Motion estimation

Computer Vision

CSE576, Spring 2005

Richard Szeliski

Why estimate visual motion?

Visual Motion can be annoying

- Camera instabilities, jitter
- Measure it; remove it (stabilize)

Visual Motion indicates dynamics in the scene

- Moving objects, behavior
- Track objects and analyze trajectories

Visual Motion reveals spatial layout

- Motion parallax

Today's lecture

Motion estimation

- image warping (skip: see handout)
- patch-based motion (optic flow)
- parametric (global) motion
- application: image morphing
- advanced: layered motion models

Readings

- Szeliski, R. *CVAA*
  - Ch. 7.1, 7.2, 7.4
- Bergen *et al. Hierarchical model-based motion estimation*. ECCV'92, pp. 237–252.
- Shi, J. and Tomasi, C. (1994). Good features to track. In CVPR'94, pp. 593–600.
- Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. IJCV, 56(3), 221–255.

Image Warping

Image Warping

image filtering: change *range* of image

$g(x) = h(f(x))$

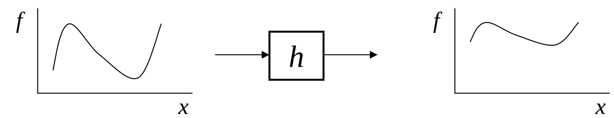


image warping: change *domain* of image

$g(x) = f(h(x))$

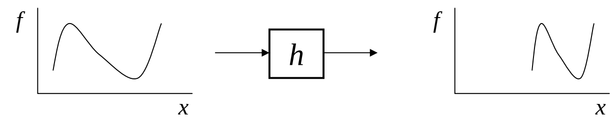


Image Warping

image filtering: change *range* of image

$g(x) = h(f(x))$

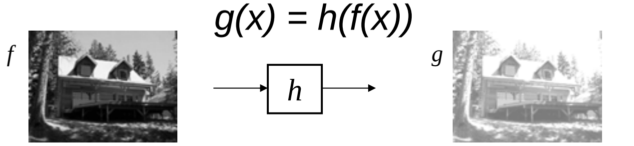
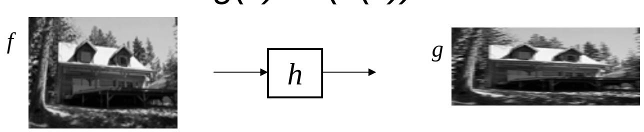


image warping: change *domain* of image

$g(x) = f(h(x))$



Parametric (global) warping

Examples of parametric warps:


translation


rotation


aspect


affine


perspective


cylindrical

2D coordinate transformations

translation: $\mathbf{x'} = \mathbf{x} + \mathbf{t}$ $\qquad \mathbf{x} = (x,y)$

rotation: $\mathbf{x'} = \mathbf{R}\,\mathbf{x} + \mathbf{t}$

similarity: $\mathbf{x'} = s\,\mathbf{R}\,\mathbf{x} + \mathbf{t}$

affine: $\mathbf{x'} = \mathbf{A}\,\mathbf{x} + \mathbf{t}$

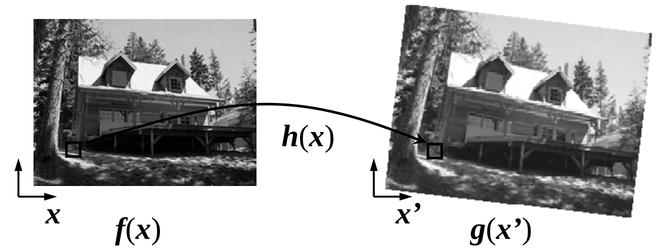
perspective: $\underline{\mathbf{x}}' \cong \mathbf{H}\,\underline{\mathbf{x}}$ $\qquad \underline{\mathbf{x}} = (x,y,1)$

($\underline{\mathbf{x}}$ is a *homogeneous* coordinate)

These all form a nested *group* (closed w/ inv.)

Image Warping

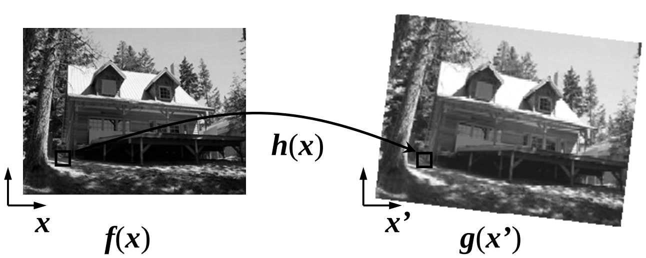
Given a coordinate transform $\mathbf{x'} = \mathbf{h}(\mathbf{x})$ and a source image $\mathbf{f}(\mathbf{x})$, how do we compute a transformed image $\mathbf{g}(\mathbf{x'}) = \mathbf{f}(\mathbf{h}(\mathbf{x}))$?



Forward Warping

Send each pixel $\mathbf{f}(\mathbf{x})$ to its corresponding location $\mathbf{x'} = \mathbf{h}(\mathbf{x})$ in $\mathbf{g}(\mathbf{x'})$

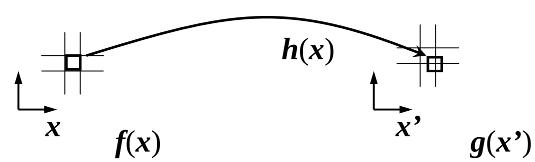
- What if pixel lands "between" two pixels?



Forward Warping

Send each pixel $\mathbf{f}(\mathbf{x})$ to its corresponding location $\mathbf{x'} = \mathbf{h}(\mathbf{x})$ in $\mathbf{g}(\mathbf{x'})$

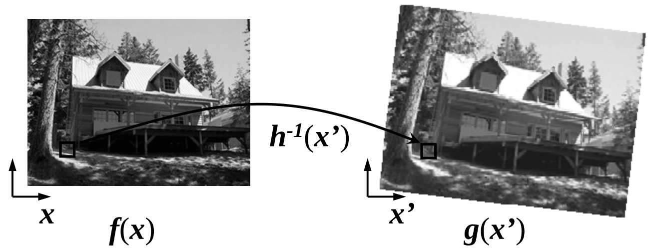
- What if pixel lands "between" two pixels?
- Answer: add "contribution" to several pixels, normalize later (*splatting*)



Inverse Warping

Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ from its corresponding location $\boldsymbol{x} = \boldsymbol{h}^{-1}(\boldsymbol{x'})$ in $\boldsymbol{f}(\boldsymbol{x})$

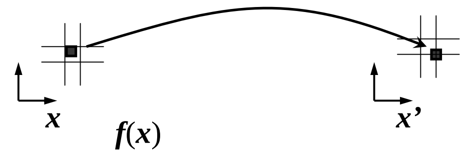
- What if pixel comes from “between” two pixels?



Inverse Warping

Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ from its corresponding location $\boldsymbol{x} = \boldsymbol{h}^{-1}(\boldsymbol{x'})$ in $\boldsymbol{f}(\boldsymbol{x})$

- What if pixel comes from “between” two pixels?
- Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Interpolation

Possible interpolation filters:

- nearest neighbor
- bilinear
- bicubic (interpolating)
- sinc / FIR

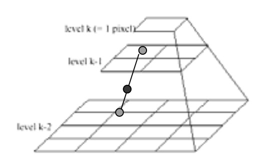
Needed to prevent “jaggies” and “texture crawl” (see demo)

Prefiltering

Essential for *downsampling* (*decimation*) to prevent *aliasing*

MIP-mapping [Williams’83]:

1. build pyramid (but what decimation filter?):
   - block averaging
   - Burt & Adelson (5-tap binomial)
   - 7-tap wavelet-based filter (better)
2. *trilinear* interpolation
   - bilinear within each 2 adjacent levels
   - linear blend *between* levels (determined by pixel size)



Prefiltering

Essential for *downsampling* (*decimation*) to prevent *aliasing*

Other possibilities:

- summed area tables
- elliptically weighted Gaussians (EWA) [Heckbert'86]

Patch-based motion estimation

Classes of Techniques

Feature-based methods

- Extract visual features (corners, textured areas) and track them
- Sparse motion fields, but possibly robust tracking
- Suitable especially when image motion is large (10s of pixels)

Direct-methods

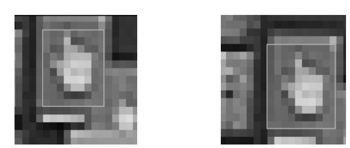
- Directly recover image motion from spatio-temporal image brightness variations
- Global motion parameters directly recovered without an intermediate feature motion calculation
- Dense motion fields, but more sensitive to appearance variations
- Suitable for video and when image motion is small (< 10 pixels)

Patch matching (revisited)

How do we determine correspondences?

- *block matching* or *SSD* (sum squared differences)

$$E(x,y;d) = \sum_{(x',y') \in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$



The Brightness Constraint

Brightness Constancy Equation:

$$J(x,y) \approx \boxed{I(x+u(x,y),\, y+v(x,y))}$$

Or, equivalently, minimize :

$$E(u,v) = (J(x,y) - I(x+u, y+v))^2$$

Linearizing (assuming small *(u,v)*)
using Taylor series expansion:

$$J(x,y) \approx \boxed{I(x,y) + I_x(x,y)\cdot u(x,y) + I_y(x,y)\cdot v(x,y)}$$

Gradient Constraint (or the Optical Flow Constraint)

$$E(u,v) = (I_x \cdot u + I_y \cdot v + I_t)^2$$

Minimizing: $\frac{\partial E}{du} = \frac{\partial E}{dv} = 0$

$$I_x(I_x u + I_y v + I_t) = 0$$

$$I_y(I_x u + I_y v + I_t) = 0$$

In general $I_x, I_y \neq 0$

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0$

Patch Translation [Lucas-Kanade]

Assume a single velocity for all pixels within an image patch

$$E(u,v) = \sum_{x,y\in\Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t\right)^2$$

Minimizing

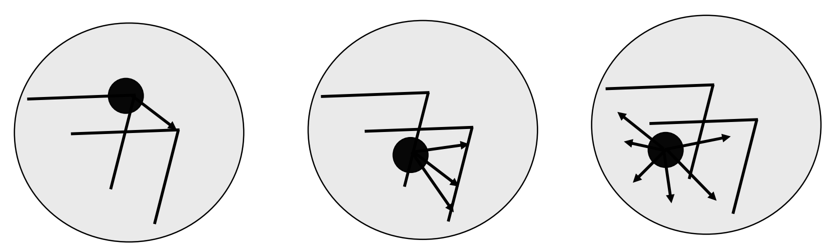
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

$$\left(\sum \nabla I \nabla I^T\right)\vec{U} = -\sum \nabla I I_t$$

LHS: sum of the 2x2 outer product of the gradient vector

Local Patch Analysis

How *certain* are the motion estimates?

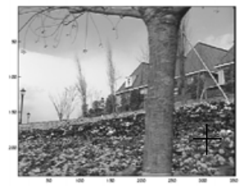
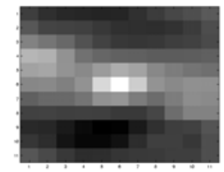
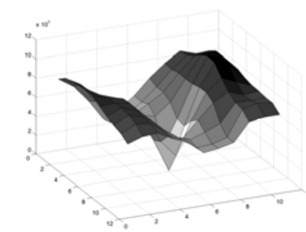


The Aperture Problem

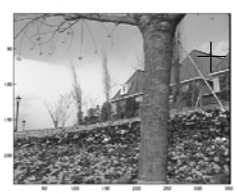
Let $M = \sum (\nabla I)(\nabla I)^T$ and $b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

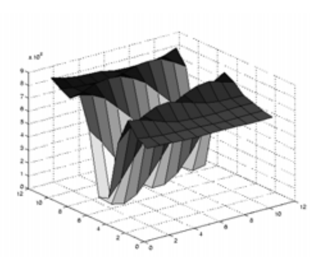
- Algorithm: At each pixel compute U by solving $MU = b$
- M is singular if all gradient vectors point in the same direction
  - e.g., along an edge
  - of course, trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only *normal flow* is available (aperture problem)
- Corners and textured areas are OK

SSD Surface – Textured area

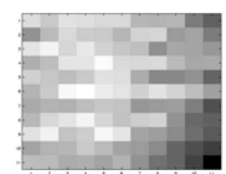
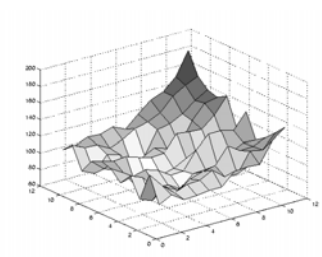




SSD Surface -- Edge





SSD – homogeneous area

Iterative Refinement

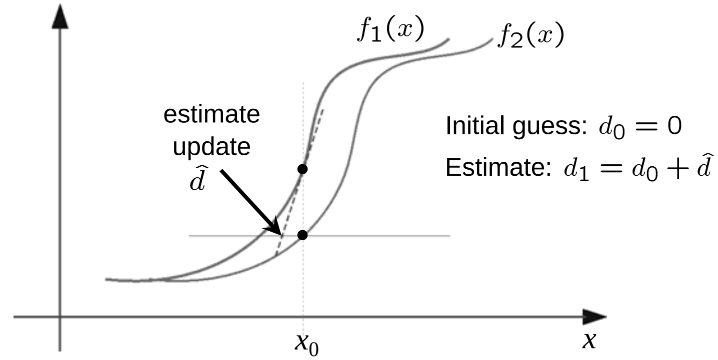
Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation

Warp one image toward the other using the estimated flow field

(easier said than done)

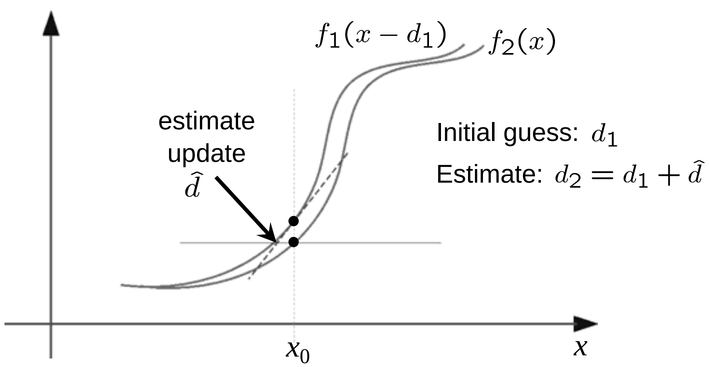
Refine estimate by repeating the process

Optical Flow: Iterative Estimation

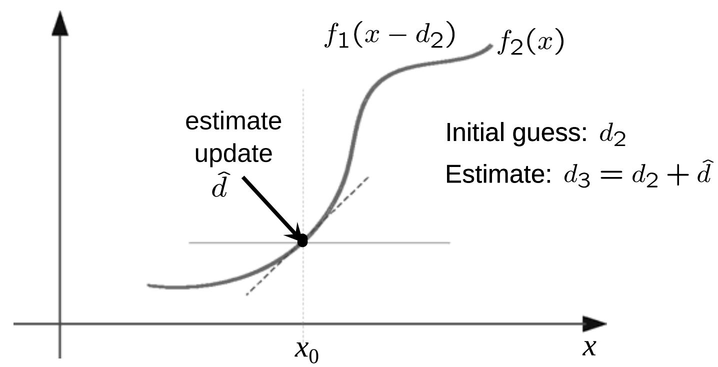


(using d for *displacement* here instead of u)

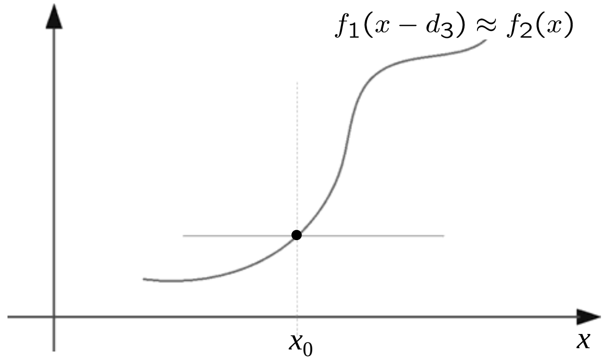
Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation

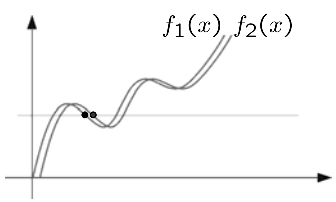
Some Implementation Issues:

- Warping is not easy (ensure that errors in warping are smaller than the estimate refinement)
- Warp one image, take derivatives of the other so you don't need to re-compute the gradient after each iteration.
- Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity)

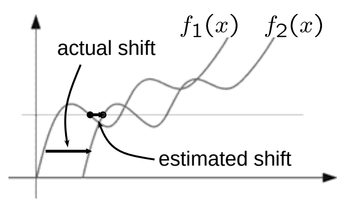
Optical Flow: Aliasing

Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity.

I.e., how do we know which 'correspondence' is correct?



nearest match is correct (no aliasing)



nearest match is incorrect (aliasing)

To overcome aliasing: coarse-to-fine estimation.

Limits of the gradient method

Fails when intensity structure in window is poor

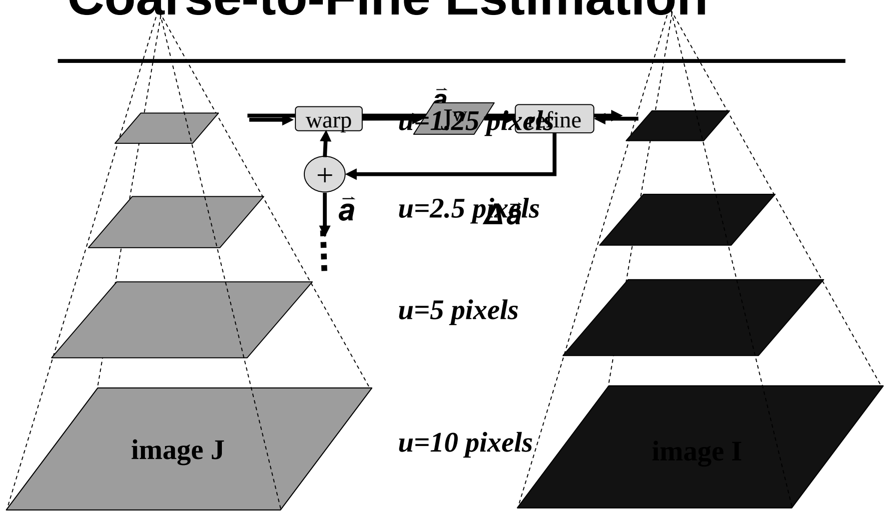
Fails when the displacement is large (typical operating range is motion of 1 pixel)

Linearization of brightness is suitable only for small displacements

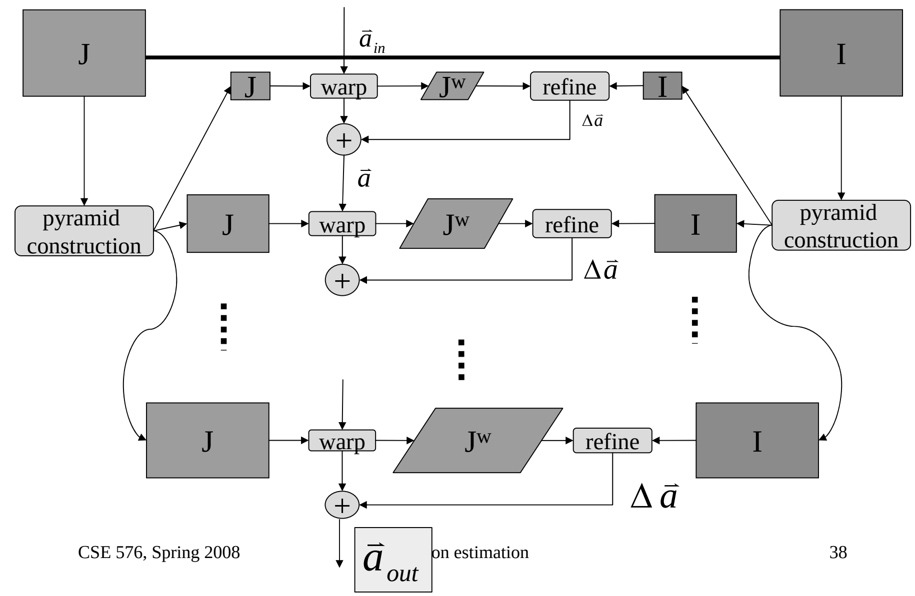
Also, brightness is not strictly constant in images

actually less problematic than it appears, since we can pre-filter images to make them look similar

Coarse-to-Fine Estimation



Coarse-to-Fine Estimation



Parametric motion estimation

Global (parametric) motion models

2D Models:

Affine

Quadratic

Planar projective transform (Homography)

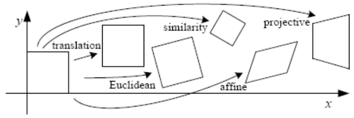
3D Models:

Instantaneous camera motion models

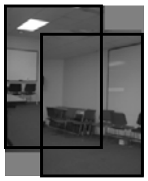
Homography+epipole

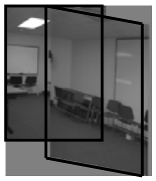
Plane+Parallax

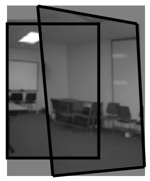
Motion models



Translation | **Affine** | **Perspective** | **3D rotation**







2 unknowns | **6 unknowns** | **8 unknowns** | **3 unknowns**

Example: Affine Motion

$$u(x,y) = a_1 + a_2 x + a_3 y$$
$$v(x,y) = a_4 + a_5 x + a_6 y$$

Substituting into the B.C. Equation:

$$\boxed{I_x (a_1 + a_2 x + a_3 y) + I_y (a_4 + a_5 x + a_6 y) + I_t \approx 0}$$

Each pixel provides 1 linear constraint in 6 *global* unknowns

Least Square Minimization (over all pixels):

$$\boxed{Err(\vec{a}) = \sum \left[I_x (a_1 + a_2 x + a_3 y) + I_y (a_4 + a_5 x + a_6 y) + I_t \right]^2}$$

Other 2D Motion Models

Quadratic – instantaneous approximation to planar motion

$$u = q_1 + q_2 x + q_3 y + q_7 x^2 + q_8 xy$$
$$v = q_4 + q_5 x + q_6 y + q_7 xy + q_8 y^2$$

Projective – exact planar motion

$$x' = \frac{h_1 + h_2 x + h_3 y}{h_7 + h_8 x + h_9 y}$$
$$y' = \frac{h_4 + h_5 x + h_6 y}{h_7 + h_8 x + h_9 y}$$

and

$$u = x' - x, \quad v = y' - y$$

3D Motion Models

Instantaneous camera motion:

Global parameters: $\boxed{\Omega_X, \Omega_Y, \Omega_Z, T_X, T_Y, T_Z}$

Local Parameter: $\boxed{Z(x,y)}$

$$u = -xy\Omega_X + (1 + x^2)\Omega_Y - y\Omega_Z + (T_X - T_Z x)/Z$$
$$v = -(1 + y^2)\Omega_X + xy\Omega_Y - x\Omega_Z + (T_Y - T_Z x)/Z$$

Homography+Epipole

Global parameters: $\boxed{h_1, \ldots, h_9, t_1, t_2, t_3}$

Local Parameter: $\boxed{\gamma(x,y)}$

$$x' = \frac{h_1 x + h_2 y + h_3 + \gamma t_1}{h_7 x + h_8 y + h_9 + \gamma t_3}$$
$$y' = \frac{h_4 x + h_5 y + h_6 + \gamma t_1}{h_7 x + h_8 y + h_9 + \gamma t_3}$$

and : $u = x' - x, \quad v = y' - y$

Residual Planar Parallax Motion

Global parameters: $\boxed{t_1, t_2, t_3}$

Local Parameter: $\boxed{\gamma(x,y)}$

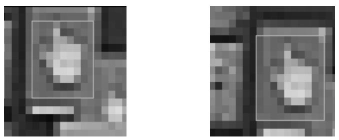
$$u = x^w - x = \frac{\gamma}{1 + \gamma t_3}(t_3 x - t_1)$$
$$v = y^w - x = \frac{\gamma}{1 + \gamma t_3}(t_3 y - t_2)$$

Patch matching (revisited)

How do we determine correspondences?

- *block matching* or *SSD* (sum squared differences)

$$E(x,y;d) = \sum_{(x',y') \in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$



Correlation and SSD

For larger displacements, do template matching

- Define a small area around a pixel as the template
- Match the template against each pixel within a search area in next image.
- Use a match measure such as correlation, normalized correlation, or sum-of-squares difference
- Choose the maximum (or minimum) as the match
- Sub-pixel estimate (Lucas-Kanade)

Discrete Search vs. Gradient Based

Consider image I translated by u_0, v_0

$$I_0(x,y) = I(x,y)$$
$$I_1(x+u_0, y+v_0) = I(x,y) + \eta_1(x,y)$$

$$\begin{aligned} E(u,v) &= \sum_{x,y} (I(x,y) - I_1(x+u, y+v))^2 \\ &= \sum_{x,y} (I(x,y) - I(x-u_0+u, y-v_0+v) - \eta_1(x,y))^2 \end{aligned}$$

The discrete search method simply searches for the best estimate.

The gradient method linearizes the intensity function and solves for the estimate

Shi-Tomasi feature tracker

1. Find good features (min eigenvalue of 2×2 Hessian)
2. Use Lucas-Kanade to track with pure translation
3. Use affine registration with first feature patch
4. Terminate tracks whose dissimilarity gets too large
5. Start new tracks when needed

Tracking results

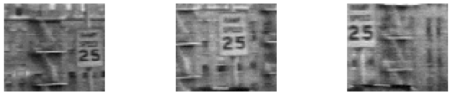


Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

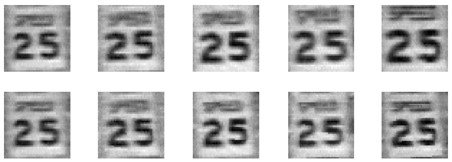


Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

Tracking - dissimilarity

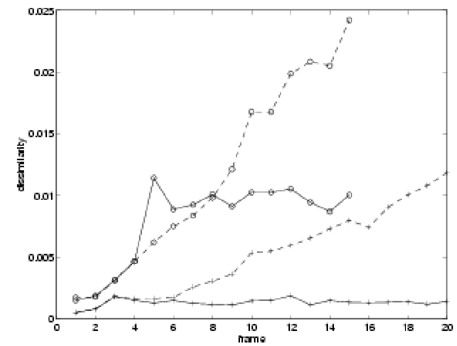


Figure 3: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 1 (plusses) and 4 (circles).

Tracking results

Figure 13: Labels of some of the features in figure 11.

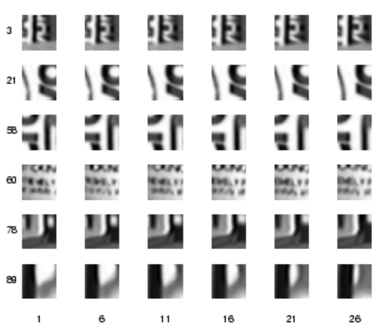


Figure 14: Six sample features through six sample frames.

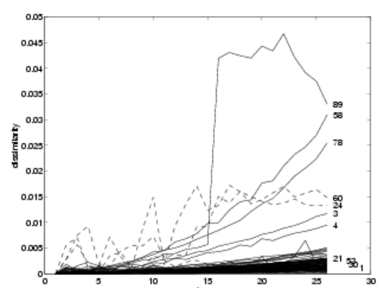


Figure 15: Affine motion dissimilarity for the features in figure 11. Notice the good discrimination between good and bad features. Dashed plots indicate aliasing (see text).

Features 24 and 60 deserve a special discussion, and

Correlation Window Size

Small windows lead to more false matches

Large windows are better this way, but…

- Neighboring flow vectors will be more correlated (since the template windows have more in common)
- Flow resolution also lower (same reason)
- More expensive to compute

Small windows are good for local search:
more detailed and less smooth (noisy?)

Large windows good for global search:
less detailed and smoother

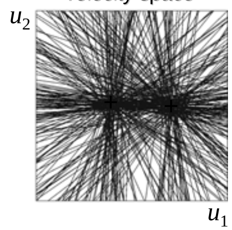
Robust Estimation

Noise distributions are often non-Gaussian, having much heavier tails. Noise samples from the tails are called outliers.

Sources of outliers (multiple motions):

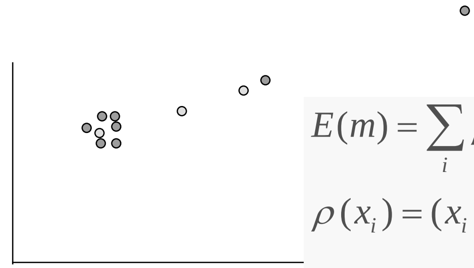
- specularities / highlights
- jpeg artifacts / interlacing / motion blur
- multiple motions (occlusion boundaries, transparency)

velocity space



Robust Estimation

Standard Least Squares Estimation allows too much influence for outlying points



$$E(m) = \sum_i \rho(x_i)$$

$$\rho(x_i) = (x_i - m)^2$$

Influence $\psi(x) = \dfrac{\partial \rho}{\partial x} = (x_i - m)$

Robust Estimation

$$E_d(u_s, v_s) = \sum \rho\left(I_x u_s + I_y v_s + I_t\right)$$ Robust gradient constraint

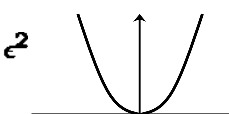
$$E_d(u_s, v_s) = \sum \rho\left(I(x, y) - J(x + u_s, y + v_s)\right)$$ Robust SSD

Robust Estimation

Problem: Least-squares estimators penalize deviations between data & model with quadratic error f^n (extremely sensitive to outliers)

error penalty function influence function

$$\rho(\epsilon) = \epsilon^2$$

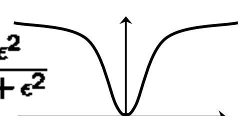


$$\psi(\epsilon) = \frac{\partial \rho(\epsilon)}{\partial \epsilon} = 2\epsilon$$

Redescending error functions (e.g., Geman-McClure) help to reduce the influence of outlying measurements.

error penalty function influence function

$$\rho(\epsilon; s) = \frac{\epsilon^2}{s + \epsilon^2}$$



$$\psi(\epsilon; s) = \frac{2\epsilon s}{(s + \epsilon^2)^2}$$

How well do these techniques work?

A Database and Evaluation Methodology for Optical Flow

Simon Baker, Daniel Scharstein, J.P Lewis, Stefan Roth, Michael Black, and Richard Szeliski

ICCV 2007

http://vision.middlebury.edu/flow/

Limitations of *Yosemite*

Only sequence used for quantitative evaluation

Yosemite

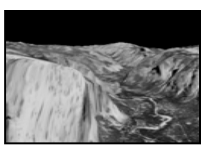


Image 7

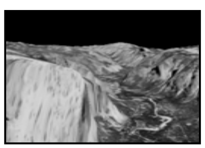
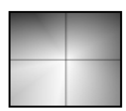


Image 8

Ground-Truth Flow



Flow Color Coding

Limitations:

- Very simple and synthetic
- Small, rigid motion
- Minimal motion discontinuities/occlusions

Limitations of *Yosemite*

Only sequence used for quantitative evaluation

Yosemite

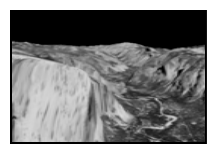


Image 7

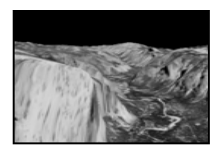
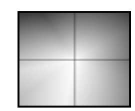


Image 8

Ground-Truth Flow



Flow Color Coding

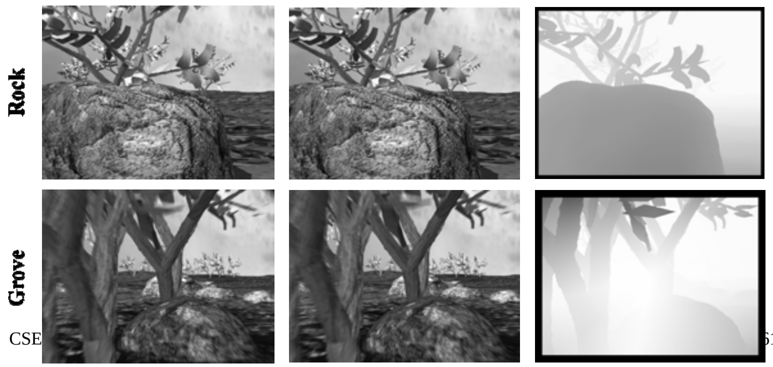
Current challenges:

- Non-rigid motion
- Real sensor noise
- Complex natural scenes
- Motion discontinuities

Need **more challenging** and **more realistic** benchmarks

Realistic synthetic imagery

- Randomly generate scenes with “trees” and “rocks”
- Significant occlusions, motion, texture, and blur
- Rendered using Mental Ray and “lens shader” plugin

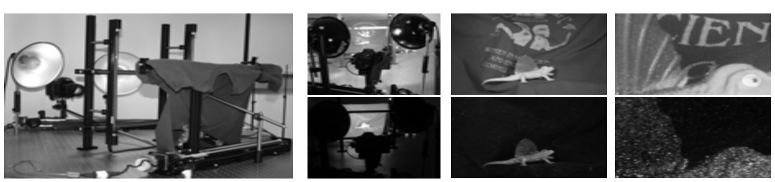


Modified stereo imagery

- Recrop and resample ground-truth stereo datasets to have appropriate motion for OF

Dense flow with hidden texture

- Paint scene with textured fluorescent paint
- Take 2 images: One in visible light, one in UV light
- Move scene in very small steps using robot
- Generate ground-truth by tracking the UV images



Visible

UV

Setup Lights Image Cropped

Experimental results

Algorithms:

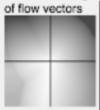
- **Pyramid LK:** OpenCV-based implementation of Lucas-Kanade on a Gaussian pyramid
- **Black and Anandan:** Author’s implementation
- **Bruhn *et al*.:** Our implementation
- **MediaPlayer^{TM}:** Code used for video frame-rate upsampling in Microsoft MediaPlayer
- **Zitnick *et al*.:** Author’s implementation

Experimental results

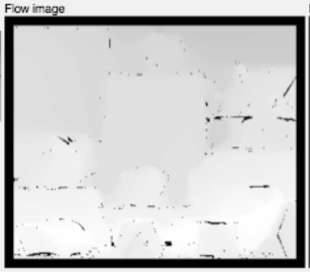
Optical flow evaluation results   Choose error measures: Average SD R1.0 R3.0 R5.0 A50 A75 A95

Average angle error	avg. rank	Dimetrodon (Hidden texture) GT im0 im1 all			Seashell (Hidden texture) GT im0 im1 all			Rock (Synthetic) GT im0 im1 all			Grove (Synthetic) GT im0 im1 all			Yosemite (Synthetic) GT im0 im1 all			Venus (Stereo) GT im0 im1 all			Moebius (Stereo) GT im0 im1 all		
		all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext
Bruhn et al.	1.6	10.99 3	**9.41** 1	14.22 3	11.09 2	19.48 2	16.21 2	**6.14** 1	**17.41** 1	12.86 2	**6.32** 1	**12.41** 1	**10.98** 1	**1.69** 1	**2.86** 1	**1.05** 1	8.73 2	31.46 2	8.15 2	**5.85** 1	10.12 2	8.80 2
Black and Anandan	2.1	**9.26** 1	10.11 3	**12.08** 1	11.20 3	19.83 3	17.01 3	7.67 3	18.44 3	16.80 4	7.89 2	13.55 2	13.96 4	2.65 2	4.18 2	1.88 2	**7.64** 1	**30.13** 1	**7.31** 1	7.05 2	**10.02** 1	**8.41** 1
Pyramid LK	2.8	10.27 2	9.71 2	13.63 2	**9.46** 1	**18.62** 1	**12.07** 1	6.53 2	18.43 2	**10.95** 1	8.14 3	15.08 3	12.78 2	5.22 3	6.64 3	4.29 3	14.61 4	36.18 4	24.67 5	12.98 5	13.85 4	20.61 5
MediaPlayer™	4.1	15.82 4	26.42 4	16.96 4	23.18 4	27.71 5	21.78 4	9.44 4	22.25 4	15.03 3	10.99 4	18.15 5	13.64 3	11.09 4	17.16 4	10.66 5	15.48 5	43.56 5	15.09 4	9.98 4	15.04 5	9.47 3
Zitnick et al.	4.2	30.10 5	34.27 5	31.58 5	29.07 5	27.55 4	21.78 4	12.38 5	23.93 5	17.59 5	12.55 5	15.56 4	17.35 5	18.50 5	28.00 5	9.41 4	11.42 3	31.46 2	11.12 3	9.88 3	12.83 3	11.28 4

Color encoding of flow vectors



Flow image



Error image

Conclusions

- **Difficulty:** Data substantially more challenging than **Yosemite**
- **Diversity: S**ubstantial variation in difficulty across the various datasets
- **Motion GT vs Interpolation:** Best algorithms for one are not the best for the other
- **Comparison with Stereo:** Performance of existing flow algorithms appears weak

Image Morphing

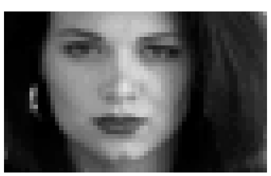
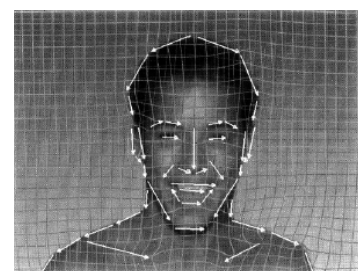


Image Warping – non-parametric

Specify more detailed warp function



Examples:

- splines
- triangles
- optical flow (per-pixel motion)

Image Warping – non-parametric

Move control points to specify spline warp

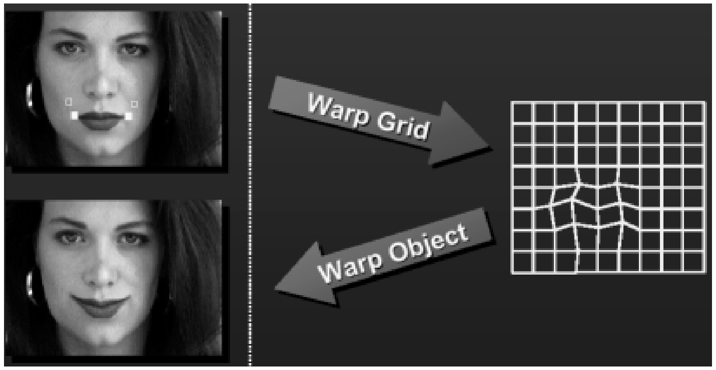
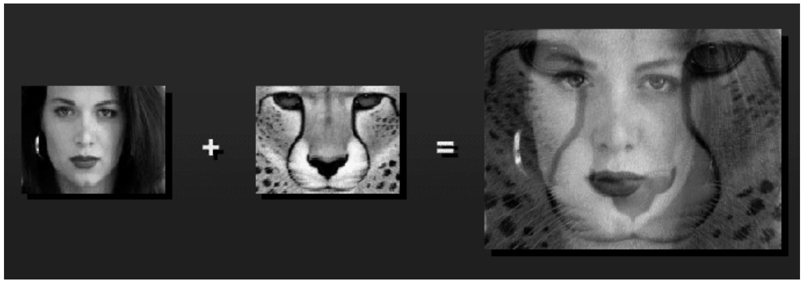


Image Morphing

How can we *in-between* two images?

1. Cross-dissolve

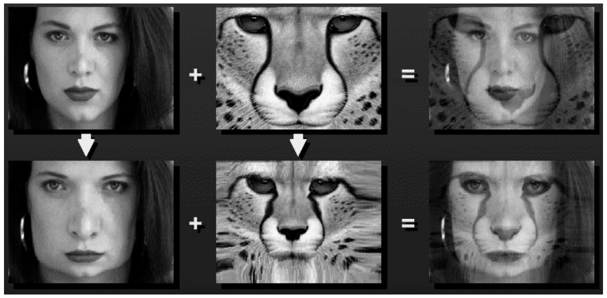


(all examples from [Gomes *et al.*'99])

Image Morphing

How can we *in-between* two images?

2. Warp then cross-dissolve = *morph*



Warp specification

How can we specify the warp?

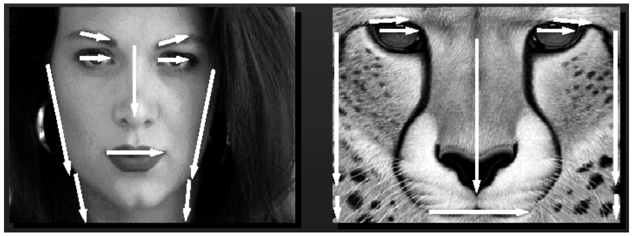
1. Specify corresponding *points*
   - *interpolate* to a complete warping function

- Nielson, *Scattered Data Modeling*, IEEE CG&A'93]

Warp specification

How can we specify the warp?

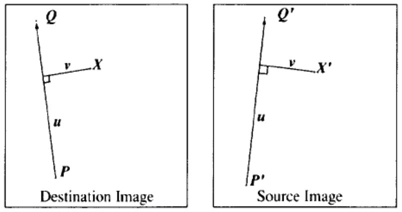
2. Specify corresponding *vectors*
   - *interpolate* to a complete warping function



Warp specification

How can we specify the warp?

2. Specify corresponding *vectors*
   - *interpolate* [Beier & Neely, SIGGRAPH'92]

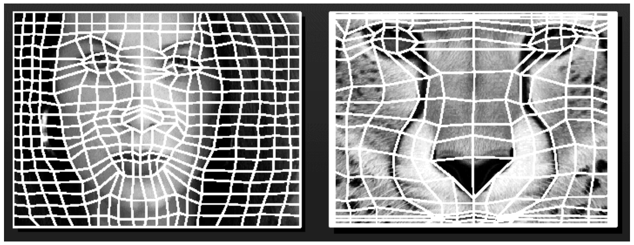


For each pixel X in the destination
  $DSUM = (0,0)$
  $weightsum = 0$
  For each line $P_i Q_i$
    calculate u,v based on $P_i Q_i$
    calculate X'_i based on u,v and $P_i'Q_i'$
    calculate displacement $D_i = X_i' - X_i$ for this line
    $dist$ = shortest distance from X to $P_i Q_i$
    $weight = (length^p / (a + dist))^b$
    $DSUM$ += $D_i * weight$
    $weightsum$ += $weight$
  $X' = X + DSUM / weightsum$
  destinationImage(X) = sourceImage(X')

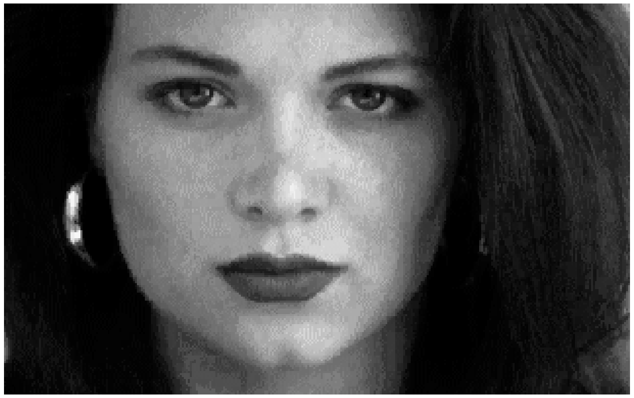
Warp specification

How can we specify the warp?

3. Specify corresponding *spline control points*
   - *interpolate* to a complete warping function



Final Morph Result



Layered Scene Representations

Motion representations

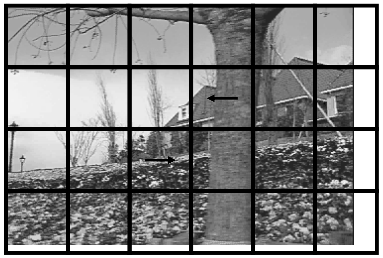
How can we describe this scene?

Block-based motion prediction

Break image up into square blocks

Estimate translation for each block

Use this to predict next frame, code difference (MPEG-2)



Layered motion

Break image sequence up into "layers":

÷

=

Describe each layer's motion

Layered motion

Advantages:

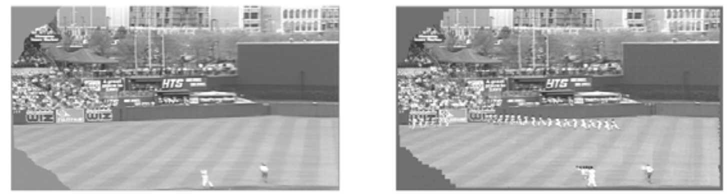
- can represent occlusions / disocclusions
- each layer's motion can be smooth
- video segmentation for semantic processing

Difficulties:

- how do we determine the correct number?
- how do we assign pixels?
- how do we model the motion?

Layers for video summarization

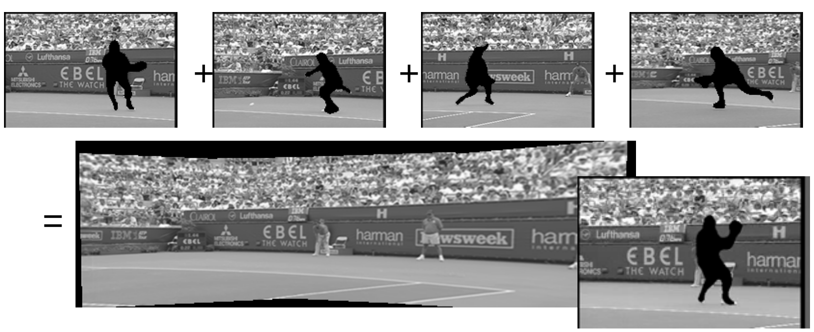
Frame 0 Frame 50 Frame 80



Background scene (players removed) Complete synopsis of the video

Background modeling (MPEG-4)

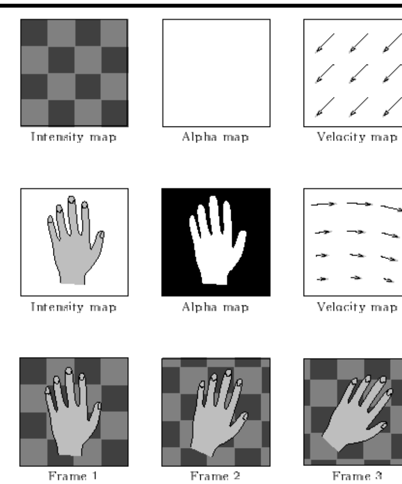
Convert masked images into a background sprite for layered video coding



What are layers?

[Wang & Adelson, 1994]

- intensities
- alphas
- velocities



How do we composite them?

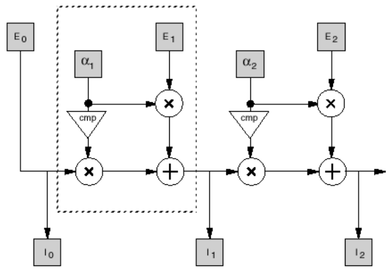


Figure 3: A flow chart for compositing a series of layers. The box labeled "cmp" generates the complement of alpha, $(1-\alpha)$

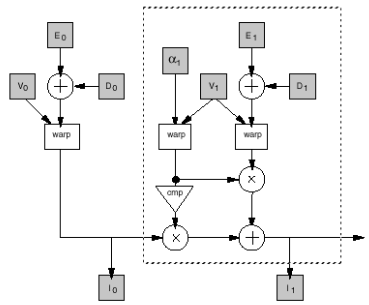
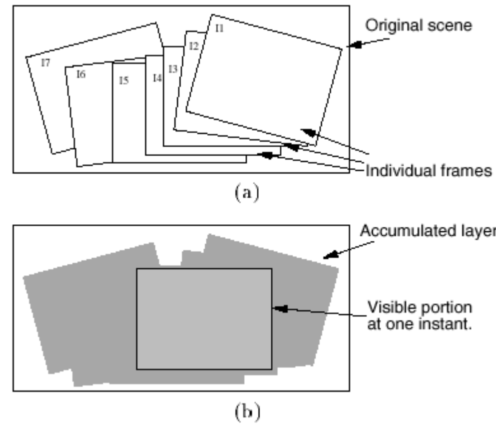


Figure 4: A flow chart for compositing that incorporates velocity maps, V, and delta maps, D.

How do we form them?



How do we form them?

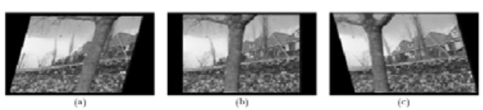


Figure 7: (a) Frame 1 warped with an affine transformation to align the flowerbed region with that of frame 15. (b) Original frame 15 used as reference. (c) Frame 30 warped with an affine transformation to align the flowerbed region with that of frame 15.

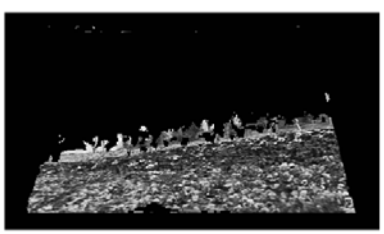
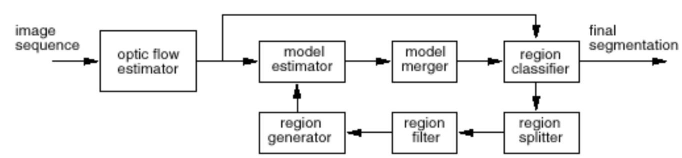


Figure 8: Accumulation of the flowerbed. Image intensities are obtained from a temporal median operation on the motion compensated images. Only the regions belonging to the flowerbed layer is accumulated in this image. Note also occluded regions are correctly recovered by accumulating data over many frames.

How do we estimate the layers?

1. compute coarse-to-fine flow
2. estimate affine motion in blocks (regression)
3. cluster with *k-means*
4. assign pixels to best fitting affine region
5. re-estimate affine motions in each region…

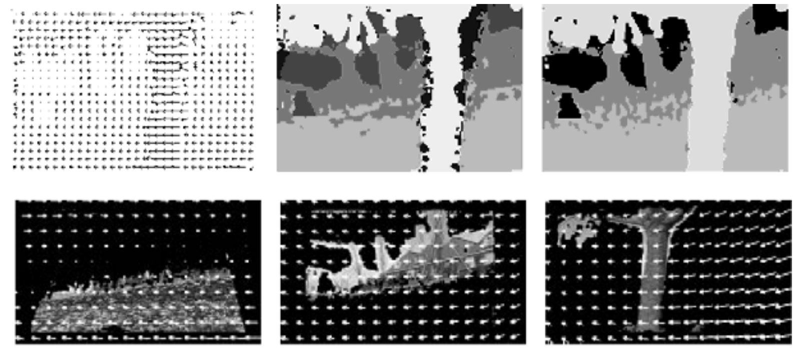


Layer synthesis

For each layer:
- stabilize the sequence with the affine motion
- compute median value at each pixel

Determine occlusion relationships

Results



Bibliography

L. Williams. *Pyramidal parametrics*. Computer Graphics, 17(3):1--11, July 1983.

L. G. Brown. *A survey of image registration techniques*. Computing Surveys, 24(4):325--376, December 1992.

C. D. Kuglin and D. C. Hines. *The phase correlation image alignment method*. In IEEE 1975 Conference on Cybernetics and Society, pages 163--165, New York, September 1975.

J. Gomes, L. Darsa, B. Costa, and L. Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999.

T. Beier and S. Neely. *Feature-based image metamorphosis*. Computer Graphics (SIGGRAPH'92), 26(2):35--42, July 1992.

Bibliography

J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In ECCV'92, pp. 237–252, Italy, May 1992.

M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. Comp. Vis. Image Understanding, 63(1):75–104, 1996.

Shi, J. and Tomasi, C. (1994). Good features to track. In CVPR'94, pages 593–600, IEEE Computer Society, Seattle.

Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework: Part 1: The quantity approximated, the warp update rule, and the gradient descent approximation. IJCV, 56(3), 221–255.

Bibliography

H. S. Sawhney and S. Ayer. Compact representation of videos through dominant multiple motion estimation. IEEE Trans. Patt. Anal. Mach. Intel., 18(8):814–830, Aug. 1996.

Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In CVPR'97, pp. 520–526, June 1997.

J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. IEEE Transactions on Image Processing, 3(5):625--638, September 1994.

Bibliography

Y. Weiss and E. H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96), pages 321--326, San Francisco, California, June 1996.

Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97), pages 520--526, San Juan, Puerto Rico, June 1997.

P. R. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representations. In Twelfth International Conference on Pattern Recognition (ICPR'94), pages 743--746, Jerusalem, Israel, October 1994. IEEE Computer Society Press

Bibliography

T. Darrell and A. Pentland. Cooperative robust estimation using layers of support. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(5):474--487, May 1995.

S. X. Ju, M. J. Black, and A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96), pages 307--314, San Francisco, California, June 1996.

M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. International Journal of Computer Vision, 12(1):5--16, January 1994.

H. S. Sawhney and S. Ayer. Compact representation of videos through dominant multiple motion estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(8):814--830, August 1996.

M.-C. Lee et al. A layered video object coding system using sprite and affine motion model. IEEE Transactions on Circuits and Systems for Video Technology, 7(1):130--145, February 1997.

Bibliography

S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In IEEE CVPR'98, pages 434--441, Santa Barbara, June 1998.

R. Szeliski, S. Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In IEEE CVPR'2000, volume 1, pages 246--253, Hilton Head Island, June 2000.

J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In Computer Graphics (SIGGRAPH'98) Proceedings, pages 231--242, Orlando, July 1998. ACM SIGGRAPH.

S. Laveau and O. D. Faugeras. 3-d scene representation as a collection of images. In Twelfth International Conference on Pattern Recognition (ICPR'94), volume A, pages 689--691, Jerusalem, Israel, October 1994. IEEE Computer Society Press.

P. H. S. Torr, R. Szeliski, and P. Anandan. An integrated Bayesian approach to layer extraction from image sequences. In Seventh ICCV'98, pages 983--990, Kerkyra, Greece, September 1999.