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 appearance consistency 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 [30] or optical flow [7]. 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 long-range and moderately dense (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).

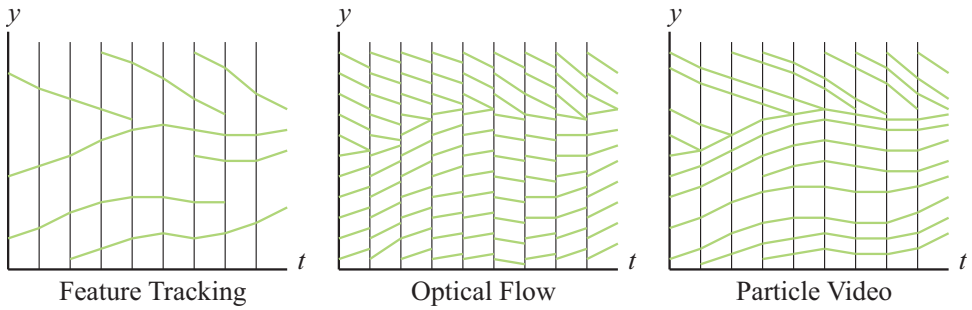


Figure 1: Each diagram represents point correspondences between frames of a hypothetical sequence. Feature tracking is long-range but sparse. Optical flow is dense but short-range. Our particle video representation is denser than feature tracking and longer range than optical flow.

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, labelling, and object removal. Goldman *et al.* [17] demonstrate interactive particle-based video annotation applications that would be difficult to create using standard motion representations.

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 [30]. 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 combines point-based appearance matching and inter-particle distortion. The algorithm 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 particle motion estimation. 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

The particle video problem can be described as dense feature tracking or long-range optical flow. We want to track the trajectory of each pixel through a given video. Ideally each trajectory would correspond to the motion of a physical real-world surface point.

A 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. Particles 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.

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.

In contrast, the particle representation allows a form of random-access motion evaluation: given a set of particles, we can easily find correspondences between any pair of frames (assuming the frames have a sufficient number of particles in common). Furthermore, unlike a sequence of motion fields, the particle representation provides discrete motion primitives, which are valuable for subsequent use of the motion information, such as interactive video matting [26] and interactive video annotation [17].

1.3 Overview

Section 2 describes related work in motion estimation. We combine several of these methods to create an optical flow algorithm described in Section 3 (with details in Appendix A). Optical flow is used as an input to our particle-based motion estimation.

The particle algorithm is described in Section 4, which explains how particles are added, propagated, linked, optimized, and pruned. These steps are performed as the algorithm sweeps back and forth across a video, constructing a complete particle representation of the video's motion.

Section 5 includes an evaluation of the particle video algorithm on a variety of real-world videos. We quantify the performance of the algorithm and possible alternatives. We provide mechanisms for visualizing the algorithm's results and measuring its performance.

This paper supersedes our CVPR paper [27] that introduced the particle video approach and a thesis [26] that covers our particle video algorithm in more detail. After the CVPR version, we removed the flow terms from the particle objective (for simplicity) and added a mechanism for handling slow particle appearance change over time (Section 4.5.5).

2 Related Work

Finding correspondences between two or more images is one of the most studied subjects in computer vision. Prior work in optical flow estimation is most closely related to our approach.

2.1 Multi-Frame Optical Flow

Most optical flow algorithms estimate correspondences between a pair of images, but some use more than two images. These methods may better disambiguate motion boundaries and may be more computationally efficient than computing flow independently for each frame pair.

Most multi-frame optical flow methods rely on some form of temporal coherence assumption [24, 6]. Black and Anandan [9] use a basic temporal smoothness constraint as part of a method that provides robustness in the data terms and spatial smoothness terms. Black [10] subsequently presents a method that adapts to temporal disruptions. Chin *et al.* [14] use an approximate Kalman filter to model temporal variations within a differential flow estimation algorithm. Elad and Feuer [15] present a differential estimation technique with decaying temporal constraints. Shi and Malik [29] use multiple frames to aid the segmentation and estimation of distinct motions.

For real-world video sequences, the temporal smoothness assumption is often violated. Some sharp motion changes (e.g. due to hand-held camera operation) can be reduced by whole-frame stabilization algorithms. However, other fast motions (such as someone walking or talking) cannot be stabilized. These motions violate temporal smoothness assumptions because of the limited time-domain sampling found in most videos.

Flow rank methods, in contrast, do not rely on assumptions of spatial or temporal smoothness. Irani [20] shows that matrices of flow components are geometrically restricted to lie in low-dimensional subspaces. Using these constraints, she presents an algorithm to simultaneously estimate flow over multiple frames. Brand [12] applies a similar approach to non-rigid scenes by describing deformable objects as linear combinations of basis shapes. Unfortunately, these constraints are only valid for weak perspective or short windows in time. Nonetheless rank constraints could be incorporated into particle video estimation.

2.2 Occlusion Detection for Optical Flow

Occlusion modelling is the most difficult part of estimating optical flow. All optical flow algorithms rely on spatial agglomeration of information, but this information may be misinterpreted when combined from both sides of an occlusion boundary. Furthermore, a core assumption of most flow algorithms is that each pixel goes somewhere, when in fact some pixels may disappear due to occlusions.

A common way of handling occlusion boundaries is robustness in the data and smoothness terms [11, 13]. This robustness allows an algorithm to cope with assumption violations that occur near flow discontinuities. In the data term, a robust distance function allows occluded pixels to mismatch. In the smoothness term, a robust distance function allows discontinuities in the flow field. Because of this robustness, these algorithms fail gracefully near occlusion boundaries, but they still fail. Methods that use anisotropic regularization (whether robust or not) [35, 6], similarly fail to model the process of occlusion.

Amiaz and Kiryati [2] use level sets (rather than standard regularization) to refine the localization of the Brox *et al.* [13] occlusion boundaries. By defining an explicitly piece-wise smooth objective, optimized as a post-process to the Brox *et al.* algorithm, the error near the boundaries is reduced. However, the algorithm still does not account for pixels that disappear.

Thompson [34] explores occlusion boundaries in more depth. He describes several of the difficulties with traditional boundary handling. He argues that, even though flow estimates are regularized, the underlying point estimates can be seriously corrupted near occlusion boundaries, because they usually have some spatial extent. (Computing a derivative always requires more than one pixel.) Also, he explains, if the boundary itself has good motion estimates, the maximal flow gradient will systematically mislocate the boundary to be over the occluded surface. Thompson proceeds by presenting an algorithm that addresses some of these problems. His algorithm explicitly identifies the direction of occlusion at each boundary. The algorithm also uses flow and boundary projection based on assumptions of temporal continuity. The main limitation of Thompson's method is that it only estimates motion at image brightness edges, ignoring valuable but subtle image textures.

Zitnick *et al.* [37] estimate optical flow using correspondences between segmented image regions. Like particles, these segments provide small, simple, discrete motion entities. The algorithm estimates blending between segments in order to model mixed pixels at occlusion boundaries. The segments provide well-defined occlusion boundaries between objects of different colors, but the algorithm fails when motion boundaries do not coincide with segment boundaries. Also, the algorithm does not account for segments that become fully occluded.

Because occluded pixels violate a basic assumption of optical flow (that each pixel goes somewhere), several methods attempt to identify occluded pixels explicitly. Silva and Victor [31] use a pixel dissimilarity measure to detect brightness values that appear or disappear over time. Alvarez *et al.* [1] present an algorithm that simultaneously computes forward and reverse flow fields, labelling pixels as occluded where the two disagree. Strecha *et al.* [32] treat occlusion labels as hidden variables in an EM optimization. In this case, pixel value mismatches (rather than flow mismatches) are used to identify occlusions. The occluded pixels modulate anisotropic regularization, such that flow values do not diffuse across occlusion boundaries.

Xiao *et al.* [36] also use pixel value mismatches to detect occluded regions across which flow diffusion is restricted. They regularize flow estimates using a bilateral filter that incorporates flow from neighboring non-occluded pixels that are similar in motion and appearance. The resulting algorithm is relatively successful at identifying occlusion boundaries and computing accurate flow on both sides of such boundaries. We incorporate some elements of this bilateral filter into the flow algorithm described in Section 3.

Like optical flow, feature tracking has difficulty with occlusion boundaries. When a feature patch lies across two independently moving surfaces, the feature cannot correctly follow both. For example, an algorithm may track what appears to be a 'T' junction, but which is in fact a pair of overlapping edges, neither of which is tracked correctly. These kinds of errors can be detected using correlation error [30, 16] or geometric constraints such as the fundamental matrix [18]. Another alternative is to adjust the region of support for a feature to fall on one side of the occlusion [28, 22].

3 Variational Optical Flow with Bilateral Filtering

Our particle video algorithm uses frame-to-frame optical flow to provide an initial guess for particle motion. 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 pair; this enables the algorithm to perform well on hand-held video with moving objects.

Our optical flow algorithm combines the variational approach of Brox *et al.* [13] with the bilateral filtering approach of Xiao *et al.* [36]. The algorithm optimizes a flow field over a sequence of increasing resolutions. At each resolution, the algorithm performs the following steps:

- optimize the flow field using a variational objective with robust data and smoothness terms (Appendix A.1),
- identify the occluded image regions using flow field divergence and pixel projection difference (Appendix A.2),
- and improve flow boundaries using an occlusion-aware bilateral filter (Appendix A.3).

The sequence of resolutions is 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.* [13]) use a larger factor ($\eta = 0.9$) to obtain better results at a cost of increased computation. We set a 0.05 lower bound on the scale factor, which results in 29 resolution levels from an NTSC video frame. The smallest level is 36 by 24 pixels. (We crop the video frame from 720x480 to 712x480 to remove left and right boundary artifacts before estimating flow.) After scaling the image, we apply a $\sigma = 1$ Gaussian smoothing filter (again following Brox *et al.* [13]).

To handle large camera motions, we add an initialization step consisting of estimating whole-frame translation. The algorithm uses the KLT [23, 3] gradient-based optimization to register the frames, in a coarse-to-fine sequence of resolutions (with a factor of 2 scale change between each resolution). At each step we perform 8 optimization iterations. This initialization process takes a fraction of a second for a full-resolution frame pair. The resulting whole-frame translational offset is used to initialize the flow field at the lowest resolution level.

4 Particle Video Algorithm

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

4.1 Top-Level Particle Video Algorithm

Our algorithm builds a particle video by moving forward and backward across the video. Moving backwards, occlusion boundaries become disocclusion boundaries, which are easier to interpret than occlusion boundaries. By moving through the video in both directions, new particles can be extended in both directions.

For each processed 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 4.3).
- **Linking.** Particle links are updated (Section 4.4).
- **Optimization.** Particle positions are optimized (Section 4.5).
- **Pruning.** Particles with high post-optimization error are pruned (Section 4.6).
- **Addition.** New particles are added in gaps between existing particles (Section 4.7).

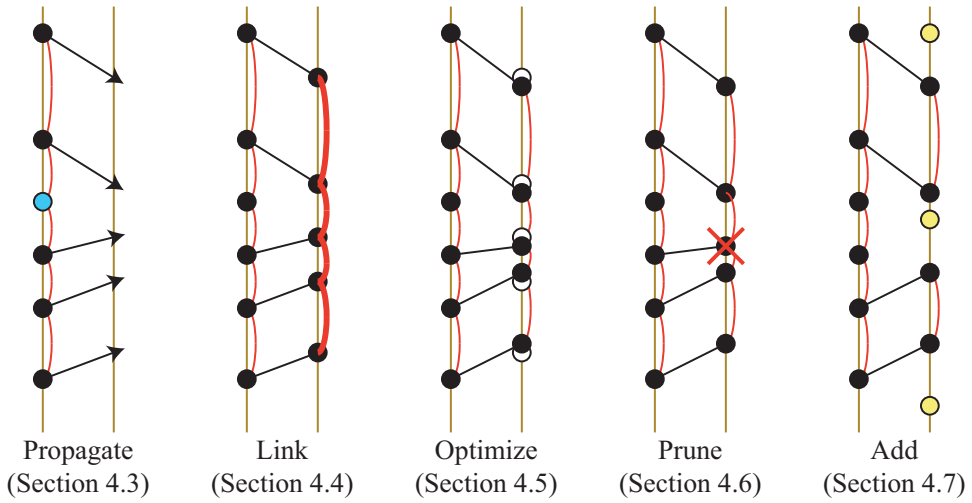


Figure 2: Each plot denotes a pair of consecutive frames. The algorithm propagates particles from one frame to the next according to the flow field, excluding particles (blue) that lie within the flow field's occluded region. The algorithm then adds links (red curves), optimizes all particle positions,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

To reduce computation, the algorithm maintains a cache of information for each video frame. This cache includes the frame itself, color and gradient channels (and gradients thereof), a scale map (Section 4.7), forward flow (and its gradient magnitude), and reverse flow.

4.2 Particle Channels

The particle video algorithm uses the same 5 channels as the flow estimation algorithm (Section A.1.1): image brightness, green minus red channel, green minus blue channel, x gradient, and y gradient. As before, k denotes the channel index; at time t the kth image channel is $I^{[k]}(t)$.

The color and gradient channels are moderately insensitive to changes in lighting and reflectance, which facilitates matching a particle with a temporally distant frame. However, these channels depend on a wider spatial area of support, which may cause mismatches for particles near occlusion boundaries. (The gradient is computed using multiple pixels and the color channel has a low spatial resolution due to common video color compression.)

To address this, we disable the gradient and color channels near occlusion boundaries, as determined by the filtered flow gradient magnitude $\hat{g}(x,y,t)$ (Section A.3). When $\hat{g}(x_i(t),y_i(t),t) > 0.01$, the particle is probably near a flow boundary, so we exclude all but brightness channel, because the other channels may be influenced by pixels on the other side of the boundary.

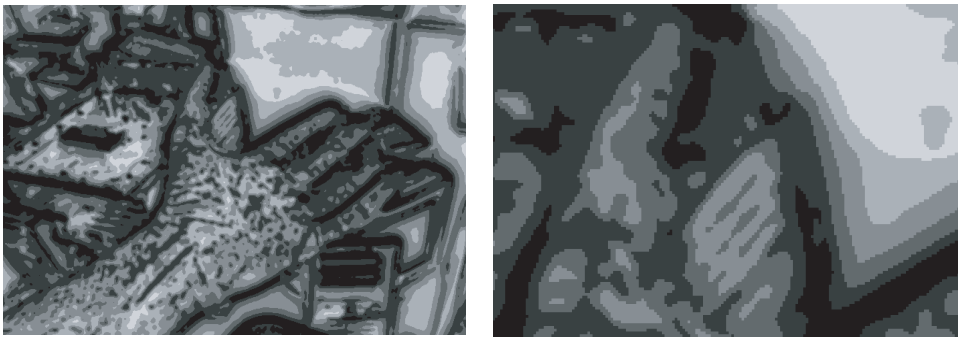
We scale the gradient and color channels by a factor of 0.1 to reduce the effects of noise in these channels. In our experiments, we find that these channels provide only a small benefit. For the sake of simplicity, others may choose to omit these channels.

4.3 Propagating Particles

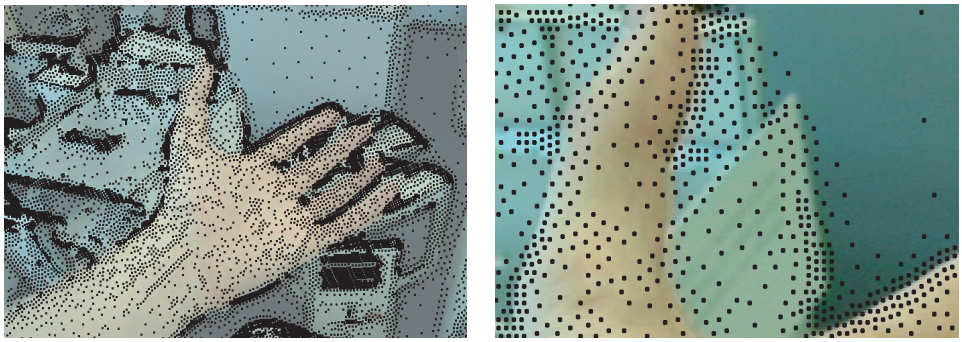
To propagate particles to a given frame, all particles defined in adjacent frames, but not defined in the given frame, are placed in the frame according to the flow fields between the frames. To propagate particle i from frame $t-1$ to t, we use the flow field $u(x,y,t-1), v(x,y,t-1)$:

$$x_i(t) \quad = \quad x_i(t-1) + u(x_i(t-1), y_i(t-1), t-1), \tag{1}$$

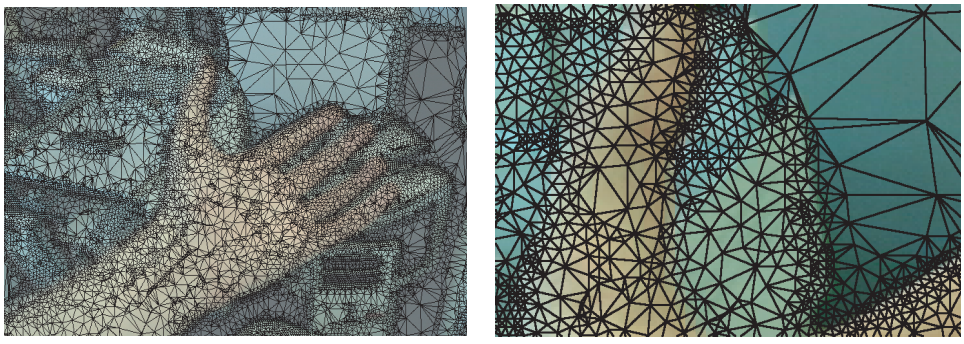
Video Frame



Scale Map



Particles



Links

Figure 3: For each video frame, the algorithm computes a scale map that determines the placement of new particles (Section 4.7). Links are added using a particle triangulation (Section 4.4). The left side shows an entire frame. The right side shows a magnified portion of the frame.

$$y_i(t) \quad = \quad y_i(t-1) + v(x_i(t-1), y_i(t-1), t-1). \tag{2}$$

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.) If the optical flow field indicates that a particle becomes occluded, the particle is not propagated.

4.4 Particle Links

To quantify relative particle motion, our algorithm creates links between particles using a constrained Delaunay triangulation [21] (Figure 3). The triangulation ensures a good directional distribution of links for each particle. This is preferable to simply linking each particle to its N nearest neighbors (which could all be in one direction from a given particle).

For any given frame, we create a particle link if the corresponding triangulation edge 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.

The algorithm assigns a weight to each link based on the difference between the trajectories of the linked particles. If the particles have similar trajectories, they are probably part of the same surface, and thus should be strongly linked. If the particles are separated by an occlusion boundary, the weight should be near zero.

The algorithm computes the mean squared motion difference between linked particles i and j over the set T of frames in which the link is defined:

$$D(i,j) = \frac{1}{|T|} \sum_{t \in T} (u_i(t) - u_j(t))^2 + (v_i(t) - v_j(t))^2. \tag{3}$$

Here we let $u_i(t) = x_i(t) - x_i(t-1)$ and $v_i(t) = y_i(t) - y_i(t-1)$. The algorithm computes the link weight using a zero-mean Gaussian prior ($\sigma_l = 1.5$):

$$l_{ij} = N(\sqrt{D(i,j)}; \sigma_l). \tag{4}$$

4.5 Particle Optimization

The core of the particle video algorithm is an optimization process that repositions particles. As described in Section 4.3, a flow field provides an initial location for each particle in a given frame; the optimization refines these positions with the goal of reducing long-range drift.

For a given particle, this optimization can modify the particle's position in any frame except for the frame in which the particle was first added. This original frame defines the particle's reference position. (The original frame will be different from the particle's start frame if it was propagated backward from the original frame.)

4.5.1 Particle Objective Function

The algorithm repositions particles to locally minimize an objective function that includes two components for each particle: a data term and a distortion term. This objective function has some similarities to the variational flow functionals described in Section 3, but it operates just on the particles, not the full set of pixels.

The energy of particle i in frame t is:

$$E(i,t) = \sum_{k \in K_i(t)} E_{Data}^{[k]}(i,t) + \alpha \sum_{j \in L_i(t)} E_{Distort}(i,j,t). \tag{5}$$

Here $K_i(t)$ denotes the set of active channels (Section 4.2), and $L_i(t)$ denotes the set of particles linked to particle i in frame t. We find that $\alpha = 1.5$ provides a reasonable trade-off between the two terms.

Given a set P of particle indices and a set F of frame indices, the complete objective function is:

$$E = \sum_{t \in F, i \in P} E(i,t). \tag{6}$$

4.5.2 Data Energy

The data term measures how well a particle's appearance (Section 4.2) matches the video frames. We allow particle appearance to change slowly over time, to cope with non-Lambertian reflectance and changes in scale. For particle i at time t the kth channel of the particle's appearance is:

$$c_i^{[k]}(t) = I^{[k]}(x_i(t), y_i(t), t). \tag{7}$$

Using a Gaussian kernel ($\sigma_c = 5$), we filter these appearance values along the time axis, producing a slowly-varying appearance denoted by $\hat{c}_i^{[k]}(t)$. For a given frame, the data term measures the difference between the observed appearance and filtered appearance:

$$E_{Data}^{[k]}(i,t) = \Psi([c_i^{[k]}(t) - \hat{c}_i^{[k]}(t)]^2). \tag{8}$$

Here Ψ is the robust norm described in Section A.1.1. Although we assume temporal appearance smoothness, we do not assume temporal motion smoothness. The data term suggests that a particle's appearance changes slowly, but does not depend on the smoothness of the particle trajectory. Alternatively, we could attempt to fit physical reflectance models to the particle appearance changes [19].

4.5.3 Distortion Energy

The data term alone does not uniquely constrain the particle positions. A distortion term spatially propagates data term constraints, such that the algorithm can jointly optimize the particle positions. This distortion term measures the relative motion of particles. If two linked particles move in different directions, they will have a larger distortion term. If they move in the same direction, they will have a smaller distortion term.

The distortion term is defined between a pair of linked particles i and j. As before, we 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:

$$E_{Distort}(i,j,t) = l_{ij}\Psi([u_i(t) - u_j(t)]^2 + [v_i(t) - v_j(t)]^2). \tag{9}$$

Note that this is symmetric: $E_{Distort}(i,j,t) = E_{Distort}(j,i,t)$.

The distortion term is modulated by the link weight l_{ij} so that a link across an occlusion boundary (i.e. a low-weight link) is allowed greater distortion for an equivalent penalty. Both the link weights and distortion term measure the relative motion of particles, but the link weights take into account entire particle trajectories whereas the distortion term refers to a single frame. By modulating the distortion term using link weights, the algorithm encourages particles that have moved together to continue moving together in the current frame, while particles that have moved differently are allowed to move differently in the current frame.

Note that the distortion term (like the data term) does not require or encourage temporal motion smoothness. It measures the relative motion of particles, so the global motion does not need to be smooth (the camera motion can be unstabilized).

The distortion term resists incorrect motions that could be caused by the data term, especially near occlusion boundaries. In the case that a particle is being occluded (but is not pruned by an occlusion mask), the data term may push the particle into an unoccluded part of the background surface (unless the particle happens to better match the foreground surface). Also, the flow field may incorrectly push or pull background pixels along with

the foreground surface. In both cases, a strong distortion term will improve the correctness of the particle motion.

However, the distortion term cannot be too strong, because this rigidity would prevent certain correct motions, such as those caused by changes in viewpoint or non-rigid object deformation. This tradeoff can be controlled by adjusting the distortion factor α in Equation 5. Limitations of this distortion approach are discussed in Section 6.

4.5.4 Constructing a Sparse Linear System

The algorithm optimizes Equation 6 in a manner similar to the variational technique described in Section A.1, using a fixed-point loop around a sparse linear system solver. In the following sections, we describe the construction of the sparse linear system. In Section 4.5.7 we provide the complete optimization algorithm.

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 $dy_i(t)$:

$$\left\{ \frac{\partial E}{\partial dx_i(t)} = 0, \frac{\partial E}{\partial dy_i(t)} = 0 \mid i \in P, t \in F \right\}. \tag{10}$$

The $dx_i(t)$ and $dy_i(t)$ values produced by solving this system are added to the current particle positions ($x_i(t)$ and $y_i(t)$).

4.5.5 Data Derivative

For the data term, we use the image linearization from Brox *et al.* [13]:

$$I_z^{[k]} = I_x^{[k]} dx_i(t) + I_y^{[k]} dy_i(t) + I^{[k]} - \hat{c}_i^{[k]}, \tag{11}$$

$$\frac{\partial E_{Data}^{[k]}(i,t)}{\partial dx_i(t)} \approx 2\Psi'([I_z^{[k]}]^2)(I_z^{[k]})I_x^{[k]}. \tag{12}$$

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

4.5.6 Distortion Derivative

For the distortion term, we use $du_i(t)$ as shorthand for $dx_i(t) - dx_i(t-1)$ and $dv_i(t)$ for $dy_i(t) - dy_i(t-1)$. This gives the following partial derivative:

$$\frac{\partial E_{Distort}(i,j,t)}{\partial dx_i(t)} = 2l_{ij}(t)\Psi'_{Distort}(i,j,t)(u_i(t) + du_i(t) - u_j(t) - du_j(t)). \tag{13}$$

Here we define:

$$\Psi'_{Distort}(i,j,t) = \Psi'([u_i(t) + du_i(t) - u_j(t) - du_j(t)]^2 + [v_i(t) + dv_i(t) - v_j(t) - dv_j(t)]^2). \tag{14}$$

The $dx_i(t)$ variable also appears in the term for link i, j at time $t+1$:

$$\frac{\partial E_{Distort}(i,j,t+1)}{\partial dx_i(t)} =$$

$$-2l_{ij}(t+1)\Psi'(i,j,t+1)(u_i(t+1)+du_i(t+1)-u_j(t+1)-du_j(t+1)). \tag{15}$$

The $dx_i(t)$ variable also appears in the terms for particle j at times t and $t+1$. These derivatives are identical (since the terms are identical via the i, j symmetry of the distortion energy), so we add an extra factor of two to the distortion derivatives.

4.5.7 Fixed-Point Scheme

Like the variational flow algorithm described in Section 3, the particle optimization iteratively solves for updates to the particle positions. The iteration terminates when the mean change in position is less than 0.005 (with an upper bound of 10 iterations). The linear system solver performs 200 iterations inside each of the loop iterations. These numbers control the tradeoff between accuracy and running time. The solver uses the SOR algorithm [5], with some conditioning and smoothing (further stability analysis would be beneficial). We limit $|dx_i(t)|$ and $|dy_i(t)|$ to be less than 2 pixels for each step.

The algorithm uses a pair of integer matrices to keep track of which sparse system variables correspond to which particles. One matrix maps variable indices to (i,t) pairs. The other maps (i,t) to variable indices.

4.6 Pruning Particles

After optimizing the particles, we prune particles that continue to have high energy values. These particles have high distortion and/or a large appearance mismatch, indicating possible occlusion.

As defined in Section 4.5.1, $E(i,t)$ denotes the objective function value of particle i in frame t 5. To reduce the impact of a single bad frame, we filter each particle's energy values using a Gaussian ($\sigma_t = 1$ frames). (Note: this Gaussian is not strictly temporal; it filters the values for the given particle, which is moving through image space.) If in any frame the filtered energy value is greater than $\delta = 5$, the particle is deactivated in that frame.

4.7 Adding Particles using Scale Maps

After optimization and pruning, the algorithm adds new particles in gaps between existing particles. The algorithm arranges for higher particle density in regions of greater visual complexity, in order to model complex motions. (Motion complexity often implies visual complexity, though the reverse is not generally true.)

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 (Figure 4). Specifically, the algorithm chooses the maximum scale index $k(x,y)$ such that $||I_j(x,y) - I_1(x,y)||_2 < \delta_s$ for all $j \leq k(x,y)$. (Here we use (r,g,b) vector distance when comparing pixel values.)

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 3 provides an example scale map.

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.

The same process is used to position all particles in the first video frame. For the first video frame, the algorithm adaptively sets the δ_s parameter that controls the creation of the scale map. The parameter is initially set to 10, then adjusted up and down until the number of created particles falls between 8000 and 12000. The same δ_s is used for the remainder of the video.

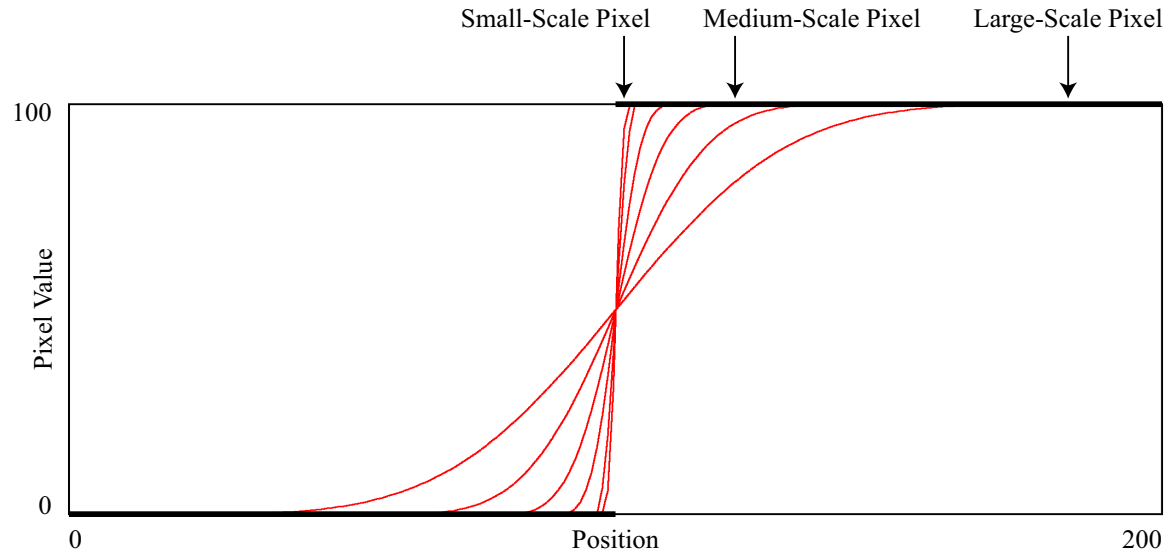


Figure 4: The algorithm computes a set of blurred images (red) for a given color channel (black). A pixel for which all of the images agree is considered a large-scale pixel. If the images disagree, it is a smaller-scale pixel.

5 Evaluation

In this section we evaluate the algorithm on a variety of videos, including footage of challenging real-world scenes and contrived cases designed to test the limits of the algorithm. We discuss quantitative evaluation measures and compare results obtained from different algorithm configurations.

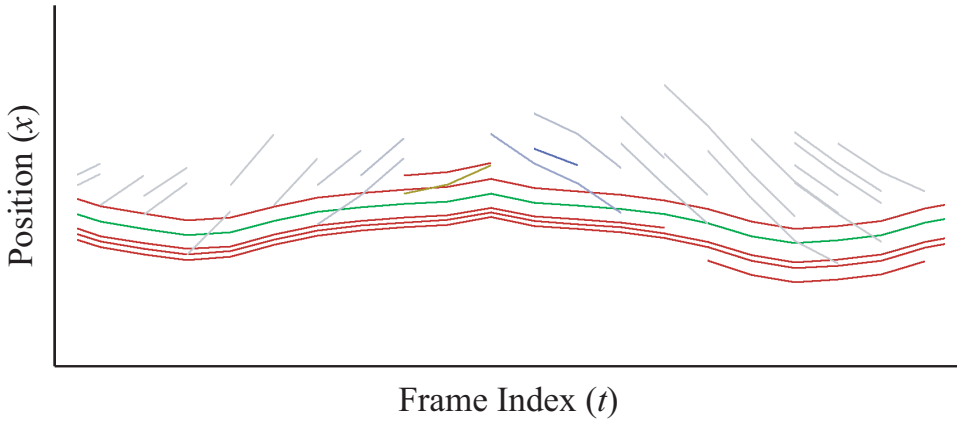


Figure 5: This space-time plot shows a single particle (green) near an occlusion boundary and other particles linked to this particle. The linked particles are shown only for frames in which the links are active. They are colored by link weight; red indicates a high weight and gray indicates a low weight.

5.1 Evaluation Measures

Objectively evaluating the algorithm's correctness 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 to long-range motion estimation.

Standard optical flow evaluation methods are not suitable because our goal is not to estimate optical flow, but instead to provide a higher-level long-range motion representation. We seek to reduce long-range drift, but this does not imply improved optical flow accuracy on a per-frame basis, in part because we do not explicitly represent the motion of every pixel. The optical flow evaluation presented by Baker *et al.* [4] measures per-frame flow accuracy; the publicly available sequences are too short for long-range evaluation.

One solution is rendering synthetic videos with long-range correspondences. To mimic challenging real-world videos, these rendered videos should include deforming objects, complex reflectance, detailed geometry, motion blur, unstabilized camera motion, optical artifacts, and video compression. All of these factors can be obtained using modern commercial rendering software, but setting up a wide variety of photo-realistic scenes would require substantial effort. In the future we envision that creating and rendering such scenes will be easy enough that researchers will produce a diverse set of ground-truth videos.

For the purposes of this paper, we quantify the algorithm's performance using videos that are constructed to return to the starting frame. We replace the second half of each evaluation video with a temporally reversed copy of the first half. We then compute the fraction of particles that survive from the start frame to the end frame (which is identical in appearance to the start frame). For each of these particles, we compute the distance between its (x,y) position in the start frame and (x,y) position in the end frame. This spatial error value should be near zero.

Like many alternative methods, this evaluation scheme is flawed. The algorithm can 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, the algorithm can both increase the survival rate and decrease the spatial error.

Another problem with this return-to-start evaluation is that the algorithm may be able to unfairly recover from mistakes. This prevents a comparison with techniques that refine concatenated flow fields; a good refinement algorithm should be able to find the trivial (zero flow) field mapping the first frame to the last frame, even if it has trouble with intermediate frames.

Because of these issues, we provide the evaluation for descriptive purposes only. These measures should not be used to compare the algorithm with future particle video algorithms.

5.2 Evaluation Videos

Our evaluation dataset consists of 20 videos, representing a range of real-world conditions and contrived test cases. These videos together include a variety of scenes, lighting conditions, camera motions, and object motions.

The videos are recorded at 29.97 non-interlaced frames per second in the MiniDV format using a Panasonic DVX100 camera. The video frames are 720 by 480 pixels with a 0.9 pixel aspect ratio (width/height). Before constructing a particle video, we crop four pixels from the left and right of each frame to remove camera artifacts.

The input videos are summarized in Table 1. For the videos of planar surfaces (VZoomIn, VZoomOut, VRotateOrtho, and VRotatePersp), we replace optical flow estimation with global parametric motion estimation.

Name	Camera Motion	Occlusion	Object Motion	Figure	Frames
VBranches	hand-held R+T	yes	none	8	50
VCars	hand-held R+T	yes	R+T	8	50
VHall	hand-held R+T	yes	none	8	50
VHand	hand-held R+T	yes	R+T; deformation	8	70
VMouth	static	yes	R+T; deformation	8	70
VPerson	tripod R	yes	R+T; deformation	9	50
VPlant	hand-held R+T	yes	none	9	70
VShelf	crane T	yes	none	9	50
VTree	hand-held R+T	yes	R+T; deformation	9	70
VTreeTrunk	hand-held R+T	yes	none	9	50
VZoomIn	static	no	none	N/A	40
VZoomOut	static	no	none	N/A	40
VRotateOrtho	static	no	R	N/A	90
VRotatePersp	static	no	R	N/A	90
VRectSlow	static	yes	R	N/A	80
VRectFast	static	yes	R	N/A	80
VRectLight	static	yes	R	N/A	80
VCylSlow	static	yes	R	N/A	50
VCylFast	static	yes	R	N/A	50
VCylLight	static	yes	R	N/A	50

Table 1: The evaluation videos include various camera motions and object motions. R denotes rotation and T denotes translation.

5.3 Particle Video Configurations

We evaluate several configurations of the particle video algorithm:

- **PVBaseline.** This uses all of the parameter settings described in Section 4 and summarized in Table 2. The following configurations are modifications, as specified, of this configuration.
- **PVSweep1.** This configuration performs a single forward sweep (whereas the baseline algorithm performs a forward sweep followed by a backward sweep).
- **PVSweep4.** This sweeps forward, backward, forward again, then backward again.
- **PVNoOcc.** This configuration ignores the occlusion maps (provided by the optical flow algorithm) during particle propagation (Section 4.3).
- **PVPruneMore.** This configuration lowers the pruning threshold to $\delta = 5$, resulting in more pruning.
- **PVPruneLess.** This configuration raises the pruning threshold to $\delta = 20$, resulting in less pruning.
- **FlowConcat.** This is a simple concatenation of flow fields (computed as described in Section 3) for each particle position in the first video frame (according to the PVBaseline configuration). The flow trajectories are terminated when they enter an occluded region, as determined by the flow algorithm.

5.4 Evaluation Results and Discussion

The return-to-start evaluation is summarized in Figures 6 and 7. In each case, the particles return to their starting positions with lower error than the trajectories formed by concatenating flow vectors. As expected, concatenated flow vectors drift. Ideally the plots should be symmetrical (since the videos are temporally symmetrical); in some cases, the particle trajectories deviate from this symmetry, suggesting occasional failures.

Variable	Description	Value	Units	Section
σ_l	motion difference prior for link weight	1.5	pixels per frame	§4.4
α	particle objective distortion factor	1.5	N/A	§4.5.1
σ_c	channel filter size	5	frames	§4.5.1
σ_t	pruning energy filter size	1	frames	§4.6
δ	pruning energy threshold	5	N/A	§4.6

Table 2: These parameter settings are used for the PVBaseline configuration.

Configuration	Return Fraction	Return Error	Mean Count	Mean Length	Run Time
FlowConcat	0.81	4.05	N/A	N/A	N/A
PVBaseline	0.65	1.12	13260	31.68	40.53
PVSweep1	0.71	0.99	11468	28.96	15.73
PVSweep4	0.66	1.24	14644	30.51	73.65
PVNoOcc	0.66	1.17	13178	32.90	57.47
PVPruneMore	0.43	0.83	14684	23.11	71.69
PVPruneLess	0.75	1.73	13304	37.15	20.11

Table 3: For each configuration, we evaluate the algorithm on videos that are constructed to return to the start frame (Section 5.1). We report the mean fraction of particles that survive to the end frame and the mean spatial distance between the each surviving particle's start and end frame positions. We also give the mean particle count, mean particle length, and mean per-frame running time. The running time does not include optical flow computation; it is a pre-process shared by all the algorithms. All statistics are averaged over the 20 videos described in Section 5.2.

The yellow lines indicate the fraction of surviving particles. For each video, particles disappear because they leave the frame boundaries or become occluded (so a 100% survival rate would be incorrect). A roughly constant survival fraction across the second half (returning to the start) indicates that few particles are lost for other (spurious) reasons.

Table 3 provides a comparison of the algorithm configurations described in Section 5.3. As expected, ignoring the occlusion masks provided by the flow algorithm results in higher error and a larger fraction of surviving particles. Also, as expected, additional pruning raises the accuracy while lowering the survival fraction. Simple flow concatenation results in a better survival rate, but also significantly higher error. (These results do not conclusively show that the flow approach is worse than the particle approach.)

Additional sweeps across the video add more particles, mostly in areas where other particles were previously pruned (the more difficult regions of the video). Thus, even though a single sweep has lower error, it is not necessarily providing a better model of the motion. (This is why, as discussed in Section 5.1, the return-to-start measure should not be used alone to evaluate particle videos.)

Table 4 gives a breakdown of the running time for each configuration. In each configuration, almost half the running time is consumed by running the sparse linear system solver. The remaining time is mostly spent constructing the linear system. The computational costs of adding, linking, and pruning particles are all relatively small.

All of the data used to generate these results, including the videos, plots, and particle trajectories are available online at `http://rvsn.csail.mit.edu/pv/`.

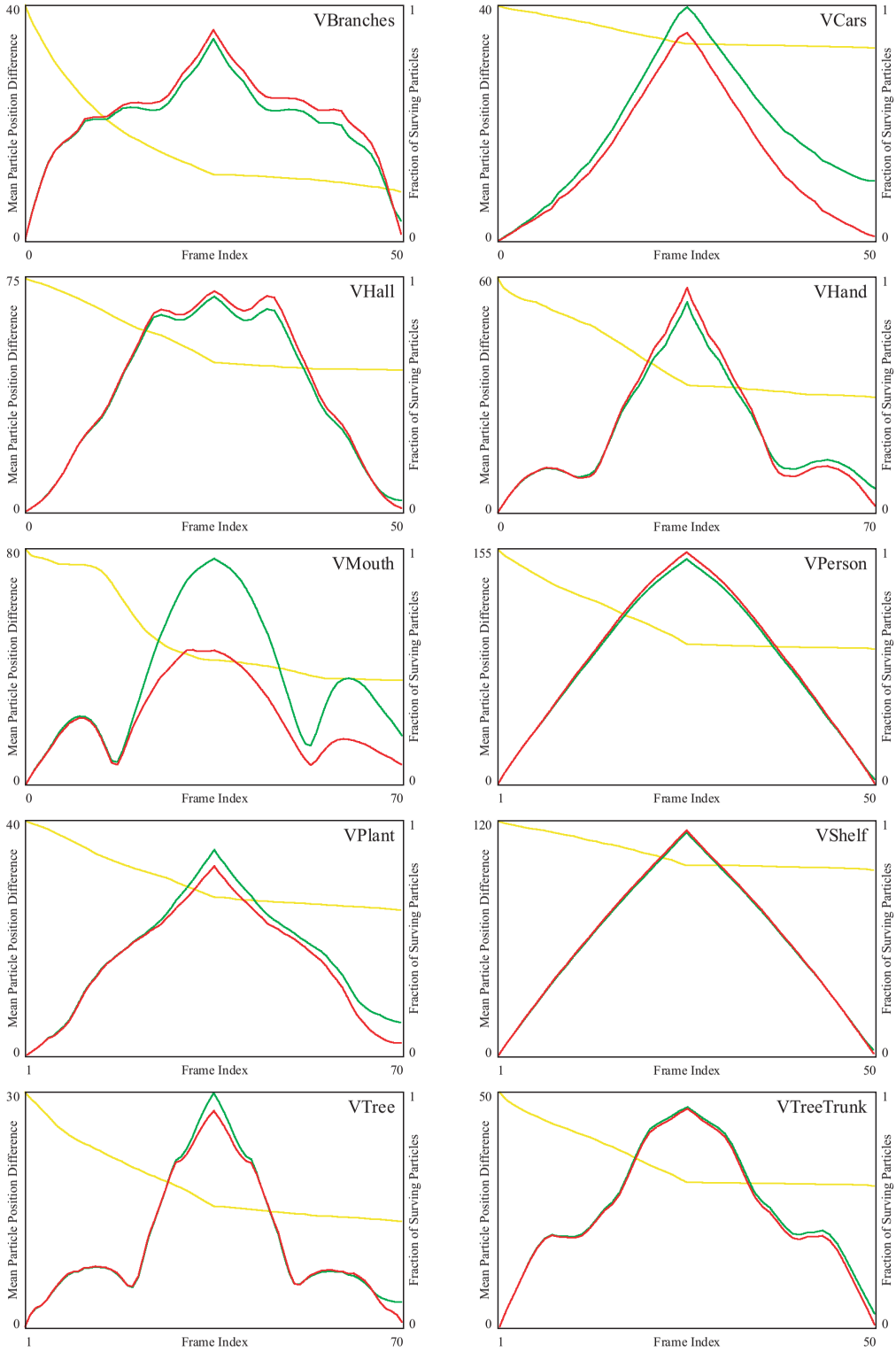


Figure 6: Each plot shows the fraction of surviving particles (yellow, right axis) and mean distance (red, left axis) of the surviving particles from their positions in the start frame. The green lines denote concatenated flow vectors. As described in Section 5.2, the videos are temporally mirrored, so we expect all unoccluded particles to return to their start positions.

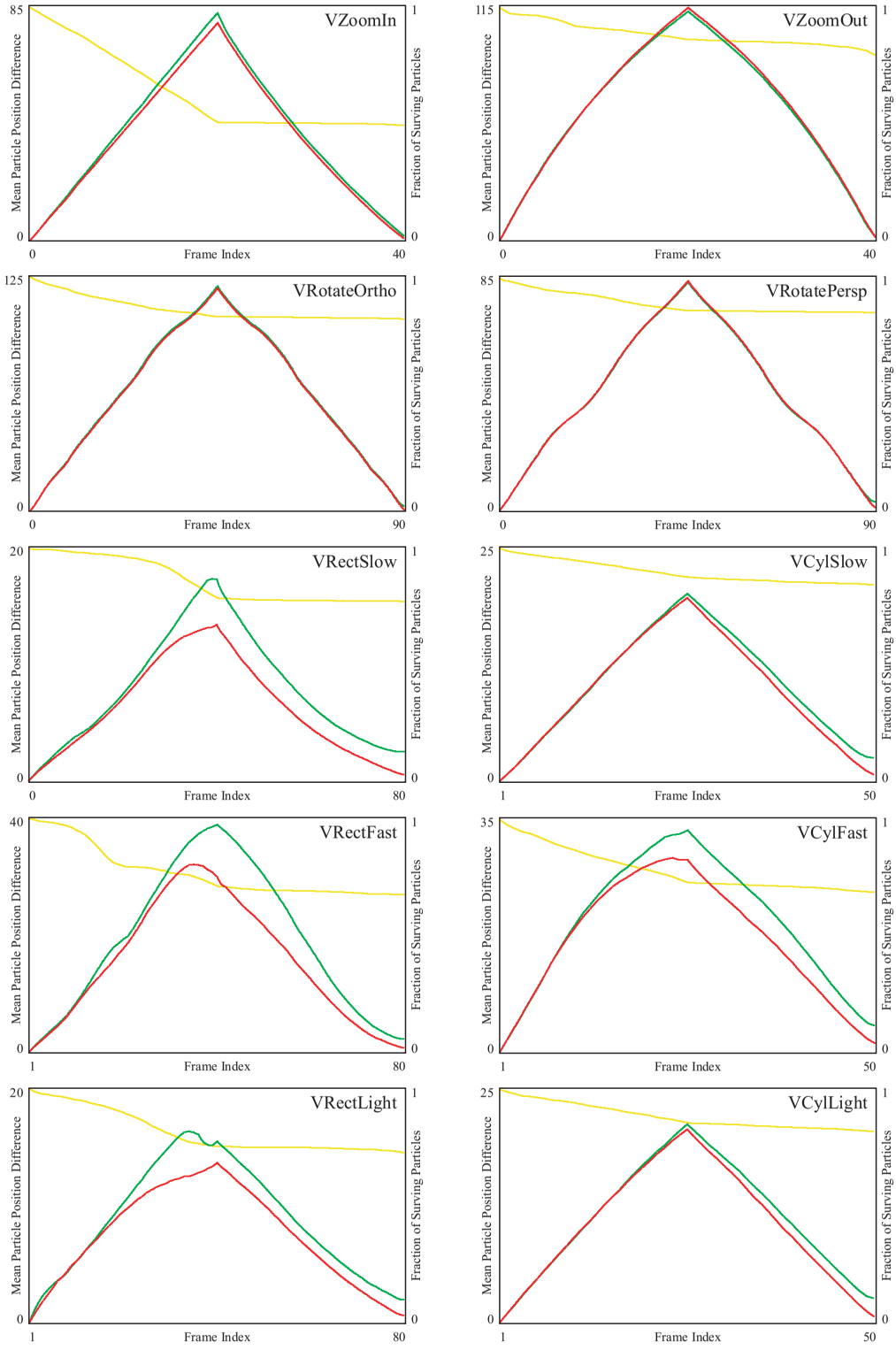


Figure 7: Each plot shows the fraction of surviving particles (yellow, right axis) and mean distance (red, left axis) of the surviving particles from their positions in the start frame. The green lines denote concatenated flow vectors. As described in Section 5.2, the videos are temporally mirrored, so we expect all unoccluded particles to return to their start positions.

Configuration	Add Time	Link Time	Opt. Time	Solver Time	Prune Time	Total Time
PVBaseline	2.99	1.02	11.57	18.54	1.39	40.53
PVSweep1	1.29	0.41	4.14	7.27	0.40	15.73
PVSweep4	6.14	2.05	21.06	31.55	3.01	73.65
PVNoOcc	3.62	0.79	17.36	27.47	2.01	57.47
PVPruneMore	4.21	3.39	21.01	32.42	3.58	71.69
PVPruneLess	2.16	0.83	4.70	8.57	0.45	20.11

Table 4: For each configuration, we report the mean per-frame running time in seconds. The **Opt.** time includes optimization overhead but not execution of the solver (which is reported in its own column). The total time includes some additional overhead, such as computing the adaptive scale map factor (Section 4.7).

6 Future Work

The largest difficulty in creating a particle video is handling occlusion boundaries. The current implementation represents occlusion boundaries using weighted links between particles. This linking scheme fails because it occasionally allows incorrect distortion or prevents correct distortion (such as that caused by non-rigid object deformation or changes in viewpoint). We hope to explore statistical and/or geometric methods for distinguishing correct and incorrect distortion.

The best approach may involve a hybrid of flow-based and particle-based occlusion handling. Flow methods provide the advantage of accounting for subtle image details, while particle methods provide easier handling of long temporal ranges (and indeed we expect occlusions to be clearest in long temporal ranges). A single optimization could include both flow and particle objectives, possibly estimating flow over a range of different temporal scales. This optimization could be directed toward occlusions by identifying high-error or high-flow-gradient manifolds in the spatiotemporal video volume.

Particle motion spaces could provide an alternative approach to the occlusion problem. Each particle could be projected into a space such that particles with similar motion are close to one another and particles with different motions are not. This would allow efficient querying to find a set of particles with motion similar to a given particle (extending beyond the set of particles linked to the given particle). One option would be assigning a particle trajectory distance to each link (as is currently done in Section 4.4) then running Isomap [33] to project all of the particles into a low-dimensional (perhaps 2D or 3D) space. Independently moving objects should appear as distinct clusters in the space (and certain motion patterns should appear as filaments or other manifolds through the motion space).

Another aspect of occlusion handling is deleting and creating particles in areas that become occluded or disoccluded. We could spatially regularize addition and pruning, but this is difficult because a slow-moving occlusion boundary may result in only a few particles being added/deleted in any given frame (in fact, we should allow singleton additions and deletions). A better option may be using the gradient of the particle motion field to modulate the density (placing more particles near occlusions boundaries and fewer in areas of uniform motion), rather than determining particle density solely by image scale.

We could also explore world-space constraints for particle optimization. We have avoided geometric constraints because the algorithm must be good at handling non-rigid cases. However, once the non-rigid cases are well-modelled, we can obtain further performance gains by using geometric rules to improve the rigid cases.

In the future we would also like to develop a stronger theoretical framework for the particle video problem. One option is to utilize a flow-based representation, which can be viewed as the derivative of a particle-based representation. The main goal of the particle video algorithm is to move beyond a flow-based representation, but it may be that our theoretical reasoning about particles will have to occur in the derivative/flow domain. Much theoretical work has already been done in the area of optical flow; one challenge would be augmenting

this with long range constraints that make statements about video properties along trajectories obtained from integrals of flow-based representations. This approach could borrow mathematical machinery from differential equations and applications of differential equations, such as fluid dynamics.

Ideally a theoretical particle framework would motivate a simple particle estimation algorithm. The current algorithm has an unsatisfying number of steps and parameters. We should aim to simplify the algorithm while simultaneously improving its accuracy.

7 Conclusion

The particle video algorithm provides a new approach to motion estimation, a central problem in computer vision. Long-range video correspondences could improve methods for many 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 approach differs from optical flow by enforcing long-range appearance consistency and motion coherence.

Current limitations of the particle video algorithm arise from our methods for positioning particles, rather than a fundamental limitation of the particle representation. Starting with the particle tools presented in this paper, we believe researchers will soon develop better particle video algorithms. By making our data and results available online, we hope others will explore the particle video problem.

A Appendix: Optical Flow Implementation

Our particle video algorithm uses optical flow fields as an input. The optical flow estimation can be substituted with any other optical flow method, but for completeness we give details of our flow algorithm here.

A.1 Variational Flow Optimization

Our variational flow optimization is adapted from Brox *et al.* [13]. The approach has proved successful because it makes relatively few simplifications of the functional.

A.1.1 Objective Function

Let $u(x,y,t)$ and $v(x,y,t)$ denote the components of an optical flow field that maps image point $I(x,y,t)$ to an image point in the next frame:

$$I(x+u(x,y,t),y+v(x,y,t),t+1). \tag{16}$$

Like many optical flow methods, the Brox *et al.* [13] objective function combines a data term and smoothness term:

$$E_{Flow}(u,v,t) = E_{FlowData}(u,v,t) + E_{FlowSmooth}(u,v,t). \tag{17}$$

Although these terms are motivated as functionals, for clarity we give them in discrete form, in which u and v are estimated at integer indices.

A.1.2 Data Term

In our algorithm, 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 A.2):

$$E_{FlowData}(u,v,t) = \sum_{x,y,k} r(x,y,t)\Psi([I^{[k]}(x+u(x,y,t),y+v(x,y,t),t+1) - I^{[k]}(x,y,t)]^2). \tag{18}$$

Here k is summed over image channels. We use the same robust norm as Brox *et al.* [13]:

$$\Psi(s^2) = \sqrt{s^2+\varepsilon^2};\ \varepsilon = 0.001. \tag{19}$$

This function, a differentiable form of the absolute value function, does not respond as strongly to outliers as the standard L^2 norm.

The original Brox *et al.* [13] formulation analytically enforces constancy of the image gradient (and optionally other linear differential operators [25]), whereas we simply treat the gradient as another image channel. Specifically, we use image brightness I (range $[0,255]$), the green minus red color component, the green minus blue color component, and the x and y derivatives of brightness (I_x and I_y). We scale the color difference channels by 0.25 to reduce the impact of color sampling/compression artifacts common in video. These additional channels do not substantially increase the algorithm's running time because they do not increase the number of terms in the sparse linear system that consumes the majority of the computation.

A.1.3 Smoothness Term

As in the Brox *et al.* [13] algorithm, the smoothness term measures the variation of the flow field using the robust norm Ψ. We modify the smoothness term to discourage flow discontinuities at locations with small image gradients:

$$E_{FlowSmooth}(u,v,t) = \sum_{x,y}(\alpha_g + \alpha_l \cdot b(x,y,t)) \cdot \Psi(u_x(x,y,t)^2 + u_y(x,y,t)^2 + v_x(x,y,t)^2 + v_y(x,y,t)^2). \tag{20}$$

Here α_g is a global smoothness factor (equivalent to the α parameter in the original Brox *et al.* [13] formulation) and α_l is a local smoothness factor, which is modulated by the local smoothness $b(x,y,t)$ (Figure 10).

We compute local smoothness using a Gaussian prior on the image gradient:

$$b(x,y,t) = N(\sqrt{\frac{\partial}{\partial x}I(x,y,t)^2 + \frac{\partial}{\partial y}I(x,y,t)^2}; \sigma_b). \tag{21}$$

Here N denotes a zero-mean non-normalized Gaussian. We set $\sigma_b = 2$, $\alpha_l = 15$, and $\alpha_g = 10$, based on a variety of flow experiments.

A.1.4 Sparse Linear System

We optimize the objective function using a fixed-point scheme, similar to the algorithms of Brox *et al.* [13] and Bergen *et al.* [8]. The optimizer iteratively updates the flow field using a sparse linear system determined by the current flow field. To construct the sparse linear system, we take discrete derivatives of the objective function, as described in Sand [26].

The algorithm solves the sparse linear system using successive over-relaxation method (SOR) [5]. At a given resolution level, the algorithm makes 3 fixed-point steps, each consisting of 500 SOR iterations.

A.2 Occlusion Detection

Handling occlusions is the most challenging aspect of building a particle video. It is also the most challenging part of optical flow estimation, stereo reconstruction, feature tracking, and motion estimation in general.

Rather than solving the problem purely with particles, we use optical flow estimation to provide information about occlusions. Ideally, the flow estimates will be able to incorporate subtle details of the surface being occluded, which are not necessarily captured by the particles.

The Brox *et al.* [13] algorithm uses the robust distance function Ψ to handle occlusions. As discussed in Section 2.2, using robustness to account for occlusion boundaries is not ideal. Rather than properly modeling the physical behavior of the occlusion boundary, the algorithm is simply allowed to fail (with a small penalty due to the robust distance function). In practice, the Brox *et al.* [13] algorithm produces flow fields that incorrectly group occluded pixels with the occluding object, because this produces a lower objective value.

Like other approaches [31, 1, 32, 36], we model occlusion by explicitly labelling occluded pixels. Once the pixels are labelled, they can be excluded from the data term, rather than incorrectly matched with non-occluded pixels. A flow field augmented with an occlusion mask correctly models the fact that some pixels disappear.

Our algorithm uses a combination of flow divergence and pixel projection difference to identify occluded pixels. The divergence of an optical flow field distinguishes between different types of motion boundaries:

$$div(x,y,t) = \frac{\partial}{\partial x}u(x,y,t) + \frac{\partial}{\partial y}v(x,y,t). \tag{22}$$

Flow divergence is positive for disoccluding boundaries, negative for occluding boundaries, and near zero for shear boundaries (Figure 11). To select occluding boundaries, but not disoccluding boundaries, we define a one-sided divergence function d:

$$d(x,y,t) = \begin{cases} div(x,y,t) & div(x,y,t) < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Flow divergence also occurs with visual expansion and contraction, but typically at a lower magnitude than arises at an occlusion boundary.

Pixel projection difference provides another occlusion cue:

$$e(x,y,t) = I(x,y,t) - I(x+u(x,y,t), y+v(x,y,t), t+1). \tag{24}$$

We combine the one-sided divergence and pixel projection using zero-mean non-normalized Gaussian priors:

$$r(x,y,t) = N(d(x,y,t);\sigma_d) \cdot N(e(x,y,t);\sigma_e). \tag{25}$$

The $r(x,y,t)$ values are near zero for occluded pixels and near one for non-occluded pixels. We set $\sigma_d = 0.3$ and $\sigma_e = 20$ based on experimental observation of occluded regions.

A.3 Bilateral Flow Filtering

Detecting occluded pixels is a key part of the occlusion modelling process, but we must still handle the mixing of pixel properties across boundaries. This mixing occurs for all types of motion boundaries: disocclusions, occlusions, and shear motions. To improve boundary sharpness, we use a bilateral filter based on the work of Xiao *et al.* [36].

Xiao and colleagues motivate the approach by pointing out an equivalence between variational smoothness optimization and Gaussian filtering of the flow fields. Using this observation, they replace traditional anisotropic

Variable	Description	Value	Units	Section
η	multi-resolution scale factor	0.9	N/A	§3
α_g	global smoothness factor	10	N/A	§A.1.1
α_l	local smoothness factor	15	N/A	§A.1.1
σ_b	image gradient prior	2	pixel value gradient	§A.1.1
σ_d	flow divergence prior	0.3	flow gradient	§A.2
σ_e	pixel mismatch prior	20	pixel values	§A.2
σ_x	bilateral filter size	4	image space	§A.3
σ_i	filter image difference	7.5	pixel values	§A.3
σ_m	filter motion difference	0.5	flow values	§A.3
σ_g	flow gradient filter	3	image space	§A.3

Table 5: We use these optical flow parameter settings for our experiments.

regularization with a filter that better separates distinct motions.

The filter sets each flow vector to a weighted average of neighboring flow vectors:

$$u'(x,y,t) = \frac{\sum_{x_1,y_1} u(x_1,y_1,t)w(x,y,x_1,y_1,t)}{\sum_{x_1,y_1} w(x,y,x_1,y_1,t)}. \tag{26}$$

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

$$\begin{aligned} w(x,y,x_1,y_1,t) \quad = \quad & N(\sqrt{(x-x_1)^2+(y-y_1)^2};\sigma_x) \\ & \cdot N(I(x,y,t)-I(x_1,y_1,t);\sigma_i) \\ & \cdot N(\sqrt{(u-u_1)^2+(v-v_1)^2};\sigma_m) \\ & \cdot r(x_1,y_1,t) \end{aligned} \tag{27}$$

Here u denotes $u(x,y,t)$ and u_1 denotes $u(x_1,y_1,t)$ (and v similarly). We set $\sigma_x = 4$, $\sigma_i = 7.5$, $\sigma_m = 0.5$, and restrict (x_1,y_1) to lie within 10 pixels of (x,y).

This filter computes weights for a neighborhood of pixels around each pixel, so it is quite computationally expensive. Thus, for efficiency, we apply the filter only near flow boundaries, which we localize using the flow gradient magnitude:

$$g(x,y,t) = \sqrt{u_x^2(x,y,t)+u_y^2(x,y,t)+v_x^2(x,y,t)+v_y^2(x,y,t)} \tag{28}$$

The algorithm filters $g(x,y,t)$ using a spatial Gaussian kernel ($\sigma_g = 3$), producing a smoothed gradient magnitude $\hat{g}(x,y,t)$. Note that, unlike the divergence, this gradient magnitude is large for all types of motion boundaries (occlusions, disocclusions, and shear boundaries). We apply the bilateral filter (Equation 26) to pixels with $\hat{g}(x,y,t) > 0.25$.

Table 5 summarizes the parameters for our complete optical flow algorithm. Figure 12 shows flow fields generated by the algorithm.

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. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004.

[4] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV*, page (to be determined), 2007.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Society for Industrial and Applied Mathematics, 1994.

[6] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994.

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

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

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

[10] M. J. Black. Recursive non-linear estimation of discontinuous flow fields. In *ECCV*, pages 138–144, 1994.

[11] M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

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

[13] 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.

[14] T. M. Chin, W. C. Karl, and A. S. Willsky. Probabilistic and sequential computation of optical flow using temporal coherence. *IEEE Trans. Image Processing*, 3(6):773–788, 1994.

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

[16] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. Improving feature tracking with robust statistics. *Pattern Analysis and Applications*, 2(4):312–320, 1999.

[17] D. B. Goldman, B. Curless, D. Salesin, and S. M. Seitz. Interactive video object annotation. Technical Report UW-CSE-2007-04-01, University of Washington, 2007.

[18] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, UK, 2000.

[19] H. W. Haussecker and D. J. Fleet. Computing optical flow with physical models of brightness variation. *PAMI*, 23(6):661–673, 2001.

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

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

[22] C. Liu, A. Torralba, W. T. Freeman, F. Durand, and E. H. Adelson. Motion magnification. *ACM Trans. Graph.*, 24(3):519–526, 2005.

[23] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[24] D. W. Murray and B. F. Buxton. Scene segmentation from visual motion using global optimization. *PAMI*, 9(2):220–228, 1987.

[25] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 67(2):141–158, 2006.

[26] P. Sand. *Long-Range Video Motion Estimation using Point Trajectories*. PhD thesis, MIT, 2006.

[27] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *CVPR*, pages 2195–2202, 2006.

[28] H. S. Sawhney, Y. Guo, K. Hanna, R. Kumar, S. Adkins, and S. Zhou. Hybrid stereo camera: an IBR approach for synthesis of very high resolution stereoscopic image sequences. In *SIGGRAPH*, pages 451–460, 2001.

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

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

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

[32] 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.

[33] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2322, 2000.

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

[35] J. Weickert, A. Bruhn, N. Papenberg, and T. Brox. Variational optic flow computation: From continuous models to algorithms. In *International Workshop on Computer Vision and Image Analysis*, pages 1–6, 2004.

[36] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *ECCV*, pages 211–224, 2006.

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

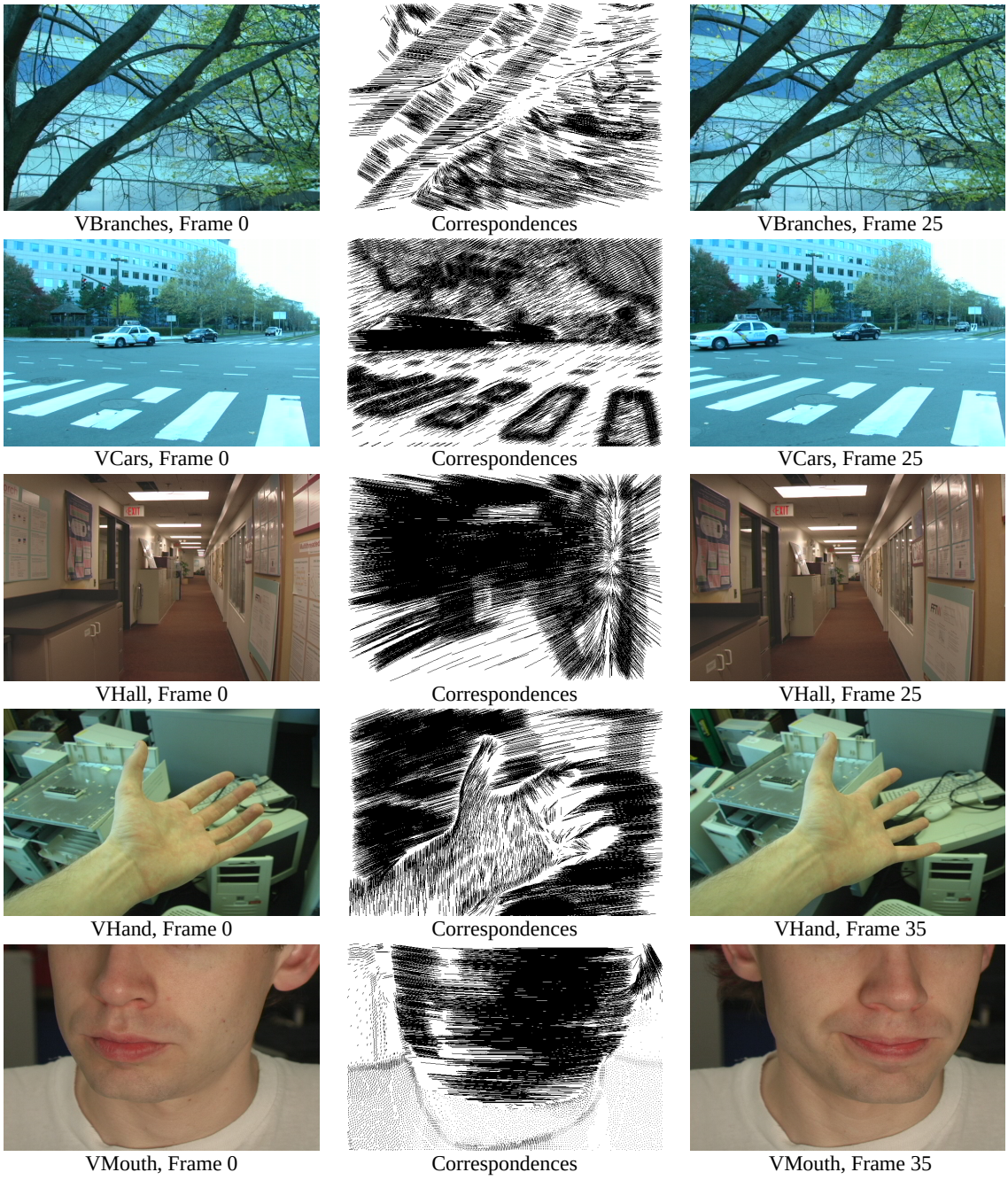


Figure 8: Each row shows a frame pair from one test video. Correspondences are shown for particles in common between the frames.

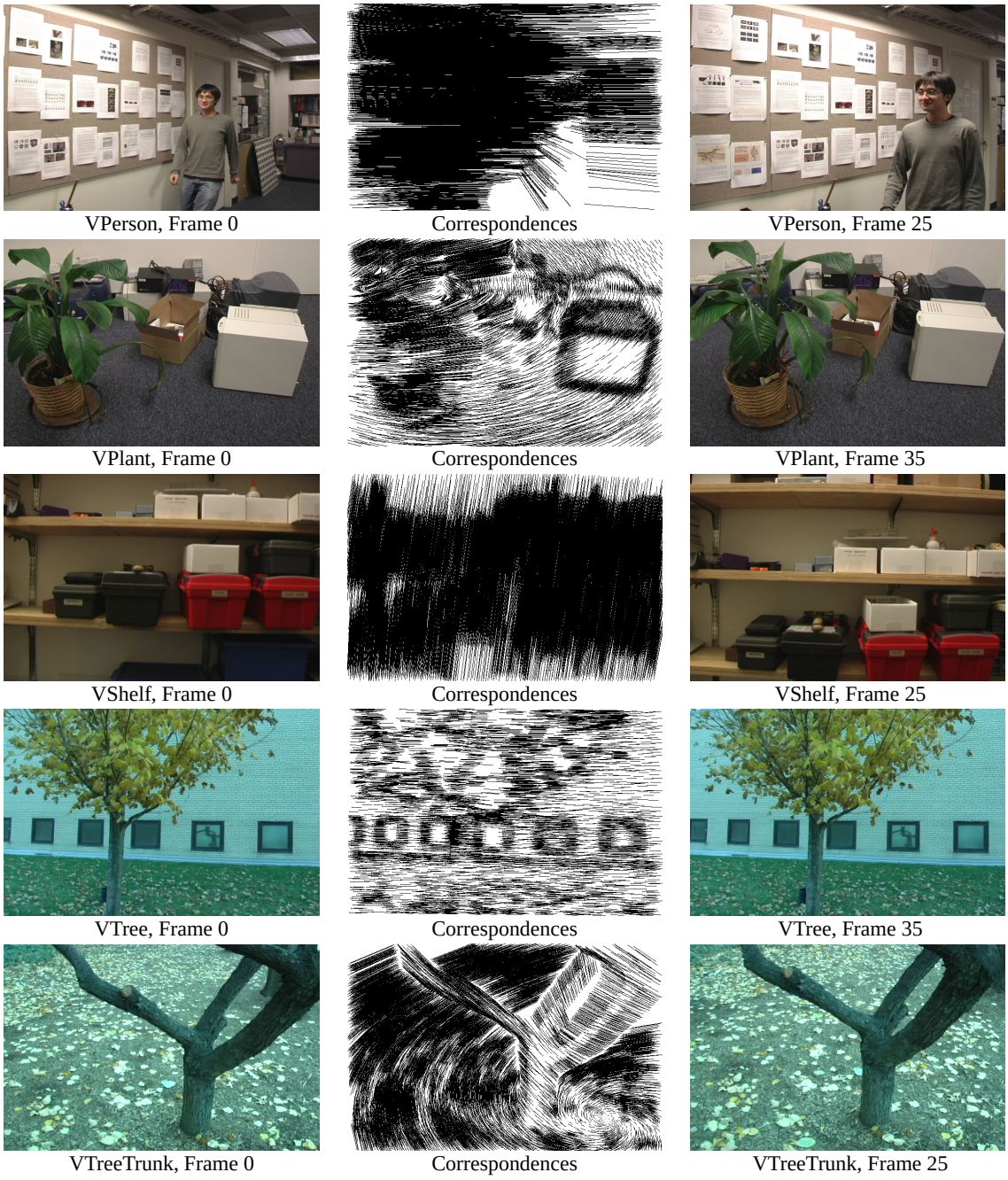


Figure 9: Each row shows a frame pair from one test video. Correspondences are shown for particles in common between the frames.

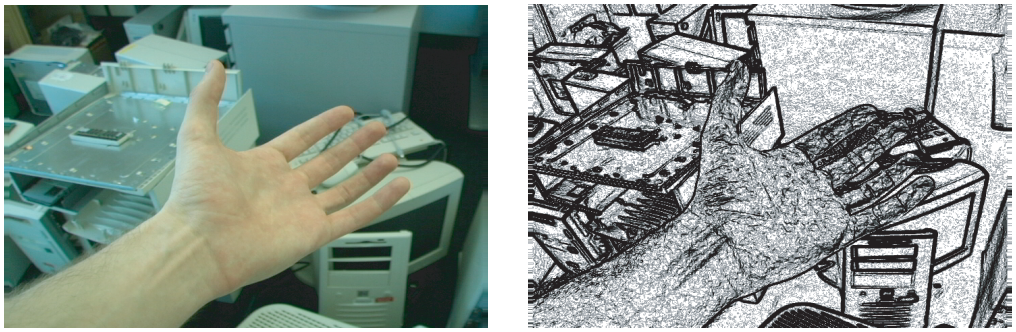


Figure 10: The local smoothness image modulates the smoothness term in the optical flow objective function. The objective discourages flow discontinuities in uniform image regions.

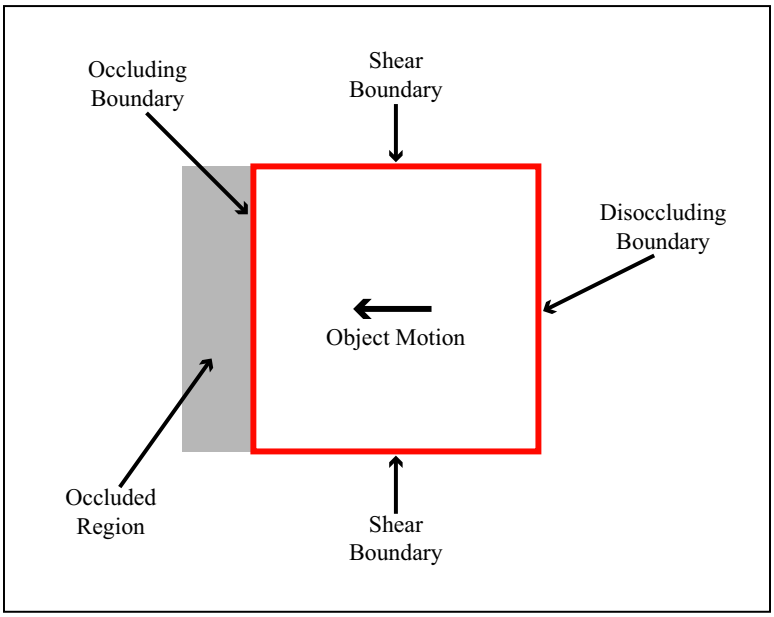


Figure 11: In this diagram, the motion discontinuities (red) include occluding boundaries, disoccluding boundaries, and shear boundaries. The occluded region is the set of pixels that are not visible in the subsequent frame.

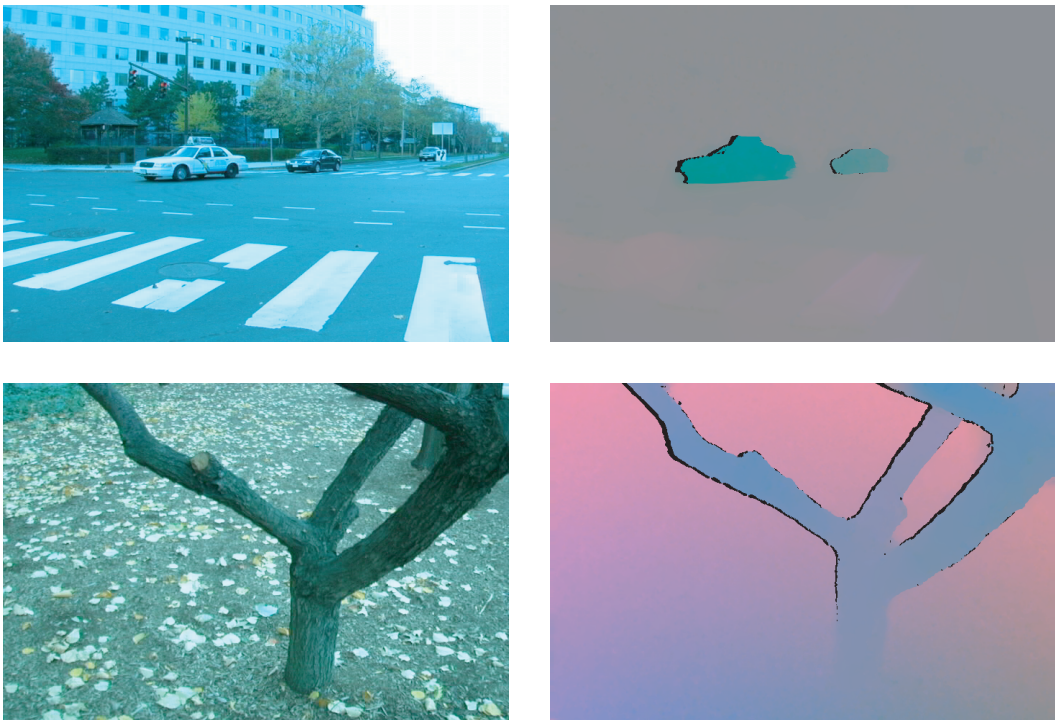


Figure 12: Each flow field is generated between a video frame (left) and the subsequent video frame. The flow field is visualized (right) using hue to denote flow direction and saturation to denote flow magnitude. The black regions are labelled by the algorithm as occluded.