. Shape parameters describe unchanging features of an observed object and capture variations in shape across objects in a target class. Motion parameters describe how an observed object changes during a tracking session. This separation produces an easier tracking problem by requiring a smaller description of object state to be estimated in each frame. It also allows information to be applied more precisely, since optical flow information only registers changes in motion parameters, while edge information figures in both shape and motion parameter estimation.

1.1 Outline

After a review of related work, in Section 3 we present some preliminaries on our deformable model framework. Then, Section 4 describes how the optical flow is treated as a hard constraint in this framework, and how a Kalman filter is used to relax this constraint to account for uncertainty. We then present a series of experiments designed to assess the generality of our approach and its quantitative validity in Section 5. These experiments extract the shape of the face, and track its motion—even in the presence of large rotations and self-occlusion. They demonstrate that our treatment of optical flow as a hard constraint, as well as our use of a Kalman filter to reconcile these constraints with the uncertainty in the optical flow, are vital for improving the performance of our system.

2 Related Work

There is a wide variety of work that relates to what is presented here concerning both the underlying techniques used, and the application of tracking a human face. Virtually all work on face tracking takes advantage of the constrained situation: instead of using a generic tracking framework which views the observed face as an arbitrarily deforming object, a *model-based* approach is favored, which incorporates knowledge about facial deformations, motions and appearance. This facial model is used in concert with a number of model-based techniques.

Model-based edges and features: A prevalent model-based approach for tracking and shape estimation uses features and edges in a sequence of images to track an object [27, 29, 33, 48, 50]. This requires aligning the model features with the data, and is typically formulated as an optimization problem where the parameter combination is sought which yields the best alignment. The alignment can be performed using either a 2D appearance model [48, 50] or on 2D features computed from a 3D model [29, 33, 47]. This optimization problem tends to be quite difficult, however, especially as the deviation between the model and data becomes large.

Model-based optical flow: Instead of computing an unconstrained flow field (a grid of arrows), a model-based approach explains the optical flow information in terms of motion parameters of the model [1, 5, 9, 23, 28, 35, 36]. While the problem is non-linear, these frameworks can use either a single step linear least-squares solution [9, 28, 36], or an iterative least-squares solution [1, 5, 23, 35]. The motion model can be a 2D model of image motion [5, 6] or a 3D model (rigid or non-rigid) of object motion [5, 9, 28] (along with a camera model to relate to the images).

It is also possible to compute an unconstrained optical flow field using standard techniques, and fit a parametric motion model to the resulting field [4, 14]. The primary downside to this approach would be that with the computation of an unconstrained flow field comes the artifacts resulting from smoothness assumptions and the problem

of finding motion discontinuities. The model-based methods above take this information from the model (instead of assuming it).

Preventing tracking drift: In this paper, we advocate the use of *both* optical flow and edges for face tracking. Face tracking methods using *only* optical flow [4, 6, 14] will suffer from tracking drift, since error will accumulate when only velocity information is used. As a result, long sequences are not tracked successfully. However, in the context of facial expression recognition (where the sequences are quite short), this might not be a serious problem.

In previous work [11], we presented a framework for combining optical flow and edge information to avoid the problems with tracking drift. Aside from this, [28] is the other notable exception of a framework which uses model-based optical flow yet avoids drift. In this work, a render-feedback loop was used to prevent drift by locally searching for the best set of parameters which aligns the rendered model with the image.

Model-based constraints and Kalman filtering: The treatment of optical flow as a hard constraint on the motion of the model in [11] not only helps prevent tracking drift, but also makes the system more robust and efficient when coupled with a Kalman filter to handle uncertainty in the constraint.

Before [11], hard constraints were used in estimation only as a modeling tool where an articulated object was modeled as a set of rigid pieces held together by geometric constraints (which model the joints) [21, 33, 42]. A method known as constraint fusion [21, 42] combines constraints with measurement data to account for the fact that the joint configurations might not be known in advance. This fusion was performed using a Kalman filter in [21], and ends up being closely related to the physics-based constraint method in [33] which adds constraint forces to data forces. More efficient means for dealing with such constraints has also been investigated [17, 33].

The use of soft constraints is much more common for fusing information. In [18], stereo and shading information are combined using a soft constraint (weighted terms from each source are added into the system energy). A physics-based sensor fusion method combining range and intensity data was presented in [51]. Using a Kalman filter [2, 26] or Bayesian methods [12] for fusion combines solutions in a similar way. Aside from this, Kalman filtering has become a standard tool for estimation in dealing with noisy data [3, 19, 30, 32, 37].

Face tracking: There is a vast body of work on tracking the human face, with applications ranging from motion capture to human-computer interaction. Among them, there are a number which bare similarity in some respect to the work presented in this paper.

Several 2D face models based on splines or deformable templates [27, 34, 50] have been developed which track the contours of a face in an image sequence. In addition to motion, these methods provide rough 2D information about the observed individual's appearance. In [6], the optical flow field is parameterized based on the motion of the face (under projection) using a set of locally defined 2D motion regions. The extracted parameters are used for expression recognition.

In [14, 47], a physics-based 3D face model (with many degrees of freedom) is used, where motion is measured in terms of muscle activations. Edge forces from snakes are used in [47], while in [14], activations are determined from an optical flow field which are later used for expression recognition. In [4], a rigid ellipsoid model of the head is used to estimate motion parameters from a flow field.

Addressing the problem of image coding, [9, 28] estimate face motion using a simple 3D model and a model-based least-squares solution to the optical flow constraint equation. [28] improves performance using motion prediction, and avoids tracking drift using a render-feedback loop.

Our system, first presented in [11], uses a combination of model-based optical flow and model-based edge tracking to estimate the shape and motion of a face. The flow and edges are combined by treating the flow as a hard constraint on the motion of the model. This combination prevents tracking error accumulation, as with the render-feedback loop in [28], although our use of a hard constraint produces a much more robust solution by making the model-based edge tracking problem easier to solve.

In addition to this, none of the previous work makes a serious attempt in extracting a detailed 3D *shape* description of the face from an image sequence. At best, only a rough shape description is derived. Furthermore, most all of these approaches fail under large head rotations due to the use of a 2D model or the inability to handle self-occlusion.

3 Deformable model dynamics

Deformable models [33, 38, 48] are parameterized shapes that deform due to forces according to physical laws. For vision applications, physics provides a useful analogy for treating shape estimation [33], where forces are determined from visual cues such as edges in an image. The deformations that result produce a shape that agrees with the data. The use of physics also makes available additional mathematical tools; for example, constraint techniques from physics will be used in Section 4 to incorporate the optical flow information. In this section, we review deformable models as presented in [33] and briefly describe our face model.

A three-dimensional deformable model $\mathbf{x}$ maps a domain Ω (of surface coordinates) to a set of points in $\mathbb{R}^3$ which form the model's surface. It is parameterized by a vector of values $\mathbf{q}$, meaning that changes in $\mathbf{q}$ register as geometric deformations of the surface. A particular point on the surface is written as $\mathbf{x}(\mathbf{q};\mathbf{u})$ with $\mathbf{u} \in \Omega$ being used to identify a specific surface location. (Note that the dependency of $\mathbf{x}$ on $\mathbf{q}$ is often omitted, for reasons of conciseness.) The goal of a shape and motion estimation process is to recover the value of $\mathbf{q}$ for each image in a sequence of frames. We now present our deformable face model; following this, the remaining discussion will refer to this face model, although the development of the techniques will apply generally.

3.1 A deformable face model

Across the human population, the faces of individuals exhibit a great deal of variation in their appearance, but they all still have a good deal of structure in common. A similar statement can be made about facial motion—while it is complex and non-rigid, the motions are still fairly constrained. We take advantage of this commonality in the construction of our model of the human face. Here, we briefly describe the face model used in the experiments in this paper.

Our deformable face model is a 3D polygon mesh, shown smoothly shaded in Figure 1(a) and wireframe in (b) in its default configuration. The model is realized using a set of parameterized deformations (which depend on $\mathbf{q}$) applied to this polygon mesh. The parameterization of this face model was constructed by hand; details concerning its construction are in Appendix B. Appendix C describes a system of anthropometric measurements of the face; we use data from published tables of these measurements to help bias the model away from producing unlikely faces during estimation. Our model does have limitations in its coverage, however. There are no means

for representing large amounts of facial hair (such as a beard or mustache) or eyeglasses. Furthermore, there are many facial motions that cannot be expressed accurately, such as many of the lip deformations produced during speech. Effects of these limitations on system performance are discussed in Section 5.6. Better and easier methods of data acquisition are becoming available, and are making it possible to build a model constructed from examples [10, 25]. It's likely that more automatic methods of model construction (using examples) will become the favored approach, as it is rather difficult to obtain a fully developed model of the face by hand.

(a) (b)

Figure 1: The deformable face model (in rest position)

3.2 Separation of shape and motion

In some applications (including face tracking), to distinguish the processes of shape estimation and motion tracking, the parameters in $\mathbf{q}$ are rearranged and separated into $\mathbf{q}_\mathrm{b}$—a *static* quantity—which describes the basic shape of the object, and into $\mathbf{q}_\mathrm{m}$—a *dynamic* quantity—which describes its motion (both rigid and non-rigid), so that $\mathbf{q} = (\mathbf{q}_\mathrm{b}^\top, \mathbf{q}_\mathrm{m}^\top)^\top$. Regarding human faces, $\mathbf{q}_\mathrm{b}$ describes the unchanging features of an observed face and captures variations in appearance across the human population, while $\mathbf{q}_\mathrm{m}$ describes how an observed face changes during a tracking session (head position, as well as facial displays and expressions). This separation produces an easier tracking problem by requiring a smaller description of object state to be estimated in each frame. This division is often built into face models [6, 28, 34, 47] to simplify model construction or estimation, while Reynard, et al. [40] use this separation to permit learning the variability of motions for a class of objects. Note that there is no guarantee that the shape and motion of some class of objects is separable; this is a simplifying assumption that we make. For human faces, this separation is quite reasonable, and results in the changes in $\mathbf{q}_\mathrm{b}$ tending to zero as the shape of the observed object is established. Once this occurs, fitting need only continue for $\mathbf{q}_\mathrm{m}$. This suggests that including as many parameters as possible in $\mathbf{q}_\mathrm{b}$ makes long-term estimation more efficient.

The model $\mathbf{x}$ is realized by applying deformation functions to an underlying shape $\mathbf{s}$. For this paper, $\mathbf{s}$ is the polygon mesh in Figure 1, and Ω is an index set used to refer to its vertices (when applications require it, we add additional structure to Ω to allow references to any point on its surface, instead of just the vertices). In other deformable model work [33], $\mathbf{s}$ is a solid primitive such as an ellipsoid given by its explicit parametric equation with its domain Ω being an appropriate rectangle in $\mathbb{R}^2$.

As with the parameters, the deformations applied to $\mathbf{s}$ are split into two separate deformation functions—one for shape ($\mathbf{T}_\mathrm{b}$) and one for motion ($\mathbf{T}_\mathrm{m}$)—as demonstrated in Figure 2. These deformation functions (which depend

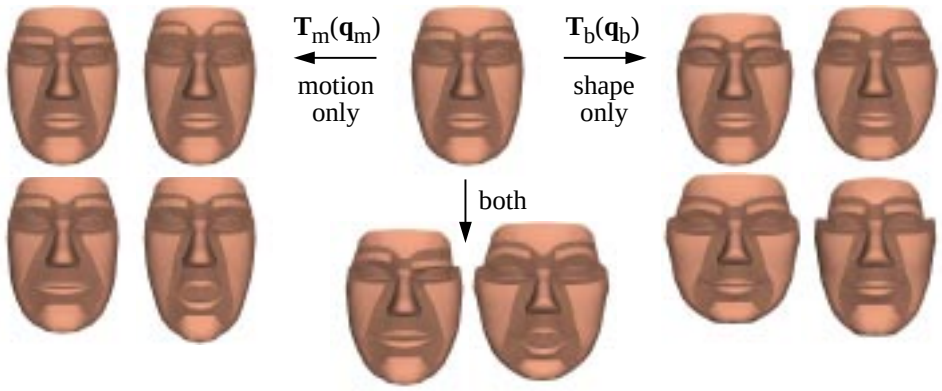


Figure 2: Example parameterized deformations of the face model (with separate shape and motion parameters)

on the parameters $\mathbf{q}$) map $\mathbb{R}^3$ to $\mathbb{R}^3$. For faces, the shape deformation $\mathbf{T}_b$ is applied to the underlying polygon mesh first (since facial motion can be seen as deviations away from a particular individual's face), so that:

$$\mathbf{x}(\mathbf{q};\mathbf{u}) = \mathbf{T}_m\left(\mathbf{q}_m;\ \mathbf{T}_b\left(\mathbf{q}_b;\ \mathbf{s}(\mathbf{u})\right)\right) \tag{1}$$

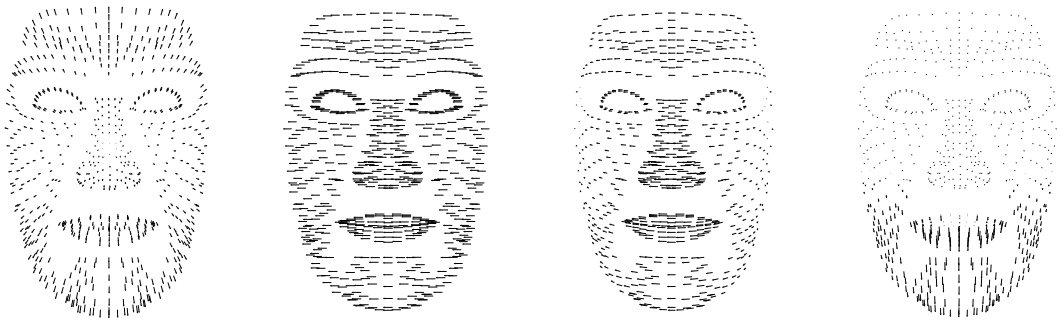
The shape deformation $\mathbf{T}_b$ uses the parameters $\mathbf{q}_b$ to deform the underlying shape $\mathbf{s}$. For faces, applying this deformation alone will produce a particular individual's face in rest position. On top of this is the motion deformation $\mathbf{T}_m$ with parameters $\mathbf{q}_m$, which includes a rigid translation and rotation, as well as non-rigid deformations (such as raising eyebrows, frowning, smiling, and opening the mouth, as shown on the left of Figure 2). Each of these deformations can be defined using a series of composed functions, allowing a more modular design (see Appendix A).

3.3 Kinematics

The kinematics of the model can be determined in terms of the parameter velocities $\dot{\mathbf{q}}$. As the shape changes, the velocity at a point $\mathbf{u}$ on the model is given by:

$$\dot{\mathbf{x}}(\mathbf{u}) = \mathbf{L}(\mathbf{q};\mathbf{u})\dot{\mathbf{q}} \tag{2}$$

where $\mathbf{L} = \partial\mathbf{x}/\partial\mathbf{q}$ is the model Jacobian [33]. For cases where $\mathbf{x}$ is defined using a sequence of deformation functions, the Jacobian can be computed using the chain rule as in Appendix A. We also partition the Jacobian (as we did with the parameters) into blocks corresponding to $\mathbf{q}_b$ and $\mathbf{q}_m$ as $\mathbf{L} = [\mathbf{L}_b\ \mathbf{L}_m]$. A geometric interpretation for $\mathbf{L}(\mathbf{u})$ comes from viewing each column of $\mathbf{L}$ as corresponding to a particular parameter in $\mathbf{q}$. Each column is a three-dimensional vector which "points" in the direction that $\mathbf{x}(\mathbf{u})$ moves as that parameter is increased. When considered over the entire model (over Ω), they form vector fields, which are shown in Figure 3 for particular motion parameters of our face model.



forward translation (a) right translation (b) turning head right (c) opening the mouth (d)

Figure 3: Sample vector fields for various motion parameters

3.4 Perspective projection of the model

When modeling an object viewed in images, $\mathbf{x}$ needs to include a camera projection, resulting in a two-dimensional model (called $\mathbf{x}_p$), which is projected flat from the original three-dimensional model. Under perspective projection (with a camera having focal length f), the point $\mathbf{x}(\mathbf{u}) = (x,y,z)^\top$ projects to the image point $\mathbf{x}_p(\mathbf{u}) = \frac{f}{z}(x,y)^\top$.

The velocities of model points projected onto the image plane, $\dot{\mathbf{x}}_p$, can be found in terms of $\dot{\mathbf{x}}$. The Jacobian $\mathbf{L}_p = \partial \mathbf{x}_p / \partial \mathbf{q}$ is given by:

$$\dot{\mathbf{x}}_p(\mathbf{u}) = \frac{\partial \mathbf{x}_p}{\partial \mathbf{x}} \dot{\mathbf{x}}(\mathbf{u}) = \left(\frac{\partial \mathbf{x}_p}{\partial \mathbf{x}} \mathbf{L}(\mathbf{q};\mathbf{u}) \right) \dot{\mathbf{q}} = \mathbf{L}_p(\mathbf{q};\mathbf{u}) \dot{\mathbf{q}} \tag{3}$$

where

$$\frac{\partial \mathbf{x}_p}{\partial \mathbf{x}} = \begin{bmatrix} f/z & 0 & -fx/z^2 \\ 0 & f/z & -fy/z^2 \end{bmatrix} \tag{4}$$

The matrix in (4) projects the columns of $\mathbf{L}$ (which are three-dimensional vectors) onto the image plane. As with $\mathbf{L}$, we partition $\mathbf{L}_p$ as $\begin{bmatrix} \mathbf{L}_{\mathrm{b}p} & \mathbf{L}_{\mathrm{m}p} \end{bmatrix}$. In fact, the vector fields in Figure 3 are just renderings of $\mathbf{L}_{\mathrm{m}p}$.

3.5 Estimation using dynamics

The models defined earlier are useful for applications such as shape and motion estimation when used in a physics-based framework [33]. These techniques are a form of optimization whereby the deviation between the model and the data is minimized. The optimization is performed by integrating differential equations derived from the Euler-Lagrange equations of motion. In a typical vision application, the equations of motion are simplified to omit the mass term, which produces a model free of inertia. From an optimization point of view, this has the desirable property that the model state no longer changes once all forces vanish or equilibrate. In addition to this, the damping matrix is set to be the identity and the stiffness term is omitted, resulting in the following simplified

dynamic equations of motion:

$$\dot{\mathbf{q}} = \mathbf{f_q} \tag{5}$$

where the applied forces $\mathbf{f_q}$ are computed from three-dimensional forces $\mathbf{f}_{\text{3D}}$ and two-dimensional image forces $\mathbf{f}_{\text{image}}$ as:

$$\mathbf{f_q} = \sum_j \left(\mathbf{L}(\mathbf{u}_j)^\top \mathbf{f}_{\text{3D}}(\mathbf{u}_j) + \mathbf{L}_p(\mathbf{u}_j)^\top \mathbf{f}_{\text{image}}(\mathbf{u}_j) \right) \tag{6}$$

The distribution of forces on the model is based in part on forces computed from the edges of an input image. We compute the image forces $\mathbf{f}_{\text{image}}(\mathbf{u}_j)$ using the intensity gradient, as in [33, 48]. Using this method, the image force applied to model point $\mathbf{u}_j$ (which corresponds by projection to pixel j) is the product of the intensity gradient and a weighting function (with range $[0,1]$) which is a "probability" that the current model configuration would produce an edge visible nearby pixel j. As in [48], we use a thresholded version of this weighting function—details on how potentially visible edges are determined for our face model are provided in Appendix D. Note that these image forces depend on $\mathbf{q}$ not only through $\mathbf{L}$, but also in the determination of likely visible edges (the weighting function). Given an adequate model initialization, these forces will align features on the model with image features, thereby determining the object parameters. Using $\mathbf{L}$ and $\mathbf{L}_p$, the applied forces are converted to forces which act on $\mathbf{q}$ and are integrated over the model to find the total parameter force $\mathbf{f_q}$. The dynamic system in (5) is solved by integrating over time, using standard (explicit) differential equation integration techniques (we use an Euler step):

$$\mathbf{q}(t+1) = \mathbf{q}(t) + \dot{\mathbf{q}}(t)\Delta t \tag{7}$$

The process used to initialize the system to determine the value of $\mathbf{q}(0)$ is described in Section 5. The next section describes how this framework is augmented to accommodate optical flow information.

4 Optical flow constraints

In the following, the use of hard optical flow constraints on deformable models is presented. The optical flow constraint equation, which expresses a constraint on the optical flow velocities, is reformulated as a system of dynamic constraints that constrain $\dot{\mathbf{q}}$, the velocity of the deformable model. The resulting information will be combined with the data forces $\mathbf{f_q}$ while leaving the constraint satisfied. The optical flow constraint equation is used at a number of select locations in the image to constrain the motion of the model, instead of explicitly computing an unconstrained optical flow field on the entire image. We will see below how the use of this constraint is related to model-based optical flow methods (which are also known as "direct methods" since they also do not explicitly compute a flow field). The use of optical flow information greatly improves the estimation of $\mathbf{q}_{\text{m}}$, the motion parameters of the deformable model.

In deformable model frameworks, estimation is accomplished through an energy optimization process via equa-

tions of motion. Hard constraints impose a global requirement on this dynamic system whose solution is enforced at each iteration (either exactly or, if the system is overconstrained, in a least-squares sense), while soft constraints (such as spring forces) only bias the behavior of the system toward a certain goal (often involving the system energy). We will discuss how hard constraints provide a means for results obtained from one data source to guide the computation of the solution to another, potentially more difficult problem. This decreases the cost of this further computation and increases the likelihood that its solution will closely reflect the true state of the observed object.

Constraints which depend only on $\mathbf{q}$ are called *holonomic* constraints, and constrain the model to a set of allowable positions. They have been used in a deformable model formulation, for instance, to add point-to-point attachment constraints between the parts of an articulated object [21, 33, 42]. A holonomic constraint $\mathbf{C}$ has the general form

$$\mathbf{C}(\mathbf{q}, \mathrm{t}) = \mathbf{0} \tag{8}$$

Non-holonomic constraints additionally depend on the velocity of the parameters, $\dot{\mathbf{q}}$, and constrain the motion of the model. A non-holonomic constraint $\mathbf{C}$ has the general form

$$\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}, \mathrm{t}) = \mathbf{0} \tag{9}$$

In the following, we show how the optical flow constraints take this form and can be incorporated into the dynamic system using Lagrange multipliers. This results in hard constraints, since the constraints will be enforced exactly (or in a least-squares way). Following this, we will describe how this constrained system is solved, and how a Kalman filter is used to relax these hard constraints.

4.1 Optical flow constraints

Given the assumption of brightness constancy of small regions in an image, the optical flow constraint equation [22] at a pixel i in the image I takes the form:

$$\nabla \mathrm{I}_i \begin{bmatrix} u_i \\ v_i \end{bmatrix} + \mathrm{I}_{t_i} = 0 \tag{10}$$

where $\nabla \mathrm{I} = [\mathrm{I}_x \;\; \mathrm{I}_y]$ are the spatial derivatives and I_t is the temporal derivative of the image intensity. u_i and v_i are the components of the optical flow velocities.

For a model under perspective projection, there exists a unique point $\mathbf{u}_i$ on the model that corresponds to the pixel i (provided it is not on an occluding boundary). The optical flow constraint equation can now be rewritten in terms of $\mathbf{q}$ with this in mind. This rewriting uses an identification of the image velocity (u_i, v_i) at pixel i with its corresponding model velocity $\dot{\mathbf{x}}_p(\mathbf{u}_i)$ from (3):

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \dot{\mathbf{x}}_p(\mathbf{u}_i) = \mathbf{L}_{\mathrm{m}p}(\mathbf{u}_i)\dot{\mathbf{q}}_{\mathrm{m}} \tag{11}$$

Direct use of the optical flow information only provides motion information, and as a result, only $\mathbf{q}_\mathrm{m}$ is affected. To clarify this: any observed motion is caused by dynamic changes in the true value of $\mathbf{q}_\mathrm{m}$. The true value of $\mathbf{q}_\mathrm{b}$ is a static quantity—the meaning of $\dot{\mathbf{q}}_\mathrm{b}$ comes from the analogy of physics, where the value of $\mathbf{q}_\mathrm{b}$ improves over the course of fitting (over time) as more data becomes available.

The non-holonomic constraint equation for the optical flow at a pixel *i* in the image can be found by rewriting the optical flow constraint equation (10) using (11):

$$\nabla \mathrm{I}_i \mathbf{L}_{\mathrm{m}p}(\mathbf{u}_i)\dot{\mathbf{q}}_\mathrm{m} + \mathrm{I}_{t_i} = 0 \tag{12}$$

Instead of using this constraint at every pixel in the image, *n* pixels are selected from the input image (where $n \gg \dim \mathbf{q}_\mathrm{m}$). Appendix E describes the criterion used to choose these particular points, and also describes how some of the known difficulties in the computation of the optical flow are avoided in this model-based approach.

For the *n* chosen pixels in the image, the system of equations based on (12) becomes:

$$\begin{bmatrix} \nabla \mathrm{I}_1 \mathbf{L}_{\mathrm{m}p}(\mathbf{u}_1) \\ \vdots \\ \nabla \mathrm{I}_n \mathbf{L}_{\mathrm{m}p}(\mathbf{u}_n) \end{bmatrix} \dot{\mathbf{q}}_\mathrm{m} + \begin{bmatrix} \mathrm{I}_{t_1} \\ \vdots \\ \mathrm{I}_{t_n} \end{bmatrix} = \mathbf{0} \tag{13}$$

which can be written compactly as

$$\mathbf{B}\dot{\mathbf{q}}_\mathrm{m} + \mathbf{I}_\mathrm{t} = \mathbf{0} \tag{14}$$

This equation is simply a model-based version of the optical flow constraint equation [1, 5, 9, 23, 28, 35, 36]. Instead of solving it on its own, however, it is used as a hard constraint on the motion of the model.

4.2 Solving the dynamic system

Constraining the equations of motion with the model-based flow equation results in the constrained system:

$$\dot{\mathbf{q}} = \mathbf{f}_\mathbf{q} \quad \text{subject to} \quad \mathbf{B}\dot{\mathbf{q}}_\mathrm{m} + \mathbf{I}_\mathrm{t} = \mathbf{0} \tag{15}$$

This is solved using the method of Lagrange multipliers [41, 45]. The Lagrange multiplier technique adds additional degrees of freedom (one for each degree of constraint), to form a larger, unconstrained system (with the constraints "built in"). The initial dynamic equation of motion (5), now split into two parts corresponding to $\mathbf{q}_\mathrm{b}$ and $\mathbf{q}_\mathrm{m}$, is modified by adding the constraint force $\mathbf{f}_\mathrm{c}$ to $\dot{\mathbf{q}}_\mathrm{m}$:

$$\dot{\mathbf{q}}_\mathrm{b} = \mathbf{f}_{\mathbf{q}_\mathrm{b}}, \qquad \dot{\mathbf{q}}_\mathrm{m} = \mathbf{f}_{\mathbf{q}_\mathrm{m}} + \mathbf{f}_\mathrm{c} \tag{16}$$

Adding a particular value of $\mathbf{f}_c$ will ensure the constraint equation is satisfied, in part by cancelling the components of $\mathbf{f}_{\mathbf{q}_m}$ that violate the constraint. This constraint force is determined using the Lagrange multiplier $\boldsymbol{\lambda}$ as:

$$\mathbf{f}_c = -\mathbf{B}^\top \boldsymbol{\lambda} \tag{17}$$

We can combine equations (14), (16) and (17) to form:

$$\mathbf{B}\mathbf{B}^\top \boldsymbol{\lambda} = \mathbf{B}\mathbf{f}_{\mathbf{q}_m} + \mathbf{I}_t \tag{18}$$

and can now determine the constraint force (by multiplying (18) on the left by $\mathbf{B}^+$, the pseudo-inverse [45] of $\mathbf{B}$):

$$\mathbf{f}_c = -\mathbf{B}^+(\mathbf{B}\mathbf{f}_{\mathbf{q}_m} + \mathbf{I}_t) = -\mathbf{B}^+\mathbf{I}_t - \mathbf{B}^+\mathbf{B}\mathbf{f}_{\mathbf{q}_m} \tag{19}$$

which results in the unconstrained dynamic system (which is solved iteratively since $\mathbf{f}_\mathbf{q}$ is highly non-linear):

$$\dot{\mathbf{q}}_b = \mathbf{f}_{\mathbf{q}_b}, \qquad \dot{\mathbf{q}}_m = -\mathbf{B}^+\mathbf{I}_t + \left[\mathbf{1} - \mathbf{B}^+\mathbf{B}\right]\mathbf{f}_{\mathbf{q}_m} \tag{20}$$

The first term of $\dot{\mathbf{q}}_m$ in (20), $-\mathbf{B}^+\mathbf{I}_t$, is a model-based linear least-squares solution to the optical flow constraint equation [28]. A model-based solution to the optical flow constraint equations attributes the flow in the image to motion parameters in the model. This works as follows. A change to any motion parameter induces a characteristic motion field in the image. Figure 3 illustrates these vector fields for particular motion parameters of our face model. The linear combination of the fields $\mathbf{L}_{m_p}(\mathbf{u})$ using the weights $-\mathbf{B}^+\mathbf{I}_t$ best satisfies (14) at the sampled pixels in a least-squares sense. The pseudo-inverse of $\mathbf{B}$ is determined by computing its singular value decomposition [39, 45]. Each motion parameter in $\mathbf{q}$ will have a corresponding singular value—a singular value near zero is interpreted as a lack of motion in that particular parameter (although this could also be caused by the failure to gather enough information from the images to sample the motion). In solving (20) iteratively, $\mathbf{f}_\mathbf{q}$ is re-evaluated upon each iteration, while $\mathbf{B}$ is not. (We have empirically found that re-evaluating $\mathbf{B}$ does not change the performance of our system when high frame-rate cameras are used. This is not surprising, as the dependency of $\mathbf{B}$ on $\mathbf{q}$ is very small for the applications here, when given moderately sized changes in $\mathbf{q}$.)

The second term in (20) contains the edge forces $\mathbf{f}_{\mathbf{q}_m}$ scaled by the matrix $[\mathbf{1} - \mathbf{B}^+\mathbf{B}]$. This projection matrix cancels the component of $\mathbf{f}_{\mathbf{q}_m}$ that violates the constraint (14) on $\dot{\mathbf{q}}_m$. Unlike the hard optical flow constraint, the edge forces act as a soft constraint, but still prevent errors in $\mathbf{q}_m$ from accumulating, since the uncancelled component of the edge forces can further adjust the solution.

As it stands, however, this method will not be robust since $\mathbf{B}$ depends on noisy data. In [11], an ad hoc method was used to relax the hard constraint by replacing the projection matrix in (20) with $[\mathbf{1} - \mathbf{B}^+\mathbf{W}\mathbf{B}]$ where $\mathbf{W}$ is a diagonal matrix whose entries (in $[0,1]$) represent the certainty of information provided by a particular pixel. A more principled approach to relaxing the constraint is described in the next section, where (20) is reformulated using a Kalman filter. But first, it's worth taking a closer look at what the hard constraint is actually doing, and what is meant by relaxing the hard constraint.

4.3 Discussion

Consider the problem of combining together the two sources of information (edges and flow) to compute $\dot{\mathbf{q}}_m$. Independently, the edges and flow produce two different solutions: $\dot{\mathbf{q}}_m = \mathbf{f}_{\mathbf{q}_m}$ and $\dot{\mathbf{q}}_m = -\mathbf{B}^+\mathbf{I}_t$. In the following discussion, we consider the different formulations that result depending on whether soft, hard, or relaxed hard constraints are used to combine the solutions.[1]

Soft constraints are the typical means for combining these solutions. Statistical methods for combination weight these together (using matrices $\mathbf{W}_{\text{flow}}$ and $\mathbf{W}_{\text{edge}}$) according to their reliability:

$$\dot{\mathbf{q}}_m^{(\text{soft})} = -\mathbf{W}_{\text{flow}}\mathbf{B}^+\mathbf{I}_t + \mathbf{W}_{\text{edge}}\mathbf{f}_{\mathbf{q}_m} \tag{21}$$

These weighting matrices are typically formed from the covariance matrices for the individual solutions [8, 12]. In non-linear situations, (21) is solved iteratively. This is also comparable to using a Kalman filter to combine sources together (or an iterated extended Kalman filter in the non-linear case). In a deformable model framework, this approach is achieved by adding together weighted combinations of forces [46, 48] or energies [18] derived from data sources. The dynamic system produces a weighted least squares estimate similar to (21) as it converges.

With hard constraints, instead of combining solutions as above, we solve a constrained system: the equation originally used to solve for flow, $\mathbf{B}\dot{\mathbf{q}}_m + \mathbf{I}_t = \mathbf{0}$, will be used as a hard constraint on the solution of $\dot{\mathbf{q}}_m = \mathbf{f}_{\mathbf{q}_m}$. What results is the following:

$$\dot{\mathbf{q}}_m^{(\text{hard})} = -\mathbf{W}_{\text{flow}}\mathbf{B}^+\mathbf{I}_t + \mathbf{W}_{\text{edge}}\left[\mathbf{1} - \mathbf{B}^+\mathbf{B}\right]\mathbf{f}_{\mathbf{q}_m} \tag{22}$$

This solution gives precedence to the flow solution in an interesting way. There is now a projection matrix $[\mathbf{1} - \mathbf{B}^+\mathbf{B}]$ which cancels the component of the edge solution which violates the constraint *before* the solutions are combined together. This makes a substantial difference when (22) must be solved iteratively when the model-edge alignment problem ($\mathbf{f}_{\mathbf{q}_m}$) is non-linear. When solved alone, it is significantly more computationally expensive than solving for the flow. In this case, however, the projection matrix cancels out a portion of the search space for the model-edge alignment that the constraint makes impossible. This results in a lower dimensional problem in solving the model-edge alignment, and can improve efficiency, as well as decrease the chances of reaching a local minimum. Alternatively, the edge solution could be used as a hard constraint on the flow; but this would lose the efficiency benefits, as the edge solution is much more expensive to solve, so that its projection matrix would not be available in time to efficiently guide the flow solution.

In practice, the hard constraint depends on noisy data, in which case it is overly restrictive to fully cancel any component of the edge forces. Furthermore, during complex motions, it is not unreasonable for the flow solution to be non-degenerate, so that the projection matrix is zero (so that it cancels everything). One way of dealing with this is to *relax* the hard constraint: to permit small violations of the hard constraint where it is noisy while still

[1] To keep this discussion informal, equations (21)-(23) are merely suggestive of the process involved in statistical combination. As we shall see in the next section, the combination process can be formalized most perspicuously by recasting the entire solution process to take into account the statistical information from both solutions. Unfortunately, this presentation distracts from the high level differences between methods.

projecting away much of the edge forces which violate it.

The ad hoc method that accomplished this in [11] (mentioned earlier) was to replace the projection matrix in (22) with $[\mathbf{1} - \mathbf{B}^{+}\mathbf{W}_{\text{constraint}}\mathbf{B}]$, using the diagonal matrix $\mathbf{W}_{\text{constraint}}$. This results in:

$$\dot{\mathbf{q}}_{\text{m}}^{(\text{relaxed})} = -\mathbf{W}_{\text{flow}}\mathbf{B}^{+}\mathbf{I}_{\text{t}} + \mathbf{W}_{\text{edge}}\left[\mathbf{1} - \mathbf{B}^{+}\mathbf{W}_{\text{constraint}}\mathbf{B}\right]\mathbf{f}_{\mathbf{q}_{\text{m}}} \tag{23}$$

The best illustration of what this relaxed solution is doing comes from the special case when $\mathbf{W}_{\text{constraint}}$ has equal entries on its diagonal, so that $\mathbf{W}_{\text{constraint}} = \alpha\mathbf{1}$ with $\alpha \in [0,1]$. Then the relaxed solution is simply a convex combination of the soft and hard constraint solutions:

$$\dot{\mathbf{q}}_{\text{m}}^{(\text{relaxed})} = \alpha\dot{\mathbf{q}}_{\text{m}}^{(\text{hard})} + (1-\alpha)\dot{\mathbf{q}}_{\text{m}}^{(\text{soft})} \tag{24}$$

The hard constraint is enforced when α is 1, with a linear compression in the constraint's null space direction that decreases until α is 0, which is the soft constraint solution. This special case of the method in [11] represents one of the simplest means of relaxing a hard constraint. The next section describes a more principled approach to relaxing the hard constraint using a Kalman filter.

4.4 Kalman filtering and hard constraints

The optical flow constraint on $\dot{\mathbf{q}}_{\text{m}}$ is imperfect due to noise and estimation errors. It is therefore desirable to have only a partial cancellation of $\mathbf{f}_{\mathbf{q}_{\text{m}}}$; one way this can be accomplished is through the use of a Kalman filter. This section describes how the computation from Section 4.2 is reformulated using an iterated extended Kalman filter.

Kalman filtering [3, 19, 30] has become a popular tool in computer vision, and the formulation here is, on the whole, similar to other applications [2, 7, 26, 32, 37]: there is a measurement equation which models the noise inherent in the data gathering process, and there is a process model, which predicts the behavior of the system based on the current state. The initialization and tuning of the filter is accomplished using standard techniques. The significant difference here, is that there is not only the edge data equation (5), which has been previously used as a filtering measurement equation [33], but there is also a data-based constraint equation (14). The first part of this section describes one reasonable way of using this constraint in the measurement equation. Alternative formulations are possible; ours corresponds to one which involves the solution of a hard constraint. The remainder of the section describes an iterated extended Kalman filter based in part on this measurement equation. First, we will describe our formulation of the filter. Following this, we will explain why this treatment allows for relaxed hard constraints.

By assuming a Gaussian noise model for both the measurements and state, a Kalman filter can maintain an estimate of the state $\mathbf{y}$ and the state covariance $\mathbf{P}$. While the assumption of Gaussian noise might not be particularly accurate in describing the actual noise in the system, it permits a much simpler solution while still capturing a large amount of the uncertainty.

The *measurement* equation for the filter relates the measurements $\mathbf{z}$ to the state $\mathbf{y}$ using the measurement matrix $\mathbf{H}$. Terms $\mathbf{v}_{\mathbf{f}_{\mathbf{q}}}$ and $\mathbf{v}_{\mathbf{I}_{\text{t}}}$ are added to represent the assumed zero-mean Gaussian noise in $\mathbf{f}_{\mathbf{q}}$ and $\mathbf{I}_{\text{t}}$; they have

covariances $\mathbf{R}_{\mathbf{f_q}}$ and $\mathbf{R}_{\mathbf{I}_t}$ respectively:

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{y}(t) + \begin{pmatrix} \mathbf{v}_{\mathbf{f_q}}(t) \\ \mathbf{v}_{\mathbf{I}_t}(t) \end{pmatrix} \tag{25}$$

where the construction of $\mathbf{H}$, $\mathbf{y}$ and $\mathbf{z}$ in (25) comes from (14), (16) and (17).

$$\mathbf{H} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{B}^\top \\ \mathbf{0} & \mathbf{B} & \mathbf{0} \end{bmatrix}, \quad \mathbf{y} = \begin{pmatrix} \dot{\mathbf{q}}_\mathrm{b} \\ \dot{\mathbf{q}}_\mathrm{m} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} \mathbf{f}_{\mathbf{q}_\mathrm{b}} \\ \mathbf{f}_{\mathbf{q}_\mathrm{m}} \\ -\mathbf{I}_\mathrm{t} \end{pmatrix} = \begin{pmatrix} \mathbf{f_q} \\ -\mathbf{I}_\mathrm{t} \end{pmatrix} = \begin{pmatrix} \sum_j \mathbf{L}_p(\mathbf{u}_j)^\top \mathbf{f}(\mathbf{u}_j) \\ -\mathbf{I}_\mathrm{t} \end{pmatrix} \tag{26}$$

The state $\mathbf{y}$ consists of the parameter velocities $\dot{\mathbf{q}}$; together with the Lagrange multipliers $\boldsymbol{\lambda}$ used in the optical flow solution. This inclusion is for presentation only, because, as will be seen later, $\boldsymbol{\lambda}$ is effectively not part of the state. The discrete update equation for the state is given by (7).

$\mathbf{z}$ consists of the parameter forces $\mathbf{f_q}$ and the temporal image derivatives $\mathbf{I}_\mathrm{t}$. Note that the spatial image derivatives are not included in the measurements (even though they are used in the formation of $\mathbf{B}$); doing so would greatly complicate the measurement equations. Similar simplifications can be found in image-based optical flow techniques [44] where the noise in the spatial image derivatives are ignored to provide a Gaussian solution. Reasonably accurate estimates of the spatial image derivatives are usually available (especially away from occlusion boundaries), making this a fairly safe assumption.

Note that $\mathbf{H}$ depends on the state $\mathbf{y}$, so that the measurement equation is non-linear, and its solution requires the use of an extended Kalman filter. We also choose to iterate the solution, due to serious non-linearities in $\mathbf{f_q}$. Recall from the previous section that while both $\mathbf{B}$ and $\mathbf{f_q}$ depend on $\mathbf{q}$, we only re-evaluate $\mathbf{f_q}$, and not $\mathbf{B}$. This iterated extended Kalman filter is implemented in the standard way [19, 31], paying heed to the usual caveats concerning linearized filter convergence.

A more standard implementation would use the Kalman filter for data integration [2, 26] (a soft constraint approach) to fuse the flow and edge solutions. This solution simply lacks the Lagrange multipliers:

$$\mathbf{H}^{\mathrm{soft}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}, \quad \mathbf{y}^{\mathrm{soft}} = \begin{pmatrix} \dot{\mathbf{q}}_\mathrm{b} \\ \dot{\mathbf{q}}_\mathrm{m} \end{pmatrix} = \dot{\mathbf{q}}, \quad \mathbf{z}^{\mathrm{soft}} = \begin{pmatrix} \mathbf{f}_{\mathbf{q}_\mathrm{b}} \\ \mathbf{f}_{\mathbf{q}_\mathrm{m}} \\ -\mathbf{I}_\mathrm{t} \end{pmatrix} = \begin{pmatrix} \mathbf{f_q} \\ -\mathbf{I}_\mathrm{t} \end{pmatrix} \tag{27}$$

However, it is the solution that contains $\boldsymbol{\lambda}$ that produces a hard constraint solution. Comparing the pseudo-inverses of these two measurement matrices shows that $\mathbf{H}^+$ results in (20):

$$\begin{aligned} \mathbf{H}^+ &= \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} - \mathbf{B}^+\mathbf{B} & \mathbf{B}^+ \\ \mathbf{0} & (\mathbf{B}^+)^\top & -(\mathbf{B}^+)^\top \mathbf{B}^+ \end{bmatrix} \\ \mathbf{H}^{\mathrm{soft}+} &= \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\mathbf{B}^\top\mathbf{B}+\mathbf{1})^{-1} & (\mathbf{B}^\top\mathbf{B}+\mathbf{1})^{-1}\mathbf{B}^\top \end{bmatrix} \end{aligned} \tag{28}$$

The inclusion of $\boldsymbol{\lambda}$ in (26) thus ensures the system reduces to the original unfiltered solution (in the presence of no a priori state information). The presence of $\boldsymbol{\lambda}$ is a result of the constraints on the dynamic system. However, it should not be considered part of the state. In fact, the Lagrange multipliers are not something that really needs to be estimated; but we must include it to effect a hard constraint solution. This decision comes with some complications. Each λ_j in $\boldsymbol{\lambda}$ is associated with a particular pixel from the optical flow computation. However, there is not necessarily any correspondence between the pixels (and hence the λ_j) across iterations. Even worse, the number of pixels used (the dimension of $\boldsymbol{\lambda}$) varies across iterations. This means a subset of the state parameters are only present at one iteration, and their predicted values at time t are not based on any previously estimated values. An alternative interpretation would be to view these parameters $\boldsymbol{\lambda}$ as having infinite observation noise, or perhaps that the "observability" of $\boldsymbol{\lambda}$ is changing.

The *discrete process* equation for the Kalman filter gives an expression for the prediction of the state $\mathbf{y}(t+1)$ given the previous estimate $\mathbf{y}(t)$. In this case, this equation states that the predicted motion of the observed subject is the same as in the previous iteration, along with the added noise $\mathbf{w}$ (assumed to be independent zero-mean Gaussian noise with covariance $\mathbf{Q}$) to form the primarily data-driven system:

$$\mathbf{y}(t+1) = \mathbf{y}(t) + \mathbf{w}(t) \qquad p(\mathbf{w}) \sim N(\mathbf{0}, \mathbf{Q}) \tag{29}$$

The prior estimates of $\mathbf{y}$ and $\mathbf{P}$ used in the computation of the estimated state and covariance at time t are denoted $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{P}}$. Since $\boldsymbol{\lambda}$ is treated as a distinct value at each iteration, only the portions of $\tilde{\mathbf{y}}(t-1)$ and $\tilde{\mathbf{P}}(t-1)$ that correspond to $\dot{\mathbf{q}}$ are retained, resulting in:

$$\begin{aligned}
\tilde{\mathbf{y}}(t) &= \begin{pmatrix} \dot{\mathbf{q}}(t-1) \\ \mathbf{0} \end{pmatrix}, \\
\tilde{\mathbf{P}}(t) &= \mathbf{P}(t-1) + \mathbf{Q}(t-1) \\
&= \begin{bmatrix} \mathbf{P}_{\dot{\mathbf{q}}}(t-1) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{\dot{\mathbf{q}}_\mathrm{m}}(t-1) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_{\boldsymbol{\lambda}}(t-1) \end{bmatrix}
\end{aligned} \tag{30}$$

(where $\mathbf{P}_{\dot{\mathbf{q}}}$ is the block of $\mathbf{P}(t-1)$ corresponding to $\dot{\mathbf{q}}$).

$\mathbf{Q}$ is the covariance of the process noise, and represents the uncertainty in the process model. As estimation of $\mathbf{q}_\mathrm{b}$ is static, its corresponding block in $\mathbf{Q}$ is zero (in practice, a small amount of stabilizing noise is needed). $\mathbf{Q}_{\dot{\mathbf{q}}_\mathrm{m}}$ models the uncertainty of the actions of the observed subject. This way, the estimation of the static quantity $\mathbf{q}_\mathrm{b}$ will eventually cease as the estimated covariance of these parameters converges, while $\mathbf{q}_\mathrm{m}$ is interpreted as a dynamic quantity by the filter. $\mathbf{Q}_{\boldsymbol{\lambda}}$ is used to relax the hard constraints; this will be explained below.

Computing the estimated mean and covariance of $\mathbf{y}$ involves forming the Kalman gain matrix, which is used to combine the solution using the current measurements with the solution from the previous iteration. In the following filtering equations, all quantities are taken at time t, but this dependence is omitted to improve readability.

The Kalman gain matrix [3, 19, 30] is computed as:

$$\mathbf{K} = \tilde{\mathbf{P}}\mathbf{H}^{\top}\left(\mathbf{H}\tilde{\mathbf{P}}\mathbf{H}^{\top} + \mathbf{R}\right)^{-1} \tag{31}$$

The covariance matrix $\mathbf{R}$ is the sum of terms resulting from the noise in $\mathbf{f_q}$ and $\mathbf{I}_t$:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{\mathbf{f_q}} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathbf{I}_t} \end{bmatrix} \tag{32}$$

The relative scale between $\mathbf{R}_{\mathbf{f_q}}$ and $\mathbf{R}_{\mathbf{I}_t}$ provides a control for tuning how much trust goes into the optical flow information relative to the edge information.

The estimated mean is computed as a sum of the current solution $\mathbf{Kz}$ and the weighted prior mean estimate $\tilde{\mathbf{y}}$, or as the sum of the prior estimate $\tilde{\mathbf{y}}$ and the innovation $(\mathbf{z} - \mathbf{H}\tilde{\mathbf{y}})$ weighted by $\mathbf{K}$:

$$\mathbf{y} = \mathbf{Kz} + (\mathbf{1} - \mathbf{KH})\,\tilde{\mathbf{y}} = \tilde{\mathbf{y}} + \mathbf{K}\,(\mathbf{z} - \mathbf{H}\tilde{\mathbf{y}}) \tag{33}$$

The estimated covariance [3, 19, 30] is computed from the prior covariance $\tilde{\mathbf{P}}$ as:

$$\mathbf{P} = (\mathbf{1} - \mathbf{KH})\,\tilde{\mathbf{P}} \tag{34}$$

4.4.1 Relaxing hard constraints

In understanding why this formulation relaxes the hard constraints, it is much clearer to consider the following alternative (and algebraically equivalent) formulation [30] of (31)-(34):

$$\begin{aligned} \mathbf{y} &= \mathbf{P}\mathbf{H}^{\top}\mathbf{R}^{-1}\mathbf{z} + \mathbf{P}\tilde{\mathbf{P}}^{-1}\tilde{\mathbf{y}} \\ \mathbf{P} &= \left(\mathbf{H}^{\top}\mathbf{R}^{-1}\mathbf{H} + \tilde{\mathbf{P}}^{-1}\right)^{-1} \end{aligned} \tag{35}$$

When written this way, it is clear how the solution of the measurement equation is being combined with the a priori information, based on their corresponding covariances.

Consider the case when $\mathbf{R} = \mathbf{1}$, $\mathbf{P}(t-1) = \mathbf{0}$, $\mathbf{Q}_{\dot{\mathbf{q}}_m}(t-1) = \mathbf{0}$ and $\mathbf{Q}_{\boldsymbol{\lambda}}(t-1) = \frac{1}{\alpha}\mathbf{1}$, with $\alpha > 0$ (using $\tilde{\mathbf{P}}^{+}$ for $\tilde{\mathbf{P}}^{-1}$ in the absence of prior information). The result simplifies to:

$$\mathbf{y} = \left(\mathbf{H}^{\top}\mathbf{H} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \alpha\mathbf{1} \end{bmatrix}\right)^{-1} \mathbf{H}^{\top}\mathbf{z} \tag{36}$$

Without the addition of $\alpha\mathbf{1}$, this would be the hard constraint solution in (20). The addition of $\alpha\mathbf{1}$ relaxes the hard constraint, with more constraint violation as α increases. This is because the Lagrange multipliers that were used to enforce the constraint are gradually driven towards zero as α increases, since they are increasingly combined with the a priori value of $\boldsymbol{\lambda}$ in $\tilde{\mathbf{y}}$ (which is zero). In fact, when α is sufficiently large, this solution approaches that

of a soft constraint solution (i.e. one without Lagrange multipliers), since:

$$\lim_{\alpha\to\infty}\left(\mathbf{H}^\top\mathbf{H}+\begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{0}\\\mathbf{0}&\mathbf{0}&\mathbf{0}\\\mathbf{0}&\mathbf{0}&\alpha\mathbf{1}\end{bmatrix}\right)^{-1}=\begin{bmatrix}\mathbf{1}&\mathbf{0}&\mathbf{0}\\\mathbf{0}&(\mathbf{B}^\top\mathbf{B}+\mathbf{1})^{-1}&\mathbf{0}\\\mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix} \tag{37}$$

which, when right multiplied by $\mathbf{H}^\top$, produces $\mathbf{H}^{\text{soft}^+}$ (with additional rows of zeros).

In the general case, $\mathbf{P}(t-1)$ and $\mathbf{Q}_{\boldsymbol{\lambda}}(t-1)$ will also cause constraint violation, but not in any controlled way. Rather, their presence causes the solution to be a balance between the measurement equation solution and the prediction in a way that doesn't respect the constraint. In other words, the Kalman filter can put more trust in the prediction (which can violate the constraint) at times when the estimate of $\dot{\mathbf{q}}_{\text{m}}$ is noisy. In practice, the form of $\mathbf{Q}_{\boldsymbol{\lambda}}(t-1)$ is still $\frac{1}{\alpha}\mathbf{1}$, with α determined during filter tuning (the determined value cancelled on average 97% of the component of the edge forces which violated the constraint, on each iteration of the extended Kalman filter). Keep in mind that the only distinction between this solution, and an ordinary use of a Kalman filter, is the inclusion of the Lagrange multipliers in the state variable, and their process update in (30). Neither of these betray the assumptions made in the derivation of the iterated extended Kalman filter [31], and as a result, the solution here has the same stability properties as an ordinary solution.

The Kalman filter solution presented here has a number of advantages over the direct solution from (20), and the commonplace use of a Kalman filter for data fusion. It makes the framework more robust to noise and small estimation errors. More importantly, it provides a valuable means for combining the edge forces and optical flow information; the optical flow constraint is now relaxed to a degree based on the error in the optical flow information in a way that makes the system more efficient and robust. Next, experiments will be presented which show that the use of a Kalman filter (in addition to treating optical flow as a hard constraint) was an important addition to the system.

5 Experiments and discussion

This section contains the results from a series of face shape and motion estimation experiments. The first three experiments exhibit the generality of our system on a variety of subjects, while the next four experiments use a common observed subject, and provide a quantitative validation of the shape and motion estimation. The last of the validation experiments compares a number of related frameworks mentioned in this paper.

5.1 Initialization

The entire estimation process is automatic, except for the initialization, which requires the manual specification of several landmark features in the first frame of the sequence (the eyebrow centers, eye corners, nose tip, and mouth corners). The subject must also be at rest, and (approximately) facing forward, as in Figure 4(a). In all the experiments, except for those used for motion validation, the shape of the face is estimated only from the images.

Using these marked features, forces are applied to the initial face model that deform the corresponding points

on the face toward the desired locations in the image. Experience has shown that the initialization process is robust to small displacements (i.e. several pixels) in the selected landmark points. The rotation and translation, as well as course-scale face shape parameters (such as those which determine the positions and sizes of the face parts) are fitted using this information, the result of which is shown in Figure 4(b). Once roughly in place, both edge and anthropometry forces are applied that pull the face into the correct shape as in Figure 4(c). The distance from the initial face to the camera is determined given the assumption that the subject's face is the same size as the model.

(a) (b) (c)

Figure 4: Model initialization

The problem of automatically locating the face and its various features has been addressed elsewhere [49, 50], and could be used to make this process automatic. No markers or make-up are used on the subject (markers are used for the validation of the method, however, as described below).

5.2 Tracking experiments

The original image sequences are 8 bit gray images at NTSC resolution (480 vertical lines). In each of the sequences, the width of the face in the image averages 200 pixels, and the range of motion of features across the image sequence is typically 80 to 100 pixels. For each of the tracking examples, several frames from the image sequence are displayed, cropped appropriately. Below each, the same sequence is shown with the estimated face superimposed. In each case, a model initialization is performed as described above. The initialization process usually takes about 2 minutes of computation. Afterwards, processing each frame (using the extended Kalman filter formulation) takes an average of 1.4 seconds each (all computation times are measured on a 175 MHz R10000 SGI O2).

The sequence shown in Figure 5 was taken on an IndyCam at 5 fps. Figure 5 shows a subject turning her head in (a) through (d) and opening her mouth from (d) to (f). Based on the good alignment of the face model with the image, it appears the face model is able to capture the shape of her face, as well as the head rotation and mouth motion. The next two sequences were taken on a higher quality camera at 30 fps. Both Figure 6 and Figure 7 show a subject smiling and moving forward in (b) and (c), opening their mouth while turning their head in (e) and (f), and turning back, closing their mouth slightly in (g). All of these motions appear to be correctly tracked based on the observed motion. These three experiments involve different subjects, having very different appearances. This suggests the verification of the face model shape parameterization (described in Appendix B) was successful. The extracted face shape is quite individualized to the subject, but not to the point that would be useful for certain applications in computer graphics. These extracted models for the three subjects here in Figures 5, 6 and 7 are on

the right side of Figure 2 (the upper-right, lower-left and lower-right, respectively).

5.3 Shape estimation validation

This experiment provides a validation of the shape estimation accuracy of our system. The extracted shape (specified by $\mathbf{q}_b$) is validated by comparing with a Cyberware range scan of the subject, shown in Figure 8(a).

The shape estimation validation experiment in Figure 9 shows the subject performing small head motions in (a) through (f) while smiling in (c) and (d), and finishing with a significant head rotation in (g). At each frame, Figure 10 shows the extracted shape results as compared against the range scan of the subject. Note that for this comparison, all motion parameters are ignored, so that only the shape is compared. The RMS error is computed using the nodes of the model, and also includes a uniform scaling of the model so that the two faces are the same scale (this eliminates the depth ambiguity—in this case, the estimated model was compared at 96% scale). The rigid alignment (translation and rotation) as well as this uniform scaling were computed using a semi-automatic alignment method (the chosen alignment had the smallest RMS error).

The RMS error, which starts at around 2 cm after initialization, shows a gradual reduction over the course of the experiment, ending around 1 cm, with the large reduction in error around frame 50 corresponding to when the subject turned his head significantly to the side in Figure 9(f) and (g), where the profile view contained good edge information to fit the face shape.

5.4 Motion estimation validation

The next three experiments use markers to allow for the validation of the motion tracking of our technique. Eleven small circular markers were placed on the face of a subject. Analysis of the accuracy of the motion estimation in $\mathbf{q}_m$ is performed using these markers on the subject, which allow for alignment verification in the image plane (ground truth motion in 3D is not available).

For these three experiments, no shape estimation is performed. Instead, the face shape is provided by an off-line fitting of the face model to the range scan in Figure 8(a)—this way, any deviation can be attributed primarily to motion error, not shape error. In addition, the fixed locations of the markers on the model are determined using some additional images taken of the subject, shown in Figure 8(b). The markers are fixed into particular locations of the polygon mesh (they have fixed coordinates in Ω). The model resulting from this fitting and marker placement is shown in Figure 8(c), with the marker locations shown as dark circles. The RMS error of the extracted model (comparing the extracted model with the range scan) is 0.26 cm.

First, the image locations of each of the markers from the image sequence is obtained using a semi-automatic tracking system. The rough location of the markers is tracked using the KLT[2] package (which is based on [43]), and was fine tuned using a deformable ellipse template. Simple calibration tests suggest this tracking technique has a variance of 0.35 pixels in measuring the center of a marker (which are usually about 8 pixels across) in the image.

[2]Stan Birchfield's KLT package is available at `http://vision.stanford.edu/~birch`

(a) (b) (c) (d) (e) (f)

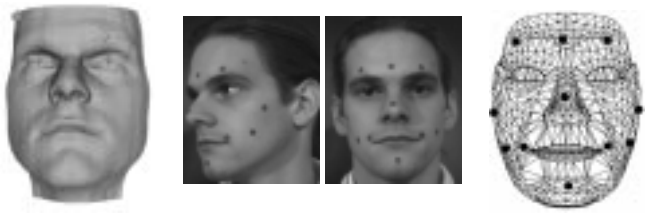
Figure 5: Motion and expression tracking example 1

(a) (b) (c) (d) (e) (f) (g)

Figure 6: Motion and expression tracking example 2

(a) (b) (c) (d) (e) (f) (g)

Figure 7: Motion and expression tracking example 3



(a) (b) (c)

Figure 8: (a) Shaded range scan of subject, (b) Marker calibration images, (c) Resulting marked model

Care was taken so that the presence of the markers did not significantly affect the motion estimation, since these markers could provide useful information for tracking. The pixel selection method for the optical flow information was modified so that no points were selected that were within 3 pixels (the radius of the spatial derivative filters) of any point on a marker. In addition, any edges used to produce edge forces were similarly limited to be distant from markers. Given that the markers were not placed directly on top of important facial features, it is unlikely that the presence of the markers detrimentally affected the experiment results.

In each of the following three motion validation experiments, there is an accompanying graph showing the displacement error for each frame. This displacement error of a marker is the Euclidean distance (in pixels) between the image location of the marker (if visible), and the predicted image location of the marker given the model (which is the projected image location of the model marker). The dark line on the graph shows the mean displacement error of all visible markers (one standard deviation is indicated by the gray region surrounding it). The dotted lines indicate the minimum and maximum displacement error.

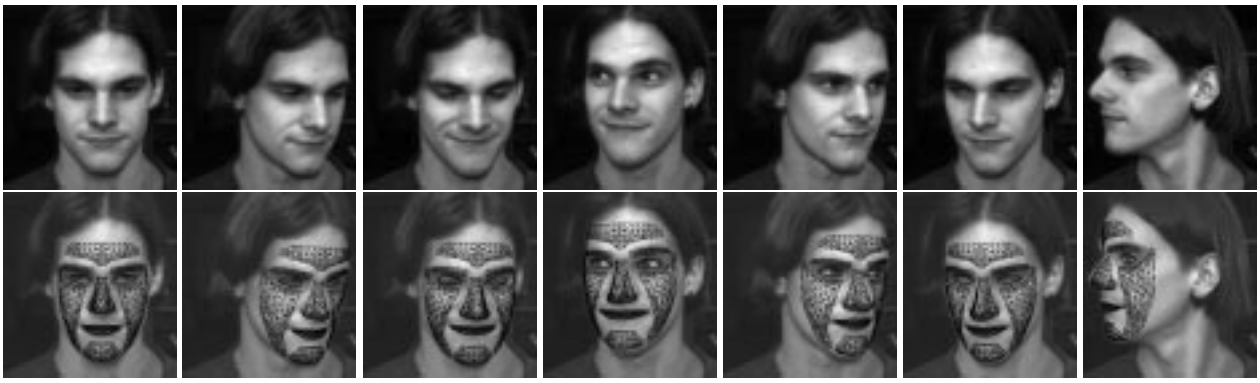
The first two sequences were taken using an IndyCam at 5 fps. The final sequence was taken on a high quality camera (Pulnix TM-9701; grayscale, progressive scan) at 30 fps. Also note that this final sequence was taken at a different time than the first two—the markers were re-applied to the subject, and their locations were determined again, as in Figure 8(b) and (c). Their new locations were roughly the same as in the earlier validation experiments (at most 1.5 cm difference).

The sequence in Figure 11 shows predominantly non-rigid motion (facial expressions). The subject moves forward and frowns his eyebrows in (b), moves back and produces a surprise expression in (d), followed by a smile in (f). The average error shown in Figure 12 is between 2 and 4 pixels, which given the face is approximately 200 pixels across in the image, amounts to less than 2%. The maximum error of around 7 pixels corresponds to around 3.5% (roughly 0.5 cm). The largest error is produced during the smile expression; possible reasons for this are discussed in the next section.

The second sequence in Figure 13 is a combination of rigid and non-rigid motions. The subject turns his head from (a) through (d) while smiling, returning to rest position in (f). The displacement error shown in Figure 14 averages from 2 to 4 pixels (but being closer to 4 for a longer period), reaching a maximum of just over 7 pixels. The largest error is produced when the smile is viewed from the side, and is concentrated in the mouth area.

The last sequence in Figure 15 is primarily a rigid-motion sequence that is significantly longer than the other experiments (760 frames). It includes head rotations in a variety of directions, as well as some large head translation (side-to-side and away from the camera). Eyebrow raises and a smile are also present. This sequence demonstrates

the ability of the system to maintain track over a long sequence, without experiencing failure due to tracking drift. In this sequence, the face is approximately 140 pixels across in the image (somewhat smaller than in the previous experiments). The average pixel deviations shown in Figure 16, range between 1.5 and 2.8 pixels, with a maximum error at 4.6 pixels, corresponding to about the same absolute distance error as with the previous experiments (roughly 0.5 cm). Hence, the apparently lower pixel deviations for this sequence amount to approximately the same error in actual distance. During the sequence, some of the motions were very close to the maximum limits of tracking speed (pixel velocities were about the same size as the derivative filter width). In particular, the turning motion at frames 250–320 is the most serious, with other occurrences at frames 430–450 and 610–620. These motions manifest themselves in Figure 16 as larger displacement errors. However, during the successive motions (which are well below this maximum velocity), the system recovers from these errors, and improves the fit using edge information, returning to the baseline deviation amount of around 2 pixels. This baseline corresponds to the maximum accuracy of model-edge alignment, and the limited precision of the marked model in Figure 8(c).



(a) Frame 1 (b) Frame 11 (c) Frame 18 (d) Frame 24 (e) Frame 35 (f) Frame 46 (g) Frame 57

Figure 9: Shape validation experiment

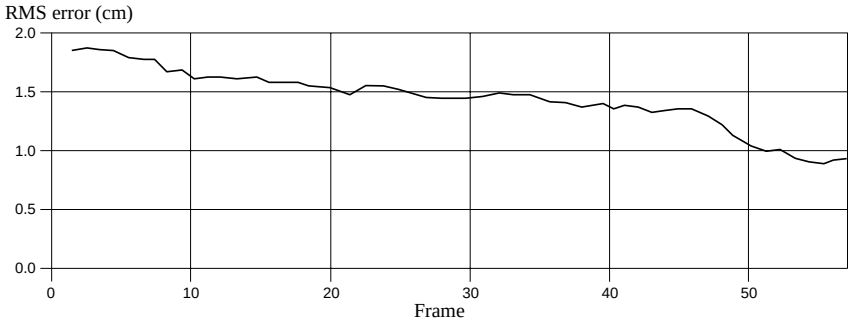


Figure 10: Results of shape validation experiment

5.5 Discussion

The successful tracking performed by this framework is primarily due to the use of optical flow as a constraint. This is empirically verified by disabling key components of our tracking system, and observing the resulting per-

(a) Frame 1 (b) Frame 11 (c) Frame 16 (d) Frame 18 (e) Frame 24 (f) Frame 29 (g) Frame 40

Figure 11: Motion validation experiment 1 (no shape estimation performed)

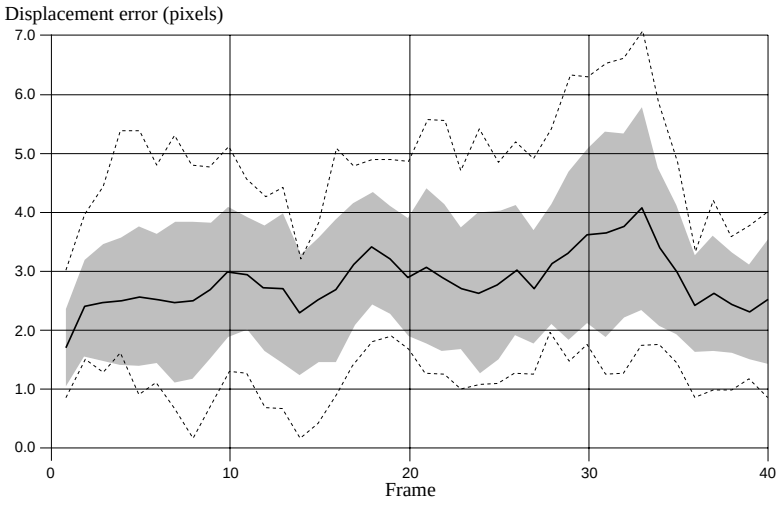


Figure 12: Results of motion validation experiment 1

(a) Frame 1 (b) Frame 16 (c) Frame 18 (d) Frame 27 (e) Frame 39 (f) Frame 43

Figure 13: Motion validation experiment 2 (no shape estimation performed)

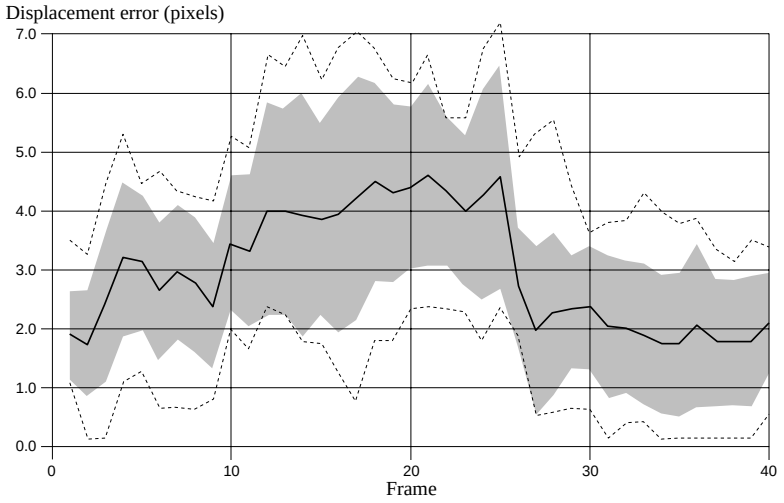


Figure 14: Results of motion validation experiment 2

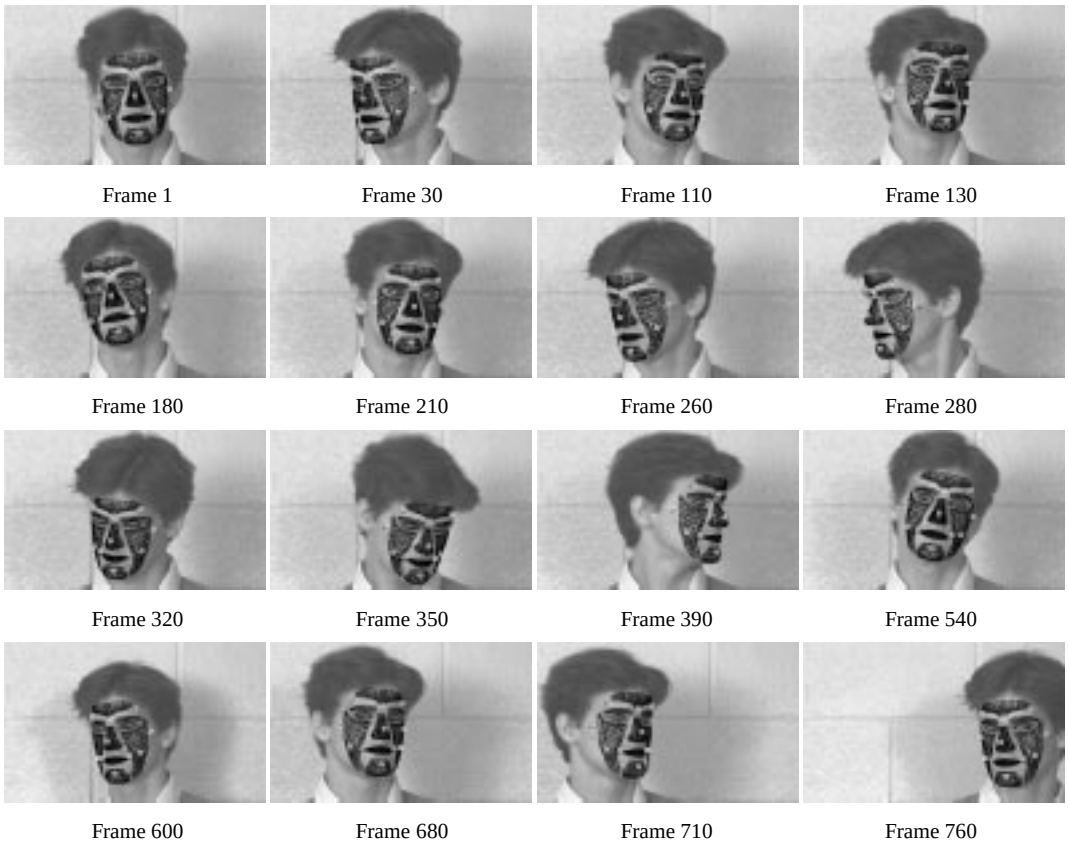


Figure 15: Motion validation experiment 3 (no shape estimation performed)

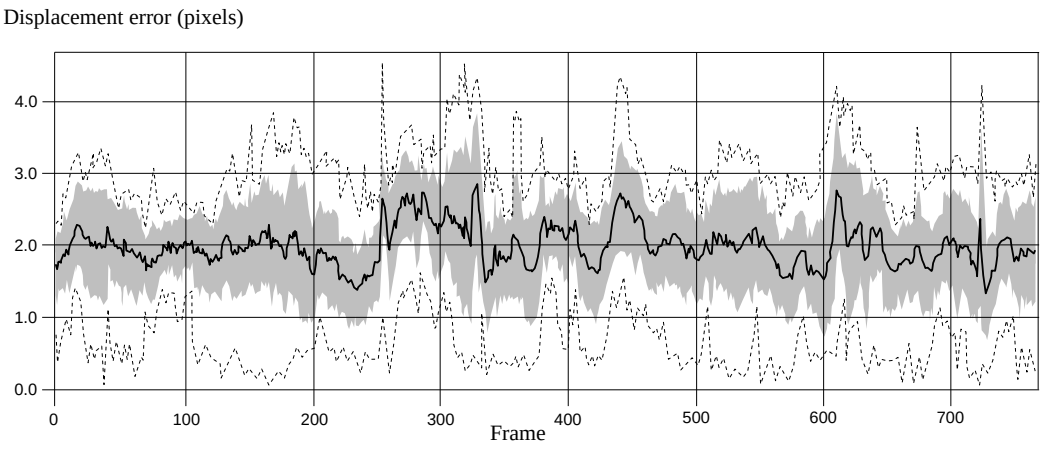


Figure 16: Results of motion validation experiment 3

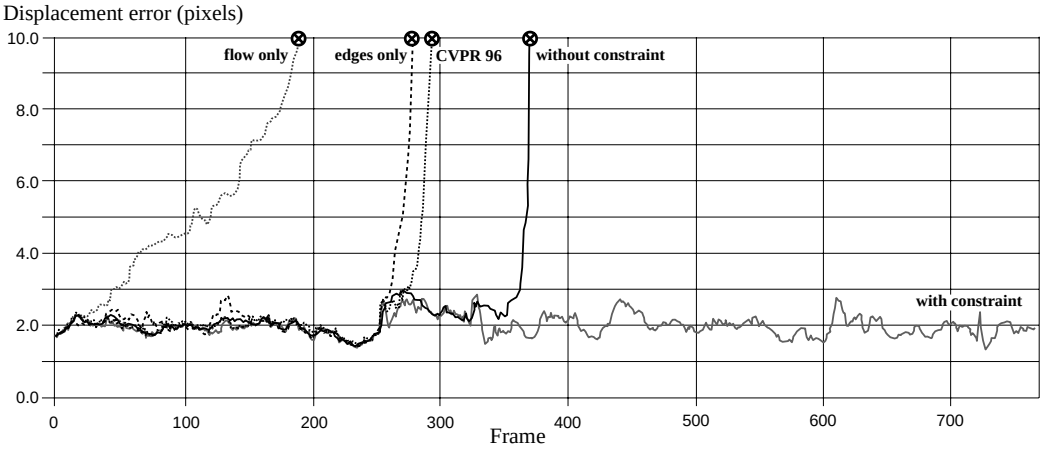


Figure 17: Tracking performance of various frameworks

Framework	Average time for entire frame	Average iterations within a frame
with constraint	1.4 seconds	2.9
CVPR 96 (constraint; no KF)	1.3 seconds	5.5
without constraint	6.5 seconds	17.7
flow only	0.34 seconds	1 (not iterative)
edges only	15 seconds	36.1

Figure 18: Timing of various frameworks (175 MHz R10000 SGI O2)

formance decrease (in the form of tracking failures[3]). The results of running the experiments in Figure 15 on the various frameworks is shown in Figure 17. The timings shown in Figure 18 include the average execution time for a single frame (for all iterations) on a 175 MHz R10000 SGI O2, along with the average number of iterations required within a frame.

The results from the constraint-based Kalman filtering framework for the third experiment are shown in Figure 16 (and also in Figure 17 as the line marked "with constraint"). The framework which uses both optical flow and edges, but uses the measurement equation in (27) which does not incorporate optical flow as a hard constraint, is shown as "without constraint" in Figure 17. This system can experience tracking failure (as in frame 370) when it encounters a difficult model-edge alignment problem (when the deviation is large, or many parameters require adjustment). It is worth noting that a constraint-based Kalman filtering method without the relaxation (a framework like "with constraint" but with $\mathbf{Q}_{\boldsymbol{\lambda}} = \mathbf{0}$) had tracking performance that was virtually the same as the "without constraint" method (although was just as fast as the method "with constraint").

The framework in [11] (labeled "CVPR 96") used an ad hoc filtering method (to soften the constraint) instead of a Kalman filter. In other words, this system used flow as a hard constraint, but did not use a Kalman filter. While each iteration took less time, more iterations were required, resulting in roughly the same timing as the method which uses a Kalman filter. However, this method is not as robust, losing track around frame 290.

[3]Tracking failure is simply defined as reaching a 10 pixel deviation–at this point the deviation typically increases, with tracking being re-gained only by luck.

It is also worthwhile to test each data source separately (but still using a Kalman filter). When edges are not used, leaving only the model-based optical flow solution, errors in the estimation of $\dot{\mathbf{q}}_m$ accumulate (since this solution is integrating a velocity), causing the model to lose track quite quickly. This method is marked as "flow only" in Figure 17. While a non-linear iterative solution [5] would improve the accuracy, it would still not prevent tracking drift.

The method using only edge information (marked "edges only" in Figure 17) often finds local minimum solutions (such as around frame 80 and frame 130), some of which can lead to tracking failure (near frame 280). As the model-image displacement increases, the model-edge alignment problem becomes quite difficult and expensive to solve. Tracking failure occurs in situations not unlike those that caused problems for the framework marked "without constraint".

While the framework using hard constraints performed well in this sequence, we can add noise to the system to determine at what point the system fails. This tracking experiment was run again (a number of times) to experimentally determine the minimum sustained deviation that causes tracking failure. After each iteration, Gaussian noise was added (of increasing variance until tracking failed) to the rigid motion parameters in $\mathbf{q}_m$ at the start of each iteration. Tracking failure became common as average displacement errors went above 4.6 pixels (the incidence of failure went from non-existent below 4.5, to prevalent by 4.7). Alternatively, adding Gaussian noise directly to the images (of increasing variance until tracking failed) produced a similar value (average displacement errors of 4.4 pixels, with a corresponding image noise variance of 15.5% of intensity). Incidences of tracking failures for the other systems (when noise was added) became noticeably worse during these tests. This suggests that our system using relaxed hard constraints has a comfortable margin of safety from tracking failure.

Considering all the experiments, the error in the tracking results can have other (non-noisy) sources, besides motion estimation error. One possibility is that it can be caused by poorly extracted marker locations (although this distance is less than a pixel). Another source can be the discrepancy between the face shape used and the shape of the observed subject. The RMS error between the face shape and the range scan for *only* the marker points is much lower than that from the whole model; it is 0.1 cm, which will cause at most 1 pixel of deviation in marker locations. Violation of the assumption of perspective projection is also a possible contributor to error, although in this case is minimal, given the small depth range of the face compared to the distance of the face to the camera. From this, it can be concluded that a significant portion of the errors present here are from motion estimation.

Upon closer examination, it can be seen that the larger errors which are present during non-rigid motions (in particular, smiling), are caused by the smile produced by the model not matching the smile on the subject. In particular, the subject's smile is more curved than the one produced by the model. Also, the smile produced by the model does not move the mouth back (into the face) far enough, which explains the fact that the most error is present when the smile is viewed from the side. These errors result from the inability to estimate the scaling constants used in (46). Attempting to estimate these constants for each individual using only edge forces does not produce reliable results.

5.6 Limitations

The many experiments in this section show the capabilities of the shape estimation and tracking systems described in this paper. On the other hand, they also say a lot about what the limitations of the system are.

First, some of the limitations of the system come directly from the assumptions made during design. Most obvious is the assumption of brightness constancy during optical flow computation. Major lighting changes can cause tracking failure. Specularities also cause small problems, but tend not to affect the entire model, since they tend to be fairly localized. In some cases, poor lighting will also lead to tracking failure. Typically, these occur in situations where edges are washed out (opening the aperture too wide on a camera will do this).

Second, is to simply exceed the maximum tracking speed (determined by the derivative filter width). This problem can be addressed by using multi-scale optical flow methods. On a related note, the motions and edges can also become too small to be estimated accurately. When the face in the image is smaller than about 40 pixels across, there is not enough edge information to maintain track reliably.

Third, are deviations from the model—where the images go past the coverage limits of the model. Attempting to track motions that are not represented produces relatively unpredictable effects. For example, lip puckering is not modeled: tracking this facial motion produces the best fit using the existing motion parameters (and can often be quite far off). This causes poor model-image alignment, which can lead to tracking failure if the unmodeled motion is very large. Note that during speech, however, the system retains good track of the head and brows, while the motion parameters affecting the mouth region are garbled. This is not surprising, as these unmodeled motions are attributed to other parameters in the same region (in a least squares way). Large occlusions produce similar problems (such as a hand passing in front of the face). And of course, since a "mask" face model is used, our framework will lose track during head motions where the mask visibility becomes too small. There is hope for detecting these problems automatically—many of these difficulties first appear as large increases in the constraint residual (localized to the region of model deviation).

Finally, are the problems associated with the tracking of multiple, simultaneous motions. In the validation experiments, situations where head rotation was accompanied by a non-rigid expression deformation often produced higher pixel deviations. On occasion, this deviation can be serious enough to cause tracking failure. This is caused by the linearization in the model-based optical flow solution, which could perhaps be alleviated by using an iterative least-squares solution [5]. There can also be situations where motions can be confused (given a particular model configuration, two parameterized motions may appear nearly identical). The problem arises when the model state changes to make the current motion estimate inconsistent. Multiple hypothesis estimation methods might provide a solution here, although it's likely the most robust solution (for some applications) would be simply to detect and recover from such a situation.

6 Conclusions

We have presented a method for treating optical flow information as a hard constraint on the motion of a deformable model. We have argued, as well as empirically demonstrated, that it was the treatment as a *hard* constraint which resulted in the benefits in efficiency and robustness. Furthermore, we showed how a hard constraint

based on noisy data can be softened using a Kalman filter while preserving these beneficial aspects. Finally, hard constraints provided a means for combining information sources which allowed edge information to be used along with optical flow in order to combat error accumulation in tracking.

Our use of a detailed three-dimensional model also helped a great deal. By accounting for the self-occlusion of the face, large amounts of head rotation can successfully be tracked. Our detailed shape model allowed for accurate descriptions of facial shape to be extracted, the parameterization for which would not have been possible to implement without the use of face anthropometry data to control model coverage. Finally, by designing the model with a separable shape and motion parameterization, we can separate the problem of estimating the shape of an individual's face from estimating their motion, resulting in a much smaller dynamic estimation problem.

The current system does have a number of limitations, however. The most significant of which is the idealization of the optical flow constraint equation. For instance, the problems of photometric variation and self-shadowing, which violate the optical flow constraint equation, are not addressed. The presence of a three-dimensional model could prove to be useful when addressing these problems. Another limitation is in tracking large motions; at the moment, motions larger than the width of the derivative filters will not be tracked correctly. Multi-scale model-based optical flow techniques [5] can be applied here to address this.

Looking to the future, investigation of the recognition of faces using the shape parameterization, or of facial expressions using the motion description is worth pursuing. Simplistic approaches that depend on a particular parameterization (such as directly using the "smile" parameter with a threshold) would not be robust. Also, having a more detailed motion parameterization will allow for the tracking of more complex facial motions. Methods for generating motion models from example data, which are becoming more commonplace as data gathering methods improve, would be particularly effective in building such a complex model.

Acknowledgments

We would like to thank Eero Simoncelli, Max Mintz, Matthew Turk, Pascal Fua, Frank Dellaert, Yaser Yacoob and the reviewers of this paper for their helpful comments and discussions. We are also grateful to Yaser Yacoob and the Center for Automation Research at the University of Maryland College Park for providing the image sequences in Figure 6 and Figure 7. This research is partially supported by ARPA DAMD-17-94-J-4486 and DAAH-049510067; ARO DURIP DAAH-049510023; NSF IRI93-09917, IRI95-04372 and NSF-MIP94-20393; and a RuCCS postdoctoral fellowship.

A Modularization of global deformations

The shape model $\mathbf{x}$ is defined through the repeated application of n global deformations $\mathbf{T}_k : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, where $k \in 1 \ldots n$, to the underlying shape $\mathbf{s}$ (which has parameters $\mathbf{q_s}$ in general deformable model frameworks) as:

$$\mathbf{x}(\mathbf{q};\mathbf{u}) = \mathbf{T}_n(\mathbf{q}_{\mathbf{T}_n}; \ldots \mathbf{T}_1(\mathbf{q}_{\mathbf{T}_1}; \mathbf{s}(\mathbf{q_s};\mathbf{u}))) \tag{38}$$

where $\mathbf{q}_{\mathbf{T}_k}$ are the parameters used by $\mathbf{T}_k$. The parameters used by all of the global deformations are accumulated into the vector $\mathbf{q}_\mathbf{T}$ as in:

$$\mathbf{q}_\mathbf{T} = (\mathbf{q}_{\mathbf{T}_1}^\top, \ldots, \mathbf{q}_{\mathbf{T}_n}^\top)^\top \tag{39}$$

so that $\mathbf{q}$ can now be grouped as:

$$\mathbf{q} = (\mathbf{q}_\mathbf{s}^\top, \mathbf{q}_\mathbf{T}^\top)^\top \tag{40}$$

For a particular set of deformation functions, closed form expressions for the resulting shape can be derived. From these complex expressions, the Jacobian matrix can be derived (see [33] for an example), although this method is tedious and not modular.

Instead of this, a single expression for the resulting shape is not derived, but rather each deformation is applied separately given the definition in (38). The Jacobian matrix can be calculated in a similar way using the chain rule. First, define the deformation τ_k as the composition of the first k deformation functions $\mathbf{T}_1$ through $\mathbf{T}_k$:

$$\tau_k(\mathbf{q}_\mathbf{T}; \mathbf{p}) = \mathbf{T}_k(\mathbf{q}_{\mathbf{T}_k}; \ldots \mathbf{T}_1(\mathbf{q}_{\mathbf{T}_1}; \mathbf{p})) \qquad \mathbf{p} \in \mathbb{R}^3, \quad k \in 1 \ldots n \tag{41}$$

with τ_0 defined to be the identity. Given this definition of τ_k, it follows how to compute $\mathbf{J}_\mathbf{x}$, the Jacobian of $\mathbf{x}$ with respect to $\mathbf{q}$, using the following recurrence:

$$\begin{aligned} \mathbf{J}_{\tau_0} &= \mathbf{J}_\mathbf{s} = \frac{\partial \mathbf{s}}{\partial \mathbf{q}_\mathbf{s}} \\ \mathbf{J}_{\tau_k} &= \left[\frac{\partial \mathbf{T}_k(\mathbf{p})}{\partial \mathbf{p}} \mathbf{J}_{\tau_{k-1}} \;\middle|\; \frac{\partial \mathbf{T}_k}{\partial \mathbf{q}_{\mathbf{T}_k}} \right] \quad k \in 1 \ldots n \end{aligned} \tag{42}$$

so that $\mathbf{J}_\mathbf{x} = \mathbf{J}_{\tau_n}$. The left block in (42) uses the chain rule, so that the matrix $\partial \mathbf{T}_k(\mathbf{p}) / \partial \mathbf{p}$ "deforms" the individual columns of the Jacobian matrix $\mathbf{J}_{\tau_{k-1}}$. The right block in (42) contains the derivatives of the outermost deformation $\mathbf{T}_k$ with respect to its parameters.

A naive technique for computing $\mathbf{J}_\mathbf{x}$ using this recurrence from the bottom-up (which starts with $\mathbf{J}_\mathbf{s}$), is particularly expensive in terms of both time and space complexity. This is particularly a problem since the Jacobian needs to be re-evaluated at each iteration, over many points on the model. Instead, the quantity $\mathbf{J}^\top \mathbf{f}$ is computed, given an applied force $\mathbf{f}$ such as in (6). The quantity $\mathbf{J}^\top \mathbf{f}$ can be computed efficiently in a top-down fashion as:

$$\mathbf{f}_n = \mathbf{f}\,, \qquad \mathbf{f}_{k-1} = \left(\frac{\partial \mathbf{T}_k(\mathbf{p})}{\partial \mathbf{p}} \right)^\top \mathbf{f}_k \quad k \in 1 \ldots n \tag{43}$$

$$\mathbf{J}_\mathbf{s}^\top \mathbf{f} = \left(\frac{\partial \mathbf{s}}{\partial \mathbf{q}_\mathbf{s}} \right)^\top \mathbf{f}_0\,, \qquad \mathbf{J}_{\mathbf{T}_k}^\top \mathbf{f} = \left(\frac{\partial \mathbf{T}_k}{\partial \mathbf{q}_{\mathbf{T}_k}} \right)^\top \mathbf{f}_k \quad k \in 1 \ldots n \tag{44}$$

If the actual columns of $\mathbf{J}_\mathbf{x}$ are required, as is the case for the optical flow computation (12), they can be found by

three applications of the above technique using the unit vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$ in the x, y and z directions, respectively, as:

$$\mathbf{J}_{\mathbf{x}}^{\top} = (\mathbf{J}_{\mathbf{x}}^{\top}\hat{\mathbf{i}})\hat{\mathbf{i}}^{\top} + (\mathbf{J}_{\mathbf{x}}^{\top}\hat{\mathbf{j}})\hat{\mathbf{j}}^{\top} + (\mathbf{J}_{\mathbf{x}}^{\top}\hat{\mathbf{k}})\hat{\mathbf{k}}^{\top} \tag{45}$$

since $\hat{\mathbf{i}}\hat{\mathbf{i}}^{\top} + \hat{\mathbf{j}}\hat{\mathbf{j}}^{\top} + \hat{\mathbf{k}}\hat{\mathbf{k}}^{\top} = \mathbf{1}$. For the optical flow computation, this construction is only required for the motion parameters in $\mathbf{q}_{\mathrm{m}}$.

Besides global deformations, it is also useful to include rigid motions (translations and rotations) and even camera projections. For the case of camera projections, however, the mapping becomes $\mathbf{T} : \mathbb{R}^3 \to \mathbb{R}^2$, and (45) uses only $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$, since the image forces are two-dimensional. The formulation of the projected Jacobians in (3) and (4) is simply an instance of the left block of (42).

This modular technique for computing the Jacobian matrix allows for significantly easier implementation at little computational expense. It is also a more modular approach, since the choice of which deformations used can be made on the fly.

B Face model deformations

In order to capture the variations seen in the shape and motion of human faces, a mixture of scaling, bending, and rigid deformations are used in the construction of the face model. This section provides details on these deformations. The model designer carefully combines the deformations to produce a parameterized face model. The result of this construction is an underlying model (the polygon mesh) which has a series of deformations functions applied to it, each having a small number of parameters, and each is applied to a particular set of face parts, ranging from a single part to the entire face.

Rigid transformations such as translation and rotation are used for the placement of parts on the face. Scaling and bending deformations, shown in Figure 19, allow for the representation of a variety of face shapes. Each of these deformations is defined with respect to particular landmark locations in the face mesh. By fixing the deformations into the mesh, the desired effect of any particular deformation is not lost due to the presence of other deformations (since the landmark points are deformed along with the rest of the mesh). Although varying degrees of continuity can be attained for these deformations, each of the following deformations are C^1 continuous.

A shape (before any deformation is applied) which contains the landmark points p_0, p_1 and c is shown in Figure 19(a). Figure 19(b) shows the effect of scaling this shape along the displayed axis. The center point c is a fixed point of the deformation, while the region between p_0 and p_1 is scaled to have length d (the parameter of the scaling deformation). Portions of the shape outside this region are rigidly translated.

Bending is applied in Figure 19(c), and shows the effect of bending the shape in (a) in a downward direction. The bending is applied to the area between p_0 and p_1, where c is the center of the bending. Outside this area, the shape is rotated rigidly. Each plane perpendicular to the bending axis is rotated by an angle determined by the distance of this plane to the center point c. The amount of bending is specified by the parameters θ_0 and θ_1, which specify the rotation angle at p_0 and p_1, respectively.

In addition, the spatial extent of each of these deformations can be localized, as shown in Figure 19(d). The

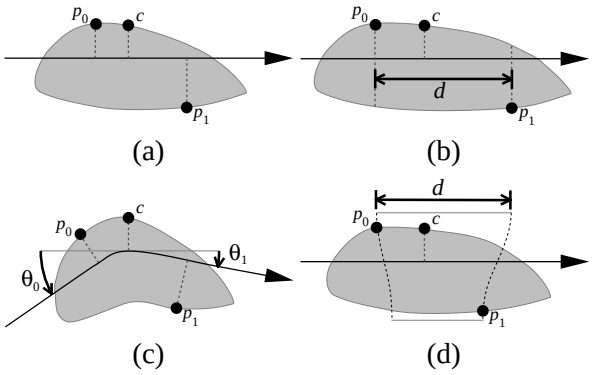


Figure 19: Scaling and bending deformations

influence of the scaling deformation varies in directions perpendicular to the axis, producing a tapering effect. Near the top of the shape, the object is fully scaled to be the length d, while the bottom of the object is unaffected by the deformation. The ability to restrict the effect of a deformation is vital in specifying the variations of shape seen in the face. We will now see how these deformations can be used to create the model.

B.1 Face shape

The underlying shape **s**, which is the polygon mesh shown in Figure 1, can take the shape of a variety of faces through the application of a number of spatial deformations. This parameterization of the model is specified by the model designer. The job of the designer is made easier by separating the face into parts, allowing each face model component to be treated separately. Instead of describing the entire model (which would be extremely lengthy and not particularly enlightening), a short description is provided which illustrates the concepts necessary for its construction.

Deformations are defined over a particular set of face model parts, although most deformations affect only one part. Example deformations that parameterize multiple parts include those affecting the lower face, which deform the chin and both cheeks. All of the deformations are specified in a particular order, and are applied in sequence to the underlying shape (see Appendix A). All of the parameters to describe the shape of the face at rest (there are approximately 80) are collected together into $\mathbf{q}_\mathrm{b}$. The shape deformations are collected together into a single deformation function $\mathbf{T}_\mathrm{b}$. Most of the parameters are independent due to spatial locality, which keeps the problem of estimation using this model fairly tractable.

Figure 20 shows some of the scaling deformations defined for the nose. Each arrow indicates a particular scaling parameter (in the vertical or horizontal direction), that affects the space between the enclosing lines. The results of applying some of the deformations to the nose are shown in Figure 21. Figure 21(a) and (d) show two views of the default nose. Figure 21(b) shows a nose deformed using vertical scaling, while the pulled-up nose in (c) is produced using a localized bending deformation. Figure 21(e) and (f) show localized scaling affecting the width of the nose in different places. Different faces produces using many deformations are shown on the right side of Figure 2.

Verification of the face parameterization produced by the model designer can be accomplished by fitting the

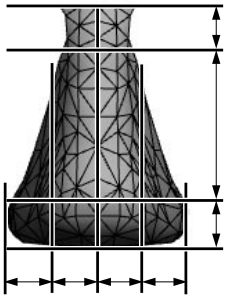
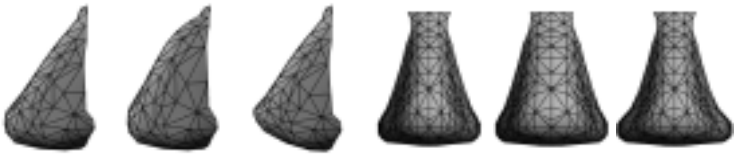


Figure 20: Scaling deformations of the nose



(a) (b) (c) (d) (e) (f)

Figure 21: Example deformations affecting the nose

model to a series of randomly generated sets of facial measurements. This is effectively a Monte Carlo method of sampling the space of face measurements. The fitting is easily accomplished, given a set of measurements, using the anthropometric forces described in Appendix C. The model designer can alter the model parameterization when a particular set of face measurements cannot be satisfied by the model. We obtained a face model capable of representing a wide variety of faces after only a few design iterations.

B.2 Face motion

The deformations corresponding to motions (such as facial expressions) are modeled using the same techniques employed for face shape. However, there is no available motion data that corresponds to anthropometric data for shape (although technology for gathering such data is becoming available [20]). The motion deformations are applied to the face in rest position—after the shape deformations, as in (1). Examples of modeled expressions are displayed on the left side of Figure 2. The model is capable of frowning or raising each eyebrow (top-left, top-right), smiling (bottom-left) or opening the mouth (bottom-right). This results in a total of 6 expression parameters (2 brow frowning, 2 brow raising, 1 smiling, 1 mouth opening), each corresponding to a particular FACS action unit [13]. In addition to this are the six parameters for rigid head motion (translation and rotation), resulting in a total of 12 parameters in $\mathbf{q}_{\mathrm{m}}$. These deformations can be applied to any face (different $\mathbf{q}_{\mathrm{b}}$), such as those on the bottom of Figure 2.

The construction of expressions is simplified by decomposing each face motion into several component deformations. For example, the mouth opening deformation is decomposed into chin/cheek bending, lip scaling and lip translation. To facilitate tracking of these expressions by reducing the number of motion parameters, there is a single *control* parameter for each expression which uniquely determines all of its component parameters. Given a particular face motion which is constructed using a series of deformations with parameters b_i, the control pa-

rameter e determines the value b_i based on the formula:

$$b_i = s_i e \tag{46}$$

where s_i is the scaling parameter used to form the linear relationship between b_i and e. These scaling parameters are the expression-shape parameters included in $\mathbf{q}_{\mathrm{b}}$ (there are about 20 in total). For situations where these parameters are not estimated, these parameters are treated as constants, average values for which are determined by the designer during construction of the model.

The set of face motion parameters $\mathbf{q}_{\mathrm{m}}$ consists of the control parameters for each of the expressions (which are initially all zero), and the rigid translation and rotation specifying the head position. The parameters b_i are not estimated, but are determined directly by (46) using the estimated value of e. The motion deformations are collected together into the deformation $\mathbf{T}_{\mathrm{m}}$.

C Anthropometry

Anthropometry is the biological science of human body measurement. Anthropometric data is used in a variety of applications that require knowledge of the distribution of measurements across human populations. For example, in medicine, quantitative comparison of anthropometric data with patients' measurements before and after surgery furthers planning and assessment of plastic and reconstructive surgery [16]. This paper proposes a similar use of anthropometry, in the construction of a face model for computer vision.

In order to develop useful statistics from anthropometric measurements, the measurements are made in a strictly defined way [24]. Particular locations on a subject, called *landmark* points, are defined in terms of visible or palpable features. A series of measurements between these landmarks is then taken using carefully specified procedures and measuring instruments (such as calipers, levels and measuring tape). A canonical coordinate system for the head and face is also defined in terms of landmarks, and provides a set of axes from which some measurements are taken. As a result, repeated measurements of the same individual (taken a few days apart) are very reliable, and measurements of different individuals can be successfully compared.

Farkas [16] describes a widely used set of measurements for describing the human face. A large amount of anthropometric data using this system is available [15, 16]. The system uses a total of 47 landmark points to describe the face, and includes the following five types of facial measurements:

- the *shortest distance* between two landmarks (such as the separation of the eyes)
- the *axial distance* between two landmarks—the distance measured along an axis (such as the vertical height of the nose)
- the *tangential distance* between two landmarks—the distance measured along a prescribed path on the surface of the face (such as the arc length of the upper lip boundary)
- the *angle of inclination* between two landmarks with respect to an axis (such as the slope of the nose bridge)
- the *angle between locations* (such as the angle formed at the tip of the nose)

Farkas describes a total of 132 measurements on the face and head.

Systematic collection of anthropometric measurements has made possible a variety of statistical investigations of groups of subjects. Subjects have been grouped on the basis of gender, race and age. Means and variances for the measurements within a group, tabulated in [16], effectively provide a set of measurements which captures virtually all of the variation that can occur within the group.

In addition to statistics on measurements, statistics on the *proportions* between measurements have also been used. Anthropometrists have found that proportions give useful information about the correlations between features, and can serve as more reliable indicators of group membership than can simple measurements [15]. These proportions are averaged over a particular population group, and means and variances are provided in [15].

C.1 Use of anthropometry data

Our face model includes representation of the anthropometric measurements described above. Given the measurement descriptions in [16], they are realized using a straightforward set of geometric operations performed using the face model: given a value of $\mathbf{q}_\mathrm{b}$, a set of measurements can be taken from the model.

Use of this data limits the coverage of a hand-crafted model to the space of faces made likely by a distribution of anthropometric measurements. Forces arising from this data are comparable to internal stiffness forces used in other deformable model work [48]. In that work, stiffness forces were used to determine a smooth surface in situations where the data was sparse or noisy. Here, anthropometric forces maintain a believable face shape, avoiding the parameter combinations that result in unlikely or impossible faces.

For a particular set of model points $\mathbf{x}_1 \ldots \mathbf{x}_n$, a measurement M_j is written as:

$$M_j(\mathbf{x}_1, \ldots, \mathbf{x}_n) \quad j \in 1..M \tag{47}$$

where M is the number of measurements in Farkas' inventory. As an example, a *shortest distance* measurement is simply the following:

$$M_{\mathrm{dist}}(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\| \tag{48}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are model points corresponding to the two landmarks used by the measurement. Note that these points depend on the shape parameters $\mathbf{q}_\mathrm{b}$, but not on the motion parameters $\mathbf{q}_\mathrm{m}$ (which is effectively zeroed when any anthropometric measurements are taken on the model—since this reflects the same "expressionless" conditions under which the data was originally gathered).

The statistical characterization of measurements and proportions can be built into the model in two ways. First, by using an average set of measurements, a set of parameters specifying the initial model is determined. This initial model is an anthropometrically "average" model, and is shown in Figure 1. Second, this characterization provides a means of biasing the face model shape parameters ($\mathbf{q}_\mathrm{b}$) toward more likely occurring individuals.

Given a particular set of population groups, average measurement values and variances are obtained from [16]

as:

$$(\mu_j, \sigma_j^2) \quad j \in 1..M \tag{49}$$

The biasing of the parameters is performed using three-dimensional spring-like forces (a soft constraint) that are applied to the polygonal face model that softly enforce a measurement on the model. First, an energy is associated with each measurement:

$$E_j = \frac{1}{2}\left(M_j(\mathbf{x}_1, \ldots, \mathbf{x}_n) - \mu_j\right)^2 \tag{50}$$

Then, the force resulting from the energy E_j, which is applied to model domain point $\mathbf{u}_i$ (which corresponds to the point $\mathbf{x}_i$ on the model surface), is obtained as:

$$\mathbf{f}_{E_j}(\mathbf{u}_i) = -\frac{\partial E_j}{\partial \mathbf{x}_i} = -\left(M_j(\mathbf{x}_1, \ldots, \mathbf{x}_n) - \mu_j\right)\frac{\partial M_j}{\partial \mathbf{x}_i} \tag{51}$$

The total anthropometric force applied to model domain point $\mathbf{u}_i$ is computed as the *weighted* sum of all measurement forces at $\mathbf{u}_i$:

$$\mathbf{f}_{\text{ant}}(\mathbf{u}_i) = \sum_{j \in 1..M} \left(1 - \frac{e^{-E_j/\sigma_j^2}}{\sqrt{2\pi\sigma}}\right)^{\rho} \mathbf{f}_{E_j}(\mathbf{u}_i) \tag{52}$$

Each force is weighted by a quantity which is a power (ρ) of how improbable the current measurement is (assuming a Gaussian distribution on the anthropometric measurements [16]). This weighting prevents the model from actually attaining the average set of measurements, but instead is simply biased towards them. For values of ρ around 10, forces on measurements within one standard deviation of the mean for that measurement are effectively ignored, making it used primarily as a prior on $\mathbf{q}_b$.

For proportions, the energy would involve two measurements as:

$$E_{jk} = \frac{1}{2}\left(M_j(\mathbf{x}_1, \ldots, \mathbf{x}_n) - p_{jk} \cdot M_k(\mathbf{x}'_1, \ldots, \mathbf{x}'_{n'})\right)^2 \tag{53}$$

where p_{jk} is the mean proportion between measurements μ_j and μ_k. As with the above for measurements, a force distribution for proportion data is obtained using this energy.

D Face feature and edge determination

The edge-based force field methods in [33, 48] require knowing which locations of the face model are likely to produce edges in an image. On the face, certain features are likely to produce edges in the image. The particular features chosen are the boundary of the lips and eyes, and the top boundary of the eyebrows. Edges in the polygon mesh which correspond to these features were manually marked during the model construction, and are shown in Figure 22(a).

(a) (b) (c)

Figure 22: Likely face features in an image

Other likely candidates for producing edges are the regions on the model of high curvature. The base of the nose and indentation on the chin are examples of high curvature edges, and can be seen in Figure 22(b). Occluding boundaries on the model also produce edges in the image, and can be determined using the three-dimensional model. The location of occlusion boundaries on the model will be useful when determining the quality of selected points for the measurement of optical flow.

Of course, for an edge to be produced in the image, the corresponding region on the face must be visible to the camera. This visibility determination is performed using the model and camera transformation. The model depth information can be used to determine the parts of the model that are visible to the camera (the frontmost regions of the model). Figure 22(b) shows visible locations of the model (features, high curvature and occluding edges) that are likely to produce edges, given the model in (c).

Once the locations on the model are known which are likely to produce image edges, we can weight the intensity gradient accordingly when forming the two-dimensional edge-based forces [33, 48] that are applied to the model. These forces contribute to the value of $\mathbf{f_q}$ (affecting parameters in both $\mathbf{q}_\mathrm{b}$ and $\mathbf{q}_\mathrm{m}$) based on (6). Over the course of fitting, these edge forces "pull" the model so that the model edges become aligned with their corresponding image edges.

E Optical flow point selection

The construction of the optical flow constraint on $\dot{\mathbf{q}}_\mathrm{m}$ required the selection of a set of image pixels from which to measure optical flow information. While it would be possible to use all pixels on the observed object, this would have two problems. Most obviously, it would be more expensive to solve the system—it is especially wasteful since it is likely that most pixels do not provide a significant amount of useful information. And second, particular points actually provide harmful information—such as those near occlusion boundaries. This section describes our method for the selection of pixels in the construction of (13).

Tomasi and Shi [43] define good features for tracking by using the following criterion. The outer product of the image gradients at pixel i is summed over a small window around that pixel:

$$\sum_{\mathrm{window}(i)} \nabla \mathrm{I}_i \, \nabla \mathrm{I}_i^\top \tag{54}$$

A feature is selected when the smaller eigenvalue of this 2×2 matrix is greater than a threshold value. These features possess significant image gradients in two orthogonal directions, which makes them reliable tracking features, as well as good sources of optical flow information. Features with one very large eigenvalue are also useful in our application, as these image points also provide good optical flow information.

However, not all pixels with significant gradient magnitude should be chosen. In particular, pixels on occlusion boundaries must be avoided, as they violate the optical flow constraint equation. The use of model-based techniques here provides a straightforward solution—assuming the model is at least roughly aligned with the image, pixels anywhere nearby the predicted occlusion boundaries of the model are simply not chosen. A detailed model, such as our face model, coupled with a method which computes occlusion boundaries (as in Appendix D), can be used to avoid these problems.

Traditional techniques for solving the optical flow constraint equation (10), often impose smoothness conditions on the flow field [22] to determine a solution. Smoothing is complicated by the fact that occlusion boundaries violate (10) and must be located to determine where to relax the smoothness conditions. The presence of a model entirely avoids the need for smoothing, as connectivity and discontinuity information is provided by the model. In addition to this, model-based optical flow techniques are more immune to the aperture problem [22], since information is combined over much larger image regions.

Besides providing the most accurate information possible, the set of chosen points must also adequately sample the motion information present in the image. The accurate measurement of a parameter in $\mathbf{q}_{\mathrm{m}}$ requires a sufficient number of pixels in the image corresponding to model points where the Jacobian of that parameter does not vanish. Note that some motion deformations may affect only a particular region of the face.

Using too few pixels in the computation results in a loss of accuracy, and can reach the point where the system loses track of the subject. Including too many pixels forces the pixel selection method to include pixels containing little useful information (such as having a small gradient magnitude). It has been determined by experimentation that 10 to 20 pixels per parameter provide sufficient accuracy and robustness for the application of face tracking (at which point the results change negligibly when more pixels are used). Since there can be considerable overlap between the sets of pixels used to measure each parameter, the total number of pixels used can be fairly small. For each of the experiments here, n is approximately 120 pixels.

References

[1] G. Adiv. Determining 3-d motion and structure from optical flow generated by several moving objects. *IEEE Pattern Analysis and Machine Intelligence*, 7(4):384–401, July 1985.

[2] N.J. Ayache and O.D. Faugeras. Building, registering, and fusing noisy visual maps. *IJRR*, 7(6):45–65, 1988.

[3] Y. Bar-Shalom and T. Fortmann. *Tracking and data association*. Academic Press, 1988.

[4] S. Basu, I. Essa, and A. Pentland. Motion regularization for model-based head tracking. In *Proceedings ICPR '96*, page C8A.3, 1996.

[5] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings ECCV '92*, pages 237–252, 1992.

[6] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proceedings ICCV '95*, pages 374–381, 1995.

[7] T.J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Pattern Analysis and Machine Intelligence*, 8(1):90–99, January 1986.

[8] A. Bryson and Y. Ho. *Applied Optimal Control*. Halsted Press, 1975.

[9] C. Choi, K. Aizawa, H. Harashima, and T. Takebe. Analysis and synthesis of facial image sequences in model-based image coding. *IEEE Circuits and Systems for Video Technology*, 4(3):257–275, 1994.

[10] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. In *ECCV98*, pages II:484–498, 1998.

[11] D. DeCarlo and D. Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proceedings CVPR '96*, pages 231–238, 1996.

[12] H.F. Durrant-Whyte. Consistent integration and propagation of disparate sensor observations. *IJRR*, 6(3):3–24, 1987.

[13] P. Ekman and W. Friesen. *The Facial Action Coding System*. Consulting Psychologist Press, Inc., 1978.

[14] I.A. Essa and A.P. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Pattern Analysis and Machine Intelligence*, 19(7):757–763, July 1997.

[15] L. Farkas. *Anthropometric Facial Proportions in Medicine.* Thomas Books, 1987.

[16] L. Farkas. *Anthropometry of the Head and Face.* Raven Press, 1994.

[17] P. Fua and C. Brechbuhler. Imposing hard constraints on deformable models through optimization in orthogonal subspaces. *Computer Vision and Image Understanding*, 65(2):148–162, February 1997.

[18] P. Fua and Y.G. Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–56, September 1995.

[19] A. Gelb. *Applied Optimal Estimation.* MIT Press, 1974.

[20] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Proceedings SIGGRAPH '98*, pages 55–66, July 1998.

[21] Y. Hel-Or and M. Werman. Constraint fusion for recognition and localization of articulated objects. *International Journal of Computer Vision*, 19(1):5–28, July 1996.

[22] B. Horn. *Robot Vision*. McGraw-Hill, 1986.

[23] B. Horn and E. Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76, June 1988.

[24] A. Hrdlicka. *Practical anthropometry*. AMS Press, 1972.

[25] R. Kaucic and A. Blake. Accurate, real-time, unadorned lip tracking. In *ICCV98*, pages 370–375, 1998.

[26] D.J. Kriegman, E. Triendl, and T.O. Binford. Stereo vision and navigation in buildings for mobile robots. *RA*, 5(6):792–803, December 1989.

[27] A. Lanitis, C.J. Taylor, and T.F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Pattern Analysis and Machine Intelligence*, 19(7):743–756, July 1997.

[28] H. Li, P. Roivainen, and R. Forchheimer. 3-D motion estimation in model-based facial image coding. *IEEE Pattern Analysis and Machine Intelligence*, 15(6):545–555, June 1993.

[29] D.G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[30] P. Maybeck. *Stochastic Models, Estimation and Control, Volume 1*. Academic Press, 1979.

[31] P. Maybeck. *Stochastic Models, Estimation and Control, Volume 2*. Academic Press, 1982.

[32] D. Metaxas. *Physics-Based Deformable Models : Applications to Computer Vision, Graphics, and Medical Imaging*. Kluwer Academic Publishers, 1996.

[33] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Pattern Analysis and Machine Intelligence*, 15(6):580–591, June 1993.

[34] Y. Moses, D. Reynard, and A. Blake. Robust real time tracking and classificiation of facial expressions. In *Proceedings ICCV '95*, pages 296–301, 1995.

[35] S. Negahdaripour and B. Horn. Direct passive navigation. *IEEE Pattern Analysis and Machine Intelligence*, 9(1):168–176, January 1987.

[36] A. Netravali and J. Salz. Algorithms for estimation of three-dimensional motion. *AT&T Technical Journal*, 64:335–346, 1985.

[37] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Pattern Analysis and Machine Intelligence*, 13(7):730–742, July 1991.

[38] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Pattern Analysis and Machine Intelligence*, 13(7):715–729, 1991.

[39] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[40] D. Reynard, A. Wildenberg, A. Blake, and J. Marchant. Learning dynamics of complex motions from image sequences. In *Proceedings ECCV '96*, pages I:357–368, 1996.

[41] A. Shabana. *Dynamics of Multibody Systems*. Wiley, 1989.

[42] R. Sharma, Y. Azoz, and L. Devi. Reliable tracking of human arm dynamics by multiple cue integration and constraint fusion. In *Proceedings CVPR '98*, 1998.

[43] J. Shi and C. Tomasi. Good features to track. In *Proceedings CVPR '94*, pages 593–600, 1994.

[44] E. Simoncelli, E. Adelson, and D. Heeger. Probability distributions of optical flow. In *Proceedings CVPR '91*, pages 310–315, 1991.

[45] G. Strang. *Linear algebra and its applications*. Harcourt, Brace, Jovanovich, 1988.

[46] D. Terzopoulos. Physically-based fusion of visual data over space, time and scale. In J. Aggarwal, editor, *Multisensor Fusion for Computer Vision*, pages 63–69. Springer-Verlag, 1993.

[47] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Pattern Analysis and Machine Intelligence*, 15(6):569–579, 1993.

[48] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, 1988.

[49] Y. Yacoob and L.S. Davis. Computing spatio-temporal representations of human faces. In *Proceedings CVPR '94*, pages 70–75, 1994.

[50] A.L. Yuille, D.S. Cohen, and P. Halliman. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8:104–109, 1992.

[51] G.H. Zhang and A. Wallace. Physical modeling and combination of range and intensity edge data. *Computer Vision, Graphics, and Image Processing*, 58(2):191–220, September 1993.