Lecture 13 – Optical Flow

With slides from R. Szeliski, S. Lazebnik, S. Seitz, A. Efros, C. Liu & F. Durand
Admin

• Assignment 3 due

• Assignment 4 out
  – Deadline: Thursday 11\textsuperscript{th} Dec
  – THIS IS A HARD DEADLINE
    (I have to hand in grades on 12\textsuperscript{th})

• Course assessment forms
Overview

• Segmentation in Video
• Optical flow
• Motion Magnification
Video

- A video is a sequence of frames captured over time
- Now our image data is a function of space \((x, y)\) and time \((t)\)
Applications of segmentation to video

• Background subtraction
  • A static camera is observing a scene
  • Goal: separate the static *background* from the moving *foreground*
Applications of segmentation to video

• Background subtraction
  • Form an initial background estimate
  • For each frame:
    – Update estimate using a moving average
    – Subtract the background estimate from the frame
    – Label as foreground each pixel where the magnitude of the difference is greater than some threshold
    – Use median filtering to “clean up” the results
Applications of segmentation to video

- Background subtraction
- Shot boundary detection
  - Commercial video is usually composed of *shots* or sequences showing the same objects or scene
  - Goal: segment video into shots for summarization and browsing (each shot can be represented by a single keyframe in a user interface)
  - Difference from background subtraction: the camera is not necessarily stationary
Applications of segmentation to video

• Background subtraction
• Shot boundary detection
  • For each frame
    – Compute the distance between the current frame and the previous one
      » Pixel-by-pixel differences
      » Differences of color histograms
      » Block comparison
    – If the distance is greater than some threshold, classify the frame as a shot boundary
Applications of segmentation to video

- Background subtraction
- Shot boundary detection
- Motion segmentation
  - Segment the video into multiple *coherently* moving objects
Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept
Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept
Uses of motion

- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning dynamical models
- Recognizing events and activities
- Improving video quality (motion stabilization)
Motion estimation techniques

- Direct methods
  - Directly recover image motion at each pixel from spatio-temporal image brightness variations
  - Dense motion fields, but sensitive to appearance variations
  - Suitable for video and when image motion is small

- Feature-based methods
  - Extract visual features (corners, textured areas) and track them over multiple frames
  - Sparse motion fields, but more robust tracking
  - Suitable when image motion is large (10s of pixels)
Motion field

• The motion field is the projection of the 3D scene motion into the image
Motion field and parallax

• \( \mathbf{P}(t) \) is a moving 3D point

• Velocity of scene point: \( \mathbf{V} = \frac{d\mathbf{P}}{dt} \)

• \( \mathbf{p}(t) = (x(t), y(t)) \) is the projection of \( \mathbf{P} \) in the image

• Apparent velocity \( \mathbf{v} \) in the image: given by components \( v_x = \frac{dx}{dt} \) and \( v_y = \frac{dy}{dt} \)

• These components are known as the *motion field* of the image
Motion field and parallax

\[ \mathbf{V} = (V_x, V_y, V_z) \quad \mathbf{p} = f \frac{\mathbf{P}}{Z} \]

To find image velocity \( \mathbf{v} \), differentiate \( \mathbf{p} \) with respect to \( t \) (using quotient rule):

\[ \mathbf{v} = f \frac{Z \mathbf{V} - V_z \mathbf{P}}{Z^2} \]

\[ v_x = \frac{f V_x - V_z x}{Z} \quad v_y = \frac{f V_y - V_z y}{Z} \]

Image motion is a function of both the 3D motion (\( \mathbf{V} \)) and the depth of the 3D point (\( Z \))
Motion field and parallax

- Pure translation: \( \mathbf{V} \) is constant everywhere

\[
\begin{align*}
    v_x &= \frac{fV_x - V_z x}{Z} \\
    v_y &= \frac{fV_y - V_z y}{Z}
\end{align*}
\]

\[
\mathbf{v} = \frac{1}{Z} (\mathbf{v}_0 - V_z \mathbf{p}),
\]

\[
\mathbf{v}_0 = \mathbf{f}V_x, fV_y.
\]
Motion field and parallax

• Pure translation: $\mathbf{V}$ is constant everywhere

\[ \mathbf{v} = \frac{1}{Z} (\mathbf{v}_0 - V_z \mathbf{p}), \]

\[ \mathbf{v}_0 = \begin{pmatrix} fV_x \\ fV_y \end{pmatrix} \]

• $V_z$ is nonzero:
  - Every motion vector points toward (or away from) $\mathbf{v}_0$, the vanishing point of the translation direction
Motion field and parallax

- Pure translation: \( \mathbf{V} \) is constant everywhere

\[
\mathbf{v} = \frac{1}{Z} (\mathbf{v}_0 - V_z \mathbf{p}),
\]

\[
\mathbf{v}_0 = \begin{pmatrix} fV_x \\ fV_y \end{pmatrix}
\]

- \( V_z \) is nonzero:
  - Every motion vector points toward (or away from) \( \mathbf{v}_0 \), the vanishing point of the translation direction

- \( V_z \) is zero:
  - Motion is parallel to the image plane, all the motion vectors are parallel

- The length of the motion vectors is inversely proportional to the depth \( Z \)
Overview

• Segmentation in Video
• Optical flow
• Motion Magnification
Optical flow

Combination of slides from Rick Szeliski, Steve Seitz, Alyosha Efros and Bill Freeman and Fredo Durand
Motion estimation: Optical flow

Will start by estimating motion of each pixel separately
Then will consider motion of entire image
Why estimate motion?

Lots of uses

- Track object behavior
- Correct for camera jitter (stabilization)
- Align images (mosaics)
- 3D shape reconstruction
- Special effects
Problem definition: optical flow

How to estimate pixel motion from image $H$ to image $I$?

- Solve pixel correspondence problem
  - given a pixel in $H$, look for nearby pixels of the same color in $I$

Key assumptions

- **color constancy**: a point in $H$ looks the same in $I$
  - For grayscale images, this is brightness constancy
- **small motion**: points do not move very far

This is called the optical flow problem
Optical flow constraints (grayscale images)

Let’s look at these constraints more closely

- brightness constancy: Q: what’s the equation?
  \[ H(x, y) = I(x+u, y+v) \]
  
- small motion: (u and v are less than 1 pixel)
  - suppose we take the Taylor series expansion of I:
    \[ I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms} \]
    
    \[ \approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v \]
Optical flow equation

Combining these two equations

\[ 0 = I(x + u, y + v) - H(x, y) \]
\[ \approx I(x, y) + I_x u + I_y v - H(x, y) \]
\[ \approx (I(x, y) - H(x, y)) + I_x u + I_y v \]
\[ \approx I_t + I_x u + I_y v \]
\[ \approx I_t + \nabla I \cdot [u \ v] \]

In the limit as \( u \) and \( v \) go to zero, this becomes exact

\[ 0 = I_t + \nabla I \cdot [\partial x \ \partial y] \]
Optical flow equation

\[ 0 = I_t + \nabla I \cdot [u \ v] \]

Q: how many unknowns and equations per pixel?

2 unknowns, one equation

Intuitively, what does this constraint mean?

- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown

This explains the Barber Pole illusion

http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm
http://www.liv.ac.uk/~marcob/Trieste/barberpole.html

http://en.wikipedia.org/wiki/Barber's_pole
Aperture problem
Aperture problem
Solving the aperture problem

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel’s neighbors have the same (u,v)
    - If we use a 5x5 window, that gives us 25 equations per pixel!

\[ 0 = I_t(p_i) + \nabla I(p_i) \cdot [u \ v] \]

\[
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= 
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
\]

\[
A \quad d \quad b
\]

25x2 \quad 2x1 \quad 25x1
How to get more equations for a pixel?

• Basic idea: impose additional constraints
  – most common is to assume that the flow field is smooth locally
  – one method: pretend the pixel's neighbors have the same \((u,v)\)
    » If we use a 5x5 window, that gives us 25*3 equations per pixel!

\[
0 = I_t(p_1)[0, 1, 2] + \nabla I(p_1)[0, 1, 2] \cdot [u \ v]
\]

\[
\begin{bmatrix}
I_x(p_1)[0] & I_y(p_1)[0] \\
I_x(p_1)[1] & I_y(p_1)[1] \\
I_x(p_1)[2] & I_y(p_1)[2] \\
\vdots & \vdots \\
I_x(p_{25})[0] & I_y(p_{25})[0] \\
I_x(p_{25})[1] & I_y(p_{25})[1] \\
I_x(p_{25})[2] & I_y(p_{25})[2]
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -
\begin{bmatrix}
I_t(p_1)[0] \\
I_t(p_1)[1] \\
I_t(p_1)[2] \\
\vdots \\
I_t(p_{25})[0] \\
I_t(p_{25})[1] \\
I_t(p_{25})[2]
\end{bmatrix}
\]

\[
A_{75 \times 2} \quad d_{2 \times 1} \quad b_{75 \times 1}
\]

Note that RGB is not enough to disambiguate because R, G & B are correlated
Just provides better gradient
Lukas-Kanade flow

Prob: we have more equations than unknowns

\[
\begin{align*}
A_{25x2} \quad d_{2x1} \quad b_{25x1} & \quad \rightarrow \quad \text{minimize} \quad \|Ad - b\|^2 \\
\end{align*}
\]

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

\[
(A^T A)_{2x2} \quad d_{2x1} \quad A^T b_{2x1}
\]

\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} =
\begin{bmatrix}
- \sum I_x I_t \\
- \sum I_y I_t
\end{bmatrix}
\]

\[
A^T A \quad A^T b
\]

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)
The gradient constraint:

\[ I_x u + I_y v + I_t = 0 \]
\[ \nabla I \cdot \mathbf{U} = 0 \]

Defines a line in the \((u,v)\) space

Normal Flow:

\[ u_\perp = -\frac{I_t}{|\nabla I|} \frac{\nabla I}{|\nabla I|} \]
Combining Local Constraints

\[ \nabla I^1 \cdot U = -I_t^1 \]

\[ \nabla I^2 \cdot U = -I_t^2 \]

\[ \nabla I^3 \cdot U = -I_t^3 \]

etc.
Conditions for solvability

- Optimal \((u, v)\) satisfies Lucas-Kanade equation

\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} = -
\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}

A^T A

A^T b

When is This Solvable?

- \(A^T A\) should be invertible
- \(A^T A\) should not be too small due to noise
  - eigenvalues \(\lambda_1\) and \(\lambda_2\) of \(A^T A\) should not be too small
- \(A^T A\) should be well-conditioned
  - \(\lambda_1 / \lambda_2\) should not be too large (\(\lambda_1 =\) larger eigenvalue)

\(A^T A\) is solvable when there is no aperture problem

\[
A^T A = \begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix} = \sum \begin{bmatrix}
I_x \\
I_y
\end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T
\]
Eigenvectors of $A^T A$

\[ A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T \]

- Recall the Harris corner detector: $M = A^T A$ is the \textit{second moment matrix}
- The eigenvectors and eigenvalues of $M$ relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it
Interpreting the eigenvalues

Classification of image points using eigenvalues of the second moment matrix:

- \( \lambda_1 \) and \( \lambda_2 \) are small: “Flat” region
- \( \lambda_1 \) and \( \lambda_2 \) are large, \( \lambda_1 \sim \lambda_2 \): “Corner”
- \( \lambda_1 \gg \lambda_2 \): “Edge”
- \( \lambda_2 \gg \lambda_1 \): “Corner”
\[ \sum \nabla I (\nabla I)^T \]

- large gradients, all the same
- large \( \lambda_1 \), small \( \lambda_2 \)
Low texture region

\[
\sum \nabla I(\nabla I)^T
\]
- gradients have small magnitude
- small \( \lambda_1 \), small \( \lambda_2 \)
High textured region

\[ \sum \nabla I (\nabla I)^T \]

- gradients are different, large magnitudes
- large \( \lambda_1 \), large \( \lambda_2 \)
Observation

This is a two image problem BUT

- Can measure sensitivity by just looking at one of the images!
- This tells us which pixels are easy to track, which are hard
  - very useful later on when we do feature tracking...
Motion models

- **Translation**: 2 unknowns
- **Affine**: 6 unknowns
- **Perspective**: 8 unknowns
- **3D rotation**: 3 unknowns
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 brightness constancy equation:

\[ I_x \cdot u + I_y \cdot v + I_t \approx 0 \]
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 brightness constancy equation:

\[ I_x (a_1 + a_2 x + a_3 y) + I_y (a_4 + a_5 x + a_6 y) + I_t \approx 0 \]

- Each pixel provides 1 linear constraint in 6 unknowns

- Least squares minimization:

\[ Err(\tilde{a}) = \sum I_x (a_1 + a_2 x + a_3 y) + I_y (a_4 + a_5 x + a_6 y) + I_t^2 \]
Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

- Suppose $A^TA$ is easily invertible
- Suppose there is not much noise in the image

When our assumptions are violated

- Brightness constancy is not satisfied
- The motion is not small
- A point does not move like its neighbors
  - window size is too large
  - what is the ideal window size?
Iterative Refinement

Iterative Lukas-Kanade Algorithm

1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp H towards I using the estimated flow field
   - *use image warping techniques*
3. Repeat until convergence
Optical Flow: Iterative Estimation

Initial guess: \( d_0 = 0 \)
Estimate: \( d_1 = d_0 + \hat{d} \)

(Using \( d \) for displacement here instead of \( u \))
Optical Flow: Iterative Estimation

Initial guess: $d_1$

Estimate: $d_2 = d_1 + \hat{d}$
Optical Flow: Iterative Estimation

Initial guess: $d_2$

Estimate: $d_3 = d_2 + \hat{d}$

$f_1(x - d_2)$

$f_2(x)$
Optical Flow: Iterative Estimation

\[ f_1(x - d_3) \approx f_2(x) \]
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)
Revisiting the small motion assumption

Is this motion small enough?
  • Probably not—it’s much larger than one pixel (2\textsuperscript{nd} order terms dominate)
  • How might we solve this problem?
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?

To overcome aliasing: coarse-to-fine estimation.
Reduce the resolution!
Coarse-to-fine optical flow estimation

Gaussian pyramid of image H

u=10 pixels
u=5 pixels
u=2.5 pixels
u=1.25 pixels

Gaussian pyramid of image I
Coarse-to-fine optical flow estimation

- Gaussian pyramid of image H
- Gaussian pyramid of image I
- Run iterative L-K
- Warp & upsample
- Run iterative L-K
- Run iterative L-K
Beyond Translation

So far, our patch can only translate in \((u,v)\)

What about other motion models?

- rotation, affine, perspective

Same thing but need to add an appropriate Jacobian

See Szeliski’s survey of Panorama stitching

\[
A^T A = \sum_i J \nabla I (\nabla I)^T J^T \\
A^T b = - \sum_i J^T l_t (\nabla I)^T
\]
Recap: Classes of Techniques

**Feature-based methods (e.g. SIFT+Ransac+regression)**
- Extract visual features (corners, textured areas) and track them over multiple frames
- Sparse motion fields, but possibly robust tracking
- Suitable especially when image motion is large (10-s of pixels)

**Direct-methods (e.g. optical flow)**
- 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)
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)
Retiming

http://www.realviz.com/retiming.htm
Layered motion

- Break image sequence into “layers” each of which has a coherent motion

What are layers?

- Each layer is defined by an alpha mask and an affine motion model

Motion segmentation with an affine model

\[ u(x, y) = a_1 + a_2 x + a_3 y \]

\[ v(x, y) = a_4 + a_5 x + a_6 y \]

Local flow estimates

Motion segmentation with an affine model

\[
\begin{align*}
    u(x, y) &= a_1 + a_2 x + a_3 y \\
    v(x, y) &= a_4 + a_5 x + a_6 y
\end{align*}
\]

Equation of a plane (parameters \(a_1, a_2, a_3\) can be found by least squares)

Motion segmentation with an affine model

\[ u(x, y) = a_1 + a_2 x + a_3 y \]
\[ v(x, y) = a_4 + a_5 x + a_6 y \]

Equation of a plane (parameters \( a_1, a_2, a_3 \) can be found by least squares).

1D example

True flow

Local flow estimate

Segmented estimate

Line fitting

“Foreground”

“Background”

Occlusion

How do we estimate the layers?

- Compute local flow in a coarse-to-fine fashion
- Obtain a set of initial affine motion hypotheses
  - Divide the image into blocks and estimate affine motion parameters in each block by least squares
    - Eliminate hypotheses with high residual error
  - Perform k-means clustering on affine motion parameters
    - Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene
- Iterate until convergence:
  - Assign each pixel to best hypothesis
    - Pixels with high residual error remain unassigned
  - Perform region filtering to enforce spatial constraints
  - Re-estimate affine motions in each region

Example result

Overview

• Segmentation in Video
• Optical flow
• Motion Magnification
Motion Magnification

Ce Liu     Antonio Torralba     William T. Freeman
Frédo Durand     Edward H. Adelson

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Motion Microscopy

How can we see all the subtle motions in a video sequence?

Original sequence

Magnified sequence
Naïve Approach

- Magnify the estimated optical flow field
- Rendering by warping
Layer-based Motion Magnification Processing Pipeline

Input raw video sequence

Video Registration

Feature point tracking

Trajectory clustering

Magnification, texture fill-in, rendering

Layer segmentation

Dense optical flow interpolation

Output magnified video sequence

User interaction

Layer-based motion analysis

Stationary camera, stationary background
Layer-based Motion Magnification

Video Registration

Input raw video sequence

Video Registration

Feature point tracking

Trajectory clustering

Layer segmentation

Dense optical flow interpolation

Magnification, texture fill-in, rendering

User interaction

Output magnified video sequence

Layer-based motion analysis

Stationary camera, stationary background
Robust Video Registration

• Find feature points with Harris corner detector on the reference frame
• Brute force tracking feature points
• Select a set of robust feature points with inlier and outlier estimation (most from the rigid background)
• Warp each frame to the reference frame with a global affine transform
Motion Magnification Pipeline

Feature Point Tracking

Input raw video sequence

Video Registration

Feature point tracking

Output magnified video sequence

Magnification, texture fill-in, rendering

Layer segmentation

Dense optical flow interpolation

Layer-based motion analysis
Challenges (1)
Adaptive Region of Support

- Brute force search

- Learn adaptive region of support using expectation-maximization (EM) algorithm

Confused by occlusion!
Challenges (2)
Trajectory Pruning

- Tracking with adaptive region of support
- Outlier detection and removal by interpolation

Nonsense at full occlusion!
Comparison

Without adaptive region of support and trajectory pruning

With adaptive region of support and trajectory pruning
Motion Magnification Pipeline

Trajectory Clustering

- Input raw video sequence
- Video Registration
- Feature point tracking
- Trajectory clustering
- Magnification, texture fill-in, rendering
- Layer segmentation
- Dense optical flow interpolation

Output magnified video sequence

User interaction

Layer-based motion analysis

SIGGRAPH 2005
Normalized Complex Correlation

- The similarity metric should be independent of phase and magnitude
- Normalized complex correlation

\[ S(C_1, C_2) = \frac{\left| \sum_t C_1(t) \overline{C_2(t)} \right|^2}{\sqrt{\sum_t C_1(t) \overline{C_1(t)}} \sqrt{\sum_t C_2(t) \overline{C_2(t)}}} \]
Spectral Clustering

Affinity matrix

Clustering

Reordering of affinity matrix

Two clusters

Trajectory
Motion Magnification Pipeline
Dense Optical Flow Field

Input raw video sequence → Video Registration → Feature point tracking → Trajectory clustering

Magnification, texture fill-in, rendering
Layer segmentation
Dense optical flow interpolation

Output magnified video sequence

User interaction
Layer-based motion analysis
From Sparse Feature Points to Dense Optical Flow Field

- Interpolate dense optical flow field using locally weighted linear regression

Cluster 1: leaves
Cluster 2: swing
Motion Magnification Pipeline

Layer Segmentation

- Input raw video sequence
- Video Registration
  - Feature point tracking
  - Trajectory clustering
- Magnification, texture fill-in, rendering
- User interaction
- Output magnified video sequence

Layer-based motion analysis

Dense optical flow interpolation
Motion Layer Assignment

• Assign each pixel to a motion cluster layer, using four cues:
  – **Motion likelihood**—consistency of pixel’s intensity if it moves with the motion of a given layer (dense optical flow field)
  – **Color likelihood**—consistency of the color in a layer
  – **Spatial connectivity**—adjacent pixels favored to belong the same group
  – **Temporal coherence**—label assignment stays constant over time

• Energy minimization using graph cuts
Segmentation Results

- Two additional layers: static background and outlier
Motion Magnification Pipeline
Editing and Rendering

Input raw video sequence → Video Registration → Feature point tracking → Trajectory clustering

Magnification, texture fill-in, rendering → Layer segmentation → Dense optical flow interpolation

Layer-based motion analysis

Output magnified video sequence → User interaction
Layered Motion Representation for Motion Processing

- **Background**
- **Layer 1**
- **Layer 2**

Layer mask

Occluding layers

Appearance for each layer before texture filling-in

Appearance for each layer after texture filling-in
Video

Motion Magnification
Is the Baby Breathing?

Original Sequence
Are the Motions Real?
Are the Motions Real?
Applications

- Education
- Entertainment
- Mechanical engineering
- Medical diagnosis
Conclusion

• Motion magnification
  – A motion microscopy technique
• Layer-based motion processing system
  – Robust feature point tracking
  – Reliable trajectory clustering
  – Dense optical flow field interpolation
  – Layer segmentation combining multiple cues
Thank you!

Motion Magnification

Ce Liu   Antonio Torralba   William T. Freeman   Frédo Durand   Edward H. Adelson

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology