Particle Video: Long-Range Motion Estimation Using Point Trajectories

Peter Sand , Seth Teller

Presented by

Jednipat Moonrinta

Computer Science and Information Management
Asian Institute of Technology
July 14, 2009

Outline

- Abstract
- Keywords
- Introduction
- Related work
- Optical Flow Algorithm
- Particle Video Algorithm
- Evaluation
- Conclusion

Abstract

- Represent video motion using a set of particles.
- Each particle is an image point with a long-duration trajectory and other properties.
- To optimize particles trajectories
  - measure consistency along the particle trajectories
  - measure distortion between particles

Keywords

- **Video motion estimation**
  - The problem for video motion estimation is how to adequately represent the changes, or differences, between two video frames.
- **Feature tracking**
  - follows a sparse set of salient image points **over many frames**.

Keywords (cont.)

- **Optical flow**
  - estimates a dense motion field from **one frame to the next**. Generally, optical flow corresponds to the motion field, but not always.

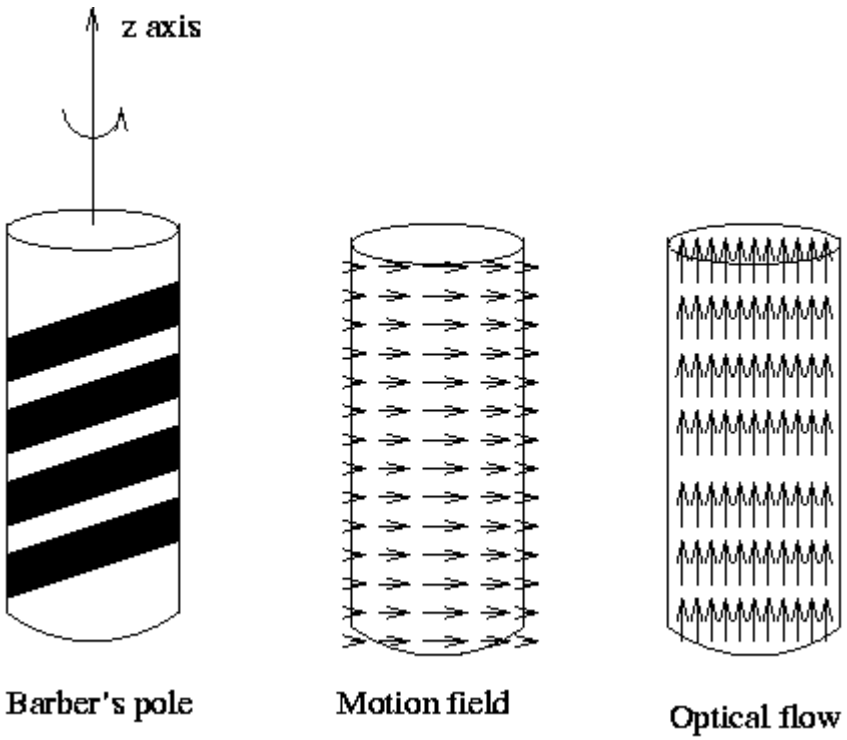


Fig.1 The motion field and optical flow of a barber's pole.

Introduction

- Normally, video motion estimation is performed using feature tracking or optical flow.
- Their goal is to combine feature tracking and optical flow to produce motion estimation that are both long-range and moderately dense.

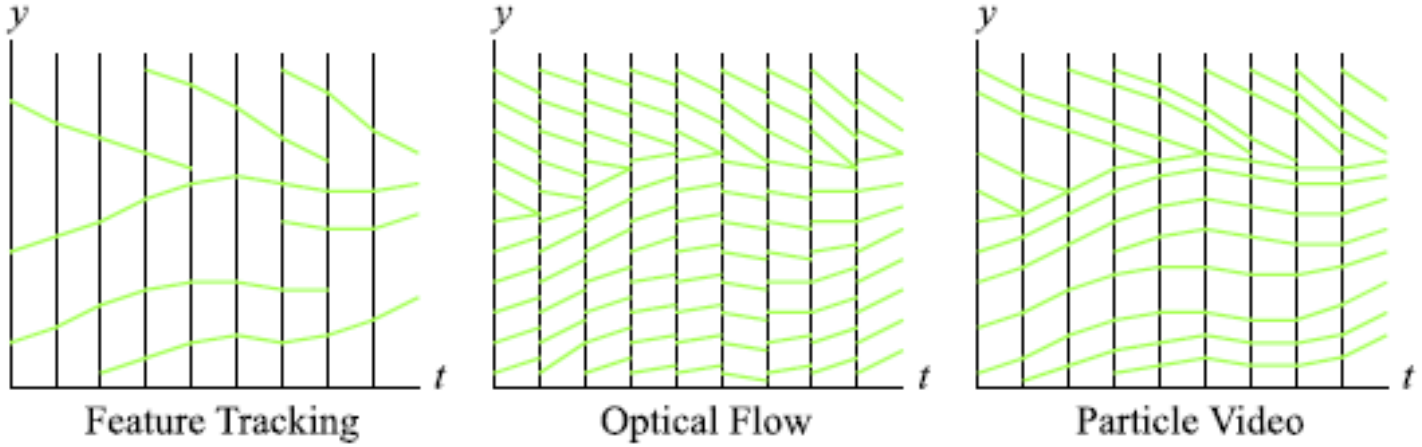


Fig.2 Each diagram represents point correspondences between frames of a hypothetical sequence. Feature tracking is long-range but sparse. Optical flow is dense but short-range. Their particle video representation is denser than feature tracking and longer range than optical flow.

Introduction (cont.)

- **Useful**

  - Multiple observations of each scene point

  can be combined for super-resolution, noise removal and segmentation.

- **Particle Video Representation**

  - Their approach represents video motion using a set of particles that move through the time.

  - Each particle denotes an interpolated image point sample with a long-duration trajectories.

Related Work

- **Multi-Frame Optical Flow**
  - Most optical flow algorithms estimate correspondences between a pair of image, but some use more than two images.
- **Occlusion Detection for Optical Flow**
  - Normally, handling occlusion boundaries is robustness in *data* and *smoothness* term.

Optical Flow Algorithm

- Their optical flow algorithm combines the variational approach of Brox et al.(2004) with bileteral filtering approach of Xiao et al.(2006).
- Their particle video algorithm uses frame-to-frame optical flow to provide an **initial guess for particle motion**.
- **They estimate optical flow independently for each frame pair**.
- The algorithm optimizes a flow field over a *sequence of resolutions*[1] (as shown in Fig.3).

1: Sequence of resolution is obtained by recursively reducing the original resolution by a factor *n=0.9* and the smallest value is *n=0.05* (i.e. if input video has resolution 720x480 pixels, the largest resolution is 648x432 and smallest resolution is 36x24 pixels)

Optical Flow Algorithm (cont.)

- After scaling image, they apply a Gaussian smoothing filter with $\sigma = 1$.

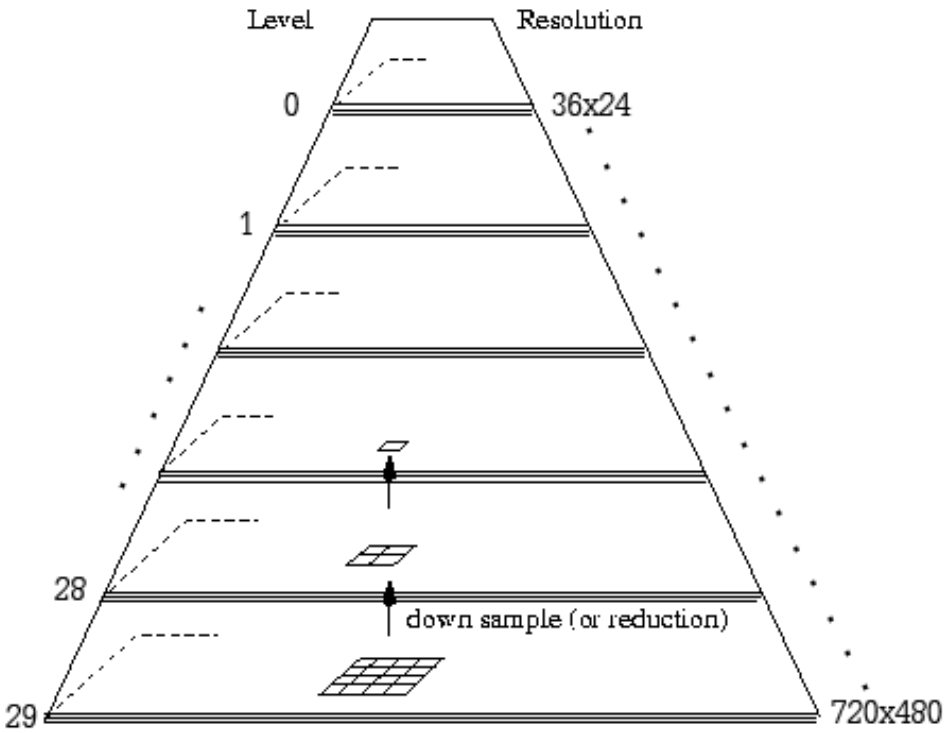


Fig.3 Sequence of resolutions

Optical Flow Algorithm (cont.)

- At each resolution, the algorithm performs the following 3 steps:

Step 1: optimize the flow field using a variational objective with robust *data* and *smoothness* terms
- *data term* measures the variation of a multi-channel image
- *smoothness term* measures the variation of the flow field using the robust norm

Optical Flow Algorithm (cont.)

Step 2: identify the occluded image regions using flow field divergence and pixel projection difference

- The divergence[1] of an optical flow field distinguishes between different types of motion boundaries.

- Pixel projection difference represents difference of image point between frame *t* and *t+1*.

1: In vector calculus, divergence is an operator that measures the magnitude of vector field. The divergence of vector field is a (signed) scalar.

Optical Flow Algorithm (cont.)

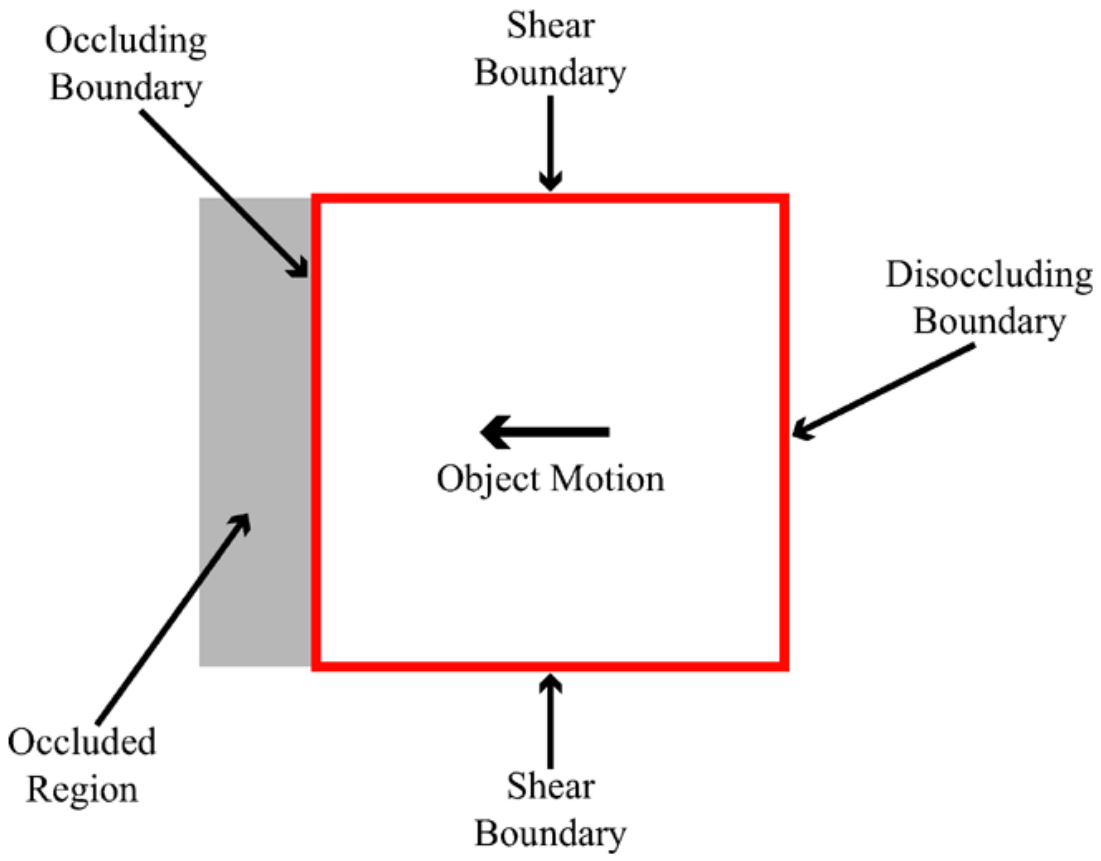


Fig.4 In this diagram, the motion discontinuities (*red) include occluding boundaries, disoccluding boundaries, and* shear boundaries. The occluded region is the set of pixels that are not visible in the subsequent frame

Optical Flow Algorithm (cont.)

Step 3: improve flow boundaries using a occlusion-aware bilateral filter

- to improve boundary sharpness
- the filter computes weights for a neighborhood of pixels around each pixel

Optical Flow Algorithm (cont.)

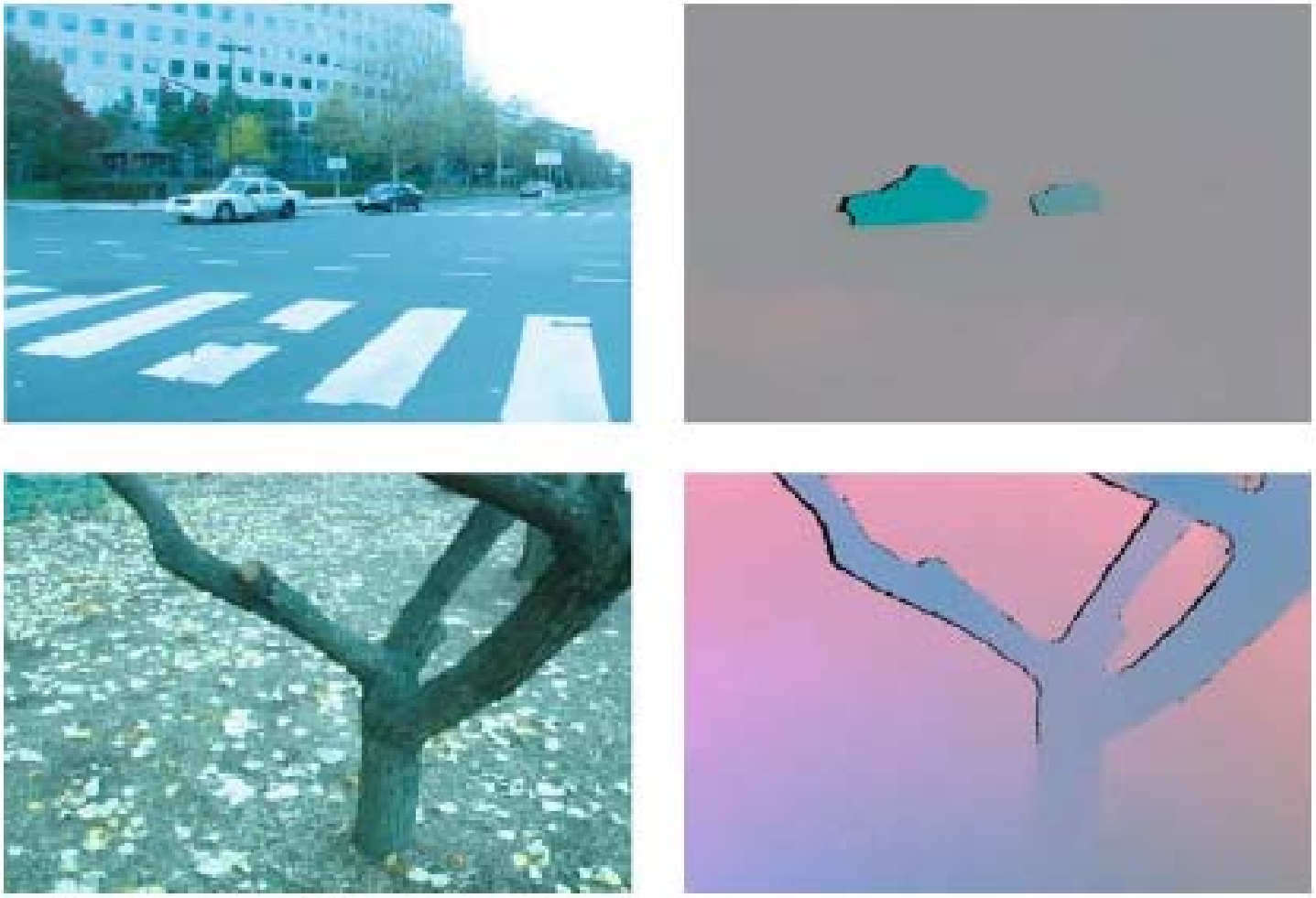


Fig.5 shows flow field generated by optical flow algorithm. Each flow field is generated between a video frame(left) and the subsequent video frame. The flow field is visualized (right) using hue to denote flow direction and saturation to denote flow magnitude. The black regions are labeled by the algorithm as occluded.

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.

Top-Level Particle Video Algorithm

- Their algorithm builds a particle video by moving forward and backward across the video.
- By moving through the video in both directions, new particles can be extended in both directions.

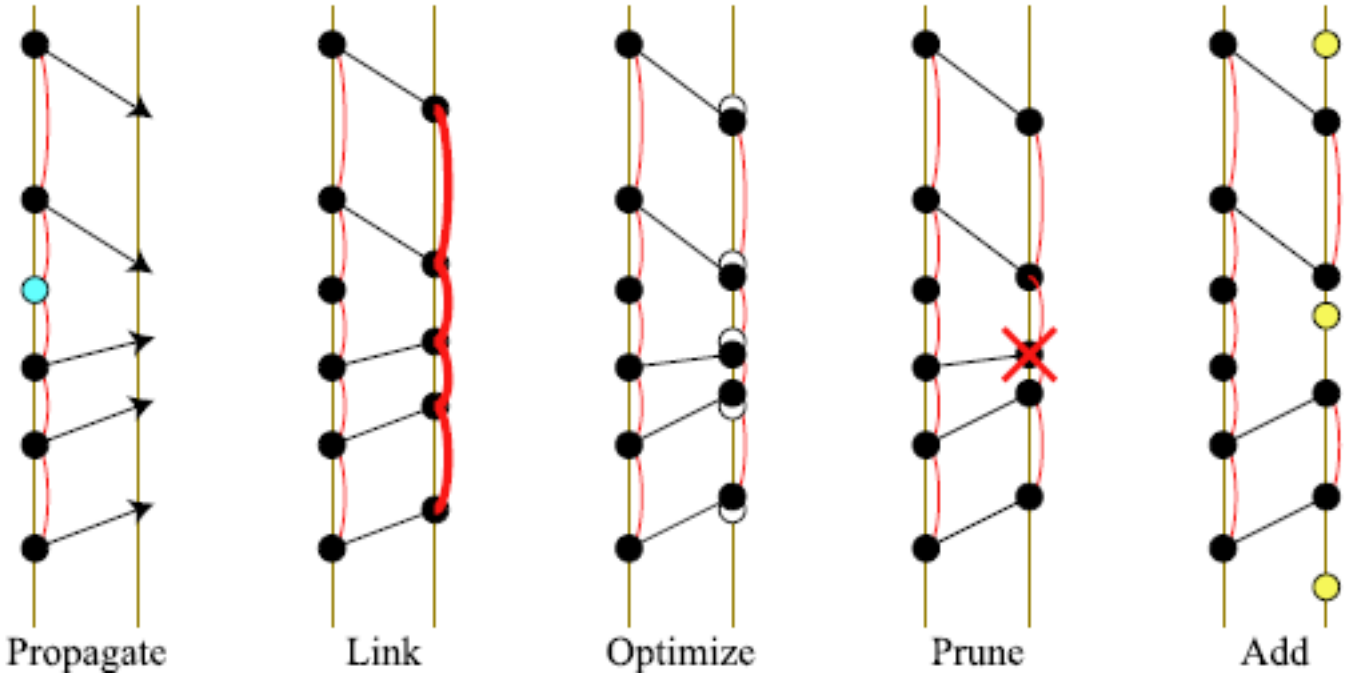
Top-Level Particle Video Algorithm (cont.)

For each processed frame, the following step are performed :

- **Propagation.** Particles terminating in adjacent frame are extended into the current frame according to forward and reverse flow field.
- **Linking.** Particle links are updated.
- **Optimization.** Particle positions are optimized.
- **Pruning.** Particles with high post-optimization error are pruned.
- **Addition.** New particles are add in gaps between existing particles.

Top-Level Particle Video Algorithm (cont.)

Fig.6 Each plot denotes a pair of consecutive frames. The algorithm propagates particles (*black)* 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.



Particle Channels

- The particle video algorithm uses the same 5 channels as the flow estimation algorithm:
  - image brightness
  - green minus red channel
  - green minus blue channel
  - x gradient
  - y gradient

  k denotes the channel index; at time *t* the *k*th image channels is $I^{[k]}(t)$.

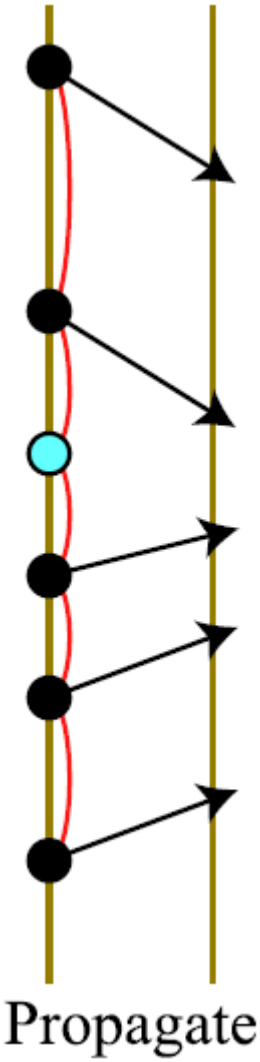
Propagating Particles

- To propagate particle *i* from frame *t-1* to *t*, they use the flow field $u(x, y, t-1)$, $v(x, y, t-1)$:

$$x_i(t) = x_i(t-1) + u(x_i(t-1), y_i(t-1), t-1)$$

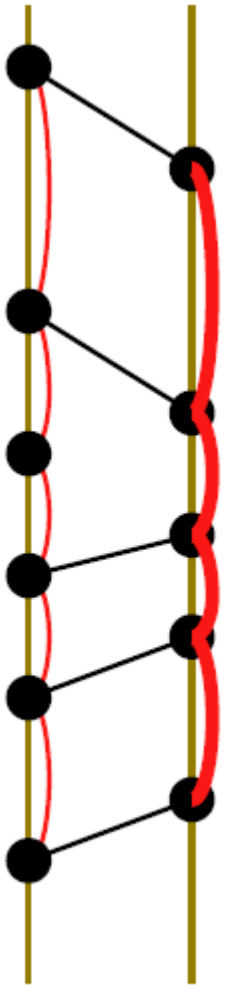
$$y_i(t) = y_i(t-1) + v(x_i(t-1), y_i(t-1), t-1)$$

- **If the optical flow field indicates that a particle becomes occluded, the particle is not propagated.**
- Backward propagation from frame *t+1* to *t* is defined analogously.



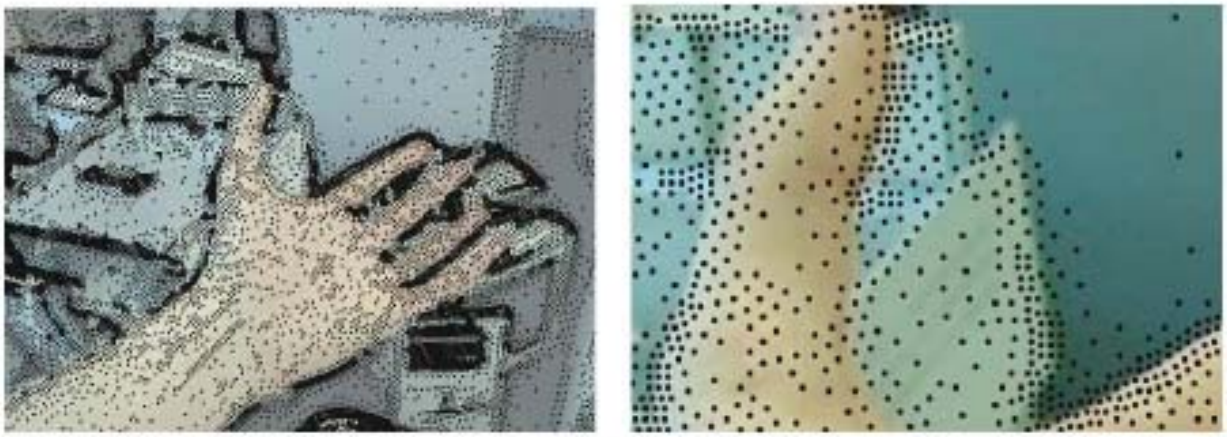
Particle Links

- Their algorithm creates links between particles using a constrained Delaunay triangulation.
- For a given frame, particle link will be created if the corresponding triangulation edge exists for the frame or adjacent frame.

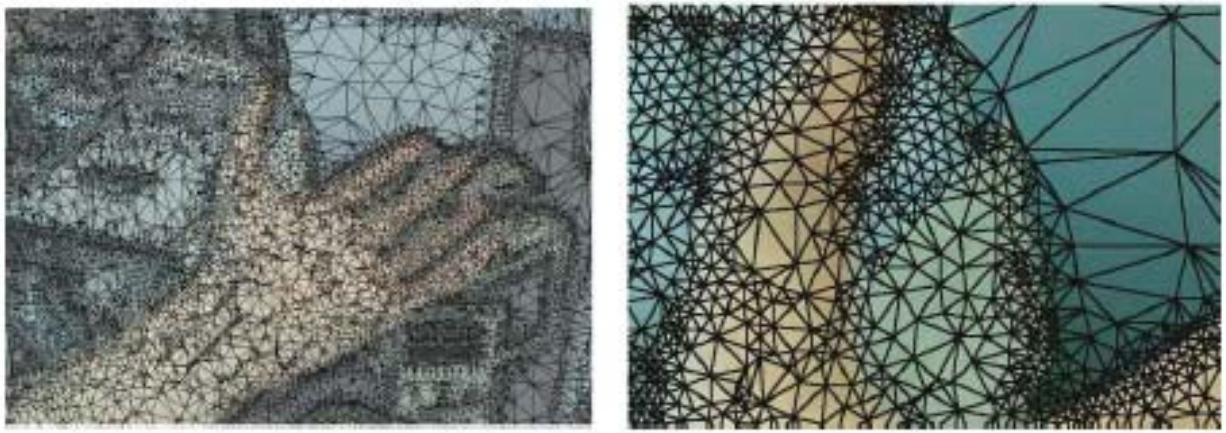


Link

Particle Links (cont.)



Particles



Links

Fig.7 Links are added using particle triangulation

Particle Links (cont.)

- The algorithm assign weight to each link based on difference between the trajectories of the linked particles.
- If particles have **similar trajectories**, they probably arise from same scene surface, and thus should be **strongly linked**.
- If the particles are separated by an occlusion boundary, the weight should be zero or near zero.

Particle Links (cont.)

- **Mean squared motion difference** between linked particles *i* and *j* over set *T* of frames is calculated by following equation:

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

where $u_i(t) = x_i(t) - x_i(t-1)$

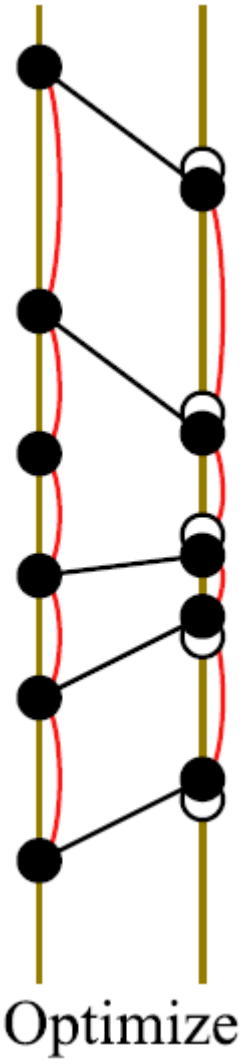
$$v_i(t) = y_i(t) - y_i(t-1)$$

Then algorithm computes link weight using zero-mean Gaussian prior ($\sigma_l = 1.5$).

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

Particle Optimization

- Flow field provides an initial location for each particle in the given video frame.
- Optimization process repositions particles.



Particle Optimization : 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.
  They define the energy of particle *i* in frame *t* as:

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

  $K_i(t)$ denotes the set of active chanels
  $L_i(t)$ denotes the set of particles linked to particle *i* in frame *t*

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

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

Particle Optimization : Data Energy

- The data energy term **measures how well a particles's appearance matches the pixel values**.
- For particle *i* at time *t* , the *k*th channel of particles's appearance is:

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

- The data term measures the difference between the observed appearance and filtered appearance:

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

where $\hat{c}_i^{[k]}(t)$ is filtered appearance which is slowly- varying apearrance

$$\psi(s^2) = \sqrt{s^2 + \varepsilon^2}; \varepsilon = 0.001$$

Particle Optimization : Distortion Energy

- The distortion term **measures the relative motion of particles**. For two linked particles which in the same frame *t*,
  - if they move in different directions , they will have large distortion
  - if they move in same direction, they will have a smaller distortion
- The **distortion term resists incorrect motions that could be caused by the data term**, especially near occlusion boundaries.

Particle Optimization : Distortion Energy (cont.)

- The distortion term is defined between a pair of linked particles *i* and *j*.

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

where

$$u_i(t) = x_i(t) - x_i(t-1)$$

$$v_i(t) = y_i(t) - y_i(t-1)$$

l_{ij} is link weight between particles *i* and *j*

Particle Optimization : Constructing a Sparse Linear System

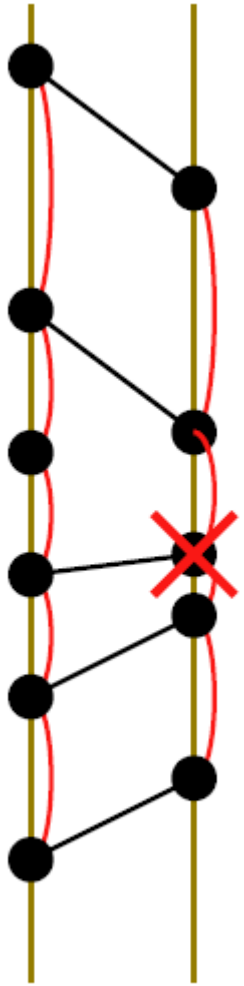
- Within objective function *E* , they substituted $dx_i(t) + x_i(t)$ for $x_i(t)$ (and instances of y accordingly).
- **Taking partial derivatives to obtain a system of equations.**

$$\left\{ \frac{\partial E}{\partial dx_i(t)} = 0, \frac{\partial E}{\partial dy_i(t)} = 0 \middle| i \in P, t \in F \right\}$$

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 $y_i(t)$).

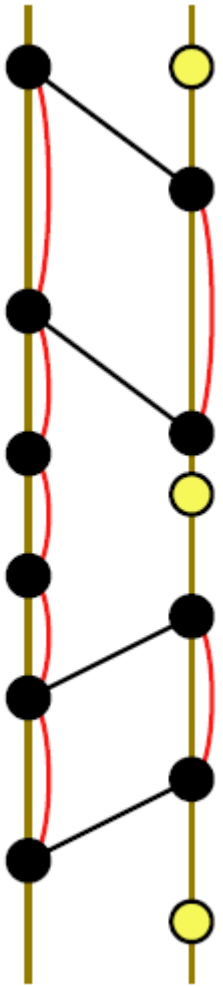
Pruning Particles

- After optimizing the particles, **the particles that continue to have high energy values will be pruned**.
- These particles have high distortion (large appearance mismatch), indicating possible occlusion.
- Each particle's energy values is filtered by using a Gaussian. **If in any frame the filtered energy value is greater than threshold, the particle is deactivated in that frame.**



Adding Particles Using Scale Maps

- After optimization and pruning, the algorithm adds new particles in gap between existing particles.
- The algorithm arranges for higher particle density in regions of greater visual complexity, in order to model complex motions.



Evaluation

- To quantify the algorithm's performance, they use video that are constructed to return to starting frame.
- They replace the second half of each evaluation video with temporally reversed copy of the first half.
- Then , they compute the fraction of particles that survive from the start frame to the end frame.

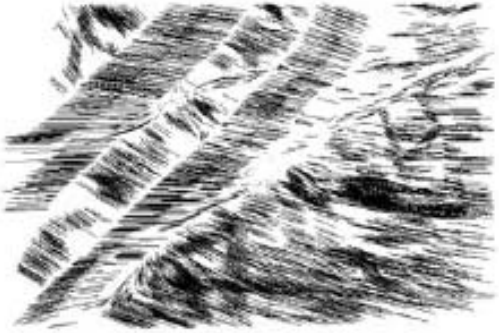
Evaluation Videos

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

Name	Camera motion	Occlusion	Object motion	Frames
VBranches	Hand-held R + T	Yes	None	50
VCars	Hand-held R + T	Yes	R + T	50
VHall	Hand-held R + T	Yes	None	50
VHand	Hand-held R + T	Yes	R + T; deformation	70
VMouth	Static	Yes	R + T; deformation	70
VPerson	Tripod R	Yes	R + T; deformation	50
VPlant	Hand-held R + T	Yes	None	70
VShelf	Crane T	Yes	None	50
VTree	Hand-held R + T	Yes	R + T; deformation	70
VTreeTrunk	Hand-held R + T	Yes	None	50
VZoomIn	Static	No	None	40
VZoomOut	Static	No	None	40
VRotateOrtho	Static	No	R	90
VRotatePersp	Static	No	R	90
VRectSlow	Static	Yes	R	80
VRectFast	Static	Yes	R	80
VRectLight	Static	Yes	R	80
VCylSlow	Static	Yes	R	50
VCylFast	Static	Yes	R	50
VCylLight	Static	Yes	R	50

Results of Particle Video Algorithm

VBranches, Frame 0



Correspondences

VBranches, Frame 25

VCars, Frame 0

Correspondences

VCars, Frame 25

VHand, Frame 0

Correspondences

VHand, Frame 35

Evaluation Results

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

PVBaseline -performs a forward sweep followed by a backward sweep.
PVSweep1- performs a single forward sweep.
PVSweep4-This sweeps forward, backward, forward again, then backward again.
PVNoOcc-This configuration ignores the occlusion maps.
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 for each particle position in the first video frame.

Conclusion

- Particle representation differs from standard motion representations such as vector field, layers and tracked feature patches.
- Their approach differs from optical flow by enforcing long-range appearance consistency and motion coherence.
- Their implementation represent occlusion boundaries using weight links between particles.

Thank you