Tracking Motion, Deformation, and Texture Using Conditionally Gaussian Processes

Tim K. Marks, *Member*, *IEEE*, John R. Hershey, *Member*, *IEEE*, and Javier R. Movellan, *Member*, *IEEE*

Abstract—We present a generative model and inference algorithm for 3D nonrigid object tracking. The model, which we call G*-flow*, enables the joint inference of 3D position, orientation, and nonrigid deformations, as well as object texture and background texture. Optimal inference under G-flow reduces to a conditionally Gaussian stochastic filtering problem. The optimal solution to this problem reveals a new space of computer vision algorithms, of which classic approaches such as optic flow and template matching are special cases that are optimal only under special circumstances. We evaluate G-flow on the problem of tracking facial expressions and head motion in 3D from single-camera video. Previously, the lack of realistic video data with ground truth nonrigid position information has hampered the rigorous evaluation of nonrigid tracking. We introduce a practical method of obtaining such ground truth data and present a new face video data set that was created using this technique. Results on this data set show that G-flow is much more robust and accurate than current deterministic optic-flow-based approaches.

Index Terms—Computer vision, generative models, motion, shape, texture, video analysis, face tracking.

1 Introduction

A central and challenging problem in computer vision is tracking the motion of a deformable object in three dimensions using a video sequence from a single camera. For human-computer interfaces, a nonrigid object of primary interest is the human face. The motion of the face, consisting of both rigid motion (changes in the position and orientation of the head) and nonrigid motion (facial expressions and other deformations), carries information about emotions, identity, and speech.

It is well known how 3D objects project to form 2D images in a camera, and we have an intuitive understanding of how a person's face can move in three dimensions. If we express this knowledge formally, we can use it to perform 3D nonrigid face tracking by incorporating it into a *generative model*. This generative model approach begins with a forward model of how the state of the face—its pose (rigid position and orientation as well as nonrigid deformation) and texture (appearance)—changes over time, and how this 3D face generates a 2D image. Given an observed image, the task is to infer those hidden state values using the forward model. In other methods such as [1], which seek to directly estimate the pose of the face from image features without a generative model of images, it is difficult to incorporate our prior knowledge.

- T.K. Marks is with Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139. E-mail: tmarks@merl.com.
- J.R. Hershey is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: jrhershe@us.ibm.com.
- J.R. Movellan is with the Machine Perception Laboratory, University of California, San Diego, Atkinson Hall (CALIT2), 2400A, 9500 Gilman Dr., Mail Code 0445, La Jolla, CA 92093-0445. E-mail: movellan@mplab.ucsd.edu.

Manuscript received 14 Mar. 2008; revised 2 Oct. 2008; accepted 6 Oct. 2008; published online 11 Nov. 2008.
Recommended for acceptance by F. Dellaert.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-03-0153.
Digital Object Identifier no. 10.1109/TPAMI.2008.278.

Generative models are frequently used in the machine learning community and have recently been applied successfully in computer vision (e.g., [2], [3], [4], [5], [6]). One advantage of generative approaches is that they force us to make explicit the structure of the problem of interest and enable us to derive optimal solutions. In this way, they help elucidate the advantages and limitations of existing algorithms. For example, the generative model proposed here reveals a new space of computer vision algorithms, of which classic approaches such as optic flow and template matching are special cases, optimal only under very special circumstances.

1.1 Overview of Existing Systems for Nonrigid 3D Face Tracking

Existing face tracking models differ in how they model the *appearance*, or *texture*, of the surfaces of the object. For example, Torresani and Hertzmann [7] and Xiao et al. [8] use an appearance model that is constant across all time. We refer to these as *template-based* appearance models. In contrast, the appearance models of [9], [10], [11] do not remain constant over time, but are completely changed as each new image is presented. In these systems, each new observed frame of video is compared with an appearance model that is based entirely on the previous frame. We call these *flow-based* appearance models. One can think of flow-based approaches as using appearance models which, rather than remaining constant over time, are reset at each time step based upon the previous observed image.

There is an inherent trade-off between template-based and flow-based approaches. The advantage of flow-based approaches is that they make few assumptions about the appearance of the object. Flow-based approaches are robust to gradual changes in appearance because the appearance model used at a given time comes only from the previous image. However, optic flow's appearance model is only as good as its alignment in the previous frame. As a result, alignment error tends to build over time, which can cause the system to completely lose track of the object. This is not a problem for template approaches because the alignment in the previous frame has no effect on the appearance model. However, template-based approaches require good knowledge of the appearance of the object and are unable to adapt when this appearance changes over time, for example, due to changes in lighting or facial expression. Template-based and flow-based appearance models can be considered as the two ends of a continuum. In the middle of the continuum would be appearance models that change slowly over time, gradually incorporating information from newly presented images into the existing appearance model.

While previous models tend to start with optic flow or template matching as an image preprocessing primitive, here we pursue a different approach. First, we formulate a generative model for video sequences, which we call *G-flow*, and then we derive the optimal solution to the tracking problem under the model. The solution reveals a space of possible computer vision algorithms that includes optic flow and template matching as special cases, which are optimal only in specific conditions.

From this point of view, many previous 3D tracking algorithms can be seen as approximations to optimal tracking. For example, the algorithms in [8], [9], [10], [11] each choose one extreme of the flow-to-template continuum and commit to a single estimate of pose and appearance at each time step. In contrast, here we maintain a probability distribution over the possible poses and textures.

Approaching the tracking problem from an optimality point of view reveals some important mathematical properties of the problem. For example, using a reasonable model of image formation, video generation is a conditionally Gaussian process [12], [13], [14], [15] given the pose process: The conditional distribution of the texture given the pose is Gaussian. This enables us to use a Rao-Blackwellized particle filter [13], [15], in which the filtering problem is partitioned into two components: a particle-based nonlinear component for pose and a linear Gaussian component for texture given a sequence of pose estimates. Our algorithm also takes advantage of the fact that the peak and covariance of the filtering distribution for pose can be estimated efficiently. We first described the essence of this approach in [16], [17].

1.2 Collecting Video with Locations of Unmarked Smooth Features

Although, there are now a number of 3D nonrigid tracking algorithms, measuring their effectiveness has been difficult. The problem lies in the lack of video data of real moving human faces (or other deformable objects) in which there are no visible marks on the face and yet for which the actual 3D positions of the features being tracked are known.

Here, we present a new method for collecting the true 3D positions of points on smooth facial features (such as points on the skin) without leaving a visible trace in the video. This technique, described in Section 5 and Appendix II, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278, utilizes infrared markings to accurately measure the 3D positions of points on the face with infrared-sensitive cameras, without the markings showing up in the video taken by visible light cameras. We present the *IR Marks* video data set, a new face motion data set that was created using this technique. This provides an accurate, objective method for testing and comparing the performance of nonrigid tracking systems. Section 6 presents our tracking results on this data set, which we use both to compare our system with other tracking systems and to measure the effects that changing parameter values have on our system's performance.

1.3 Notation

Throughout this paper, we use standard probabilistic notation: Uppercase letters represent random variables and random vectors and lowercase letters represent the values taken by these variables, no matter whether the values are scalars, vectors, or matrices. We represent sequences using a subscripted colon, for instance, $u_{1:t} \overset{\text{def}}{=} \{u_1, u_2, \ldots, u_t\}$. We use I_m to represent the $m \times m$ identity matrix.

2 The Generative Model for G-Flow

The task of recovering the 3D structure of a moving, deforming object from sequences of 2D images is a very challenging problem, partly due to the fact that a given 2D image may be consistent with a large number of 3D explanations. The approach we pursue here takes advantage of the fact that it is well understood how a known 3D world generates 2D images (in a camera, for example). This allows us to formulate the much harder problem of going from 2D images to 3D structure as Bayesian inference, enabling us to study the mathematical structure of the problem.

In this section, we lay out the forward model: how a 3D deformable object generates a video (a sequence of 2D images). Then, in Section 3, we describe how to use Bayesian inference to tackle the inverse problem: determining the pose (nonrigid and rigid motion) of the 3D object from an image sequence.

Our generative model incorporates the assumptions that objects occlude backgrounds and that object and background texture are statistically independent. It is assumed that, at the time of object tracking, the system has had sufficient experience to have good estimates of the following: models of the 3D geometry of the object and its possible deformations, pose dynamics, observation noise (the amount of uncertainty when rendering a pixel from a given texture value), and texture process noise (how quickly each *texel*, or texture element, of the foreground and background appearance models varies over time). The following values are left as unknowns and inferred in a causal manner by the tracking algorithm: object texture (gray-scale appearance), background texture, rigid pose (3D orientation and translation), and nonrigid pose (object deformation, such as facial expression).[1]

2.1 Modeling 3D Deformable Objects

We represent object structure using a 3D Morphable Model (3DMM) [18]. A nonrigid object's structure is specified by the 3D locations of n points, which we refer to as *vertices*. To model nonrigid deformations (e.g., facial expressions), the

1. Note that throughout this paper, we use the term *pose* to include not only the rigid 3D pose of an object, but also its nonrigid deformations.

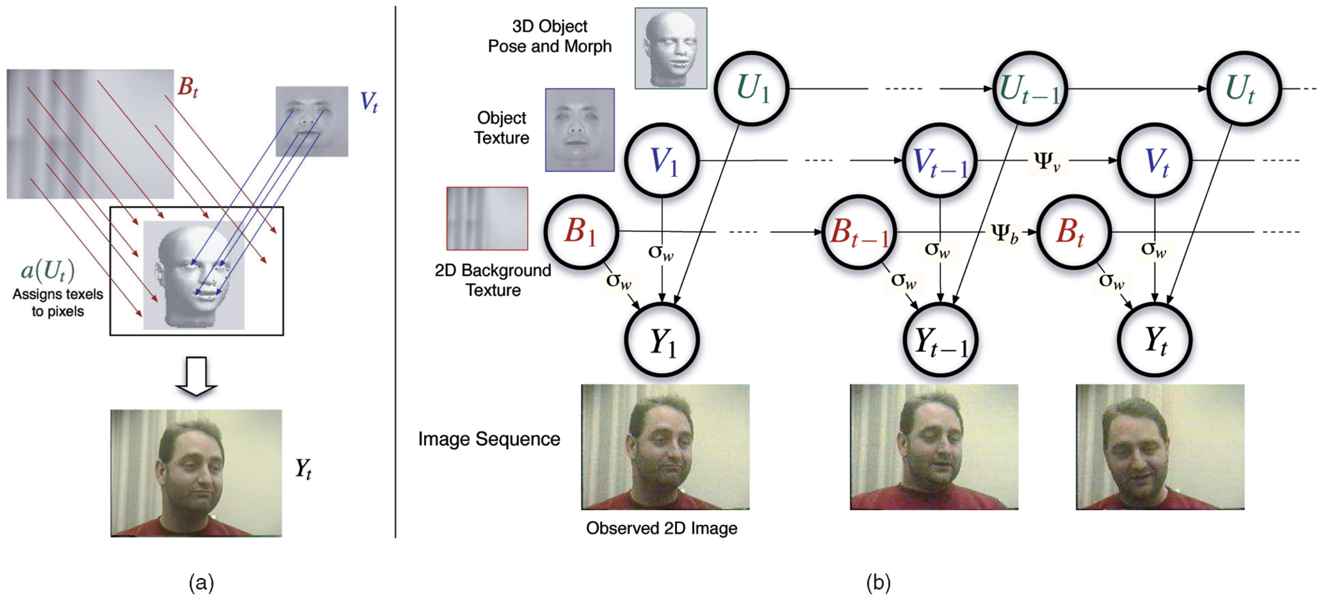

Fig. 1. (a) Each pixel in image Y_t is rendered using the color or gray-scale value of either an object texel (a pixel in the object texture map V_t) or a background texel (a pixel in the background texture map B_t). The projection function $a(U_t)$, where U_t is the pose at time t, determines which texel is responsible for rendering each image pixel. (b) G-flow video generation model. At time t, the object's 3D pose U_t is used to project the object texture V_t into the 2D image. This projection is placed in front of the background texture B_t to generate the observed image Y_t. The goal is to make inferences about the hidden variables (U_t, V_t, B_t) based on the observed video sequence $\{Y_1, \ldots, Y_t\}$.

locations of the vertices are constrained to be a linear combination of k fixed 3D basis shapes, which we call *morph bases*, with coefficients $c = [c_1, c_2, \ldots, c_k]^T$. This linear combination may then undergo rigid motion (rotation and translation) in 3D. Finally, a projection model (such as weak perspective projection) is used to map the 3D vertex locations onto 2D coordinates in the image plane.

Let the fixed $3 \times k$ matrix h_i contain the position of the ith vertex in all k morph bases. Thus, in 3D object-centered coordinates, the location of the ith vertex is $h_i c$. Scale changes are accomplished by multiplying all k morph coefficients by the same scalar. In practice, the first morph basis is often the mean shape, and the other $k-1$ morph bases are deformations of that base shape (e.g., the results of applying principal component analysis (PCA) to the 3D locations of the vertices in several key frames). In this case, the first morph coefficient c_1 can be used as a measure of scale. The transformation from object-centered coordinates to image coordinates consists of a rotation, weak perspective projection, and translation. Thus, x_i, the 2D location of the ith vertex on the image plane, is $x_i = grh_ic + l$, where r is a 3×3 rotation matrix, l is a 2×1 translation vector, and

$$g = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

is the projection matrix.

The object *pose* at time t, denoted by u_t, comprises the parameters of both rigid motion (rotation r and translation l) and nonrigid motion (morph coefficients c): $u_t = \{r_t, l_t, c_t\}$. To avoid clutter, we usually omit the subscript t from the components $r, l,$ and c of the pose parameter.

2.2 Modeling Video Sequences

We model the image sequence Y as a stochastic process generated by three hidden causes, U, V, and B, as shown in the graphical model (Fig. 1b). The $m \times 1$ random vector Y_t represents the m-pixel image at time t. The $n \times 1$ random vector V_t represents the n-texel object texture,[2] and the $m \times 1$ random vector B_t represents the m-texel background texture. As illustrated in Fig. 1a, the object pose U_t determines which object and background texels render which image pixels at time t.

For our derivations, we assume that the object has n vertices, one corresponding to each texel in v_t, the object texture map: The ith texel in v_t corresponds to the ith vertex of the object. We further assume that the mapping (which is a function of the pose, u_t) from object texels (in v_t) to image locations (in y_t) is one-to-one, i.e., that every object texel renders exactly one pixel in the image (unless an object texel is self-occluded, in which case it does not render any image pixels). Of the m pixels in each observed image, at most n of the pixels are rendered by the object (one pixel per unoccluded texel in v_t), and the remaining image pixels are rendered by the background. In practice, we actually use fewer vertices than pixels; each pixel of the object is rendered by interpolating the location in the object texture map from the surrounding vertex positions. The simplifying assumption that the rendering process is a one-to-one discrete function from vertices/texels to pixels is a convenient fiction, in that it simplifies the derivations without affecting the underlying mathematics.

The ith texel in the object texture map renders the pixel at $x_i(u_t)$, which is the image location to which vertex i projects. From Section 2.1, this image location is given by:

2. We use the term *texel* (short for *tex*ture *el*ement) to refer to each individual element in a texture map, just as pixel (*pic*ture *el*ement) refers to an individual element of an image.

$x_i(u_t) = grh_ic + l$. To avoid clutter, we let the dependency of r, c, and l on u_t be implicit in our notation. Moreover, we index pixels in the image vector y_t by their 2D image locations. Thus, $y_t\big(x_i(u_t)\big)$ represents the gray-scale value of the image pixel that is rendered by object vertex i. In our image generation model, this is equal to the gray-scale value of texel i in the object texture map plus Gaussian observation noise:

$$\textbf{Foreground pixel } \boldsymbol{x_i(u_t)}\text{:}\quad Y_t\big(x_i(u_t)\big) = V_t(i) + W_t\big(x_i(u_t)\big), \quad W_t\big(x_i(u_t)\big) \sim N\big(0, \sigma_w^2\big), \tag{1}$$

where the random variables $W_t(j)$ represent the independent, identically distributed observation noise (rendering noise) at every image pixel $j \in \{1, \ldots, m\}$. This model for texture is reasonable because it models the changes in pixel values due to sensor noise, which is Gaussian to first approximation.

The background texture map b_t is the same size as the image y_t. Every image pixel j that is not rendered by a foreground texel is rendered by background texel j. The gray-scale value of each pixel rendered by the background is equal to the gray-scale value of the corresponding texel in the background texture map plus Gaussian observation noise:

$$\textbf{Background pixel } \boldsymbol{j}\text{:}\quad Y_t(j) = B_t(j) + W_t(j), \quad W_t(j) \sim N\big(0, \sigma_w^2\big). \tag{2}$$

To combine the scalar equations (1) and (2) into a single vector equation, we introduce the projection function $a(U_t)$, which is determined by the imaging model (e.g., weak perspective projection). For a given pose u_t, the projection $a(u_t)$ is a block matrix, $a(u_t) \overset{\text{def}}{=} [a_v(u_t) \;\; a_b(u_t)]$. Here, $a_v(u_t)$, the object projection function, is an $m \times n$ matrix of 0s and 1s that tells onto which image pixel each object vertex projects, e.g., a 1 at row j, column i means that the ith object vertex projects onto image pixel j. Matrix $a_b(u_t)$, an $m \times m$ diagonal matrix of 0s and 1s, plays the same role for background pixels. Assuming that the foreground mapping is one-to-one (each texel projects to exactly one pixel) it follows that $a_b(u_t) = I_m - a_v(u_t)[a_v(u_t)]^T$, expressing the occlusion constraint that every image pixel is rendered by object or background, but not both. Then, we can express the observation model given in (1) and (2) by a single matrix equation:

$$Y_t = a(U_t)\begin{bmatrix} V_t \\ B_t \end{bmatrix} + W_t, \quad W_t \sim N\big(\mathbf{0}, \sigma_w^2 I_m\big), \tag{3}$$

where $\sigma_w > 0$ is a scalar.

The system dynamics describe how the foreground and background texture maps change over time. For $t \geq 2$, the system dynamics are given by

$$\begin{aligned} V_t &= V_{t-1} + Z_{t-1}^v, \quad \text{with} \quad Z_{t-1}^v \sim N(\mathbf{0}, \Psi_v), \\ B_t &= B_{t-1} + Z_{t-1}^b, \quad \text{with} \quad Z_{t-1}^b \sim N(\mathbf{0}, \Psi_b), \end{aligned} \tag{4}$$

where the covariance matrices Ψ_v and Ψ_b are diagonal, and Z^v, Z^b, W are independent over time, independent of each other, and independent of the initial conditions. The n values on the diagonal of Ψ_v represent the variance of the process noise for each of the n texels of the object texture map. Similarly, the m values on the diagonal of Ψ_b represent the variance of the process noise for the m background texels. We let the pose U_t be a Markov process with a known pose transition distribution: $p(u_t \mid u_{1:t-1}, y_{1:t-1}) = p(u_t \mid u_{t-1})$. The form of the pose distribution is left unspecified since our algorithm does not require the pose distribution or the pose dynamics to be Gaussian.

2.3 Model Parameters

The initial conditions consist of a distribution for object pose U_1 and Gaussian distributions of object and background texture, V_1 and B_1, all of which are independent. The covariance matrix of V_1 is diagonal and the covariance of B_1 is a scalar times the identity matrix. The other model parameters are: diagonal covariance matrices for the state transitions (process noise), Ψ_v and Ψ_b; the pose transition distribution, $p(u_t \mid u_{t-1})$; and the variance of the image rendering noise at each pixel, σ_w^2. Using standard methods, the above parameters could be either learned from training data, or inferred dynamically during tracking to adjust to the observed data. For the results described in this paper, we simply chose generic parameter values that work well. For example, we use a weak model for the pose transition distribution $p(u_t \mid u_{t-1})$ that simply constrains object pose to not change too much in adjacent frames and penalizes facial expressions whose morph coefficients exceed the range observed in the training data.

3 Inference in G-Flow

Nonrigid 3D tracking is a difficult nonlinear filtering problem, both because the image location of each vertex is a nonlinear function of the pose, i.e., $a(u_t)$ depends nonlinearly on u_t, and because the image intensity is a nonlinear function of image location, i.e., $y_t(x)$ is a nonlinear function of x. Fortunately, the problem has a rich structure that can be exploited: It is reasonable to approximate video generation as a conditionally Gaussian process [12], [13], [14], [15], as we do in the G-flow model: If the specific values taken by the pose sequence $u_{1:t}$ were known, then the texture processes V and B and the image process Y would be jointly Gaussian, with a time-varying parameter $a(u_t)$ determined by the known pose u_t. Hence, given the poses $u_{1:t}$, we could use a Kalman filter [19] to optimally infer the object and background texture maps from the observed images over time. Essentially, the model makes the reasonable assumption that if the pose and appearance of the object were known, then the only source of variability left in the rendering process would be Gaussian sensor noise.

3.1 Rao-Blackwellized Particle Filtering

Stochastic filtering is the process of sequentially estimating the posterior distributions of some target variables given a sequence of observations. In our case, the observable sequence is the collection of pixel values, and the target variables are the pose (rigid motion and nonrigid deformation) of the object and the texture (appearance) of the object and the background.

Particle filtering is a popular approach to obtain approximate solutions to nonlinear stochastic filtering problems. Unfortunately, standard particle filtering alone would be insufficient to solve the 3D nonrigid tracking problem because of the very high dimensionality of the state space (the joint distribution over pose and texture). Fortunately, the conditionally Gaussian structure of the task makes the problem tractable: We can use a particle filter to estimate the pose distribution, then use an analytical solution to obtain the texture appearance given the pose distribution. The use of a Monte Carlo method on the analytically intractable components, while integrating out the tractable components, is known in the particle filtering literature as *Rao-Blackwellization* [13], [15]. This approach is also known in the statistics community as a Monte Carlo filtering solution for conditionally Gaussian processes [12], [14]. In a Rao-Blackwellized particle filter, the portion of the system's state that must be sampled is of lower dimension than the full model. This reduces the sampling variance, and hence, fewer samples (fewer particles) are required in order to achieve the same level of performance.

3.2 Overview of Inference in G-Flow

Our goal is to find an expression for the *filtering distribution* $p(u_t, v_t, b_t \mid y_{1:t})$, i.e., the posterior distribution of pose and texture given the observed sequence of images. In addition, we want the filtering distribution to be updated in a recursive manner—knowing the filtering distribution at time $t-1$ and given a new image y_t at time t, we want to find the filtering distribution at time t, without requiring the past images to be stored. The following equation provides the key to doing so. Using the law of total probability, we can derive:

$$\overbrace{p(u_t, v_t, b_t \mid y_{1:t})}^{\substack{\text{Filtering} \\ \text{Distribution}}} = \int \underbrace{p(u_{1:t-1} \mid y_{1:t})}_{\substack{\text{Credibility} \\ \text{of expert}}} \underbrace{p(u_t \mid u_{1:t-1}, y_{1:t})}_{\substack{\text{Pose Opinion} \\ \text{of expert}}} \underbrace{p(v_t, b_t \mid u_{1:t}, y_{1:t})}_{\substack{\text{Texture Opinion} \\ \text{of expert (Distribution} \\ \text{of texture given pose)}}} du_{1:t-1}. \quad (5)$$

We can think of the integral in (5) as a weighted sum over a distribution of particles. Each particle represents a pose sequence $u_{1:t-1}$. Given its pose sequence, each particle can be thought of as an *expert* that is endowed with: 1) a *pose opinion*, i.e, a predictive distribution for the current pose, $p(u_t \mid u_{1:t-1}, y_{1:t})$; 2) a *texture opinion*, i.e., a posterior distribution for the current object and background textures, $p(v_t,\ b_t \mid u_{1:t},\ y_{1:t})$; and 3) a scalar *credibility* value, $p(u_{1:t-1} \mid y_{1:t})$. Because the image generation model is conditionally Gaussian given the pose, we can perform exact inference using Kalman filtering to determine the foreground and background texture distributions for each particle.

We refer to each particle as an *expert* and use the terms *pose opinion*, *texture opinion*, and *credibility* as shorthand to describe how G-flow factorizes the filtering distribution (5). These terms are just mnemonic handles for referring to the components of this probability distribution. Table 1 summarizes how we compute each term in the integral. We cover pose opinion (predictive pose distribution) in Section 3.4, texture opinion (distribution of texture given pose) in Section 3.3, and credibility in Section 3.5.

3.3 Expert Texture Opinions

Because the image generation process is conditionally Gaussian, the distribution of texture given a pose sequence $u_{1:t}$ and an image sequence $y_{1:t}$ is Gaussian. The mean of this texture opinion, which can be thought of as an expert's texture template, consists of a single texture value for each texel. The covariance matrix of the texture opinion, which can be shown to be diagonal, represents the expert's uncertainty about the texture values at each texel. The problem of updating the texture mean and uncertainty maps given a sequence of poses is equivalent to solving a time-dependent Kalman filtering problem. The expected values and covariance matrices of the predictive texture maps (the texel means and variances) are denoted as follows:

TABLE 1
Overview of Inference in G-Flow

Pose opinion (Section 3.4)	→	Efficient Monte Carlo sampling. We use importance sampling, starting with a Gaussian proposal distribution obtained by Laplace's method under the assumption of a white-noise background.
Texture opinion (Section 3.3)	→	(Object and background dynamic texture maps for each expert.) Analytical solution given the estimated pose opinion distribution.
Credibility (Section 3.5)	→	Analytical solution given the estimated pose opinion distribution.

To infer the filtering distribution at time t, we evaluate the integral in (5) as a sum over particles (a.k.a. experts). For each expert, we compute the three terms in the integral using the method indicated.

predictive texture maps from time $t-1$

$$\begin{aligned} \hat{v}_{t|t-1}(i) &\overset{\text{def}}{=} i^{th} \text{ element of } E(V_t \mid u_{1:t-1}, y_{1:t-1}), \\ \hat{b}_{t|t-1}(j) &\overset{\text{def}}{=} j^{th} \text{ element of } E(B_t \mid u_{1:t-1}, y_{1:t-1}), \end{aligned} \quad (6)$$

$$\begin{aligned} \mathcal{V}_{t|t-1}(i) &\overset{\text{def}}{=} i^{th}\text{diagonal elem. of } Cov(V_t \mid u_{1:t-1}, y_{1:t-1}), \\ \mathcal{B}_{t|t-1}(j) &\overset{\text{def}}{=} j^{th}\text{diagonal elem. of } Cov(B_t \mid u_{1:t-1}, y_{1:t-1}). \end{aligned} \quad (7)$$

We refer similarly to the process noise for each texel:

$$\begin{aligned} \Psi_v(i) &\overset{\text{def}}{=} i^{th}\text{diagonal element of } \Psi_v, \\ \Psi_b(j) &\overset{\text{def}}{=} j^{th}\text{diagonal element of } \Psi_b. \end{aligned} \quad (8)$$

Just as the subscript $t|t-1$ above refers to texture estimates at time t conditioned on the poses and images up to time $t-1$, we will use the subscript $t \mid t$ to refer to the estimate at time t conditioned on the pose and image history up to time t. For example,

$$\hat{v}_{t|t}(i) \overset{\text{def}}{=} i^{th}\text{element of } E(V_t \mid u_{1:t}, y_{1:t}). \quad (9)$$

3.3.1 *Kalman Equations for Dynamic Update of Texel Maps*

Assume that by time t, the system already has the means and variances of the predictive texture maps from the previous time step: $\{\hat{v}_{t|t-1}, \hat{b}_{t|t-1}, \mathcal{V}_{t|t-1}, \mathcal{B}_{t|t-1}\}$. By applying the Kalman filtering equations, we obtain the predictive texture maps at time t. The Kalman gain for texel i is the scalar quantity

TABLE 2
Summary of the G-Flow Inference Algorithm

Inference in G-flow utilizes a Rao-Blackwellized particle filter. We refer to each particle as an *expert*. Before time step t, each expert has a single hypothesis about $u_{1:t-1}$, the object pose at all previous time steps. Given this pose history and the new image, y_t, we estimate the expert's *pose opinion* distribution, which is the predictive probability distribution over the expert's pose at time t. Each particle's new pose u_t is sampled from this distribution.

At each time step (each frame of video), the procedure is to first obtain the filtering distribution for pose (Step II), then find the distribution of texture given those pose experts (Step III).

I. Initialization ($t = 1$). Each expert's pose, $\{u_1^{(d)} : d = 1, \ldots, \eta\}$, is sampled from $p(u_1)$. At $t = 1$, every expert has the same predictive texture distributions for object and background:

$$\hat{v}_{1|0}^{(d)} = \hat{v}_{1|0}, \quad \mathcal{V}_{1|0}^{(d)} = \mathcal{V}_{1|0}, \quad \hat{b}_{1|0}^{(d)} = \hat{b}_{1|0}, \quad \mathcal{B}_{1|0}^{(d)} = \mathcal{B}_{1|0}.$$

Observe image y_1. Use (17) to calculate each expert's weight at $t = 1$:

$$w_1^{(d)} \propto p(y_1 \mid u_1^{(d)}), \qquad \text{with} \qquad \sum_{d=1}^{\eta} w_1^{(d)} = 1.$$

II. Filtering distribution for pose
The filtering distribution at time t is estimated as a weighted sum of the poses of η experts:

$$p(u_t \mid y_{1:t}) \approx \sum_{d=1}^{\eta} w_t^{(d)} \delta(u_t - u_t^{(d)}).$$

We resample experts periodically, calling those video frames (those time steps) in which we resample "resampling frames," and calling the other time steps "continuation frames." After observing image y_t, each expert's pose $u_t^{(d)}$ and weight $w_t^{(d)}$ are found using A or B below.

A. Resampling frames

1. Use Gauss-Newton to estimate the peak of each expert's pose opinion distribution, $\hat{u}_t(u_{1:t-1}^{(d)})$. (See Section 3.4.1 and Appendix VII-A.)
2. Use Laplace's method to estimate the covariance of the expert's pose opinion distribution, $\mathcal{U}\big(\hat{u}_t(u_{1:t-1}^{(d)})\big)$. Generate λ independent samples $u_t^{(d,e)}$, where $e \in \{1, \ldots, \lambda\}$, from the Gaussian proposal distribution:
$$\pi(u_t \mid u_{1:t-1}^{(d)}, y_{1:t}) \overset{\text{def}}{=} N\Big(\hat{u}_t(u_{1:t-1}^{(d)}),\ \alpha\,\mathcal{U}\big(\hat{u}_t(u_{1:t-1}^{(d)})\big)\Big).$$
(See Section 3.4.1 and Appendix VII-B.)
3. For each sample $u_t^{(d,e)}$, use (17) to calculate the value of the predictive distribution, $p(y_t \mid u_{1:t-1}^{(d)}, u_t^{(d,e)}, y_{1:t-1})$. Then compute the sample's importance weight:
$$\xi_t(d,e) = \frac{p(u_t^{(d,e)} \mid u_{t-1}^{(d)})\, p(y_t \mid u_{1:t-1}^{(d)}, u_t^{(d,e)}, y_{1:t-1})}{\pi(u_t^{(d,e)} \mid u_{1:t-1}^{(d)}, y_{1:t})}.$$
4. Estimate the expert's credibility, $\tilde{w}_{t-1|t}^{(d)} \approx p(u_{1:t-1}^{(d)} \mid y_{1:t})$:
$$\tilde{w}_{t-1|t}^{(d)} \propto w_{t-1}^{(d)} \sum_{e=1}^{\lambda} \xi_t(d,e), \quad \text{with} \quad \sum_{d=1}^{\eta} \tilde{w}_{t-1|t}^{(d)} = 1.$$
5. Generate ν next-generation pose experts, $\{u_{1:t}^{(j)} : j = 1, \ldots, \nu\}$, as follows:
   - **a.** Randomly select a parent expert, d, who will sire the child. The probability of selecting expert d is $\tilde{w}_{t-1|t}^{(d)}$.
   - **b.** Randomly select the child's pose $u_t^{(d,e)}$ from the parent's pose opinion (the samples $\{u_t^{(d,1)}, u_t^{(d,2)}, \ldots\}$), with probability proportional to $\xi_t(d,e)$.
   - **c.** The new child (the next-generation expert) is:
   $$u_{1:t}^{(j)} = \{u_{1:t-1}^{(d)}\,,\, u_t^{(d,e)}\}.$$
6. Give every new pose expert equal weight:
$$w_t^{(j)} = \tfrac{1}{\nu}\,.$$

B. Continuation frames
Follow Steps 1–5 from A (above), with $\lambda = 1, \alpha = 0, \nu = 1$. Each pose expert d has exactly one child, also labeled d. The new expert's weight equals its credibility:

$$w_t^{(d)} = \tilde{w}_{t-1|t}^{(d)}\,.$$

III. Distribution of texture given pose
For each new expert d, the Gaussian predictive distribution of texture given the pose history, $u_{1:t}^{(d)}$, is obtained from the previous frame's predictive texture distribution using the Kalman equations for each texel:

object texel i:

$$\hat{v}_{t+1|t}^{(d)}(i) = \mathcal{K}_t^{(d)}(i)\, y_t\big(x_i(u_t^{(d)})\big) + \big[1 - \mathcal{K}_t^{(d)}(i)\big]\, \hat{v}_{t|t-1}^{(d)}(i)$$
$$\mathcal{V}_{t+1|t}^{(d)}(i) = \big[1 - \mathcal{K}_t^{(d)}(i)\big]\, \mathcal{V}_{t|t-1}^{(d)}(i) + \Psi_v(i)$$
$$\text{where} \quad \mathcal{K}_t^{(d)}(i) = \begin{cases} \dfrac{\mathcal{V}_{t|t-1}^{(d)}(i)}{\mathcal{V}_{t|t-1}^{(d)}(i) + \sigma_w^2}, & \text{if texel } i \text{ is visible} \\ 0, & \text{if texel } i \text{ is occluded.} \end{cases}$$

background texel j:

$$\hat{b}_{t+1|t}^{(d)}(j) = \mathcal{K}_t^{(d)}(j)\, y_t(j) + \big[1 - \mathcal{K}_t^{(d)}(j)\big]\, \hat{b}_{t|t-1}^{(d)}(j)$$
$$\mathcal{B}_{t+1|t}^{(d)}(j) = \big[1 - \mathcal{K}_t^{(d)}(j)\big]\, \mathcal{B}_{t|t-1}^{(d)}(j) + \Psi_b(j)$$
$$\text{where} \quad \mathcal{K}_t^{(d)}(j) = \begin{cases} \dfrac{\mathcal{B}_{t|t-1}^{(d)}(j)}{\mathcal{B}_{t|t-1}^{(d)}(j) + \sigma_w^2}, & \text{if texel } j \text{ is visible} \\ 0, & \text{if texel } j \text{ is occluded.} \end{cases}$$

$$\mathcal{K}_t(i) \overset{\text{def}}{=} \begin{cases} \dfrac{\mathcal{V}_{t|t-1}(i)}{\mathcal{V}_{t|t-1}(i) + \sigma_w^2}, & \text{if texel } i \text{ is visible,} \\ 0, & \text{if texel } i \text{ is occluded.} \end{cases} \tag{10}$$

The predictive distribution for the foreground texture map is

$$\hat{v}_{t+1|t}(i) = \mathcal{K}_t(i)\ y_t(x_i(u_t)) + [1 - \mathcal{K}_t(i)]\ \hat{v}_{t|t-1}(i), \tag{11}$$

$$\mathcal{V}_{t+1|t}(i) = [1 - \mathcal{K}_t(i)]\ \mathcal{V}_{t|t-1}(i) + \Psi_v(i). \tag{12}$$

The update equations for the background texture map are identical, but with the object texture means and variances replaced by the background texture means and variances (see Table 2).

3.3.2 Interpreting the Kalman Equations

The updated texture map is a combination of the previous texture map and the new image, and the Kalman gain dynamically determines the relative contributions of each. It follows from (10) that

$$\mathcal{V}_{t|t-1}(i) \gg \sigma_w^2 \Longrightarrow \mathcal{K}_t(i) \approx 1, \quad \mathcal{V}_{t|t-1}(i) \ll \sigma_w^2 \Longrightarrow \mathcal{K}_t(i) \approx 0. \tag{13}$$

In other words, if the uncertainty (variance) of a texel in the object texture map is much larger than the image rendering noise, then the Kalman gain will be close to 1, and thus, the information from the texture map will be ignored in favor of the (more accurate) information from the current image.

On the other hand, if a texel's variance is much less than the image rendering noise, then the Kalman gain will be close to 0, and thus, the information from the current image will be ignored in favor of the (more accurate) information from the texture map.

3.4 Expert Pose Opinions

Our goal is to estimate the *pose opinion* distribution,

$$p(u_t \mid u_{1:t-1}, y_{1:t}), \quad \leftarrow \text{Pose opinion distribution}, \tag{14}$$

which is the belief of expert $u_{1:t-1}$ about the pose at time t. Since the effect of the pose u_t on the observed image y_t is nonlinear, exact analytical solutions are not available. However, the spatiotemporal smoothness of video signals can be exploited to obtain efficient estimates: First approximate the pose opinion of each expert as a Gaussian distribution, then use importance sampling to correct for the potential bias introduced by this approximation.

3.4.1 Gaussian Approximation of Each Expert's Pose Opinion

Laplace's method approximates an arbitrary probability density function (pdf) by a Gaussian function that is centered at the peak of the distribution and has covariance matrix proportional to the inverse of the Hessian matrix at the peak. This is equivalent to approximating the logarithm of the pdf with a second-order Taylor series approximation centered about the peak of the distribution. We apply Laplace's method to obtain a Gaussian approximation to the pose opinion distribution of each expert. We then use this Gaussian approximation as our proposal distribution for importance sampling. The method requires: 1) finding the peak of the distribution and 2) estimating the spread of the pose opinion (the Hessian matrix of the pdf) at the peak.

Finding the peak of an expert's pose opinion. We can write the pose opinion distribution of expert $u_{1:t-1}$ as:

$$\begin{aligned} p(u_t \mid u_{1:t-1}, y_{1:t}) &= \frac{p(u_t \mid u_{1:t-1}) p(y_{1:t} \mid u_{1:t})}{p(y_{1:t} \mid u_{1:t-1})} \\ &= \frac{p(y_{1:t-1} \mid u_{1:t-1})}{p(y_{1:t} \mid u_{1:t-1})} p(u_t \mid u_{t-1}) p(y_t \mid u_{1:t}, y_{1:t-1}). \end{aligned} \tag{15}$$

To find the peak of this distribution, we need to maximize (15). Eliminating terms that do not depend on u_t, we have

$$\begin{aligned} \hat{u}_t(u_{1:t-1}) &\overset{\text{def}}{=} \arg\max_{u_t} p(u_t \mid u_{1:t-1}, y_{1:t}) \\ &= \arg\max_{u_t} p(u_t \mid u_{t-1})\, p(y_t \mid u_{1:t}, y_{1:t-1}). \end{aligned} \tag{16}$$

The following expression for the final term in (16), the *predictive distribution* $p(y_t \mid u_{1:t}, y_{1:t-1})$, is derived (99) in Appendix VI, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278:

$$\begin{aligned} \log p(y_t \mid u_{1:t}, y_{1:t-1}) &= -\frac{m}{2} \log 2\pi \\ &- \frac{1}{2}\sum_{i=1}^{n}\left[\log\left(\mathcal{V}_{t|t-1}(i) + \sigma_w^2\right) + \frac{[y_t(x_i(u_t)) - \hat{v}_{t|t-1}(i)]^2}{\mathcal{V}_{t|t-1}(i) + \sigma_w^2}\right] \\ &- \frac{1}{2}\sum_{j \notin \mathcal{X}(u_t)}\left[\log\left(\mathcal{B}_{t|t-1}(j) + \sigma_w^2\right) + \frac{[y_t(j) - \hat{b}_{t|t-1}(j)]^2}{\mathcal{B}_{t|t-1}(j) + \sigma_w^2}\right], \end{aligned} \tag{17}$$

where $x_i(u_t)$ is the image pixel rendered by the ith object vertex when the object has pose u_t, and $\mathcal{X}(u_t)$ is the set of all image pixels rendered by the object in pose u_t. In (17), the first sum corresponds to the foreground pixels and the second sum corresponds to the background pixels.

We can now convert these sums over both foreground and background pixels into a sum over only foreground pixels. We substitute (17) into (16), eliminating terms that do not depend on u_t (see Appendix VI, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278, for details):

$$\begin{aligned} \hat{u}_t(u_{1:t-1}) = \arg\min_{u_t} \Bigg(&- \log p(u_t \mid u_{t-1}) \\ &+ \frac{1}{2}\sum_{i=1}^{n}\Bigg[\overbrace{\frac{[y_t(x_i(u_t)) - \hat{v}_{t|t-1}(i)]^2}{\mathcal{V}_{t|t-1}(i) + \sigma_w^2} + \log[\mathcal{V}_{t|t-1}(x_i(u_t)) + \sigma_w^2]}^{\text{Foreground terms}} \\ &\underbrace{- \frac{[y_t(x_i(u_t)) - \hat{b}_{t|t-1}(x_i(u_t))]^2}{\mathcal{B}_{t|t-1}(x_i(u_t)) + \sigma_w^2} - \log[\mathcal{B}_{t|t-1}(x_i(u_t)) + \sigma_w^2]}_{\text{Background terms}} \Bigg]\Bigg). \end{aligned} \tag{18}$$

The sum in (18) is difficult to optimize analytically in the general case. However, an efficient approximation can be obtained under the following assumptions: 1) If there are no self-occlusions (i.e., the object texels are never occluded), then the second foreground term $\log[\mathcal{V}_{t|t-1}(x_i(u_t)) + \sigma_w^2]$ is constant with respect to pose, and thus, may be ignored; 2) if the background terms are ignored, which essentially corresponds to a white noise background model; and 3) if the transition probability is Gaussian, then the prior term $\log p(u_t \mid u_{t-1})$ becomes a quadratic penalty term. Under these conditions, the optimization reduces to a nonlinear least-squares problem, to which we apply the Gauss-Newton algorithm. We explain the details of this method for estimating the peak of the pose opinion in Appendix VII-A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278. Computer vision algorithms such as optic flow and template matching are special cases of this optimization, as we will show in Section 4. Whereas previous algorithms use the result of this optimization as their final pose estimate, in our system, it is merely the first step in approximating the posterior distribution over pose.

Hereafter, we index each individual expert by d, where $d \in \{1, 2, \ldots, \eta\}$. Then, $u_{1:t-1}^{(d)}$ refers to the pose history of expert d, and $\hat{u}_t(u_{1:t-1}^{(d)})$ refers to our estimate of the peak of that expert's pose opinion, obtained by using Gauss-Newton to minimize (18).

Estimating the spread of the expert's pose opinion. The Laplace estimate of the covariance matrix of the pose opinion of expert d is the inverse of the Hessian matrix of (18) with respect to u_t, evaluated at the peak $\hat{u}_t(u_{1:t-1}^{(d)})$. This can be obtained analytically under the same assumptions listed above for obtaining the peak. (A full derivation is in Appendix VII-B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.) We call this estimated covariance matrix of the pose opinion $\mathcal{U}(\hat{u}_t(u_{1:t-1}^{(d)}))$.

Our initial estimate of the pose opinion of expert d is a Gaussian distribution whose mean is $\hat{u}_t(u_{1:t-1}^{(d)})$ and whose covariance is proportional to the Laplace estimate $\mathcal{U}(\hat{u}_t(u_{1:t-1}^{(d)}))$:

$$\pi\big(u_t \mid u_{1:t-1}^{(d)}, y_{1:t}\big) \overset{\text{def}}{=} N\Big(\hat{u}_t\big(u_{1:t-1}^{(d)}\big),\ \alpha\, \mathcal{U}\big(\hat{u}_t\big(u_{1:t-1}^{(d)}\big)\big)\Big). \quad (19)$$

The parameter $\alpha > 0$ controls the width of the distribution. (Note that letting $\alpha \to 0$ is simply equivalent to setting the new pose u_t equal to the peak of the estimated pose opinion distribution.)

3.4.2 Importance Sampling Correction of the Gaussian Approximation

Importance sampling [20] is a method for improving the accuracy of Monte Carlo estimates, by using a *proposal distribution* to focus the samples on significant regions, then weighting the samples to compensate for not sampling from the true distribution. The basic method is outlined in Appendix I, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.

Here, we use importance sampling to improve our initial Gaussian estimate (19) of the pose opinion of each expert. For each expert $u_{1:t-1}^{(d)}$, we generate a set of λ independent samples $\{u_t^{(d,1)}, \ldots, u_t^{(d,\lambda)}\}$ from the proposal distribution $\pi(u_t \mid u_{1:t-1}^{(d)}, y_{1:t})$. By (15), the pose opinion that we wish to estimate $p(u_t \mid u_{1:t-1}^{(d)}, y_{1:t})$ is proportional to the joint distribution over the current pose and image given the pose and image history:

$$p\big(y_t, u_t^{(d)} \mid u_{1:t-1}^{(d)}, y_{1:t-1}\big) = p\big(u_t^{(d,e)} \mid u_{t-1}^{(d)}\big) p\big(y_t \mid u_{1:t-1}^{(d)}, u_t^{(d,e)}, y_{1:t-1}\big). \quad (20)$$

The first term on the right-hand side is the pose transition probability, which is a parameter of the image generation model, and the second term is the predictive probability that is expressed in (17). We compute the unnormalized importance weights,

$$\xi_t(d,e) = \frac{p\big(u_t^{(d,e)} \mid u_{t-1}^{(d)}\big)\, p\big(y_t \mid u_{1:t-1}^{(d)}, u_t^{(d,e)}, y_{1:t-1}\big)}{\pi\big(u_t^{(d,e)} \mid u_{1:t-1}^{(d)}, y_{1:t}\big)}, \quad (21)$$

then normalize these importance weights so that they sum to 1: $\tilde{\xi}_t(d,e) = \xi_t(d,e) / \sum_{f=1}^{\lambda} \xi_t(d,f)$. The pose opinion distribution is thus estimated as a set of λ weighted samples:

$$\hat{p}\big(u_t \mid u_{1:t-1}^{(d)}, y_{1:t}\big) = \sum_{e=1}^{\lambda} \delta\big(u_t - u_t^{(d,e)}\big)\, \tilde{\xi}_t(d,e), \quad (22)$$

where δ is the Dirac delta function.

3.5 Expert Credibility

Each expert d has its own precise belief about the poses from time 1 to time $t-1$, denoted as $u_{1:t-1}^{(d)}$. The *credibility* of this expert, which we denote as $\tilde{w}_{t-1|t}^{(d)}$, is an estimate of $p(u_{1:t-1}^{(d)} \mid y_{1:t})$, which measures how well this expert's hypothesis about the poses from time 1 to time $t-1$ fits with the image observed at time t:

$$\tilde{w}_{t-1|t}^{(d)} \approx p\big(u_{1:t-1}^{(d)} \mid y_{1:t}\big). \quad (23)$$

At each time step t, we determine this credibility by updating the expert's weight from the previous time step, $w_{t-1}^{(d)}$, which is an estimate of $p(u_{1:t-1}^{(d)} \mid y_{1:t-1})$:

$$w_{t-1}^{(d)} \approx p\big(u_{1:t-1}^{(d)} \mid y_{1:t-1}\big). \quad (24)$$

The credibility $\tilde{w}_{t-1|t}^{(d)}$ is then used to compute the expert's weight at the current time step, $w_t^{(d)}$, which is an estimate of $p(u_{1:t}^{(d)} \mid y_{1:t})$. We now give the details of these updates.

After observing image y_t, find the credibility using Bayes rule:

$$\underbrace{p\big(u_{1:t-1}^{(d)} \mid y_{1:t}\big)}_{\substack{\text{Credibility}\\ \text{of expert } d,\\ \tilde{w}_{t-1|t}^{(d)}}} \propto \underbrace{p\big(u_{1:t-1}^{(d)} \mid y_{1:t-1}\big)}_{\substack{\text{Prior weight}\\ \text{of expert } d,\\ w_{t-1}^{(d)}}} p\big(y_t \mid u_{1:t-1}^{(d)}, y_{1:t-1}\big). \quad (25)$$

We have already generated a set of λ pose samples $\{u_t^{(d,e)} : e = 1, \ldots, \lambda\}$ to estimate the pose opinion of the dth expert. We can now use these same samples to estimate the final term in (25). Note that

$$p\big(y_t \mid u_{1:t-1}^{(d)}, y_{1:t-1}\big) = \int \frac{p\big(y_t, u_t \mid u_{1:t-1}^{(d)}, y_{1:t-1}\big)}{\pi\big(u_t \mid u_{1:t-1}^{(d)}, y_{1:t}\big)} \pi\big(u_t \mid u_{1:t-1}^{(d)}, y_{1:t}\big) du_t. \quad (26)$$

Since the samples $u_t^{(d,e)}$ were drawn from the proposal distribution $\pi(u_t \mid u_{1:t-1}^{(d)}, y_{1:t})$, we can use them to estimate this integral:

$$\begin{aligned} p\big(y_t \mid u_{1:t-1}^{(d)}, y_{1:t-1}\big) &\approx \frac{1}{\lambda} \sum_{e=1}^{\lambda} \frac{p\big(y_t, u_t^{(d,e)} \mid u_{1:t-1}^{(d)}, y_{1:t-1}\big)}{\pi\big(u_t^{(d,e)} \mid u_{1:t-1}^{(d)}, y_{1:t}\big)} \\ &= \frac{1}{\lambda} \sum_{e=1}^{\lambda} \xi_t(d,e), \end{aligned} \quad (27)$$

where the last step follows from (21). Substituting (27) into (25), we obtain a consistent estimate for the credibility of expert d:

$$\tilde{w}_{t-1|t}^{(d)} \approx p\big(u_{1:t-1}^{(d)} \mid y_{1:t}\big) \propto w_{t-1}^{(d)} \sum_{e=1}^{\lambda} \xi_t(d,e), \quad (28)$$

where we normalize so that the credibilities of all experts sum to 1: $\sum_{d=1}^{\eta} \tilde{w}_{t-1|t}^{(d)} = 1$.

Let $u_{1:t}^{(d,e)}$ represent the concatenation of the pose history of expert d with pose sample e of the same expert: $u_{1:t}^{(d,e)} \overset{\text{def}}{=} \{u_{1:t-1}^{(d)}, u_t^{(d,e)}\}$. Note that

$$p\big(u_{1:t}^{(d,e)} \mid y_{1:t}\big) = p\big(u_{1:t-1}^{(d)} \mid y_{1:t}\big)\, p\big(u_t^{(d,e)} \mid u_{1:t-1}^{(d)}, y_{1:t}\big). \quad (29)$$

Thus, from (23) and (22), a consistent estimator of the expert's weight at time t is

$$p\big(u_{1:t}^{(d,e)} \mid y_{1:t}\big) \approx \tilde{w}_{t-1|t}^{(d)}\, \tilde{\xi}_t(d,e). \tag{30}$$

3.6 Estimating the New Filtering Distribution

If there are η experts, $u_{1:t-1}^{(d)}$, where $d \in \{1,\ldots,\eta\}$, and each of these experts has λ pose samples $u_t^{(d,e)}$, where $e \in \{1,\ldots,\lambda\}$, then there are a total of $\eta\lambda$ potential next-generation pose experts, which we call seeds. The ν next-generation experts are obtained by sampling ν times, with replacement, from the set of all $\eta\lambda$ seeds, where, on each draw, the probability of choosing seed $u_{1:t}^{(d,e)}$ is given by our estimate of $p(u_{1:t}^{(d,e)} \mid y_{1:t})$ from (30). After this resampling step, known in the literature as a *selection step* [21], the filtering distribution is represented by ν equally weighted pose experts $u_{1:t}^{(d)}$, where $d \in \{1,\ldots,\nu\}$, and $w_t^{(d)} = \frac{1}{\nu}$.

The weighted collection of next-generation experts is a consistent estimator of the filtering distribution for pose at time t:

$$p(u_t \mid y_{1:t}) = \int p(u_{1:t} \mid y_{1:t}) d(u_{1:t-1}) \approx \sum_{d=1}^{\nu} \delta\big(u_t - u_t^{(d)}\big) w_t^{(d)}, \tag{31}$$

where δ is the Dirac delta function. For each new expert $u_{1:t}^{(d)}$, the texture opinion is the Gaussian described in Section 3.3. Thus, our estimator of the filtering distribution (the joint distribution of pose, object texture, and background texture, given the images up to time t) is

$$\begin{aligned} p(u_t, v_t, b_t \mid y_{1:t}) &= \int p(u_{1:t}, v_t, b_t \mid y_{1:t}) d(u_{1:t-1}) \\ &\approx \sum_d \delta\big(u_t - u_t^{(d)}\big)\, w_t^{(d)} \mathcal{N}(v_t; \hat{v}_{t|t}, \mathcal{V}_{t|t}) \mathcal{N}(b_t; \hat{b}_{t|t}, \mathcal{B}_{t|t}). \end{aligned} \tag{32}$$

As is common in particle filters, we do not resample at every time step [22]. Rather, we resample periodically. We call video frames at which we resample "resampling frames," and call the others "continuation frames." Table 2 summarizes the entire G-flow inference algorithm that was presented in Section 3.

4 Relation to Optic Flow and Template Matching

In template matching, the same time-invariant texture map is used to track every frame of video. Optic flow can be seen as template matching in which the template is completely reset at each frame for use in the next frame. Optimal inference in G-flow involves a combination of optic flow-based and template-based tracking, in which the texture template gradually evolves as new images are presented. Pure optic flow and template matching both emerge as special cases.

4.1 Steady-State Texel Variances

For unoccluded texels, the Kalman filter asymptotically approaches a steady state. From (12) and (10), the steady-state variance $\mathcal{V}_\infty(i)$ and Kalman gain $\mathcal{K}_\infty(i)$ of object texel i satisfy the following two equations:

$$\mathcal{V}_\infty(i) = \big[1 - \mathcal{K}_\infty(i)\big]\, \mathcal{V}_\infty(i) + \Psi_v(i), \quad \mathcal{K}_\infty(i) = \frac{\mathcal{V}_\infty(i)}{\mathcal{V}_\infty(i) + \sigma_w^2}. \tag{33}$$

Combining these two equations yields the algebraic Ricatti equation, which we can solve to obtain the steady-state texel variance $\mathcal{V}_\infty(i)$ and Kalman gain $\mathcal{K}_\infty(i)$:

$$\mathcal{V}_\infty(i) = \frac{\Psi_v(i) + \sqrt{\Psi_v^2(i) + 4\sigma_w^2 \Psi_v(i)}}{2}, \tag{34}$$

$$\mathcal{K}_\infty(i) = \frac{-\Psi_v(i) + \sqrt{\Psi_v^2(i) + 4\sigma_w^2 \Psi_v(i)}}{2\sigma_w^2}. \tag{35}$$

A useful descriptive parameter is the variance of the predictive distribution for the intensity of the image pixel rendered by texel i: $\tau_t(i) \overset{\text{def}}{=} \mathcal{V}_{t|t-1}(i) + \sigma_w^2$. We call $\tau_t(i)$ the *temperature* of object texel i at time t in analogy to statistical mechanics; it measures the amount of noise in the texture value of each image pixel. Note that $\tau_t(i)$ is the denominator of the first foreground term in the G-flow objective function (18). We define the *steady-state temperature* of texel i by $\tau_\infty(i) = \mathcal{V}_\infty(i) + \sigma_w^2$.

4.2 Optic Flow as a Special Case

Here, we show that optic flow, a classic computer vision algorithm, can be seen as an approximate solution to the G-flow inference equation (18). In the process, we see that optic flow is optimal only under very special circumstances. Suppose that the pose transition probability $p(u_t \mid u_{t-1})$ is uninformative and that the background is uninformative. Then the only terms remaining to minimize in (18) are the two foreground terms. If we further suppose that there are no self-occlusions (no object texel is ever occluded), then it follows from (10) and (12) that the second foreground term $\log[\mathcal{V}_{t|t-1}(x_i(u_t)) + \sigma_w^2]$ is constant with respect to pose and thus may be ignored. In this case, the only term remaining to minimize in the G-flow objective function (18) is the first foreground term:

$$\hat{u}_t(u_{1:t-1}) = \arg\min_{u_t} \frac{1}{2} \sum_{i=1}^{n} \frac{\big[y_t(x_i(u_t)) - \hat{v}_{t|t-1}(i)\big]^2}{\mathcal{V}_{t|t-1}(i) + \sigma_w^2}. \tag{36}$$

Suppose we further restrict our model so that the process noise is the same at every texel and is much larger than the image rendering noise:

$$\Psi_v = \sigma_v^2 I_n, \quad \text{where} \quad \sigma_v^2 \gg \sigma_w^2. \tag{37}$$

Once the system has settled to its steady state, it follows from (34) that every texel will have the same variance $\mathcal{V}_\infty \approx \sigma_v^2$, and thus, by (33), every texel's Kalman gain will be $\mathcal{K}_\infty(i) \approx 1$. As a result, the equation for the texture map means (11) reduces to: $\hat{v}_{t|t-1}(i) = y_{t-1}\big(x_i(u_{t-1})\big)$, i.e., the mean of texel i of the object texture map is simply the texture value taken from the location of vertex i in the previous image. Hence, the G-flow objective function (36) further reduces to

$$\hat{u}_t(u_{1:t-1}) = \arg\min_{u_t} \frac{1}{2} \sum_{i=1}^{n} \big[y_t(x_i(u_t)) - y_{t-1}(x_i(u_{t-1}))\big]^2. \tag{38}$$

With this objective function, the Gauss-Newton minimization that we use to find the peak of the pose opinion (see Section 3.4.1 and Appendix VII-A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278) reduces to exactly the Gauss-Newton minimization performed in optic flow. Thus, constrained optic flow [9], [10], [11] is simply a special limiting case of optimal inference under G-flow, with a single expert ($\eta = 1$) and with sampling parameter $\alpha \to 0$. (For a full explanation of constrained optic flow, see Appendix V, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.)

4.2.1 Interpretation

Optic flow is only optimal in the extreme conditions given by (37), which essentially amount to the object's texture changing so fast that, at every point on the object (at every vertex), the texture value varies a great deal from each frame to the next. However, optic flow is typically applied in situations in which the object's texture changes very little from each frame to the next. In such situations, the optimal solution calls not for optic flow but for a texture map that integrates information across multiple frames.

4.3 Template Matching as a Special Case

Here, we show that template matching, another classic computer vision algorithm, can also be derived as a special solution to the G-flow inference problem. Like optic flow, template matching is optimal only under very special circumstances. Suppose, as in the optic flow case above, that the pose transition probability $p(u_t \mid u_{t-1})$ is uninformative, the background is uninformative, and no object texel is ever occluded. Then, the G-flow objective function (18) again reduces to (36). We further restrict our model so that the image rendering noise is much greater than the process noise at every object texel, i.e.,

$$\sigma_w^2 \gg \Psi_v(i) \quad \text{for all } i \in \{1, \dots, n\}. \tag{39}$$

By (35), the Kalman gain is $\mathcal{K}_\infty(i) \approx 0$, so by (11), the texture map means are constant over time. By denoting this fixed texture template as $\hat{v}_\infty(i) \overset{\text{def}}{=} \hat{v}_{t|t-1}(i)$, the objective function (36) further reduces to the error function that is minimized by template matching algorithms:

$$\hat{u}_t(u_{1:t-1}) = \arg\min_{u_t} \frac{1}{2} \sum_{i=1}^{n} [y_t(x_i(u_t)) - \hat{v}_\infty(i)]^2. \tag{40}$$

4.3.1 Interpretation

Template matching is an optimal solution only under the extreme conditions expressed by key assumption (39). This amounts to supposing that, in comparison to the image rendering noise, the object's texture value at every point (at every vertex) is basically constant from each frame to the next. This is rarely true in real data, and G-flow provides a principled way to relax this unrealistic assumption of template methods.

4.4 General Case

In general, if the background is uninformative, then minimizing the G-flow objective function (18) results in a weighted combination of optic flow and template matching, with the weight of each approach depending on the current level of uncertainty about (i.e., the current variance of) each texel in the object texture map.

Not only template matching and optic flow but also a whole new continuum of inference algorithms in between these two extremes are special cases of G-flow under the assumption of an uninformative background. When there is useful information in the background, G-flow can additionally infer a model of the background to improve tracking.

5 Empirical Evaluation

Although there are now a number of 3D nonrigid tracking algorithms [7], [8], [9], [10], [11], [23], [24], measuring their effectiveness with deformable objects has been difficult. Currently, evaluation of algorithms is primarily based on demonstration videos in which the system appears to do well in a qualitative sense, or on the system's tracking of toy data. Quantitative evaluations using real data are not possible due to the lack of video data sets of real moving human faces (or other deformable objects) in which there are no visible marks on the face, and yet for which the actual 3D positions of the features being tracked are known. To our knowledge, no publicly available data set yet exists that provides both the original video and simultaneous ground truth data during natural head and nonrigid face motion.

We developed a new data collection method, utilizing an infrared marking pen that is visible under infrared light but not under visible light (see Fig. 2). This involved setting up a rig of visible light cameras (to which the infrared marks were not visible) for collecting the test video, plus three infrared-sensitive (IR) cameras (to which the infrared marks were clearly visible), and calibrating all of the cameras both spatially and temporally. We collected video sequences simultaneously in all cameras and reconstructed the 3D ground truth information by hand-labeling several key frames from the IR cameras in each video sequence. We use this data set to rigorously test our system's performance and compare it to other systems. We are making this data set, called *IR Marks*, freely available to other researchers in the field with the publication of this paper. Fig. 2 shows a single frame of video from the *IR Marks* data set. For more details on the collection method and the data set, see Appendix II, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.

6 Results

In this section, we evaluate the G-flow inference algorithm (described in Section 3). In the experiments described here, we have assumed an uninformative background, and thus, have not implemented the background texture model. For these experiments, we simplified the G-flow objective function (18) by eliminating both background terms (essentially assuming a white noise background) and the second foreground term. The resulting model is still broad enough to encompass as special cases optic flow, template matching, and an entire continuum in between.

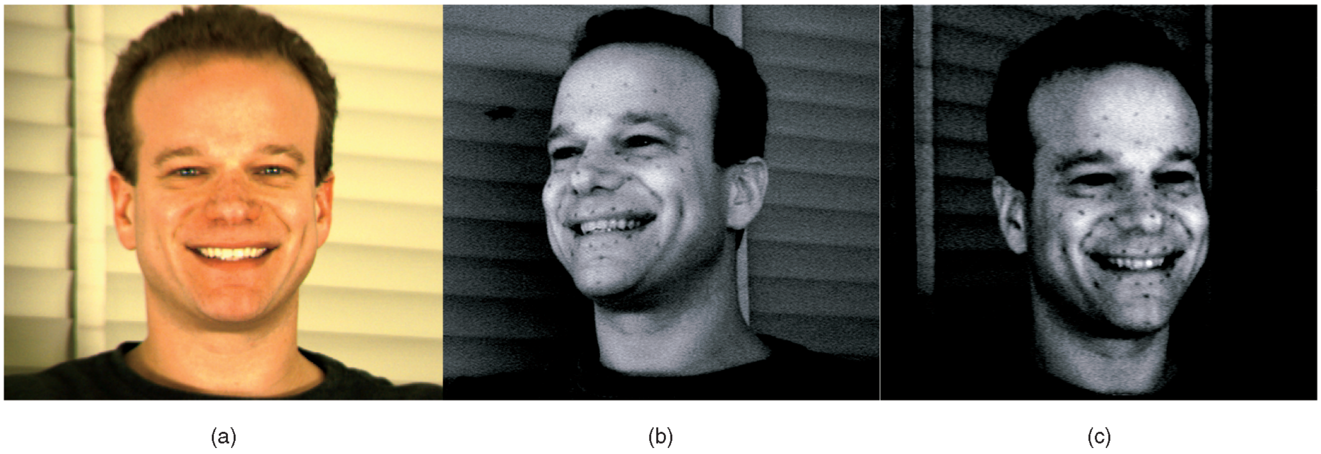

Fig. 2. A single frame of video from the *IR Marks* data set. Before video collection, the subject's face was marked using an infrared marking pen. The figure shows the same frame of video simultaneously captured by (a) one of the four visible light cameras and (b) and (c) two of the three infrared-sensitive cameras. The infrared marks are clearly visible using the infrared cameras, but they are not visible in the image from the visible light camera.

6.1 Comparison with Constrained Optic Flow: Varying the Number of Experts

Constrained optic flow [9], [10], [11] represents the filtering distribution for pose using a single point estimate, which results in a great deal of drift in difficult sequences. (For a full explanation of constrained optic flow, see Appendix V, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.) In contrast, G-flow uses multiple experts (particles), each of which has its own point estimate for pose, to represent an arbitrary filtering distribution for pose. To test the value of maintaining a full estimate of the pose distribution, we performed tracking runs using varying numbers of experts on the *Emote* video sequence (the most difficult of the three video sequences) of the IR Marks data set. The graph in Fig. 3a shows a plot of the average error (in pixels) of each vertex at several different time points in the sequence. For these results, we used a texture model of small circular patches around each vertex and a steady-state Kalman gain of $\mathcal{K}_\infty(i) \approx 1$ (the optic flow limit). In these conditions, the 1-expert case corresponds exactly to constrained optic flow [9], [10], [11]. Constrained optic flow (the line labeled "1 expert") completely loses track of the face. However, increasing the number of experts to 5 improves the tracking performance significantly, and using 10 or more experts results in dramatic improvement.

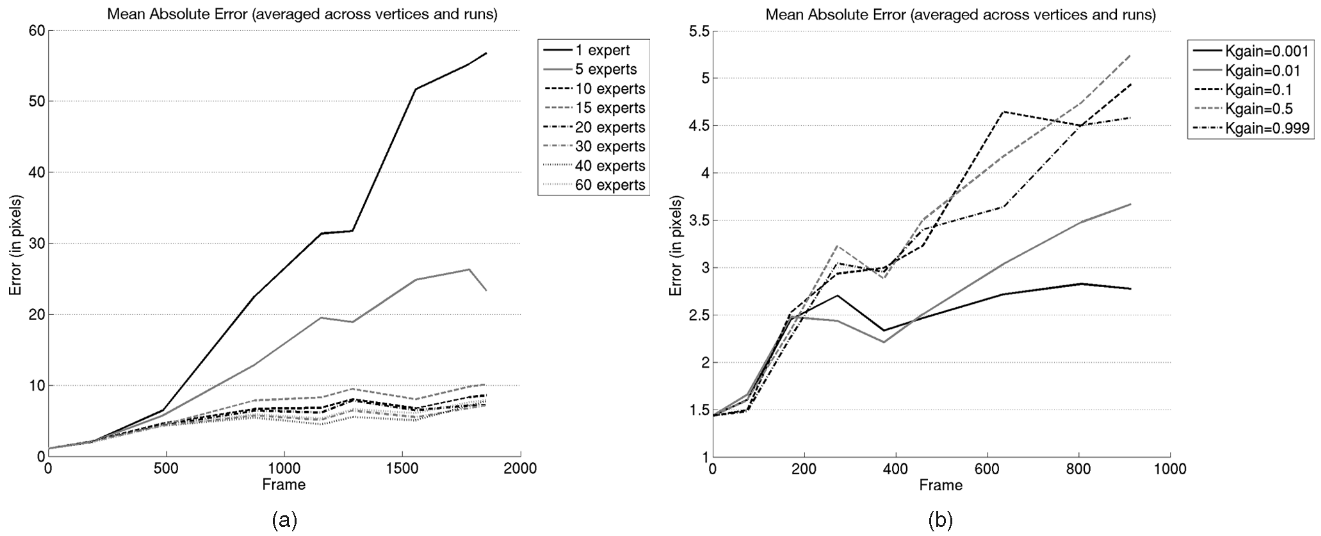

Fig. 3. Each line represents an average of several runs, showing the error of the vertex locations of the mean expert, averaged across vertices and runs. (a) The advantage of multiple experts. Constrained optic flow [9], [10], [11], labeled as "1 expert" in the graph, quickly loses track of a difficult video sequence (the *Emote* sequence of the IR Marks data set). Multiple experts provide a Monte Carlo estimate of an entire pose distribution, rather than collapsing the distribution to a single pose estimate. The graph shows that in the optic flow limit, $\mathcal{K}_\infty(i) \approx 1$, multiple experts greatly improve tracking performance. (b) Varying the Kalman gain. The results of tracking the *Talk1* sequence of the IR Marks data set using 20 experts. The steady-state Kalman gain was varied from $\mathcal{K}_\infty(i) = 0.001$ for every object texel (near the template matching limit) to $\mathcal{K}_\infty(i) = 0.999$ for every object texel (near the optic flow limit). The steady-state temperature $\tau_\infty(i)$ was kept constant across all values of Kalman gain in order to have the same overall level of noise in all runs. (The results of tracking the *Talk2* sequence are similar to those for *Talk1*; the *Talk2* results are demonstrated in a video available in the online supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.)

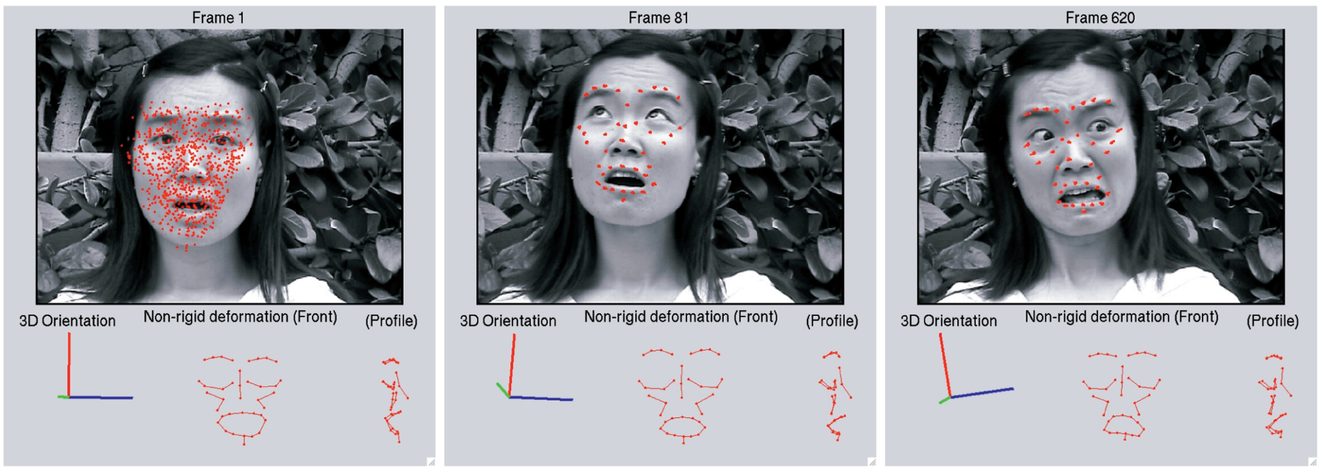

Fig. 4. G-flow tracking an outdoor video. Results are shown for frames 1, 81, and 620. In each frame, the algorithm simultaneously estimates the rigid pose of the head and the nonrigid face deformations. There are 20 experts, each of which has a hypothesis about the 3D locations of 38 points on the face. The set of all 20 experts is represented by $20 \times 38 = 760$ red dots superimposed on the figure at the top of each frame. Despite the large initial uncertainty about the pose in frame 1, G-flow determines the correct pose by frame 81 and maintains accurate tracking for the remainder of the sequence. (Video is available in the online supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.)

6.2 Initialization and Error Recovery

The results in Fig. 3a show that having multiple experts greatly reduces the tendency of flow-based systems to drift out of alignment. In addition, multiple experts improve initialization, as Fig. 4 demonstrates. We collected a video (30 frames/sec) of a subject in an outdoor setting who made a variety of facial expressions while moving her head. A later motion-capture session was used to create a 3D morphable model of her face, using five morph bases ($k = 5$).

Twenty experts were initialized randomly near the correct pose on frame 1 of the video and propagated using G-flow inference (with 20 experts, $\mathcal{K}_\infty(i) \approx 1$, small circular patches texture model). Fig. 4 shows the distribution of experts for three frames. In each frame, every one of the 20 experts has a hypothesis about the pose (both rigid and nonrigid). The 38 vertices are projected into the image according to each expert's pose, yielding 760 red dots in each frame. In each frame, the mean of the experts gives a single hypothesis about the 3D nonrigid deformation of the face (shown from frontal and profile view in the lower right) as well as the rigid pose of the face (rotated 3D-axes in the lower left). Notice G-flow's ability to recover from error: Bad initial hypotheses are quickly weeded out, leaving only good hypotheses (video is available in the online supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278).

6.3 Exploring the Continuum from Template to Flow: Varying the Kalman Gain

The object texture process noise $\Psi_v(i)$ and the image rendering noise (observation noise) σ_w^2 provide two degrees of freedom with which to tune the tracker. Using the equations in Section 4.1 as a guide, we can choose these two values in order to get any desired values for the steady-state Kalman gain $\mathcal{K}_\infty(i)$ and the steady-state temperature $\tau_\infty(i)$. (Though note that, since σ_w^2 is the same for every pixel in the image, $\mathcal{K}_\infty(i)$ and $\tau_\infty(i)$ cannot be varied independently for each texel.) To assess the effect of varying the Kalman gain on tracking performance, we performed several runs, keeping the steady-state temperature $\tau_\infty(i)$ (the overall contribution of noise) constant across all runs and all texels. We varied the steady-state Kalman gain of all object texels from $\mathcal{K}_\infty(i) = 0.001$ (near the template limit) to $\mathcal{K}_\infty(i) = 0.999$ (near the optic flow limit). For each value of $\mathcal{K}_\infty(i)$, we performed several tracking runs on the *Talk1* video sequence of the IR Marks data set (with 20 experts, dense 3D triangular mesh texture model). The results, shown in Fig. 3b, demonstrate that for this video sequence, the template end of the continuum is best. The results for the *Talk2* video sequence are similar; a video demonstrating the tracking results on the *Talk2* sequence is available in the online Supplemental Material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.278.

7 Discussion

7.1 Relation to Previous Work

We restrict our comparisons with previous work to methods that address the same problem as G-flow: tracking deformable objects in 3D from an image sequence. Hence, we do not discuss nonrigid structure-from-motion algorithms [25], [26], [27], which use 2D point correspondences rather than image sequences as input.

7.1.1 Relation to Other Algorithms for Tracking 3D Deformable Objects

Table 3 compares the features of G-flow and a number of existing approaches to monocular nonrigid 3D tracking.

Batch versus online processing. Of the existing monocular image-based 3D nonrigid trackers, some [7], [11] operate in batch mode (with the entire image sequence available throughout the tracking process) to simultaneously learn the shape model (learn the morph bases) and infer the morph parameters (track the vertices). Others operate in an online fashion (incorporate the information from each new frame as it appears), which enables them to be considered for

TABLE 3
Approaches to 3D Tracking of Deformable Objects

	Constrained optic flow	2D+3D AAM	3DGT	G-flow
Structure model (3D Morphable Model)	3DMM	3DMM	3DMM	3DMM
Inference uses Online vs. Batch processing	Online	Online	Batch	Online
Appearance model: Flow-based vs. Template-based	Flow	Template subspace (linear combina-tion of templates)	Template	Template or Flow, plus entire continuum in between. Adjusts dynamically, independently for each texel.
Appearance model: Small windows around vertices vs. Dense triangular mesh	Small windows	Dense mesh	Small windows	Small windows or Dense mesh
Generative model for image sequence	No	No	Yes	Yes
Filtering distribution for nonrigid pose (morph coefficients)	Single hypothesis (point estimate)	Single hypothesis (point estimate)	Gaussian distribution	Arbitrary distribution (Monte Carlo approximation)
Filtering distribution for rigid pose (rotation, translation)	Single hypothesis (point estimate)	Single hypothesis (point estimate)	Single hypothesis (point estimate)	Arbitrary distribution (Monte Carlo approximation)
Occlusion handling	No	Not permitted	Good (but assumes oc-clusions are independent of pose)	Yes
Background model	No	No	No	Yes

The table compares the features of G-flow with those of other approaches. The approaches compared are (left to right): Constrained optic flow [9], [10], [11]; 2D+3D active appearance models [8], [28]; the 3D generative template model of [7]; and our G-flow model.

real-time and memory-intensive applications. These online algorithms either have limited ability to learn the shape model [9] or assume that these morph bases are already known before the tracking process begins [8], [10]. Here, we have formulated our approach as an online algorithm.

Because our G-flow system has a well-defined conditionally Gaussian graphical model for generating image sequences (Fig. 1b), we could alternatively use the EM algorithm to learn the shape basis in batch mode while simultaneously inferring a posterior distribution over the pose parameters (while tracking the vertices). The implementation details would be similar to those in [7], with some added complications since we have a conditionally Gaussian model rather than a simple Gaussian model.

Relation to constrained optic flow and texture template models. In existing optic flow-based trackers [9], [10], [11], the goal of inference is to explain the flow vectors. These systems use optic flow only because the models are explicitly designed to explain optic flow. Thus, these systems can be viewed as models for generating flow vectors. In contrast, G-flow is explicitly designed as a model for generating images. In G-flow, the goal of inference is to explain the observed images (all of the pixel intensities) as well as possible. In our model, optic flow emerges naturally as part of the inference solution, rather than being the observable data in the model. As we explained in Section 4 , trackers based on constrained optic flow, as well as trackers based on appearance template matching, are entirely subsumed into the G-flow model as special cases. G-flow can be considered as a generalization of both types of models in which the entire continuum from template matching to optic flow can be utilized independently by each texel in the appearance model. Texels whose texture values remain roughly constant over time may be near the template end of the tracking spectrum, whereas texels whose values change rapidly may be near the optic flow end of the spectrum.

Relation to Torresani and Hertzmann's 3D generative template model. The system of [7] is in many ways similar to the template special case of G-flow. Although they use raw optic flow on the image sequence to initialize their system, their inference and learning procedures assume a template model. Their approach is based on a generative model for images that closely resembles the template special case of G-flow, and they too perform Bayesian inference on their generative model. To make inference tractable, they assume a purely Gaussian distribution over the nonrigid pose parameters. In contrast, G-flow makes inference tractable using Rao-Blackwellization to maintain a particle-based non-Gaussian pose distribution, and using Kalman filtering to enable an appearance model that can span the entire continuum from templates to optic flow. Finally, the EM algorithm of [7] treats rigid pose (rotation) as a parameter (with a single value), rather than as a random variable (which can take on an entire distribution of values). Thus, while their model accommodates uncertainty in the morph parameters, it cannot accommodate uncertainty in (cannot maintain a distribution over) the rigid motion parameters. The tracker of [7] handles self-occlusion quite well by using an indicator variable that enables texels to be occluded (or outliers) some fraction of the time. We may incorporate a similar indicator variable into G-flow in the future, if we implement batch EM learning. In comparison to G-flow's online inference, the system of [7] is a slow batch processing system: It requires the entire sequence of images before the tracking process can begin.

Relation to 2D+3D active appearance models. Standard active appearance models (2D AAMs) [29], [30] are not directly comparable to G-flow because they are purely 2D models, which track in the 2D image plane without regard to the 3D configuration of the vertices. However, there is a 3D extension of the active appearance model, the so-called combined 2D+3D active appearance model (2D+3D AAM) [8], [28], which is a real-time, online 3D tracking system. We classify it as a template-based model because its appearance model does not change over time.[3] The $2D + 3D$ AAM assumes that the 3D morphable shape model (3DMM) and a corresponding 2D AAM are known. The system can only track vertex positions that are consistent with both the 3DMM and the 2D AAM. Perhaps this system's biggest weakness is that it cannot handle self-

3. Note that the 2D+3D AAM appearance model is not a simple template, but a multidimensional subspace of templates. Rather than choosing the warp of the observed image that causes it to best match a single texture template, AAMs choose the warp of the observed image that causes it to best match any linear combination of a small number of appearance templates, i.e., that causes the warped image to be as close as possible to the linear subspace formed by this set of appearance templates.

occlusions (triangular facets of the 3D model are not able to face backward, or to partially occlude other patches), because self-occlusions are not consistent with the 2D AAM. In contrast, G-flow's fully 3D shape model can easily model self-occlusions.

Furthermore, since the $2D + 3D$ AAM is not a generative model for image sequences (unlike G-flow or [7]), it is not clear how it could be extended to learn models automatically. One of the most appealing features of AAMs is their impressive speed, which results from the efficiency of the *inverse compositional* form of the Lucas-Kanade image alignment algorithm [30], [31]. Because AAMs use a fixed template, they can precompute the computationally demanding steps on the template (the gradient of the template and the Jacobian of incremental warpings of the template) before tracking begins. Because the G-flow texture map is not fixed except in the template limit, we are not able to precompute the gradient of the texture map. However, we might be able to accelerate G-flow by precomputing the incremental warp Jacobian.

Modeling object texture: discrete image patches versus dense triangular mesh. The appearance models of [10], [11] consist of the pixel intensity values in a small neighborhood around each vertex. At each new frame, the pixel values in a small patch surrounding the predicted position of the vertex in the current frame are compared with the pixel values in a small patch surrounding each vertex's estimated position in the previous frame. The patch motion from one image to another is assumed to be pure translation (no warping), and subpixel displacements are handled using image interpolation. The texture models of [7], [9] similarly consist of small windows (image patches) around the vertex locations, though each patch is permitted to undergo affine transformation. One disadvantage of these approaches is that the models do not properly account for overlapping patches, and are thus unable to handle self-occlusions based on the object pose.

In contrast, the appearance model of [8] is a 2D triangular mesh connecting the object vertices, with an associated dense texture map. Each new image is warped in a piecewise-affine manner (with dense triangular patches defined by the predicted vertex locations in the image) to compare with the appearance model. However, the triangular mesh of [8] is not permitted to model self-occlusions (such as when the nose obscures part of the face), so the system has difficulty in tracking large out-of-plane rotations.

We implemented G-flow using two types of texture model: small windows (circular patches) around each vertex and a dense 3D triangular mesh connecting the vertices. We analyzed results from both in Section 6.

Modeling uncertainty in the filtering distribution. The algorithms of [8], [9], [10], [11] commit to a single solution for pose and appearance at each time step; that is, they do not model uncertainty. But image information can be ambiguous, and G-flow's explicit modeling of that uncertainty makes it more robust and less likely to lose track of the object.

Similarly to [8], [9], [10], [11], the system of [7] models *rigid* pose as a single point estimate. However, Torresani and Hertzmann [7] do maintain a Gaussian probability distribution over *nonrigid* pose parameters, which affords their system some ability to accommodate uncertain data. Still, their Gaussian uncertainty model limits the system to unimodal distributions over the pose parameters. When tracking objects, more than one pose may fit the image equally well at a given point in time. In such cases, Gaussian approaches place maximum certainty on the *average* of the high-probability poses. This can be dangerous if the average location is in a region of low probability. In contrast, G-flow's particle-based pose distribution enables it to accurately estimate arbitrary distributions over both rigid and nonrigid pose parameters.

7.1.2 Relation to Jacobian Images of Texture Maps

After developing G-flow, we learned that we were not the first to use dynamic texture maps. A similar idea was proposed in [32] as a means for obtaining super-resolution images of an object. Their basic approach can be seen as a special case of G-flow inference, in which the background is uninformative, there is only one expert, the proposal distribution collapses to a single point ($\alpha \to 0$), and the object is rigid ($k = 1$).

7.1.3 Relation to Other Rao-Blackwellized Particle Filters

Virtually, all particle filters suffer from a problem known as sample impoverishment [22], [33]. In most particle filters, the samples of the hidden variable depend only on the past samples of that variable, that is, the samples of U_t are drawn from the prior (pose transition) distribution $p(u_t \mid u_{t-1})$. The samples that are selected may be inadequate to represent the posterior distribution $p(u_t \mid y_{1:t})$ because most (or all) of the samples may be wasted in regions of u_t in which $p(u_t \mid y_{1:t})$ is negligible. Thus, very few of the particles will have high weight, so, at the next time step, these few particles will each be sampled several times.

A large part of this problem stems from the fact that standard particle filters have proposal distributions that rely only on observations up to time $t-1$, but ignore the already available sensor information from time t. In G-flow, not only does the filtering distribution for the new pose U_t depend upon the past poses $U_{1:t-1}$, as in conventional particle filters, but it also depends upon the new image y_t. In order to get our tracking scheme to work, our system obtains more informed hypotheses about the pose U_t by looking "ahead" to the current observation, using the observation y_t to help determine the location of our samples of U_t. That is, we use the information from time t to help decide which of the particles at time $t-1$ (our experts $u_{1:t-1}$) should be propagated forward to become estimates at time t, thus minimizing the problem of sample impoverishment. Particle filters like this one that look "ahead" to the current observation have been called as *look-ahead Rao-Blackwellized Particle Filters* [33].

As explained in Section 3 , we combine the Gauss-Newton method and Laplace's method to look ahead to the current image before sampling. This look-ahead results in an optic-flow-like algorithm that distinguishes inference in G-flow from other look-ahead Rao-Blackwellized particle filters.

Rao-Blackwellized particle filters have not previously been applied to 3D tracking of deformable objects, though they have been used to track 2D rigid objects, such as honeybees [34]. In [34], the sample space over poses was much lower dimensional than our pose space. As a result, they did not need look ahead and could simply sample from their prior distribution for location, a lower dimensional analog to our prior distribution $p(u_t \mid u_{t-1})$.

7.2 Future Work

7.2.1 Varying the Kalman Gain for Different Texels

Previous tracking systems use either a flow-based approach [9], [10], [11] or a template-based approach [7], [8] to appearance modeling, but G-flow is the first system to combine the advantages of both approaches in a principled manner, encompassing a continuous range of models from template-based to flow-based within a unified framework. This theoretical framework makes it possible to vary the level of template-ness or flow-ness dynamically over time and also across space. Thus, some texels (or some particles) can be more flow-like, whereas others can be more template-like. Ultimately, the optimal parameters of each texel should be inferred automatically according to the demands of the data.

7.2.2 Improved Imaging Model

To simplify the derivations in this paper, we made the assumption that there is a one-to-one mapping from object texels to image pixels. In practice, to find the texture values at image pixels that are rendered by the object, we interpolate the texture from the nearest object texels as in [8], [9], [10], [11], [30], [31]. In the future, however, we would like to incorporate interpolation directly into our imaging model (our model of how texels map to pixels), using an approach similar to that of [32].

7.2.3 Background Tracking

In our simulations, we did not use the background terms of the predictive distribution (17) and objective function (18). This is essentially equivalent to assuming a white noise background, and the mathematical analysis in this paper has demonstrated that other tracking algorithms (e.g., optic flow) make this assumption implicitly. Implementation of the background model (which is fully described and derived in this paper) will no doubt improve performance.

7.2.4 Sophisticated Texture Models

The current G-flow model uses a simple model for texture: an expected texture map and associated variance parameters for each texel. More sophisticated models are possible in which object texture is produced by a linear combination of texture maps in the same way that geometric deformations are modeled as a linear combination of key shapes. Subspace texture models have previously been used effectively for 3D face tracking [8], [28], for 2D tracking with Rao-Blackwellized particle filters [34], and for 2D tracking using a dynamic texture map [35]. Shape and texture parameters could also be permitted to be correlated. This would, for example, allow the system to use movements of eyebrow vertices to help detect the presence of wrinkles in the forehead, or to use the presence of wrinkles to help detect movements of parts of the face.

7.2.5 Learning Shape Models

The current version of G-flow assumes that the geometry of the object of interest is already known via prior experience. We and others (e.g., [7], [36]) have demonstrated methods for learning such models in a principled manner, and these could be integrated within the G-flow perspective.

7.3 Conclusion

We introduced a probabilistic formulation of the video generation process for 3D deformable objects and derived an equation for optimal tracking. The approach revealed an entire space of possible algorithms to solve this problem and helped clarify that classic computer vision algorithms such as optic flow and template matching are optimal only under very special circumstances. This opens up possibilities for a whole spectrum of new algorithms. We presented one such new algorithm that takes advantage of the conditionally Gaussian structure of the problem, resulting in significant improvements over previous methods while pointing out a rich range of possibilities for further improvements.

Unlike previous systems, G-flow incorporates a background model that enables information in the background to be utilized. The other systems cited above include only a foreground model, which (as we demonstrated in this paper) makes implicit assumptions about the background (essentially assuming that the background is white noise). In addition, G-flow uses Rao-Blackwellized particles (experts) to maintain an unrestricted (nonlinear, non-Gaussian) distribution over the pose parameters. This is more flexible than other models, which either collapse the pose distribution to a single point at every time step [8], [9], [10], [11] or use a single point estimate of rigid pose parameters (rotation) and restrict the distribution over nonrigid pose parameters to be Gaussian [7].

We refer to G-flow's Rao-Blackwellized particles as *experts* because, in addition to a single pose estimate, each expert maintains a corresponding Gaussian distribution over texture, inferred using a Kalman filter, plus a predictive pose distribution that "looks ahead" to the next frame. As a result, not nearly as many experts are required as in a standard particle filter. Indeed, existing optic-flow-based trackers can be seen as limited implementations of G-flow that use only a single expert. As demonstrated in Section 6, on the order of 10 experts is enough to produce a considerable improvement over existing tracking systems.

Acknowledgments

Earlier versions of this work appeared in [16], [17], [36]. Tim K. Marks was supported by US National Science Foundation (NSF) grants IIS-0223052 and DGE-0333451. John R. Hershey was supported by UCDIMI grant D00-10084. Javier R. Movellan was supported by NSF grants ECCS-0622229 and IIS-0808767. Tim K. Marks performed this research at the Departments of Cognitive Science and Computer Science and Engineering at the University of California, San Diego.

References

[1] M. Osadchy, Y. LeCun, and M. Miller, "Synergistic Face Detection and Pose Estimation with Energy-Based Models," *J. Machine Learning Research,* vol. 8, pp. 1197-1215, 2007.

[2] A. Torralba, K.P. Murphy, W.T. Freeman, and M. Rubin, "Context-Based Vision System for Place and Object Recognition," *Proc. IEEE Int'l Conf. Computer Vision,* 2003.

[3] G. Hinton, S. Osindero, and K. Bao, "Learning Causally Linked Markov Random Fields," *Proc. Int'l Workshop Artificial Intelligence and Statistics,* 2005.

[4] I. Fasel, B. Fortenberry, and J.R. Movellan, "A Generative Framework for Real-Time Object Detection and Classification," *Computer Vision and Image Understanding,* vol. 98, pp. 182-210, 2005.

[5] M. Beal, N. Jojic, and H. Attias, "A Graphical Model for Audio-Visual Object Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 7, pp. 828-836, July 2003.

[6] N. Jojic and B. Frey, "Learning Flexible Sprites in Video Layers," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 199-206, 2001.

[7] L. Torresani and A. Hertzmann, "Automatic Non-Rigid 3D Modeling from Video," *Proc. European Conf. Computer Vision,* 2004.

[8] J. Xiao, S. Baker, I. Matthews, and T. Kanade, "Real-Time Combined 2D+3D Active Appearance Models," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2004.

[9] L. Torresani, D. Yang, G. Alexander, and C. Bregler, "Tracking and Modeling Non-Rigid Objects with Rank Constraints," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 493-500, 2001.

[10] M. Brand and R. Bhotika, "Flexible Flow for 3D Nonrigid Tracking and Shape Recovery," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2001.

[11] M. Brand, "Morphable 3D Models from Video," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2001.

[12] H. Chen, P. Kumar, and J. van Schuppen, "On Kalman Filtering for Conditionally Gaussian Systems with Random Matrices," *Systems and Control Letters,* vol. 13, pp. 397-404, 1989.

[13] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks," *Proc. 16th Conf. Uncertainty in Artificial Intelligence,* pp. 176-183, 2000.

[14] R. Chen and J. Liu, "Mixture Kalman Filters," *J. Royal Statistical Soc.: Series B,* vol. 62, pp. 493-508, 2000.

[15] A. Doucet and C. Andrieu, "Particle Filtering for Partially Observed Gaussian State Space Models," *J. Royal Statistical Soc.: Series B,* vol. 64, pp. 827-838, 2002.

[16] T.K. Marks, J. Hershey, J.C. Roddey, and J.R. Movellan, "3D Tracking of Morphable Objects Using Conditionally Gaussian Nonlinear Filters," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, Workshop Generative Model Based Vision,* 2004.

[17] T.K. Marks, J. Hershey, J.C. Roddey, and J.R. Movellan, "Joint Tracking of Pose, Expression, and Texture Using Conditionally Gaussian Filters," *Advances in Neural Information Processing Systems,* vol. 17, pp. 889-896, MIT Press, 2005.

[18] V. Blanz and T. Vetter, "A Morphable Model for the Synthesis of 3D Faces," *Proc. ACM SIGGRAPH '99,* pp. 187-194, 1999.

[19] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME-J. Basic Eng. D,* vol. 82, pp. 35-45, 1960.

[20] G.S. Fishman, *Monte Carlo Sampling: Concepts Algorithms and Applications.* Springer-Verlag, 1996.

[21] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan, "An Introduction to MCMC for Machine Learning," *Machine Learning,* vol. 50, nos. 1/2, pp. 5-43, 2003.

[22] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for On-Line Non-Linear/Non-Gaussian Bayesian Tracking," *IEEE Trans. Signal Processing,* vol. 50, no. 2, pp. 174-188, 2002.

[23] C. Bregler, A. Hertzmann, and H. Biermann, "Recovering Non-Rigid 3D Shape from Image Streams," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2000.

[24] L. Torresani, A. Hertzmann, and C. Bregler, "Learning Non-Rigid 3D Shape from 2D Motion," *Advances in Neural Information Processing Systems,* vol. 16, MIT Press, 2004.

[25] M. Brand, "A Direct Method for 3D Factorization of Nonrigid Motion Observed in 2D," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2005.

[26] J. Xiao, J. Chai, and T. Kanade, "A Closed-Form Solution to Non-Rigid Shape and Motion Recovery," *Proc. European Conf. Computer Vision,* 2004.

[27] L. Torresani, A. Hertzmann, and C. Bregler, "Nonrigid Structure-from-Motion: Estimating Shape and Motion with Hierarchical Priors," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 5, pp. 878-892, May 2008.

[28] I. Matthews, J. Xiao, and S. Baker, "2D versus 3D Deformable Face Models: Representational Power, Construction, and Real-Time Fitting," *Int'l J. Computer Vision,* vol. 75, no. 1, pp. 93-113, 2007.

[29] T. Cootes, G. Edwards, and C. Taylor, "Active Appearance Models," *Proc. European Conf. Computer Vision,* vol. 2, pp. 484-498, 1998.

[30] I. Matthews and S. Baker, "Active Appearance Models Revisited," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 135-164, 2004.

[31] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *Int'l J. Computer Vision,* vol. 56, no. 3, pp. 221-255, 2004.

[32] F. Dellaert, S. Thrun, and C. Thorpe, "Jacobian Images of Super-Resolved Texture Maps for Model-Based Motion Estimation and Tracking," *Proc. IEEE Workshop Applications of Computer Vision,* pp. 2-7, 1998.

[33] N. de Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole, "Diagnosis by a Waiter and a Mars Explorer," *Proc. IEEE,* special issue on sequential state estimation, vol. 92, no. 3, pp. 455-468, Mar. 2004.

[34] Z. Khan, T. Balch, and F. Dellaert, "A Rao-Blackwellized Particle Filter for Eigentracking," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2004.

[35] J. Ho, K.-C. Lee, M.-H. Yang, and D.J. Kriegman, "Visual Tracking Using Learned Linear Subspaces," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 782-789, 2004.

[36] T.K. Marks, "Facing Uncertainty: 3D Face Tracking and Learning with Generative Models," PhD dissertation, Univ. of California San Diego, 2006.

Tim K. Marks received the AB degree in physics from Harvard University in 1991 and the MS and PhD degrees in cognitive science from the University of California, San Diego (UCSD) in 2001 and 2006, respectively. From 2006 to 2008, he was a postdoctoral researcher in computer science and engineering at UCSD, where he worked in collaboration with the computer vision group at the NASA/Caltech Jet Propulsion Laboratory (JPL). Since 2008, he has been a research scientist at Mitsubishi Electric Research Laboratories in Cambridge, Massachusetts. His main research interests include applications of machine learning to computer vision. He is a member of the IEEE.

John R. Hershey received the BS degree in cognitive science from the University of California, Los Angeles, in 1992, and the PhD degree in cognitive science in 2004 from the University of California, San Diego (UCSD), where he was a founding member of the Machine Perception Laboratory. His thesis explored the use of generative graphical models for speech enhancement, face tracking, and combinations of the two. During his time at UCSD, he interned at Mitsubishi Electric Research Laboratories in Cambridge, Massachusetts, in 2001, and in the Machine Learning and Applied Statistics Group at Microsoft Research, Seattle, in 2003. In 2004, he spent a year as a visiting researcher in the Speech Group at Microsoft Research. Since 2005, he has been at the IBM T.J. Watson Research Center in New York, where he is a research staff member in the Speech Algorithms and Engines Group. He is a member of the IEEE.

Javier R. Movellan received the PhD degree from the University of California, Berkeley, in 1990. He founded the Machine Perception Laboratory (MPLab) at the University of California, San Diego (UCSD), where he is currently a research professor. The mission of the MPLab is to learn about intelligent behavior by developing systems that operate in the uncertain but sensory rich conditions typically faced by the brain. His research interests include machine learning, machine perception, automatic analysis of human behavior, and social robots. Prior to his UCSD position, he was a fulbright scholar at UC Berkeley and a research associate at Carnegie Mellon University (1990-1993). He is a member of the IEEE.