Particle Video: Long-Range Motion Estimation using Point Trajectories

Peter Sand and Seth Teller
MIT Computer Science and Artificial Intelligence Laboratory
{sand,teller}@csail.mit.edu

Abstract

This paper describes a new approach to motion estimation in video. We represent video motion using a set of particles. Each particle is an image point sample with a long-duration trajectory and other properties. To optimize these particles, we measure point-based matching along the particle trajectories and distortion between the particles. The resulting motion representation is useful for a variety of applications and cannot be directly obtained using existing methods such as optical flow or feature tracking. We demonstrate the algorithm on challenging real-world videos that include complex scene geometry, multiple types of occlusion, regions with low texture, and non-rigid deformations.

1. Introduction

Video motion estimation is often performed using feature tracking [12] or optical flow [3]. Feature tracking follows a sparse set of salient image points over many frames, whereas optical flow estimates a dense motion field from one frame to the next. Our goal is to combine these two approaches: to produce motion estimates that are both spatially dense and temporally long-range (Figure 1). For any image point, we would like to know where the corresponding scene point appears in all other video frames (until the point leaves the field of view or becomes occluded).

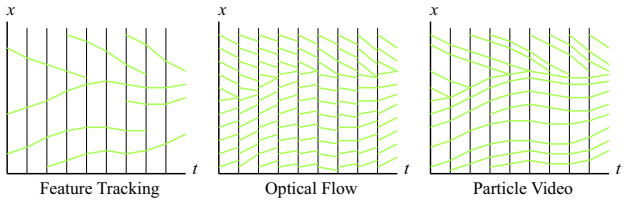

Figure 1. Each diagram represents point correspondences between frames of a hypothetical sequence. Feature tracking is long-range but sparse. Optical flow is dense but short-range. Our particle video representation is dense and long-range.

This form of motion estimation is useful for a variety of applications. Multiple observations of each scene point can be combined for super-resolution, noise removal, segmentation, and increased effective dynamic range. The correspondences can also improve the temporal coherence of image filters that operate independently on each frame. Additionally, long-range motion estimation can simplify interactive video manipulation, including matting, rotoscoping, and object removal.

1.1. Particle Video Representation

Our approach represents video motion using a set of particles that move through time. Each particle denotes an interpolated image point sample, in contrast to a feature patch that represents a neighborhood of pixels [12]. Particle density is adaptive, so that the algorithm can model detailed motion with substantially fewer particles than pixels.

The algorithm optimizes particle trajectories using an objective function that incorporates point-based image matching, inter-particle distortion, and frame-to-frame optical flow. The algorithm also extends and truncates particle trajectories to model motion near occlusion boundaries.

Our contributions include posing the particle video problem, defining the particle video representation, and presenting an algorithm for building particle videos. We provide a new motion optimization scheme that combines variational techniques with an adaptive motion representation. The algorithm uses weighted links between particles to implicitly represent grouping, providing an alternative to discrete layer-based representations.

1.2. Design Goals

Our primary goal is the ability to model complex motion and occlusion. We want the algorithm to handle general video, which may include close-ups of people talking, hand-held camera motion, multiple independently moving objects, textureless regions, narrow fields of view, and complicated geometry (e.g. trees or office clutter).

A particle approach provides this kind of flexibility. Par-

ticles can represent complicated geometry and motion because they are small; a particle's appearance will not change as rapidly as the appearance of a large feature patch, and it is less likely to straddle an occlusion boundary. Particles represent motion in a non-parametric manner; they do not assume that the scene consists of planar or rigid components.

Any flexible system needs to be augmented with constraints, which motivates another design decision: consistency is more important than correctness. If the scene includes arbitrary deforming objects with inadequate texture, finding the true motion may be hopeless. Typically, this problem is addressed with geometric assumptions about scene rigidity and camera motion. Instead, we simply strive for consistency; for example, that red pixels from one frame are matched with red pixels in another frame. For many applications, this kind of consistency is sufficient (and certainly more useful than outright failure due to non-uniqueness).

This flexibility in modelling complex motion can also be achieved by optical flow, but the optical flow representation is best suited to successive pairs of frames, not to long sequences. Frame-to-frame flow fields can be concatenated to obtain longer-range correspondences, but the resulting multi-frame flow must be refined at each step to avoid drift.

1.3. Related Work

Much of our work focuses on handling occlusion. Occlusion modelling is a difficult part of optical flow, stereo, tracking, and motion estimation in general. Thompson [15] describes issues caused by occlusion, including mixed pixels and systematic boundary localization errors. Zitnick *et al.* [19] estimate optical flow using correspondences between segmented image regions, explicitly modelling mixed pixels at occlusion boundaries. Amiaz and Kiryati [2] use level sets to refine variational motion boundaries.

Because occluded pixels violate a major assumption of optical flow (that each pixel goes somewhere), several methods attempt to identify occluded pixels explicitly. Silva and Victor [13] use a pixel dissimilarity measure to detect brightness values that appear or disappear over time. Alvarez *et al.* [1] simultaneously compute forward and reverse flow fields, labelling pixels as occluded where the two disagree. Strecha *et al.* [14] treat occlusion labels as hidden variables in an EM optimization. Xiao *et al.* [18] regularize flow estimates using a bilateral filter that incorporates flow from neighboring pixels that are similar in motion and appearance and that lie outside occluded regions.

Many algorithms propagate optical flow estimates from one frame to the next using a temporal smoothness assumption [4, 7, 11]. Brox *et al.* [6, 10] estimate optical flow simultaneously over multiple frames using an objective function that provides robust spatiotemporal regularization.

Other long-range optical flow algorithms do not assume temporal smoothness. Wills and Belongie [17] find dense correspondences (between widely-separated image pairs) using a layered representation initialized with sparse feature correspondences. Irani [8] describes linear subspace constraints for flow across multiple frames. Brand [5] applies a similar approach to non-rigid scenes. These rank-based methods assume less temporal smoothness than other methods, but are not designed to handle large-scale occlusions.

The primary difference between our algorithm and existing optical flow algorithms is that we represent motion in a scene-centric fashion, rather than as a vector field from each frame to the next. Simple concatenation of frame-to-frame flow fields will cause an accumulation of error. To solve this, we refine motion estimates in order to enforce long-range consistency.

2. Variational Optical Flow with Occlusion

Our particle video algorithm uses frame-to-frame optical flow to help guide the particles. The algorithm treats flow estimation as a black box that can be replaced with an alternate flow algorithm. Rather than assuming temporal smoothness, we estimate optical flow independently for each frame; this enables the algorithm to perform well on hand-held video with moving objects.

We use a combination of the optical flow estimation methods of Brox *et al.* [6] and Xiao *et al.* [18]. We estimate flow over a sequence of resolutions obtained by recursively reducing the original resolution by a factor η. A standard image pyramid uses $\eta = 0.5$ whereas we (following Brox *et al.* [6]) use a larger factor ($\eta = 0.9$) to obtain better results at a cost of increased computation.

At each resolution level, we run a variational flow update (Section 2.1) then alternate 4 times between estimating occluded regions (Section 2.2) and improving occlusion boundaries using a bilateral filter (Section 2.3).

2.1. Variational Flow Refinement

Our variational update scheme is based on the Brox *et al.* [6] algorithm. In our implementation, we replace the scalar-valued image I with a multi-channel image $I^{[k]}$. We also modulate the data term by a visibility term $r(x, y, t)$ (described in Section 2.2):

$$E_{FlowData}(\bar{u}, \bar{v}, t) = \sum_{x,y,k} r(x, y, t)\Psi([I^{[k]}(x+\bar{u}, y+\bar{v}, t+1) - I^{[k]}(x, y, t)]^2). \quad (1)$$

Here $\bar{u}$ and $\bar{v}$ denote the flow field (evaluated at (x, y, t)) and k is summed over image channels. We use the same robust norm as Brox *et al.* [6]:

$$\Psi(s^2) = \sqrt{s^2 + \epsilon^2}; \quad \epsilon = 0.001. \tag{2}$$

The original Brox *et al.* [6] formulation analytically enforces constancy of the image gradient (and optionally other linear differential operators [10]), whereas we simply treat the gradient as another image channel. Specifically, we use image brightness I (range $[0, 255]$), the x and y derivatives of brightness (I_x and I_y), the green minus red color component, and the green minus blue color component. We scale the color difference channels by 0.25 to reduce the impact of color sampling artifacts common in video.

We modify the Brox *et al.* [6] smoothness term using a smoothness factor α_l modulated by the image gradient:

$$\begin{aligned} b(x,y,t) &= N(\sqrt{I_x(x,y,t)^2 + I_y(x,y,t)^2}; \sigma_b), \\ E_{FlowSmooth}(\bar{u},\bar{v},t) &= \\ &\sum_{x,y} (\alpha_g + \alpha_l \cdot b(x,y,t)) \cdot \Psi(\bar{u}_x^2 + \bar{u}_y^2 + \bar{v}_x^2 + \bar{v}_y^2). \end{aligned} \tag{3}$$

Here N denotes a zero-mean non-normalized Gaussian. We set $\sigma_b = 1.5$, the local smoothness $\alpha_l = 10$, and the global smoothness $\alpha_g = 5$ (based on a variety of flow experiments). We optimize the functional as described in Brox *et al.* [6], using 100 linear solver iterations inside 3 fixed-point iterations at a given resolution level.

2.2. Occlusion Labelling

Optical flow divergence distinguishes between different types of motion boundaries:

$$div(x,y,t) = \frac{\partial}{\partial x}\bar{u}(x,y,t) + \frac{\partial}{\partial y}\bar{v}(x,y,t). \tag{4}$$

The divergence of a flow field is positive for disoccluding boundaries, negative for occluding boundaries, and near zero for shear boundaries. To select occluding boundaries, but not disoccluding boundaries, we define a one-sided divergence function d:

$$d(x,y,t) = \begin{cases} div(x,y,t) & div(x,y,t) < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Pixel projection difference provides another occlusion cue:

$$e(x,y,t) = I(x,y,t) - I(x + \bar{u}(x,y,t), y + \bar{v}(x,y,t), t+1). \tag{6}$$

We combine the modified divergence and pixel projection using zero-mean Gaussian priors:

$$r(x,y,t) = N(d(x,y,t);\sigma_d) \cdot N(e(x,y,t);\sigma_e). \tag{7}$$

We set $\sigma_d = 0.3$ and $\sigma_e = 20$ based on experimental observation of occluded regions.

2.3. Bilateral Flow Filtering

To improve boundary sharpness, we use a bilateral filter based on the work of Xiao *et al.* [18]. The filter sets each flow vector to a weighted average of neighboring flow vectors:

$$\bar{u}'(x,y) = \frac{\sum_{x_1,y_1} \bar{u}(x_1,y_1) w(x,y,x_1,y_1)}{\sum_{x_1,y_1} w(x,y,x_1,y_1)}. \tag{8}$$

The update for $\bar{v}$ is analogous. The algorithm weights the neighbors according to spatial proximity, image similarity, motion similarity, and occlusion labelling:

$$\begin{aligned} w(x,y,x_1,y_1) \quad = \quad & N(\sqrt{(x-x_1)^2 + (y-y_1)^2}; \sigma_x) \\ & \cdot N(I(x,y) - I(x_1,y_1); \sigma_i) \\ & \cdot N(\sqrt{(\bar{u}-\bar{u}_1)^2 + (\bar{v}-\bar{v}_1)^2}; \sigma_m) \\ & \cdot r(x_1,y_1)). \end{aligned} \tag{9}$$

Here $\bar{u}$ denotes $\bar{u}(x,y)$ and $\bar{u}_1$ denotes $\bar{u}(x_1,y_1)$ (and $\bar{v}$ similarly). We set $\sigma_x = 4$, $\sigma_i = 20$, and $\sigma_m = 1$, and restrict (x_1,y_1) to lie within 10 pixels of (x,y).

For efficiency, we apply the filter only near flow boundaries, which we localize using the flow gradient magnitude:

$$g(x,y,t) = \sqrt{\bar{u}_x^2 + \bar{u}_y^2 + \bar{v}_x^2 + \bar{v}_y^2}. \tag{10}$$

The algorithm filters $g(x,y,t)$ using a spatial Gaussian kernel ($\sigma_g = 3$), producing a smoothed gradient magnitude $\hat{g}(x,y,t)$. We apply the bilateral filter (Equation 8) to pixels with $\hat{g}(x,y,t) > 0.05$.

3. Particle Video Overview

A particle video is a video and corresponding set of particles. Particle i has a time-varying position $(x_i(t), y_i(t))$ that is defined between the particle's start and end frames.

We build a particle video by sweeping forward, then backward, across a video. For each frame, the following steps are performed (Figure 2):

- **Propagation.** Particles terminating in an adjacent frame are extended into the current frame according to the forward and reverse flow fields (Section 3.1).
- **Linking.** Particle links are updated (Section 3.2).
- **Optimization.** Particle positions are optimized (Section 4).
- **Pruning.** Particles with high post-optimization error are pruned (Section 4.3).
- **Addition.** New particles are added in gaps between existing particles (Section 5).

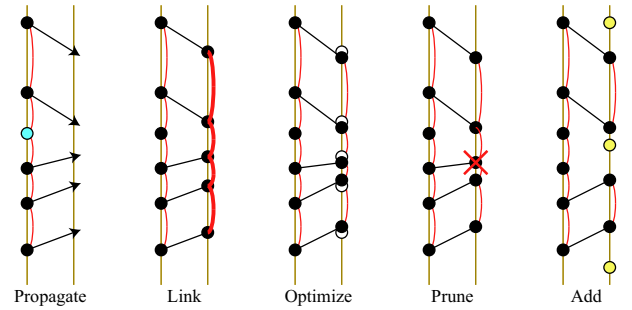

Figure 2. Each plot denotes a pair of adjacent frames. The algorithm propagates particles from one frame to the next according to the flow field, excluding particles (blue) that lie within the flow field's occluded region. The algorithm then adds links (red curves), optimizes all particle positions, and prunes particles with high error after optimization. Finally, the algorithm inserts new particles (yellow) in gaps between existing particles.

3.1. Particle Propagation

When propagating particles to a given frame, all particles defined in adjacent frames, but not defined in the given frame, are placed in the frame according to the flow fields between the frames. To propagate particle i from frame $t-1$ to t, we use the flow field $\bar{u}(x, y, t-1), \bar{v}(x, y, t-1)$:

$$\begin{aligned} x_i(t) &= x_i(t-1) + u(x_i(t-1), y_i(t-1), t-1), \\ y_i(t) &= y_i(t-1) + v(x_i(t-1), y_i(t-1), t-1). \end{aligned} \quad (11)$$

Backward propagation from frame $t+1$ to t is defined analogously. (When making the first forward pass through the video, there are no particles to propagate backward.)

3.2. Particle Linking

To quantify relative particle motion, our algorithm creates links between particles using a constrained Delaunay triangulation [9] (Figure 3). For any given frame, we create a particle link if the corresponding triangulation link exists for the frame or an adjacent frame. Using links from adjacent frames reduces temporal linking variability, while still allowing links to appear and disappear as particles pass by one another.

For each link (i, j), we compute a squared motion difference according to the flow vectors:

$$\begin{aligned} D(i,j,t) &= (\bar{u}(x_i(t), y_i(t), t) - \bar{u}(x_j(t), y_j(t), t))^2 \\ &+ (\bar{v}(x_i(t), y_i(t), t) - \bar{v}(x_j(t), y_j(t), t))^2. \end{aligned} \quad (12)$$

We assign a weight to the link using a Gaussian prior ($\sigma_l = 0.5$) on the motion difference:

$$l_{ij}(t) = N(\sqrt{D(i,j,t)}; \sigma_l^2). \quad (13)$$

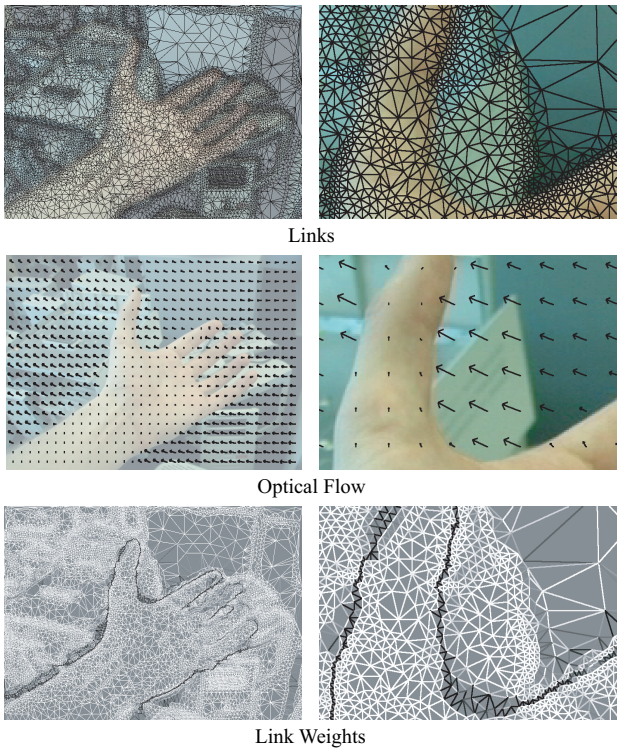

Figure 3. The algorithm uses an optical flow field (Section 2) to compute link weights $l_{ij}(t)$ (darker values are lower). The left side shows an entire frame. The right side shows a magnified portion of the frame.

3.3. Particle Channels and Weights

By design, each particle is intended to be an image point sample. A particle should have a limited spatial extent so that it does not cross an occlusion boundary. However, if the particle is not near an occlusion boundary, we may obtain better particle positioning by using a larger spatial extent.

We repeat the 5 channels used for flow estimation (Section 2.1) at three different scales (via a Gaussian filter with $\sigma = \{1, 2, 4\}$), for a total of 15 channels. We assign particle i a 15-dimensional appearance vector $\{c_i^{[k]}\}$ from the 15-channel frame $\{I^{[k]}\}$ in which it is added.

The algorithm weights each particle according to the filtered flow gradient magnitude (Section 2.3):

$$p_i(t) = N(\hat{g}(x_i(t), y_i(t), t); \sigma_p^2). \quad (14)$$

When $\hat{g}(x_i(t), y_i(t), t)$ is more than one standard deviation ($\sigma_p = 0.05$) from zero, the particle is probably near a flow boundary, so we exclude all but the first channel, because the other channels may be influenced by pixels on the other side of the boundary.

4. Particle Trajectory Optimization

The algorithm repositions particles to locally minimize an objective function that includes three parts: a data term, a distortion term, and a flow-matching term.

4.1. Particle Objective Function

The data term measures how well a particle's initial appearance vector matches the current frame. For particle i at time t the data term for channel k is:

$$E_{Data}^{[k]}(i,t) = \Psi([I^{[k]}(x_i(t), y_i(t), t) - c_i^{[k]}]^2). \quad (15)$$

The distortion term measures the relative motion of linked particles. Let $u_i(t) = x_i(t) - x_i(t-1)$ and $v_i(t) = y_i(t) - y_i(t-1)$. The larger the difference between these motion values, the larger the distortion term. The term is modulated by the link's weight $l_{ij}(t)$ (Section 3.2), so that a link across an occlusion boundary (i.e. a low-weight link) is allowed greater distortion for an equivalent penalty:

$$E_{Distort}(i,j,t) = l_{ij}(t)\Psi([u_i(t) - u_j(t)]^2 + [v_i(t) - v_j(t)]^2). \quad (16)$$

The flow-matching term measures how well each particle trajectory matches the frame-to-frame flow estimates $\bar{u}(x,y,t), \bar{v}(x,y,t)$ (Section 2). It can be viewed as regularization (constraining the data term) or as a secondary data term (where the flow is provided as data). We modulate this term according to the particle's weight $p_i(t)$ (Equation 14); if the particle is close to an occlusion boundary, matching the flow is less important:

$$E_{Flow}(i,t) = p_i(t)\Psi([\bar{u}(x_i(t-1), y_i(t-1), t-1) - u_i(t)]^2 + [\bar{v}(x_i(t-1), y_i(t-1), t-1) - v_i(t)]^2). \quad (17)$$

The combined energy of a particle at a given time is:

$$\begin{aligned} E(i,t) &= \sum_{k \in K_i(t)} E_{Data}^{[k]}(i,t) \\ &+ \alpha_d \sum_{j \in L_i(t)} E_{Distort}(i,j,t) \\ &+ \alpha_f E_{Flow}(i,t). \end{aligned} \quad (18)$$

Here $K_i(t)$ denotes the set of active channels (Section 3.3), and $L_i(t)$ denotes the set of particles linked to particle i in frame t. We find that $\alpha_d = 4$ and $\alpha_f = 0.5$ provide a reasonable tradeoff between the three terms. Given a set P of particle indices and a set F of frame indices, the complete objective function is:

$$E = \sum_{i \in P, t \in F} E(i,t). \quad (19)$$

4.2. Optimization Algorithm

Our algorithm optimizes Equation 19 in a manner similar to the Brox *et al.* [6] algorithm, using a pair of nested loops around a sparse linear system solver. The outer loop iteratively updates $x_i(t)$ and $y_i(t)$ using increments $dx_i(t), dy_i(t)$ computed by the inner loop.

Within the objective function E, we substitute $dx_i(t) + x_i(t)$ for $x_i(t)$ (and instances of y accordingly). Taking partial derivatives, we obtain a system of equations, which the algorithm solves for $dx_i(t)$ and $dy_i(t)$:

$$\left\{ \frac{\partial E}{\partial dx_i(t)} = 0, \frac{\partial E}{\partial dy_i(t)} = 0 \mid i \in P, t \in F \right\}. \quad (20)$$

For the data term, we use the image linearization from Brox *et al.* [6]:

$$I_z^{[k]} = I_x^{[k]} dx_i(t) + I_y^{[k]} dy_i(t) + I^{[k]} - c_i^{[k]}, \quad (21)$$

$$\frac{\partial E_{Data}^{[k]}(i,t)}{\partial dx_i(t)} \approx 2\Psi'([I_z^{[k]}]^2) I_z^{[k]} I_x^{[k]}. \quad (22)$$

We omit the $(x_i(t), y_i(t), t)$ indexing of I, I_x, I_y, and I_z. Ψ' is the derivative of Ψ with respect to its argument s^2. Note that this linearization occurs inside the inner fixed-point loop; the algorithm is still optimizing the original non-linearized objective function.

For the distortion term, the derivative is more straightforward. We use $du_i(t)$ as shorthand for $dx_i(t) - dx_i(t-1)$ and $dv_i(t)$ for $dy_i(t) - dy_i(t-1)$:

$$\begin{aligned} &\frac{\partial E_{Distort}(i,j,t)}{\partial dx_i(t)} = \\ &2l_{ij}(t)\Psi'([u_i(t) + du_i(t) - u_j(t) - du_j(t)]^2 \\ &\qquad + [v_i(t) + dv_i(t) - v_j(t) - dv_j(t)]^2) \\ &\cdot (u_i(t) + du_i(t) - u_j(t) - du_j(t)). \end{aligned} \quad (23)$$

The flow matching term uses the same linearization as the data term:

$$\begin{aligned} U &= \bar{u}_x dx_i(t-1) + \bar{u}_y dy_i(t-1) + \bar{u} - (u_i(t) + du_i(t)), \\ V &= \bar{v}_x dx_i(t-1) + \bar{v}_y dy_i(t-1) + \bar{v} - (v_i(t) + dv_i(t)), \end{aligned}$$

$$\frac{\partial E_{Flow}(i,t)}{\partial dx_i(t)} \approx 2p_i(t)\Psi'(U^2 + V^2)U. \quad (24)$$

Here we omit the $(x_i(t-1), y_i(t-1), t-1)$ indexing of $\bar{u}$, $\bar{v}$, and their spatial derivatives. Note that $dx_i(t)$ also appears in derivatives of $E(i, t+1)$.

Each of these partial derivatives is linear in $dx_i(t)$ and $dy_i(t)$ except the Ψ' factors. For each iteration of the inner loop, we compute the Ψ' terms, then hold them constant within the linear system solver:

Loop (4 Times)
  Compute $I_x^{[k]}(i,t), I_y^{[k]}(i,t)$
  $dx_i(t), dy_i(t) \leftarrow 0$
  Loop (4 Times)
    Compute Ψ' terms
    Solve system to update $dx_i(t)$, $dy_i(t)$
      (200 SOR iterations)
  End Loop
  $x_i(t) \leftarrow x_i(t) + dx_i(t)$
  $y_i(t) \leftarrow y_i(t) + dy_i(t)$
End Loop

4.3. Particle Pruning

After optimizing the particles, we prune particles that continue to have high energy $E(i,t)$. To reduce the impact of a single bad frame, we filter each particle's energy values using a Gaussian ($\sigma_t = 1$ frames). Particle i obtains a sequence of filtered objective values $\hat{E}(i,t)$. (Note: this Gaussian is not strictly temporal; it filters the values for the given particle, which is moving through image space.)

If $\hat{E}(i,t) > 10$, the particle is marked as incorrect in frame t. Given this marking, the algorithm prunes the particle to the longest contiguous interval of correct frames. If the remaining duration of the particle is less than 3 frames (and the particle's nearest endpoint is more than 3 frames from the current frame), the particle is deleted.

5. Particle Addition

After optimization and pruning, the algorithm adds new particles in gaps between existing particles. (The same process is used to create all particles in the first video frame.) The algorithm arranges for higher particle density in regions of greater visual complexity, in order to model complex motions.

To add new particles to a given frame, the algorithm determines a scale value $s(x,y)$ for each pixel. The scale values are discrete, taken from the set $\{\sigma(j) = 1.9^j \mid 0 \leq j \leq 5\}$. To compute the scale map, we start by filtering the image using a Gaussian kernel for each scale $\sigma(j)$, producing a set of images $\{I_j\}$.

Then, for each pixel, we find the range of scales over which the blurred pixel value does not change substantially. If the pixel has the same color in a large scale image as in all smaller scale images, it is a large scale pixel. Specifically, the algorithm chooses the maximum scale index $k(x,y)$ such that $||I_j(x,y) - I_1(x,y)||_2 < 10$ for all $j \leq k(x,y)$. (Here we use (r,g,b) vector distance.)

These scale indices are filtered with a spatial Gaussian ($\sigma_s = 2$), producing a blurred scale index map $\hat{k}(x,y)$ (which we round to integer values). We then set the scale values from the indices: $s(x,y) = \sigma(\hat{k}(x,y))$. Figure 4 provides an example scale map.

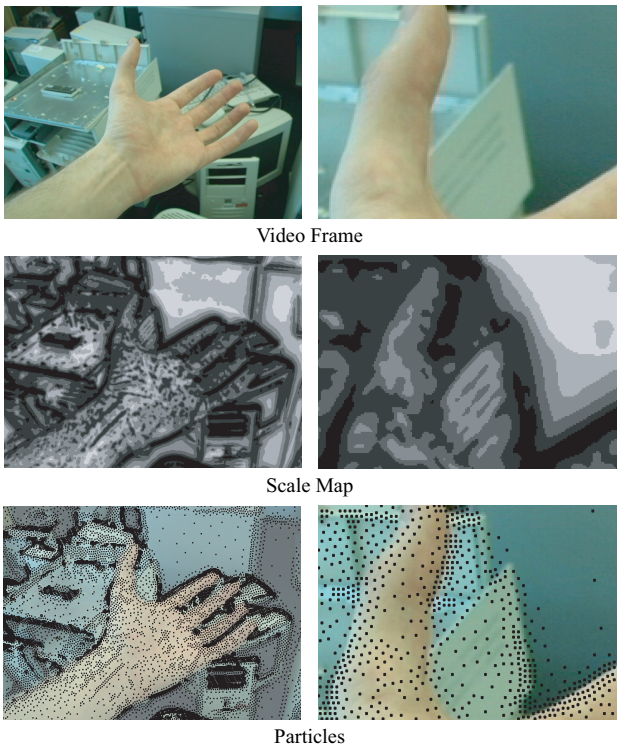

Figure 4. For each video frame, the algorithm computes a scale map that determines the placement of new particles (Section 5). The left side shows an entire frame. The right side shows a magnified portion of the frame.

Given the scale map, we iterate over the image adding particles. For each pixel, if the distance to the nearest particle is greater than $s(x,y)$, we add a particle at that pixel. The algorithm does this efficiently in time (linear in the number of particles) by creating an occupancy map at each scale.

6. Results

We evaluate the algorithm using a set of 20 videos, which together include a variety of scenes, camera motions, and object motions. Figure 6 shows particle correspondences for several of the videos. Complete videos and results are available at `http://rvsn.csail.mit.edu/pv/`.

On a typical 720 by 480 video frame, the algorithm spends about 7 seconds optimizing particles, 0.2 seconds linking particles, 0.05 seconds pruning particles, and 2 seconds adding particles (each of these figures is given per sweep). The flow algorithm (Section 2) requires about 100 seconds per frame pair. In our experiments, we use one forward sweep followed by one backward sweep.

Objectively evaluating the correctness of the algorithm is difficult given the lack of ground-truth data. The ideal evaluation measurement should allow comparison with future particle video algorithms and with non-particle approaches

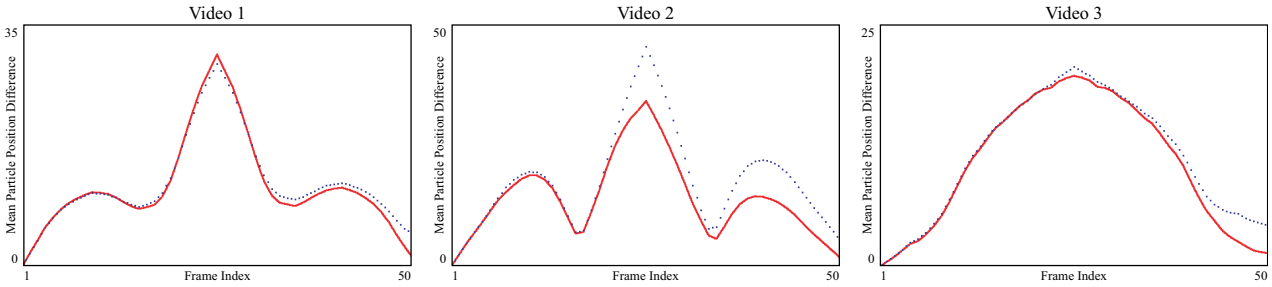

Figure 5. Each plot shows the mean distance of surviving particles from their positions in the start frame. As described in Section 6, the videos are temporally mirrored, so we expect all unoccluded particles to return to their start positions. The dotted lines denote concatenated flow vectors. In each case, particles return closer to their starting positions than concatenated flow vectors. Both algorithms have some trouble capturing the face pose change in Video 2.

to long-range motion estimation.

For our evaluation, we construct videos that return to the starting configuration by replacing the second half of each video with a temporally reversed copy of the first half. For such a video, we expect particles from the start frame to return to their starting positions in the end frame. For each particle that survives from the start frame to the end frame, the algorithm computes the spatial distance (error) between the particle's start position and end position (Figure 5).

As a simplistic point of comparison, for each particle position in the start frame, we concatenate flow vectors, computed as described in Section 2. These flow trajectories are terminated when they enter an occluded region, as determined by the flow algorithm.

This evaluation approach is flawed for several reasons. A particle video algorithm could easily obtain a lower spatial error by pruning more particles (at the cost of a lower particle survival rate). Furthermore, by allocating fewer particles near occlusions and more particles in other regions, an algorithm could both increase the survival rate and decrease the spatial error. Thus, we provide the evaluation for descriptive purposes only. This evaluation should not be used to compare the results with future methods.

In the future, we envision that researchers will create photo-realistic rendered videos with known ground truth correspondences. These rendered videos should include deforming objects, complex reflectance, detailed geometry, motion blur, unstable camera motion, and optical artifacts.

7. Conclusion

The particle video algorithm provides a new approach to motion estimation, a central problem in computer vision. Dense long-range video correspondences could improve methods for many existing vision problems, in areas ranging from robotics to filmmaking.

Our particle representation differs from standard motion representations, such as vector fields, layers, and tracked feature patches. Some existing optical flow algorithms incorporate constraints from multiple frames (often using a temporal smoothness assumption), but they do not enforce long-range correspondence consistency. Our algorithm improves frame-to-frame optical flow by enforcing long-range appearance consistency and motion coherence.

Our algorithm does have several limitations. Like most motion algorithms, the approach sometimes fails near occlusions. Also, the approach has difficulty with large changes in appearance due to non-Lambertian reflectance and major scale changes. (Our use of color and gradient channels provides some robustness to reflectance, but not enough to handle all cases correctly.)

In the future, we will focus on these issues. We plan to explore more sophisticated occlusion localization and interpretation, possibly using simultaneous particle and flow optimization. We will also experiment with allowing slow, smooth changes in a particle's appearance over time, to permit larger reflectance and scale changes. We may obtain better performance by incorporating invariant feature descriptors (for particles away from occlusions), spatiotemporal segmentation [16], and geometric constraints (for rigid portions of the scene).

Current limitations of the particle video algorithm arise from our methods for positioning particles, rather than a fundamental limitation of the particle representation. By making our data and results available online, we hope others will explore the particle video problem.

References

[1] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sanchez. Symmetrical dense optical flow estimation with occlusion detection. In *ECCV*, pages 721–735, 2002.

[2] T. Amiaz and N. Kiryati. Dense discontinuous optical flow via contour-based segmentation. In *ICIP*, pages 1264–1267, 2005.

[3] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1995.

[4] M. Black and P. Anandan. Robust dynamic motion estimation over time. In *CVPR*, pages 296–302, 1991.

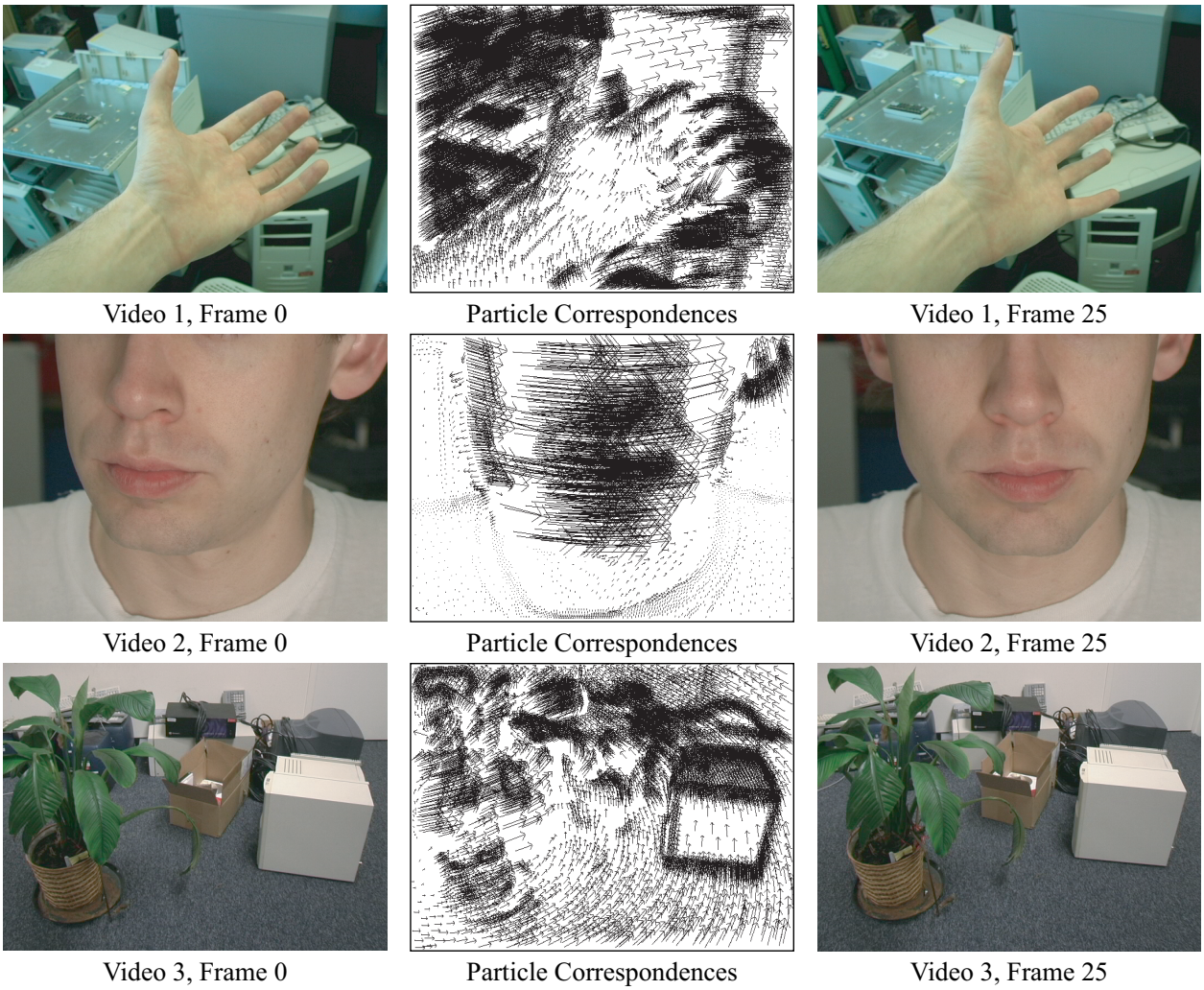

Figure 6. Each row shows a frame pair from one test video. Correspondences are shown for particles in common between the frames.

[5] M. Brand. Morphable 3D models from video. In *CVPR*, pages 456–463, 2001.

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36, 2004.

[7] M. Elad and A. Feuer. Recursive optical flow estimation–adaptive filtering approach. *Visual Communication and Image Representation*, 9(2):119–138, 1998.

[8] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *ICCV*, pages 626–633, 1999.

[9] D. Lischinski. *Graphics Gems IV*, chapter Incremental Delaunay Triangulation, pages 47–59. Academic Press, 1994.

[10] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 2006 (to appear).

[11] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, pages 1154–1160, 1998.

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

[13] C. Silva and J. Santos-Victor. Motion from occlusions. *Robotics and Autonomous Systems*, 35(3–4):153–162, 2001.

[14] C. Strecha, R. Fransens, and L. V. Gool. A probabilistic approach to large displacement optical flow and occlusion detection. In *Statistical Methods in Video Processing*, pages 71–82, 2004.

[15] W. Thompson. Exploiting discontinuities in optical flow. *IJCV*, 30(3):163–174, 1998.

[16] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *ECCV*, pages 238–249, 2004.

[17] J. Wills and S. Belongie. A feature-based approach for determining dense long range correspondences. In *ECCV*, pages 170–182, 2004.

[18] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *ECCV*, 2006 (to appear).

[19] C. L. Zitnick, N. Jojic, and S. B. Kang. Consistent segmentation for optical flow estimation. In *ICCV*, pages 1308–1315, 2005.