Supplemental Material

APPENDIX I
IMPORTANCE SAMPLING

Importance sampling is a Monte Carlo technique that was designed in the 1940s for efficient estimation of expected values [A1]. The approach works as follows. Suppose we want to estimate the expected value of a function h of a random variable X, whose probability density function is $p(x)$:

$$E\big(h(X)\big) = \int p(x)h(x)dx \tag{41}$$

One way to do so is to take n samples $\{x^{(1)}, \cdots, x^{(n)}\}$ from X and average the values taken by h on those samples

$$E\big(h(X)\big) \approx \frac{1}{n}\sum_{i=1}^{n} h\big(x^{(i)}\big) \tag{42}$$

Instead of sampling directly from $p(x)$, which may be intractable or computationally costly, in some cases it is desirable to obtain samples from a *proposal distribution*, $\pi(x)$, from which samples can be drawn efficiently, and compensate by using *importance weights* that are proportional to the ratio of $p(x)$ to $\pi(x)$. To see why, note that

$$\begin{aligned} E\big(h(X)\big) = \int p(x)h(x)dx &= \int \pi(x)\frac{p(x)}{\pi(x)}h(x)dx \\ &= \int \pi(x)\xi(x)h(x)dx \end{aligned} \tag{43}$$

where the importance weight, $\xi(x)$, is the ratio of the values taken by the desired distribution and the proposal distribution:

$$\xi(x) = \frac{p(x)}{\pi(x)}. \tag{44}$$

Thus, the corresponding importance sampling estimate would be

$$E\big(h(X)\big) \approx \frac{1}{n}\sum_{i=1}^{n} \xi\big(x^{(i)}\big)h\big(x^{(i)}\big) \tag{45}$$

where now the samples $\{x^{(1)}, \cdots, x^{(n)}\}$ are taken from the proposal distribution $\pi(x)$. Since the approach provides estimates for expected values of arbitrary functions h, it can be interpreted as providing an estimate for the distribution of X itself, i.e.,

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \xi(x)\delta(x - x^{(i)}), \tag{46}$$

where $\delta(\cdot)$ is the Dirac delta function.

When $p(x)$ is only known up to a proportionality constant, then the importance weights $\xi(x)$ are also known up to a proportionality constant. In this case we can rewrite (43) as follows:

$$E\big(h(X)\big) = \frac{\int \pi(x)\xi(x)h(x)dx}{\int \pi(x)\xi(x)dx} \tag{47}$$

where $\xi(x) \propto \frac{p(x)}{\pi(x)}$ is now known up to a normalizing constant. Since the ratio of two consistent estimators is also consistent, a standard estimate in this case is as follows [A2]:

$$E\big(h(X)\big) \approx \frac{\frac{1}{n}\sum_{i=1}^{n} \xi\big(x^{(i)}\big)h\big(x^{(i)}\big)}{\frac{1}{n}\sum_{i=1}^{n} \xi\big(x^{(i)}\big)} = \sum_{i=1}^{n} \tilde{\xi}\big(x^{(i)}\big)h\big(x^{(i)}\big), \tag{48}$$

where as in (45), the samples $\{x^{(1)}, \cdots, x^{(n)}\}$ are taken from the proposal distribution $\pi(x)$, and

$$\tilde{\xi}\big(x^{(i)}\big) = \frac{\xi\big(x^{(i)}\big)}{\sum_{j=1}^{n} \xi\big(x^{(j)}\big)} \tag{49}$$

is the normalized importance weight. The corresponding estimate of the distribution of X is as follows:

$$\hat{p}(x) = \tilde{\xi}(x)\delta(x - x^{(i)}). \tag{50}$$

For the estimation process to be efficient, the proposal distribution $\pi(x)$ must be as close as possible to the original distribution, $p(x)$. For our application, we found that in addition to Rao-Blackwellization, it was critical to use the Gauss-Newton method and Laplace's method to generate good proposal distributions for importance sampling. As we explained in Section 3.4.1, we accomplish this in the context of G-flow by performing an optic flow-like optimization.

APPENDIX II
USING INFRARED TO LABEL SMOOTH FEATURES INVISIBLY

Until now, there has been no publicly available video data set of a real human face moving that is up to the task of measuring the effectiveness of 3D nonrigid tracking systems. Progress in the field requires publicly available video data sets with ground truth information about the true 3D locations of the facial features that are to be tracked. In addition to facilitating refinement of one's own algorithm during development, such data sets will provide standards for performance comparison with other systems.

In this appendix, we describe a new method for collection of such data sets, using infrared marks that are visible by near-infrared (NIR) cameras but not by standard video cameras. We used this technique to collect a face motion data set that we are making freely available to researchers in the field with the publication of this paper. We describe the data collection method in detail in the following section of this appendix (Appendix II-A), and the new data set, which we call *IR Marks*, in the subsequent section (Appendix II-B).

A. Details of the data collection method

Subjects' faces were marked using ink that absorbs light in the near-infrared (NIR) range of the spectrum (see Figure 5). Three infrared-sensitive video cameras (which picked up the NIR markings) were placed close to the face for maximum spatial resolution. We also used four visible-light video cameras (whose images do not show the NIR markings), of which two were far away from the face and zoomed in (for testing systems that use orthographic projection or weak-perspective projection), and two were close up (for testing systems that use perspective projection). We used two, rather than just one, visible-light cameras at each distance in order to accommodate any systems that track using video from multiple cameras.

1) Infrared marking: Subjects' faces were marked with an infrared pen, the IR1 pen from *maxmax.com* (*http:// maxmax.com/ aXRayIRInks.asp*). This pen has ink that absorbs near-infrared (NIR) light with peak wavelength 793 nm, outside the visible range.[4]

2) Visible-light cameras: We used four *Firefly* cameras from Point Grey Research (*http:// www.ptgrey.com*), which we chose due to their ability to record synchronized video. These cameras captured 640×480 color video at 30 frames per second.

The ink from the IR1 pen can show up as a very faint yellow/green that is close to the threshold of visibility to the naked eye. We thus took care to apply the marks without pressing too hard with the pen (which can make visible indentations in the skin), and to avoid oversaturating the marks with ink. After some trial and error, we succeeded in applying the ink so that it showed up in the NIR cameras but not in the visible-light cameras.

[4]The human eye can see light with wavelengths from roughly 400 nm to 700 nm. Ultraviolet light has wavelengths shorter than 400 nm, whereas infrared light has wavelengths longer than 700 nm. Over the lens of each NIR-sensitive camera, we used a filter that blocks visible light: the X-Nite 715 infrared filter, also from *maxmax.com*. The ink in the IR1 pen is in an alcohol base. We cannot vouch for the safety of the ink for use on the skin, except to say that we used it liberally on the faces of two different people, without any adverse reaction.

Fig. 5. 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 four visible-light cameras (top) and three infrared-sensitive cameras (bottom). The infrared marks are clearly visible using the infrared-sensitive cameras, but are not visible in the images from the visible-light cameras.

3) Infrared-sensitive cameras: In order to reconstruct the 3D locations of the infrared-marked points, they must be visible simultaneously by at least two NIR-sensitive cameras. We used three NIR cameras, one directly in front of the face, another slightly to the left, and a third slightly to the right. The goal was to ensure that even when the head was turned, as many of the marked points as possible would be visible in at least two of the three cameras, to enable 3D reconstruction of the point locations.

The NIR cameras were standard Sony camcorders in Nightshot mode. There are some difficulties due to the limited aperture/shutter settings of these cameras in this mode, but overall, they are a reasonable alternative to more expensive NIR systems. Because we were forced to use a fully open aperture, we could not use incandescent lights to illuminate the scene, as they would overexpose the image. Instead, we used fluorescent lights to provide ample illumination for our visible light cameras, but at the same time provide only a small amount of illumination that could penetrate the X-Nite 715 filters, so as not to overexpose the NIR images.[5]

As a result of the fully open aperture and slow shutter speed required by Sony's Nightshot mode, as well as the lightness of the infrared marks that we drew, the images obtained with the NIR cameras were somewhat noisy, and also somewhat blurry during motion. To counter this effect, we used time-averaging and contrast-enhancement to make the NIR marks more visible, which resulted in some marks not being reliably visible in frames during which the head was moving quickly. As a result, we could only obtain reliable ground truth information for the frames in which the head was moving slowly, or frames in which a fast-moving head was temporarily slowed due to a change in direction.

4) Camera synchronization: The four visible-light cameras were well synchronized, as they were all triggered by an electrical pulse to capture each frame in synchrony. However, our three NIR cameras were inexpensive off-the-shelf cameras that recorded to digital videotape, without the ability to synchronize electronically. We later uploaded the video from these tapes by capturing the video frames as a sequence of digital images. At several times during the video session, the subject held up an LED connected to a switch, and switched it on. The onset of the light, which was visible in all of the visible-light and NIR cameras, was used as a synchronizing signal for temporal alignment of the frames from the three NIR cameras with each other and with the visible-light cameras. The alignment was implemented as a piecewise-linear function to map the frame number from each NIR camera to the corresponding frame number from the visible-light cameras. This resulted in a temporal alignment error on the order of just one-half of a frame $\left(\frac{1}{60}\ \mathrm{sec}\right)$.

5) 3D camera calibration: During collection of the video, we held a calibration object (a checkerboard of known dimensions) at several different 3D orientations, making sure that at each orientation, the entire checkerboard was visible in every one of the seven cameras (4 visible-light and 3 NIR). We used the stereo calibration tool from Jean-Yves Bouguet's camera calibration toolbox for Matlab ($www.vision.caltech.edu/bouguetj/calib\_doc/$) to simultaneously calibrate one of the visible-light cameras with one of

[5]Our approach was to adjust the lighting to match the camera, since the limitations of Sony's Nightshot mode did not allow us to adjust the camera to match the lighting. Another approach (which could provide more flexibility) would be to use a camera that is fully enabled in the near-infrared range.

the NIR cameras. We repeated this process three times, each time with a different NIR camera but the same visible-light camera. As a result of this process, we had each of the NIR cameras calibrated with respect to the same visible-light camera (and hence with respect to each other).

6) Hand-labeling of infrared points: We chose several frames in which to hand-label the locations of each near-infrared (NIR) dot in the image from each of the NIR-sensitive cameras. After the hand-labeling, we knew the 2D image location of each NIR dot in three images, one from each of the three NIR cameras.

7) 3D reconstruction of labeled points: The camera calibration algorithm provided extrinsic and intrinsic camera parameters for each NIR camera. The labeled 2D position of an NIR dot in an image corresponds to a 3D line on which the point lies in 3D world coordinates. If a point were visible in just two NIR cameras, we could find its 3D world position by calculating the intersection of the lines extending from the two cameras. Due to errors in calibration and labeling, however, the two lines will not quite intersect in general, in which case we need to find the single 3D point that is closest to the two lines. If an NIR dot is visible (and labeled) in images from all three NIR cameras, then the goal is to find the single 3D point that is closest to the three lines extending from the three cameras.

This is a least-squares problem (finding the 3D location in the world that minimizes the sum of squared differences corresponding to the two or three labeled 2D positions in the NIR images). The 3D location of each point was found using the method of homogeneous least squares [A3].

B. The IR Marks data set for 3D face tracking

This section describes the new data set, called *IR Marks*, which we are making available for free to the research community with the publication of this paper (go to *http://mplab.ucsd.edu* and follow the link to *Databases*). The data set consists of a training sequence (which includes 9 labeled training frames) and 3 test sequences, of a human face that exhibits both nonrigid motion (making facial expressions and talking) and rigid motion (head rotation and translation, both in-image-plane and out-of-image-plane). We used these three test sequences to rigorously test the performance of our tracking system, and to compare it to other systems (see Section 6). In addition, having ground truth information available (and thus having an accurate quantitative measure of error, rather than just "eyeballing" it approximately), has been invaluable to us in debugging our code, as well as in choosing values of parameters for optimal tracking.

Data sets like this one could be used to evaluate and compare the performance, not only of 3D nonrigid tracking systems (its primary purpose), but also of systems for determining 3D nonrigid structure-from-motion. We encourage others to collect their own data sets using our method and to similarly make them available to other researchers. We believe that the existence of standard data sets with good ground truth information is crucial for the field to progress. This data set is a first step in that direction.

1) Training sequence: We drew a number of infrared dots on a male subject. In addition to frames (of a checkerboard object) for spatial calibration and (of an LED being switched on) for temporal calibration of the cameras, we collected a training video sequence whose purpose was to learn a 3D morphable model of the subject that would later be used to track the face in the test sequences. The training sequence consists of the subject, always from frontal view (with minimal rigid head motion), making a series of facial expressions. We chose one frame of each facial expression (and two neutral frames, near the beginning and the end of the sequence)—a total of 9 frames—for hand-labeling:

$$\begin{array}{c}\text{neutral, closed-mouth smile, happy, sad,}\\ \text{angry, disgusted, afraid, surprised, neutral}\end{array} \tag{51}$$

For each of these 9 frames, the 2D image locations of 58 infrared dots were hand-labeled twice, in images from each of the three NIR-sensitive cameras. Of the 58 dots, 6 did not have the property of being visible in at least two NIR cameras for all 9 chosen training frames. These 6 vertices were removed, leaving 52 dots (vertices) whose 3D locations were reconstructed in all 9 training frames. The reconstruction error is on the order of 1 mm. In the experiments described in this paper, our only use of the training sequence was to obtain a 3D morphable model from the 3D locations of the vertices in the 9 labeled training frames, as described in Appendix III-A.

2) Three test sequences: We then collected three test video sequences simultaneously in all cameras. For each sequence, we chose several key frames for which we hand-labeled the images from all three NIR cameras, from which we reconstructed the 3D ground truth information for those frames. Of the 52 infrared dots in the training sequence, 7 did not have the property of being visible in at least two NIR cameras for all of the selected test frames. This leaves 45 vertices whose ground truth information is known for all of the chosen frames in all of the test sequences. The *IR Marks* data set consists of the training sequence and three test sequences: *Talk1*, *Talk2*, and *Emote*.

The first test sequence, which we call the *Talk1* sequence, is empirically the easiest to track (probably because it is the shortest sequence and has the least out-of-image-plane rotation). The sequence consists of the subject talking and naturally moving the head and face in a conversation. We labeled 9 key frames from this sequence twice, and used these labelings to compute the ground truth 3D locations of all of the vertices in each of these 9 frames. From the first labeled frame to the last labeled frame, the Talk1 sequence is 914 frames (over 30 sec) long.

The second test sequence, which we call the *Talk2* sequence, is of medium difficulty (it is almost twice as long as the Talk1 sequence, and includes larger out-of-image-plane rotations). Like the Talk1 sequence, the Talk2 sequence consists of the subject talking and naturally moving the head and face in a conversation. We labeled 11 key frames from the Talk2 sequence, and used these labelings to compute the ground truth 3D locations of all of the vertices in each of these 11 frames. From the first labeled frame to the last labeled frame, the Talk2 sequence is 1716 frames (over 57 sec) long.

The third test sequence, which we call the *Emote* sequence, consists of the subject making each of the facial expressions that are listed above in (51), and holding each facial expression while making fast rigid head motions that include large out-of-image-plane rotations. The Emote sequence is empirically the most difficult to track, probably due to its greater length, its greater extremes of facial expressions, its fast motion, fast transitions from one extreme expression to another, and large out-of-plane rotations. We labeled 9 key frames from this sequence, and used these labelings to compute the ground truth 3D locations of all of the vertices in each of these 9 frames. From the first labeled frame to the last labeled frame, the Emote sequence is 1855 frames (over 61 sec) long.

APPENDIX III
IMPLEMENTATION DETAILS

A. Obtaining the 3D morphable model

The 3D ground truth locations of 52 vertices were obtained for the 9 labeled frames of the IR Marks data set's training sequence, as explained in Appendix II. We then needed to find a 3D morphable model, consisting of morph bases that could be linearly combined to give (or closely approximate) the 3D locations of the points in all 9 frames.

First we used the Procrustes method [A4] to subtract any rigid motion between the frames (to rotate all of the frames in 3D to align with each other). We did not need to rescale any of the frames, since the camera calibration meant that our 3D data were already scaled to the correct absolute size (in mm). Once the frames were rigidly aligned, the only remaining differences between the frames were the nonrigid motion (due to changes in facial expressions). We performed principal component analysis (PCA) on the rigidly aligned 3D data for the 9 training frames, to obtain 9 morph bases (the mean plus 8 variational modes) that make up the 3D morphable model.

B. Parameter settings

The original 3D morphable model had 9 morph bases, but for our tracking results, we reduced this to the mean and top 4 modes of variation, for a total of 5 morph bases: $k = 5$.

The spread of the proposal distribution for pose opinion, adjusted using the parameter α (see Section 3.4.1), changes the balance between exploration and exploitation: a wider proposal distribution (larger value of α) gives the system more opportunity to find more optimal pose values. We found that a relatively large spread, $\alpha = 50$, works well. We used a steady-state temperature (see Section 4.1) of $\tau_\infty = 1000$. We chose resampling frames to occur every 25 frames. Unless stated otherwise, we performed tracking using 20 experts, $\eta = 20$, and used 5 samples for each expert's pose opinion: $\lambda = 5$.

For determining the proposal distribution, we implemented our algorithm using small circular windows (diameter 15 pixels) around each vertex. For the importance sampling correction to the proposal distribution, we usually used a dense triangular mesh, but occasionally used small circular windows (when stated explicitly, for the optic flow limit).

We tracked the *Talk1* sequence of the IR Marks data set (See Appendix II) using all 45 of the vertices that were visible in every labeled frame of the three test sequences. Because we used small windows around each vertex (rather than a dense triangular mesh) to obtain our proposal distributions, we had some difficulty when tracking the *Talk2* and *Emote* sequences in handling points on the far right edge of the face (points that were self-occluded in extreme out-of-image-plane rotations). This difficulty with tracking the *Talk2* and *Emote* sequences could be minimized by eliminating three vertices from the mesh that are on the rightmost edge of the face: the right dimple, and the two rightmost vertices under the right eye (RD, RE4, and RE5).

In the experiments described in this paper, we have assumed an uninformative background. Thus we have not implemented the background texture model. As a result, for the experiments described in this paper, we eliminated both background terms and the second of the two foreground terms from the G-flow objective function (18).

C. Graphics hardware acceleration

We implemented G-flow using both types of texture model: small windows (circular patches) around each vertex, and a dense triangular mesh connecting the vertices. For the dense triangular mesh, we took advantage of graphics hardware acceleration for texture mapping to speed up the image warping and image comparison. As mentioned above, in our current implementation of the triangular mesh texture model, we actually use small circular patches around the vertices to obtain the proposal distribution, then use the full dense triangular mesh to obtain the importance weights.

Appendix IV
Exponential rotations and their derivatives

Before we can derive the constrained optic flow algorithm (in Appendix V), we must discuss exponential rotations. Taking derivatives with respect to rotations is not trivial. The straightforward approach of differentiating a 3×3 rotation matrix with respect to each of the elements in the matrix individually (used, for example, in [A5]) is not ideal. The problem is that although there are 9 elements in the matrix, there are nonlinear constraints between the 9 values.

In fact, there are only 3 degrees of freedom in a rotation matrix. These 3 degrees of freedom are expressed more naturally using the 3 exponential rotation parameters, described below. We begin this appendix by deriving the widely known first derivative of rotations with respect to the rotation parameters [A6].

Then we derive the second derivative with respect to rotation parameters. To our knowledge, this paper is the first time that this second derivative has been derived or used in the fields of machine learning, computer vision, or robotics. This second derivative has numerous potential applications in these fields; we explain its use in G-flow in Appendix VII-B.2.

An arbitrary rotation in 3D can be specified by the axis of rotation and the angle of rotation about this axis. Let δ be the vector in the direction of the axis of rotation whose magnitude is the angle of rotation (in radians), and let Δ be the skew-symmetric matrix that is defined from δ as follows:

$$\delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix}, \qquad \Delta = \begin{bmatrix} 0 & -\delta_3 & \delta_2 \\ \delta_3 & 0 & -\delta_1 \\ -\delta_2 & \delta_1 & 0 \end{bmatrix}. \tag{52}$$

Then the rotation matrix is given by e^Δ, which is defined using the matrix exponential:

$$e^\Delta = I_3 + \Delta + \frac{\Delta^2}{2!} + \mathcal{O}(\delta^3) \tag{53}$$

where I_3 is the 3×3 identity matrix, and $\mathcal{O}(\delta^3)$ represents the third-order and higher-order terms of the Taylor series.

Define γ_1, γ_2, and γ_3 to be the matrices given by:

$$\gamma_j = \frac{\partial \Delta}{\partial \delta_j}. \tag{54}$$

Then the γ_j are the following skew symmetric matrices:

$$\gamma_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad \gamma_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad \gamma_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{55}$$

A. Derivatives of rotations

To find the first derivatives of the rotation matrix e^Δ around $\delta = \mathbf{0}$, we just need the first two terms in the Taylor series (53) for e^Δ.

$$\frac{\partial e^\Delta}{\partial \delta_j} = \frac{\partial}{\partial \delta_j}(I_3 + \Delta) = \frac{\partial \Delta}{\partial \delta_j} = \gamma_j. \tag{56}$$

To find the second derivatives of the rotation matrix e^Δ around $\delta = \mathbf{0}$, we need the first three terms in the Taylor series (53) for e^Δ.

$$\begin{aligned} \frac{\partial^2 e^\Delta}{\partial \delta_j \partial \delta_k} &= \frac{\partial}{\partial \delta_j}\frac{\partial}{\partial \delta_k}\left(I_3 + \Delta + \frac{\Delta^2}{2}\right) \\ &= \frac{\partial}{\partial \delta_j}\left(\gamma_k + \frac{1}{2}(\Delta\gamma_k + \gamma_k\Delta)\right) \\ &= \frac{\gamma_j\gamma_k + \gamma_k\gamma_j}{2}. \end{aligned} \tag{57}$$

To find the derivative of any 3×3 rotation matrix r with respect to δ, we consider the composite rotation $e^\Delta r$, and evaluate its derivative at $\delta = \mathbf{0}$.

$$\frac{\partial r}{\partial \delta_j} = \frac{\partial\, e^\Delta r}{\partial \delta_j} = \left(\frac{\partial e^\Delta}{\partial \delta_j}\right) r = \gamma_j\, r. \tag{58}$$

We use the same approach to find the second derivatives of any 3×3 rotation matrix, r, with respect to δ (to find the terms of the Hessian matrix):

$$\frac{\partial^2 r}{\partial \delta_j \partial \delta_k} = \frac{\partial^2 e^\Delta r}{\partial \delta_j \partial \delta_k} = \left(\frac{\partial^2 e^\Delta}{\partial \delta_j \partial \delta_k}\right) r = \left(\frac{\gamma_j\gamma_k + \gamma_k\gamma_j}{2}\right) r. \tag{59}$$

B. Derivatives of a vertex location in the image

We are now in a position to take derivatives of the vertex locations in our 3D morphable model for shape with respect to rotation angle (with respect to the rotation parameters, δ). Taking the first derivative of the equation (from Section 2.1) for x_i, the image coordinates of the ith vertex, gives:

$$\frac{\partial x_i}{\partial \delta_j} = \frac{\partial}{\partial \delta_j}(grh_ic + l) = g\frac{\partial r}{\partial \delta_j}h_ic = g\gamma_j r h_i c, \tag{60}$$

where for the final step we used (58). We can find the second derivatives of x_i with a similar approach using (59) :

$$\frac{\partial^2 x_i}{\partial\delta_j\partial\delta_k} = g\left(\frac{\partial^2 r}{\partial\delta_j\partial\delta_k}\right)h_i c = g\left(\frac{\gamma_j\gamma_k+\gamma_k\gamma_j}{2}\right)rh_i c. \quad (61)$$

APPENDIX V
CONSTRAINED OPTIC FLOW FOR DEFORMABLE 3D OBJECTS

In this section we derive the constrained optic flow algorithm, which is essentially equivalent to the tracking algorithms presented in [A7], [A5]. The goal of constrained optic flow is to find the value of the pose parameters, u_t, that minimizes the sum of squared differences between the appearance of each vertex at frame t and the appearance of the same vertex at frame $t-1$. In other words, the goal is to find $\hat{u}_t$, the value of u_t that minimizes $\rho(u_t)$:

$$\hat{u}_t = \underset{u_t}{\operatorname{argmin}}\,\rho(u_t), \quad (62)$$

where

$$\rho(u_t) = \frac{1}{2}\sum_{i=1}^{n}\left[y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\right]^2 \quad (63)$$

We call this algorithm *constrained optic flow*, because the vertex locations in the image are constrained by a global model (the 3D morphable model). In the special case in which the $x_i(u_t)$ are neighboring points that move with the same 2D displacement, constrained optic flow reduces to the standard Lucas-Kanade optic flow algorithm [A8], [A9].

We will minimize the sum of squares in (63) using the Gauss-Newton method. First rewrite $\rho(u_t)$ as follows:

$$\rho(u_t) = \frac{1}{2}\mathbf{f}^T\mathbf{f} = \frac{1}{2}\sum_{i=1}^{n} f_i^2 \quad (64)$$

where the vector $\mathbf{f}$,

$$\mathbf{f} = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}^T, \quad (65)$$

is analogous to the difference image in standard optic flow. Each term f_i is the difference in texture values between the pixel rendered by vertex i at time t and the pixel rendered by the same vertex at time $t-1$:

$$f_i = y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big), \quad (66)$$

Before giving the Gauss-Newton update rules, we first calculate the derivatives of $\mathbf{f}$ with respect to each of the motion parameters in u_t.

A. Derivatives with respect to translation

Let $J_l(u_t)$ be the Jacobian matrix (matrix of first derivatives) of $\mathbf{f}$ with respect to the 2×1 translation vector, l:

$$J_l(u_t) = \frac{\partial\mathbf{f}^T}{\partial l}(u_t) = \begin{bmatrix}\frac{\partial f_1}{\partial l}(u_t) & \frac{\partial f_2}{\partial l}(u_t) & \cdots & \frac{\partial f_n}{\partial l}(u_t)\end{bmatrix}. \quad (67)$$

The ith column in this Jacobian matrix is

$$\frac{\partial f_i}{\partial l}(u_t) = \left(\frac{\partial x_i(u_t)^T}{\partial l}\right)\frac{\partial y_t}{\partial x_i(u_t)} \quad (68)$$

$$= \left(\frac{\partial(grh_i c + l)^T}{\partial l}\right)\frac{\partial y_t}{\partial x}\big(x_i(u_t)\big) \quad (69)$$

$$= I_2\,\frac{\partial y_t}{\partial x}\big(x_i(u_t)\big) \quad (70)$$

$$= \nabla y_t\big(x_i(u_t)\big), \quad (71)$$

where we define the 2×1 vector $\nabla y_t(x)$ as the spatial gradient of the observed image y_t at pixel location x. In practice, we blur the image y_t before computing ∇y_t to ensure that this spatial gradient is well-behaved.

The following two products will be needed later for the Gauss-Newton update rule for translation (87):

$$J_l(u_t)[J_l(u_t)]^T = \sum_{i=1}^{n}\left[\nabla y_t\big(x_i(u_t)\big)\right]\left[\nabla y_t\big(x_i(u_t)\big)\right]^T, \quad (72)$$

$$J_l(u_t)\mathbf{f}(u_t) = \sum_{i=1}^{n}\left[\nabla y_t\big(x_i(u_t)\big)\right]\left[y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\right]. \quad (73)$$

B. Derivatives with respect to morph coefficients

Let $J_c(u_t)$ be the Jacobian matrix of $\mathbf{f}$ with respect to c, the $k\times 1$ vector of morph coefficients:

$$J_c(u_t) = \frac{\partial\mathbf{f}^T}{\partial c}(u_t) = \begin{bmatrix}\frac{\partial f_1}{\partial c}(u_t) & \frac{\partial f_2}{\partial c}(u_t) & \cdots & \frac{\partial f_n}{\partial c}(u_t)\end{bmatrix}. \quad (74)$$

The ith column in this Jacobian matrix is

$$\frac{\partial f_i}{\partial c}(u_t) = \left(\frac{\partial x_i(u_t)^T}{\partial c}\right)\frac{\partial y_t}{\partial x_i(u_t)} \quad (75)$$

$$= \left(\frac{\partial(grh_i c + l)^T}{\partial c}\right)\nabla y_t\big(x_i(u_t)\big) \quad (76)$$

$$= (grh_i)^T\,\nabla y_t\big(x_i(u_t)\big). \quad (77)$$

The following two products will be needed later for the Gauss-Newton update rule for morph coefficients (88):

$$J_c(u_t)[J_c(u_t)]^T = \sum_{i=1}^{n}(grh_i)^T\left[\nabla y_t\big(x_i(u_t)\big)\right]\left[\nabla y_t\big(x_i(u_t)\big)\right]^T grh_i, \quad (78)$$

$$J_c(u_t)\mathbf{f}(u_t) = \sum_{i=1}^{n}(grh_i)^T\left[\nabla y_t\big(x_i(u_t)\big)\right]\left[y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\right]. \quad (79)$$

C. Derivatives with respect to rotation

Let $J_\delta(u_t)$ be the Jacobian matrix of $\mathbf{f}$ with respect to δ, the 3×1 vector of rotation parameters (see Appendix IV for more details on exponential coordinates for rotation):

$$J_\delta(u_t) = \frac{\partial\mathbf{f}^T}{\partial\delta}(u_t) = \begin{bmatrix}\frac{\partial f_1}{\partial\delta}(u_t) & \frac{\partial f_2}{\partial\delta}(u_t) & \cdots & \frac{\partial f_n}{\partial\delta}(u_t)\end{bmatrix} \quad (80)$$

The ith column in this Jacobian matrix is

$$\frac{\partial f_i}{\partial\delta}(u_t) = \left(\frac{\partial x_i(u_t)^T}{\partial\delta}\right)\frac{\partial y_t}{\partial x_i(u_t)} \quad (81)$$

$$= \begin{bmatrix}\left(\frac{\partial x_i(u_t)}{\partial\delta_1}\right)^T \\ \left(\frac{\partial x_i(u_t)}{\partial\delta_2}\right)^T \\ \left(\frac{\partial x_i(u_t)}{\partial\delta_3}\right)^T\end{bmatrix}\nabla y_t\big(x_i(u_t)\big). \quad (82)$$

We then use (56) to get:

$$\frac{\partial f_i}{\partial\delta}(u_t) = \begin{bmatrix}g\gamma_1 rh_i c & g\gamma_2 rh_i c & g\gamma_3 rh_i c\end{bmatrix}^T\nabla y_t\big(x_i(u_t)\big) = \begin{bmatrix}(g\gamma_1 rh_i c)^T \\ (g\gamma_2 rh_i c)^T \\ (g\gamma_3 rh_i c)^T\end{bmatrix}\nabla y_t\big(x_i(u_t)\big) \quad (83)$$

The following two products will be needed later for the Gauss-Newton update rule for rotation (89):

$$J_\delta(u_t)[J_\delta(u_t)]^T = \sum_{i=1}^{n} \begin{bmatrix} (g\gamma_1 r h_i c)^T \\ (g\gamma_2 r h_i c)^T \\ (g\gamma_3 r h_i c)^T \end{bmatrix} \big[\nabla y_t\big(x_i(u_t)\big)\big] \big[\nabla y_t\big(x_i(u_t)\big)\big]^T \begin{bmatrix} (g\gamma_1 r h_i c)^T \\ (g\gamma_2 r h_i c)^T \\ (g\gamma_3 r h_i c)^T \end{bmatrix}^T, \tag{84}$$

$$J_\delta(u_t)\mathbf{f}(u_t) = \sum_{i=1}^{n} \begin{bmatrix} (g\gamma_1 r h_i c)^T \\ (g\gamma_2 r h_i c)^T \\ (g\gamma_3 r h_i c)^T \end{bmatrix} \big[\nabla y_t\big(x_i(u_t)\big)\big] \big[y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\big]. \tag{85}$$

D. The Gauss-Newton update rules

As each new video frame, y_t, arrives, the goal of the constrained optic flow algorithm is to find $\hat{u}_t$, the value of the pose parameters, u_t, that minimizes the sum in (63). The algorithm begins with an initial guess for u_t, namely $u_t^0 = \{r_t^0, l_t^0, c_t^0\}$. This initial guess may be set equal to the pose of the previous frame: $u_t^0 = u_{t-1}$. Alternatively, this initial guess may be extrapolated from the poses of the previous two frames, $u_{t-1} = \{r_{t-1}, l_{t-1}, c_{t-1}\}$ and $u_{t-2} = \{r_{t-2}, l_{t-2}, c_{t-2}\}$, in a manner similar to the following:

$$\begin{aligned} l_t^0 &= l_{t-1} + \big(l_{t-1} - l_{t-2}\big) \\ r_t^0 &= r_{t-1}\big[r_{t-1}(r_{t-2})^{-1}\big] \\ c_t^0 &= c_{t-1} \end{aligned} \tag{86}$$

Once the initial guess is determined, we use the Gauss-Newton method to minimize the sum in (63), by iterating the following update rules. The following update rules tell us how to perform step s of our iterative estimation of u_t. They are used to get from the estimate $u_t^s = \{r_t^s, l_t^s, c_t^s\}$ to the next estimate, $u_t^{s+1} = \{r_t^{s+1}, l_t^{s+1}, c_t^{s+1}\}$.

$$l_t^{s+1} = l_t^s - \Big[J_l(u_t^s)\big(J_l(u_t^s)\big)^T\Big]^{-1} J_l(u_t^s)\mathbf{f}(u_t^s) \tag{87}$$

$$c_t^{s+1} = c_t^s - \Big[J_c(u_t^s)\big(J_c(u_t^s)\big)^T\Big]^{-1} J_c(u_t^s)\mathbf{f}(u_t^s) \tag{88}$$

$$\begin{aligned} r_t^{s+1} &= e^{\Delta} r_t^s, \quad \text{where} \\ \delta &= -\Big[J_\delta(u_t^s)\big(J_\delta(u_t^s)\big)^T\Big]^{-1} J_\delta(u_t^s)\mathbf{f}(u_t^s), \end{aligned} \tag{89}$$

and Δ is defined from δ using (52). The rest of the terms in these update rules are written out in full in Equations (72–73), (78–79), and (84–85).

Each of the update rules includes the term $\nabla y_t\big(x_i(u_t^s)\big)$ as part of the Jacobian matrix and also includes the term $f_i(u_t^s) = \big[y_t\big(x_i(u_t^s)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\big]$. Evaluating these terms involves obtaining patches from the current image and its gradient images, which can be computationally expensive. As a result, we typically update the values of $\nabla y_t\big(x_i(u_t^s)\big)$ and $y_t\big(x_i(u_t^s)\big)$ only once every few iterations of the update rules (87–89). In practice, we only need to update the values of $\nabla y_t\big(x_i(u_t^s)\big)$ and $y_t\big(x_i(u_t^s)\big)$ twice for each frame of video.

APPENDIX VI
THE PREDICTIVE DISTRIBUTION FOR Y_t

We now tackle the predictive distribution for the current image, y_t, given the past and current poses of the object, $u_{1:t}$, and the past images, $y_{1:t-1}$:

$$p(y_t \mid u_{1:t}, y_{1:t-1}). \tag{90}$$

First note that from the conditional independence relationships defined by the graphical model (Figure 1, right),

$$p\left(\begin{bmatrix} v_t \\ b_t \end{bmatrix} \,\middle|\, u_{1:t}, y_{1:t-1}\right) = p\left(\begin{bmatrix} v_t \\ b_t \end{bmatrix} \,\middle|\, u_{1:t-1}, y_{1:t-1}\right) \tag{91}$$

$$= \mathcal{N}\left(\begin{bmatrix} v_t \\ b_t \end{bmatrix} ; \begin{bmatrix} \hat{v}_{t|t-1} \\ \hat{b}_{t|t-1} \end{bmatrix}, \begin{bmatrix} \mathcal{V}_{t|t-1} & \mathbf{0} \\ \mathbf{0} & \mathcal{B}_{t|t-1} \end{bmatrix}\right) \tag{92}$$

where (92) follows from definitions (6) and (7).

From these equations and (3), we can derive the mean and variance of the predictive distribution, which is the Gaussian distribution $p(y_t \mid u_{1:t}, y_{1:t-1})$. First the mean:

$$\begin{aligned} E(Y_t \mid u_{1:t} y_{1:t-1}) &= E\left(a(u_t)\begin{bmatrix} V_t \\ B_t \end{bmatrix} + W_t \,\middle|\, u_{1:t}, y_{1:t-1}\right) \\ &= a(u_t) E\left(\begin{bmatrix} V_t \\ B_t \end{bmatrix} \,\middle|\, u_{1:t}, y_{1:t-1}\right) + E(W_t) \\ &= a(u_t)\begin{bmatrix} \hat{v}_{t|t-1} \\ \hat{b}_{t|t-1} \end{bmatrix} \qquad (93) \\ &= a_v(u_t)\hat{v}_{t|t-1} + a_b(u_t)\hat{b}_{t|t-1}, \qquad (94) \end{aligned}$$

where (93) follows from (92), and (94) follows from the definition of the projection matrix $a(u_t)$ in Section 2.2. Equation (94) merely asserts that the expected value of each image pixel is the value of the texel in the foreground or background texture template that (according to u_t) renders that pixel.

Now we compute the variance of the predictive distribution:

$$\begin{aligned} Var(Y_t \mid u_{1:t} y_{1:t-1}) &= Var\left(a(u_t)\begin{bmatrix} V_t \\ B_t \end{bmatrix} + W_t \,\middle|\, u_{1:t}, y_{1:t-1}\right) \\ &= a(u_t) Var\left(\begin{bmatrix} V_t \\ B_t \end{bmatrix} \,\middle|\, u_{1:t}, y_{1:t-1}\right) a(u_t)^T + Var(W_t) \\ &= a(u_t)\begin{bmatrix} \mathcal{V}_{t|t-1} & \mathbf{0} \\ \mathbf{0} & \mathcal{B}_{t|t-1} \end{bmatrix} a(u_t)^T + \sigma_w^2 I_m \qquad (95) \\ &= a_v(u_t)\mathcal{V}_{t|t-1} a_v(u_t)^T + a_b(u_t)\mathcal{B}_{t|t-1} a_b(u_t)^T + \sigma_w^2 I_m, \qquad (96) \end{aligned}$$

where (95) follows from (92), and (96) follows from the definition of the projection matrix $a(u_t)$. Equation (96) simply asserts that the variance of each image pixel is σ_w^2 plus the variance of the texel in the foreground or background texture map that (according to u_t) renders that pixel.

Since the covariance matrices are all diagonal, each pixel is rendered by an independent Gaussian. Thus, we can consider the mean and variance of each pixel independently. If an image pixel is (according to u_t) rendered by foreground texel i, then by (94) and (96), the pixel value will have the following normal distribution:

Foreground pixel $x_i(u_t)$:

$$p\big(y_t(x_i(u_t)) | u_{1:t}, y_{1:t-1}\big) = \mathcal{N}\big(y_t(x_i(u_t)); \hat{v}_{t|t-1}(i), \mathcal{V}_{t|t-1}(i) + \sigma_w^2\big). \tag{97}$$

On the other hand, if an image pixel is (according to u_t) rendered by background texel j, then by (94) and (96), the pixel value will have the following normal distribution:

Background pixel j :

$$p\big(y_t(j) \mid u_{1:t}, y_{1:t-1}\big) = \mathcal{N}\big(y_t(j); \hat{b}_{t|t-1}(j), \mathcal{B}_{t|t-1}(j) + \sigma_w^2\big). \tag{98}$$

Taking the product of the probability distributions of all of the image pixels from (97) and (98) yields the probability distribution for the entire image, i.e., the predictive distribution. Taking the log of this product yields an expression for the log of the predictive

distribution:

$$\log p(y_t \mid u_{1:t}, y_{1:t-1}) = -\frac{m}{2} log\, 2\pi \tag{99}$$
$$- \frac{1}{2}\sum_{i=1}^{n}\left[\log\big(\mathcal{V}_{t|t-1}(i)+\sigma_w^2\big) + \frac{[y_t(x_i(u_t)) - \hat{v}_{t|t-1}(i)]^2}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2}\right]$$
$$- \frac{1}{2}\sum_{j\notin\mathcal{X}(u_t)}\left[\log\big(\mathcal{B}_{t|t-1}(j)+\sigma_w^2\big) + \frac{[y_t(j) - \hat{b}_{t|t-1}(j)]^2}{\mathcal{B}_{t|t-1}(j)+\sigma_w^2}\right],$$

where $\mathcal{X}(u_t)$ is the set of all image pixels rendered by the object under pose u_t. The first sum in (99) corresponds to the foreground pixels, while the second sum in (99) corresponds to the background pixels.

Substituting (99) into (16) and eliminating the terms that don't depend on u_t yields an expression for the peak of the pose opinion distribution:

$$\hat{u}_t(u_{1:t-1}) = \operatorname*{argmax}_{u_t} p(u_t \mid u_{t-1}) p(y_t \mid u_{1:t} y_{1:t-1})$$
$$= \operatorname*{argmin}_{u_t}\Bigg(-\log p(u_t \mid u_{t-1}) \tag{100}$$
$$+ \frac{1}{2}\sum_{i=1}^{n}\Big[\frac{[y_t(x_i(u_t)) - \hat{v}_{t|t-1}(i)]^2}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2} + \log[\mathcal{V}_{t|t-1}\big(x_i(u_t)\big)+\sigma_w^2]\Big]$$
$$+ \frac{1}{2}\sum_{j\notin\mathcal{X}(u_t)}\Big[\frac{[y_t(j) - \hat{b}_{t|t-1}(j)]^2}{\mathcal{B}_{t|t-1}(j)+\sigma_w^2} + \log[\mathcal{B}_{t|t-1}(j)+\sigma_w^2]\Big]\Bigg).$$

We can subtract any constant that does not depend on u_t from the expression inside the large parentheses above without changing the value of the argmin function. We are thus justified in subtracting $\frac{1}{2}\sum_{j=1}^{m}\Big[\frac{[y_t(j)-\hat{b}_{t|t-1}(j)]^2}{\mathcal{B}_{t|t-1}(j)+\sigma_w^2} + \log[\mathcal{B}_{t|t-1}(j)+\sigma_w^2]\Big]$ from the expression inside the large parentheses, which converts the sums over both foreground and background pixels into a sum over only foreground pixels:

$$\hat{u}_t(u_{1:t-1}) = \operatorname*{argmin}_{u_t}\Bigg(-\log p(u_t \mid u_{t-1}) \tag{101}$$
$$+ \frac{1}{2}\sum_{i=1}^{n}\overbrace{\Big[\frac{[y_t(x_i(u_t)) - \hat{v}_{t|t-1}(i)]^2}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2} + \log[\mathcal{V}_{t|t-1}(x_i(u_t))+\sigma_w^2]}^{\text{Foreground terms}}$$
$$\underbrace{- \frac{[y_t(x_i(u_t)) - \hat{b}_{t|t-1}(x_i(u_t))]^2}{\mathcal{B}_{t|t-1}\big(x_i(u_t)\big)+\sigma_w^2} - \log[\mathcal{B}_{t|t-1}\big(x_i(u_t)\big)+\sigma_w^2]\Big]}_{\text{Background terms}}\Bigg).$$

APPENDIX VII
ESTIMATING THE POSE OPINION DISTRIBUTION

In this appendix, we describe how to obtain a Gaussian estimate of an expert's pose opinion distribution, which we use as the proposal distribution for importance sampling. In the first section, Appendix VII-A, we describe how to find the peak of this proposal distribution. Then, in Appendix VII-B, we explain how to find the covariance matrix of this proposal distribution. Finally, in Appendix VII-C, we describe how we sample from this proposal distribution.

A. Estimating the peak of the pose opinion

We now explain how to estimate $\hat{u}_t(u_{1:t-1})$, the peak of an expert's pose opinion distribution, from the G-flow objective function (18). To estimate the peak of the pose opinion, we employ the Gauss-Newton method in a manner quite similar to constrained optic flow (Appendix V). In practice, we ignore the two background terms in (18), which is essentially equivalent to assuming a white noise background, and additionally ignore the second foreground term in (18) (this is one of the assumptions that is implicitly made by optic flow—see Section 4.2). In other words, we use Gauss-Newton to efficiently find the pose u_t that minimizes the sum over the first foreground (object) term in (18), and use that as our estimate $\hat{u}_t(u_{1:t-1})$:

$$\hat{u}_t(u_{1:t-1}) = \operatorname*{argmin}_{u_t} \rho_{\text{obj}}(u_t), \tag{102}$$

where

$$\rho_{\text{obj}}(u_t) = \frac{1}{2}\sum_{i=1}^{n}\frac{[y_t(x_i(u_t)) - \hat{v}_{t|t-1}(i)]^2}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2}, \tag{103}$$

the sum over the first foreground term in (18). It would also be possible to minimize the sum over the background terms using Gauss-Newton, and then combine that pose estimate with the estimate obtained from the foreground term, though Gauss-Newton cannot be used to minimize them both simultaneously (together, they are a difference—not a sum—of squares).

The Gauss-Newton method to iteratively minimize (103) is virtually identical to constrained optic flow, as derived in Appendix V, except that Equation (66) is replaced by:

$$f_i = y_t\big(x_i(u_t)\big) - \hat{v}_{t|t-1}(i). \tag{104}$$

As a result, the Gauss-Newton update rules (87–89) remain unchanged except that in each of Equations (72, 73, 78, 79, 84, and 85), the expression in the sum is divided by $\mathcal{V}_{t|t-1}(i)+\sigma_w^2$. Thus, Equations (72) and (73) become:

$$J_l(u_t)[J_l(u_t)]^T = \sum_{i=1}^{n}\frac{\big[\nabla y_t\big(x_i(u_t)\big)\big]\big[\nabla y_t\big(x_i(u_t)\big)\big]^T}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2} \tag{105}$$

$$J_l(u_t)\mathbf{f}(u_t) = \sum_{i=1}^{n}\frac{\big[\nabla y_t\big(x_i(u_t)\big)\big]\big[y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\big]}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2}, \tag{106}$$

Equations (78) and (79) become:

$$J_c(u_t)[J_c(u_t)]^T = \sum_{i=1}^{n}\frac{(grh_i)^T\big[\nabla y_t\big(x_i(u_t)\big)\big]\big[\nabla y_t\big(x_i(u_t)\big)\big]^T grh_i}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2} \tag{107}$$

$$J_c(u_t)\mathbf{f}(u_t) = \sum_{i=1}^{n}\frac{(grh_i)^T\big[\nabla y_t\big(x_i(u_t)\big)\big]\big[y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\big]}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2}, \tag{108}$$

and Equations (84) and (85) become:

$$J_\delta(u_t)[J_\delta(u_t)]^T = \tag{109}$$
$$\sum_{i=1}^{n}\frac{\begin{bmatrix}(g\gamma_1 rh_i c)^T\\(g\gamma_2 rh_i c)^T\\(g\gamma_3 rh_i c)^T\end{bmatrix}\big[\nabla y_t\big(x_i(u_t)\big)\big]\big[\nabla y_t\big(x_i(u_t)\big)\big]^T\begin{bmatrix}(g\gamma_1 rh_i c)^T\\(g\gamma_2 rh_i c)^T\\(g\gamma_3 rh_i c)^T\end{bmatrix}^T}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2}$$

$$J_\delta(u_t)\mathbf{f}(u_t) = \tag{110}$$
$$\sum_{i=1}^{n}\frac{\begin{bmatrix}(g\gamma_1 rh_i c)^T\\(g\gamma_2 rh_i c)^T\\(g\gamma_3 rh_i c)^T\end{bmatrix}\big[\nabla y_t\big(x_i(u_t)\big)\big]\big[y_t\big(x_i(u_t)\big) - y_{t-1}\big(x_i(u_{t-1})\big)\big]}{\mathcal{V}_{t|t-1}(i)+\sigma_w^2}.$$

B. Gaussian estimate of the pose opinion distribution

For each expert $u_{1:t-1}^{(d)}$, we compute the peak of the pose distribution at time t according to that expert, $\hat{u}_t(u_{1:t-1}^{(d)}) = \operatorname*{argmin}_{u_t}\rho_{\text{obj}}(u_t)$,

using the method explained in Appendix VII-A. The Laplace estimate of the covariance matrix of the expert's pose opinion is the inverse of the Hessian matrix (second derivatives) of (103) evaluated at this peak. For simplicity, we take the Hessian matrix with respect to each of the pose parameters independently (neglecting cross-derivatives $\frac{\partial^2 \rho}{\partial c \partial \delta}$).

1) Hessian matrix of ρ_{obj} with respect to l and c: The Hessian with respect to translation and the Hessian with respect to morph coefficients are equal to their Gauss-Newton approximations from (105) and (107):

$$\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial l^2} = J_l(u_t)[J_l(u_t)]^T \tag{111}$$

$$\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial c^2} = J_c(u_t)[J_c(u_t)]^T \tag{112}$$

2) Hessian matrix of ρ_{obj} with respect to δ: Because the rotation, r, is not a linear function of the rotation parameters, δ, the Hessian with respect to δ is not equal to its Gauss-Newton approximation from (109). Instead, we actually compute the full Hessian with respect to δ, using the method we introduced in Appendix IV.

The Hessian with respect to the rotation parameters, δ, is:

$$\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial \delta^2} = J_\delta(u_t)[J_\delta(u_t)]^T + \sum_{i=1}^{n} f_i(u_t) \frac{\partial^2 f_i(u_t)}{\partial \delta^2}. \tag{113}$$

The first term on the right side of (113) is just the Gauss-Newton approximation to $\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial \delta^2}$ from (109), and the second term on the right side of (113) can be thought of as a correction term to that Gauss-Newton approximation. We examine the latter term, the Hessian matrix of $f_i(u_t)$ with respect to δ, one element at a time:

$$\frac{\partial^2 f_i(u_t)}{\partial \delta_j \partial \delta_k} \tag{114}$$
$$= \left(\frac{\partial^2 x_i(u_t)}{\partial \delta_j \partial \delta_k}\right)^T \nabla y_t\big(x_i(u_t)\big) + \left(\frac{\partial x_i(u_t)}{\partial \delta_j}\right)^T \mathcal{H}_{y_t} \left(\frac{\partial x_i(u_t)}{\partial \delta_k}\right)$$

where $\mathcal{H}_{y_t}$ is the Hessian of the current image, y_t, with respect to spatial coordinates, evaluated at $x_i(u_t)$. Since the image Hessian, $\mathcal{H}_{y_t}$, can be sensitive to noise and/or computationally expensive, we replace it by its Gauss-Newton approximation:

$$\mathcal{H}_{y_t} \stackrel{\text{def}}{=} \frac{\partial^2 y_t}{\partial x^2}\big(x_i(u_t)\big) \approx \big[\nabla y_t\big(x_i(u_t)\big)\big]\big[\nabla y_t\big(x_i(u_t)\big)\big]^T \tag{115}$$

Now using the exponential rotation derivatives from (60) and (61), we have the following formula for element (j,k) of the Hessian matrix of ρ_{obj} with respect to the rotation parameters:

$$\begin{aligned}\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial \delta_j \partial \delta_k} &\approx \Big[J_\delta(u_t)[J_\delta(u_t)]^T\Big]_{jk} \\ &+ \sum_{i=1}^{n} f_i(u_t) \left[\left(g\left(\frac{\gamma_j \gamma_k + \gamma_k \gamma_j}{2}\right) r h_i c\right)^T \nabla y_t\big(x_i(u_t)\big)\right] \\ &+ \sum_{i=1}^{n} f_i(u_t) \Big[(g\gamma_j r h_i c)^T \big[\nabla y_t\big(x_i(u_t)\big)\big]\big[\nabla y_t\big(x_i(u_t)\big)\big]^T (g\gamma_k r h_i c)\Big].\end{aligned} \tag{116}$$

By rewriting the right side of this equation, we obtain:

$$\begin{aligned}\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial \delta_j \partial \delta_k} &\approx \Big[J_\delta(u_t)[J_\delta(u_t)]^T\Big]_{jk} + \Big[J_\delta(u_t)\mathcal{F}(u_t)[J_\delta(u_t)]^T\Big]_{jk} \\ &+ \sum_{i=1}^{n} f_i(u_t) \left[\left(g\left(\frac{\gamma_j \gamma_k + \gamma_k \gamma_j}{2}\right) r h_i c\right)^T \nabla y_t\big(x_i(u_t)\big)\right],\end{aligned} \tag{117}$$

where

$$\mathcal{F}(u_t) \stackrel{\text{def}}{=} \text{diag}\big(\mathbf{f}(u_t)\big).$$

C. Sampling from the proposal distribution

The Laplace approximation to the covariance matrices of each of l, c, and δ is the inverse of the corresponding Hessian matrix of ρ_{obj} from (111), (112), or (117):

$$\begin{aligned}\mathcal{U}_l(u_t) &= \left[\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial l^2}\right]^{-1} \\ \mathcal{U}_c(u_t) &= \left[\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial c^2}\right]^{-1} \\ \mathcal{U}_\delta(u_t) &= \left[\frac{\partial^2 \rho_{\text{obj}}(u_t)}{\partial \delta^2}\right]^{-1}.\end{aligned} \tag{118}$$

We generate a set of λ independent samples,

$$u_t^{(d,e)} = \{r^{(d,e)}, l^{(d,e)}, c^{(d,e)}\}, \tag{119}$$

where $e \in \{1, 2, \cdots, \lambda\}$, and the $u_t^{(d,e)}$ are drawn from a Gaussian distribution with mean $\hat{u}_t(u_{1:t-1}^{(d)}) = \{r^{(d)}, l^{(d)}, c^{(d)}\}$ and variance proportional to the Laplace covariance matrices (118):

$$\begin{aligned} l^{(d,e)} &\text{ is sampled from } \quad N\Big(l^{(d)},\ \alpha \mathcal{U}_l\big(\hat{u}_t(u_{1:t-1}^{(d)})\big)\Big), \\ c^{(d,e)} &\text{ is sampled from } \quad N\Big(c^{(d)},\ \alpha \mathcal{U}_c\big(\hat{u}_t(u_{1:t-1}^{(d)})\big)\Big), \\ r^{(d,e)} &= e^{\Delta}\, r^{(d)}, \\ \text{where } \quad \delta &\text{ is sampled from } \quad N\Big(0,\ \alpha \mathcal{U}_\delta\big(\hat{u}_t(u_{1:t-1}^{(d)})\big)\Big), \end{aligned} \tag{120}$$

and Δ is defined from δ using (52). The parameter $\alpha > 0$ determines the sharpness of the sampling distribution. (Note that letting $\alpha \to 0$ would be equivalent to simply setting the new pose equal to the peak of the pose opinion, $u_t^{(d,e)} = \hat{u}_t(u_{1:t-1}^{(d)})$.)

In Section 3, we used the single pose sample $u_t^{(d,e)}$ as shorthand for samples of all three pose parameters, and we used the sampling covariance $\mathcal{U}$ as shorthand for the sampling covariances of all three pose parameters:

$$u_t^{(d,e)} \stackrel{\text{def}}{=} \big\{r^{(d,e)}, l^{(d,e)}, c^{(d,e)}\big\}, \tag{121}$$

$$\begin{aligned}&\mathcal{U}\big(\hat{u}_t(u_{1:t-1}^{(d)})\big) \stackrel{\text{def}}{=} \\ &\qquad \Big\{\mathcal{U}_l\big(\hat{u}_t(u_{1:t-1}^{(d)})\big),\ \mathcal{U}_c\big(\hat{u}_t(u_{1:t-1}^{(d)})\big),\ \mathcal{U}_\delta\big(\hat{u}_t(u_{1:t-1}^{(d)})\big)\Big\}.\end{aligned} \tag{122}$$

We used a similar shorthand when we discussed sampling from the distribution in (19). For example, the statement

$$\begin{aligned}&u_t^{(d,e)} \text{ is sampled from} \\ &\qquad \pi(u_t \,|\, u_{1:t-1}^{(d)}, y_{1:t}) = N\Big(\hat{u}_t(u_{1:t-1}^{(d)}),\ \alpha \mathcal{U}\big(\hat{u}_t(u_{1:t-1}^{(d)})\big)\Big)\end{aligned}$$

is merely shorthand for all of the statements in (120).

REFERENCES

[A1] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

[A2] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, no. 1–2, pp. 5–43, 2003.

[A3] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[A4] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1989.

[A5] M. Brand and R. Bhotika, "Flexible flow for 3D nonrigid tracking and shape recovery," in *CVPR*, 2001.

[A6] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton: CRC Press, 1994.

[A7] L. Torresani, D. Yang, G. Alexander, and C. Bregler, "Tracking and modeling non-rigid objects with rank constraints," in *CVPR*, 2001, pp. 493–500.

[A8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," in *Proc. 7th Int. Joint Conf. Artificial Intelligence (IJCAI 81)*, 1981, pp. 674–679.

[A9] 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.