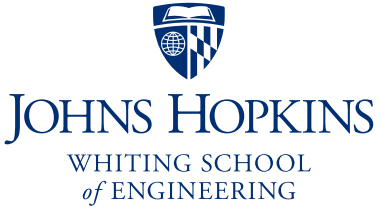
Johns Hopkins Engineering

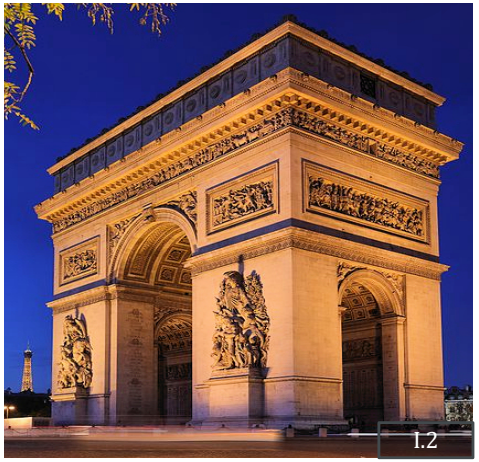
Computer Vision

Motion and Optical Flow



From Two to …

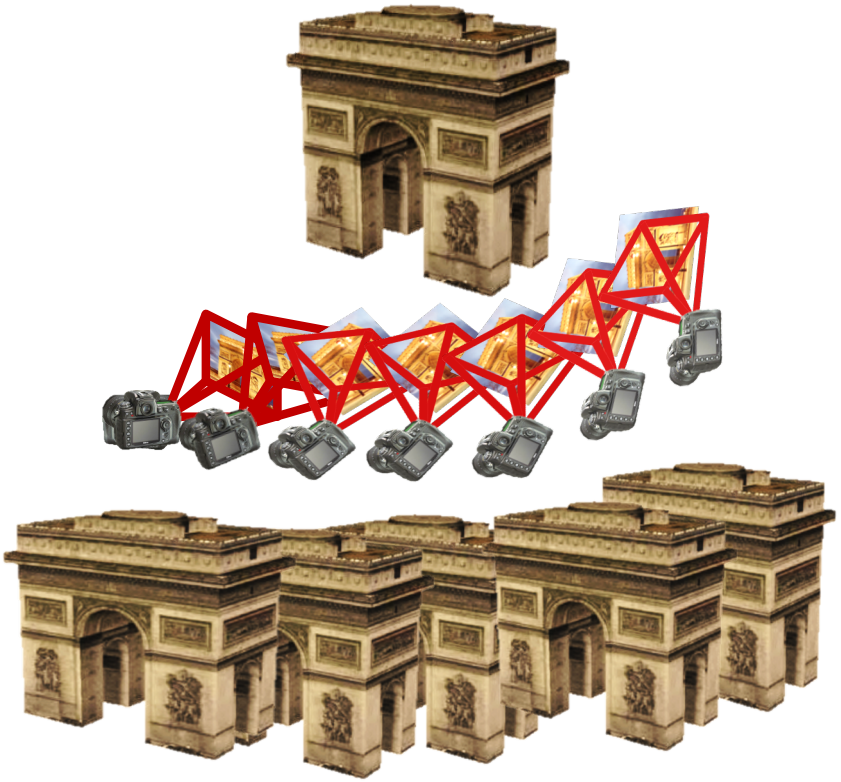
Left Image

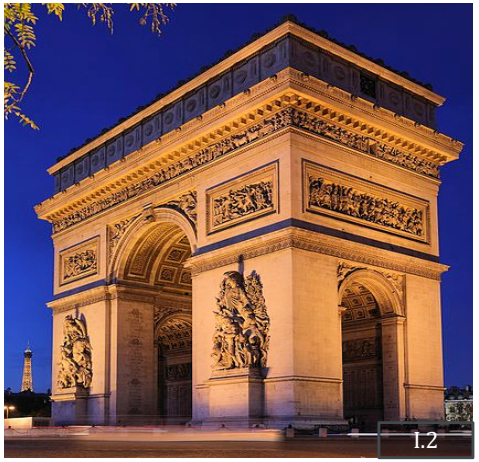


Right Image

From Two to … Many Images (In Sequence)

Left Image





Right Image

Motion and Optical Flow

Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

Topics:

- The motion field
- Optical flow
- Calculating Optical Flow
- Applications of Motion

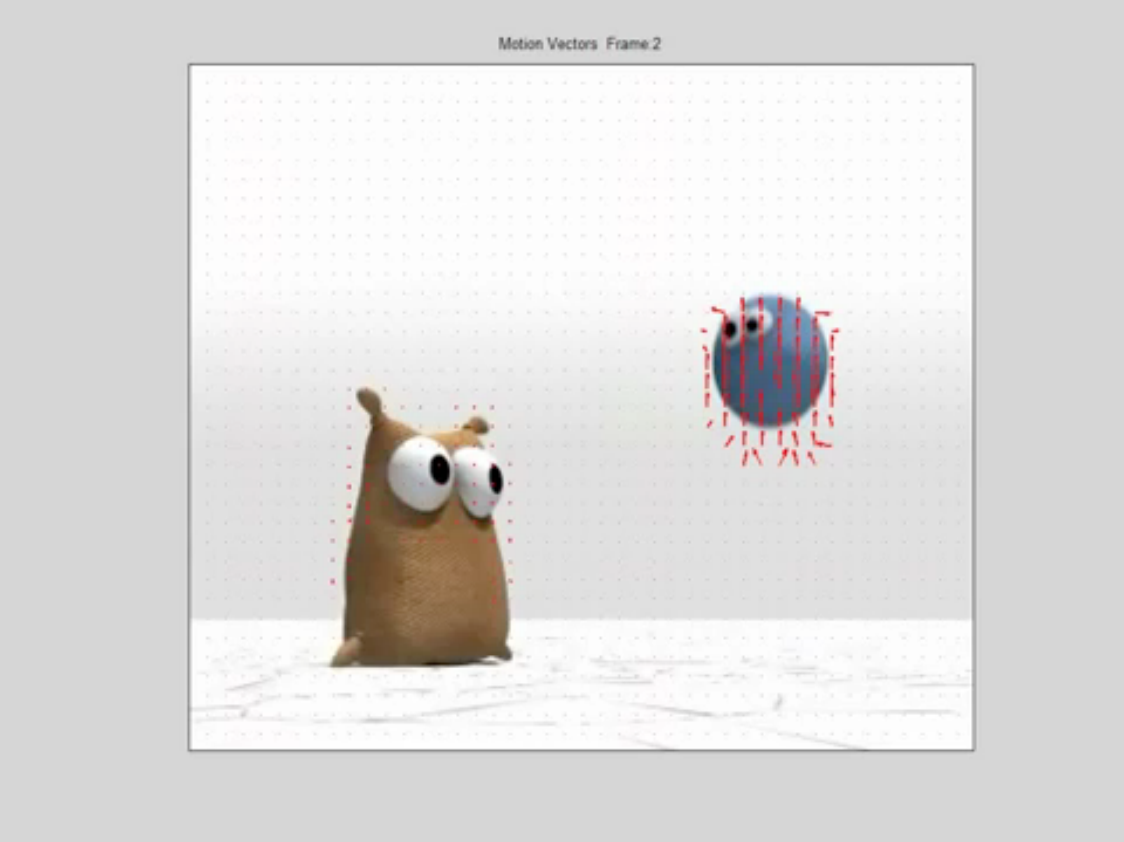
Motion and Optical Flow

Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

Topics:

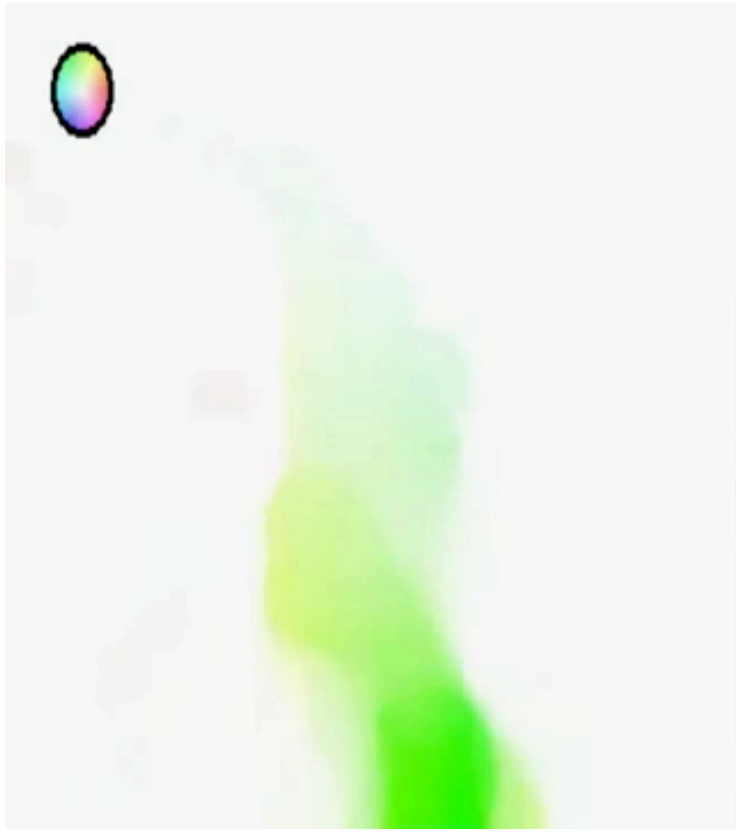
- The motion field
- Optical flow

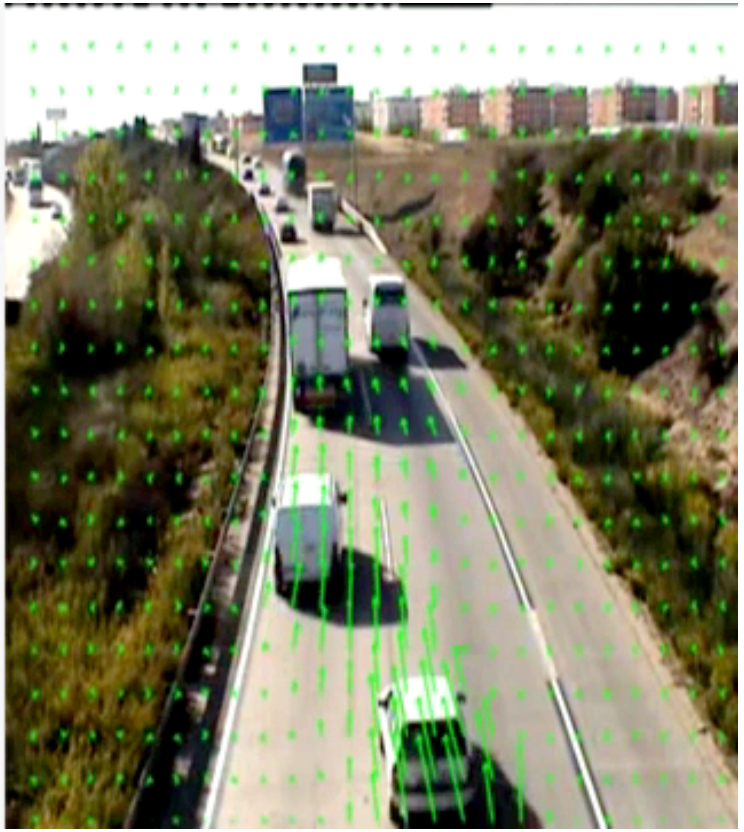
Motion Vectors



https://www.youtube.com/watch?v=BDtvjKMl30w

Motion Vectors (cont.)

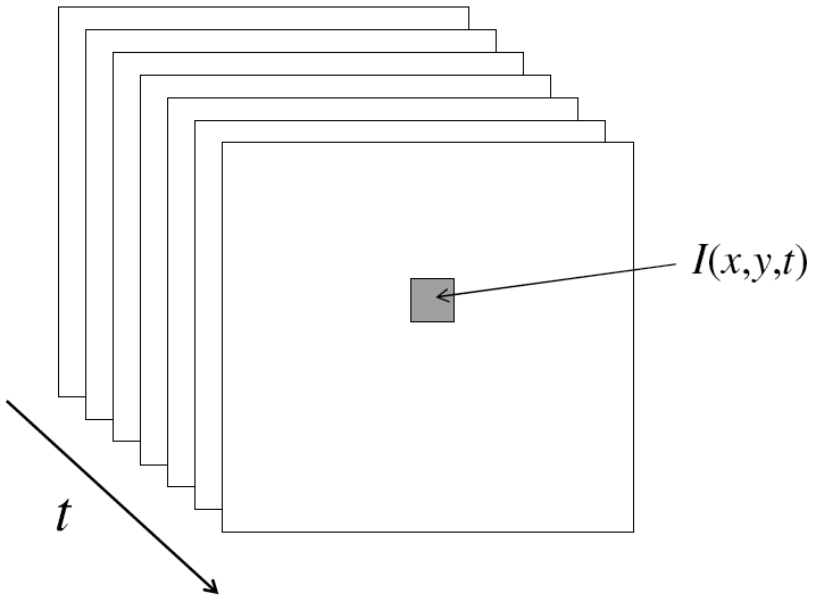




https://www.youtube.com/watch?v=5rR_9YIcg_s

Video

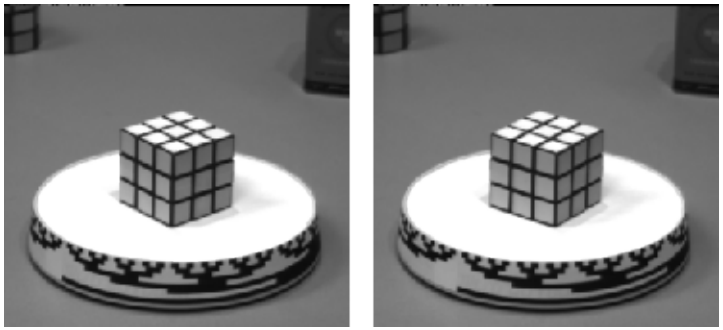
- A video is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and time (t)

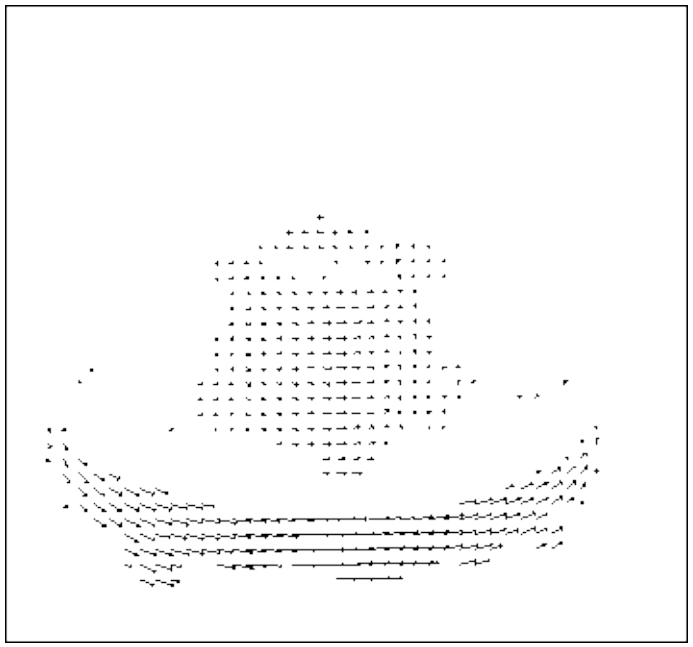


Slide adapted from K. Grauman.

Motion field

- The motion field is the projection of the 3D scene motion into the image

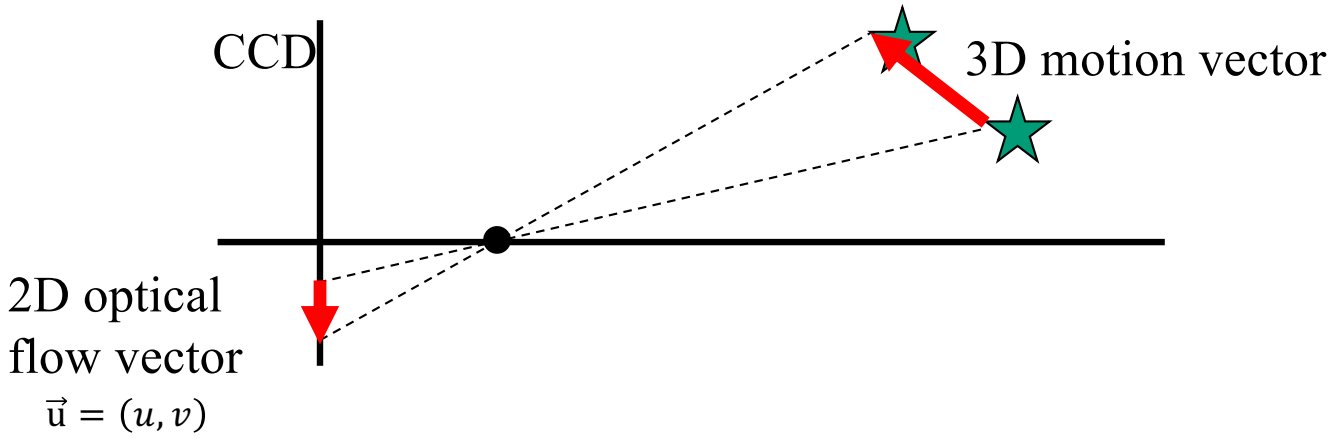




Slide adapted from K. Grauman; images from Russel and Norvig.

Motion Field & Optical Flow Field

- Motion Field = Real world 3D motion
- Optical Flow Field = Projection of the motion field onto the 2d image (The apparent motion in the image due to the motion field)

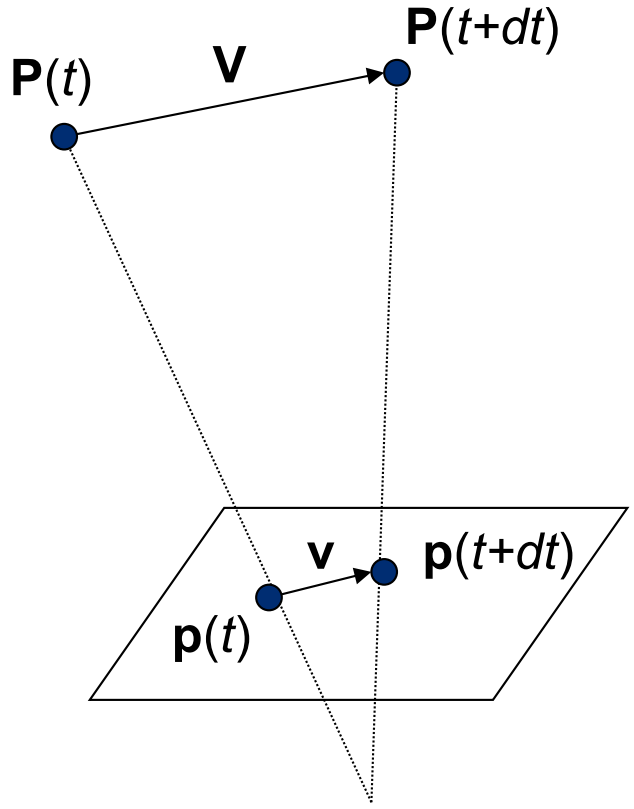


Slide adapted from Savarese.

Motion field and parallax

- $\mathbf{P}(t)$ is a moving 3D point
- Velocity of scene point: $\mathbf{V} = \mathrm{d}\mathbf{P}/\mathrm{d}t$
- $\mathbf{p}(t) = (x(t),y(t))$ is the projection of **P** in the image
- Apparent velocity **v** in the image: given by components

  $v_x = \mathrm{d}x/\mathrm{d}t$ and $v_y = \mathrm{d}y/\mathrm{d}t$
- These components are known as the *motion field* of the image



Slide adapted from K. Grauman.

Motion field and parallax

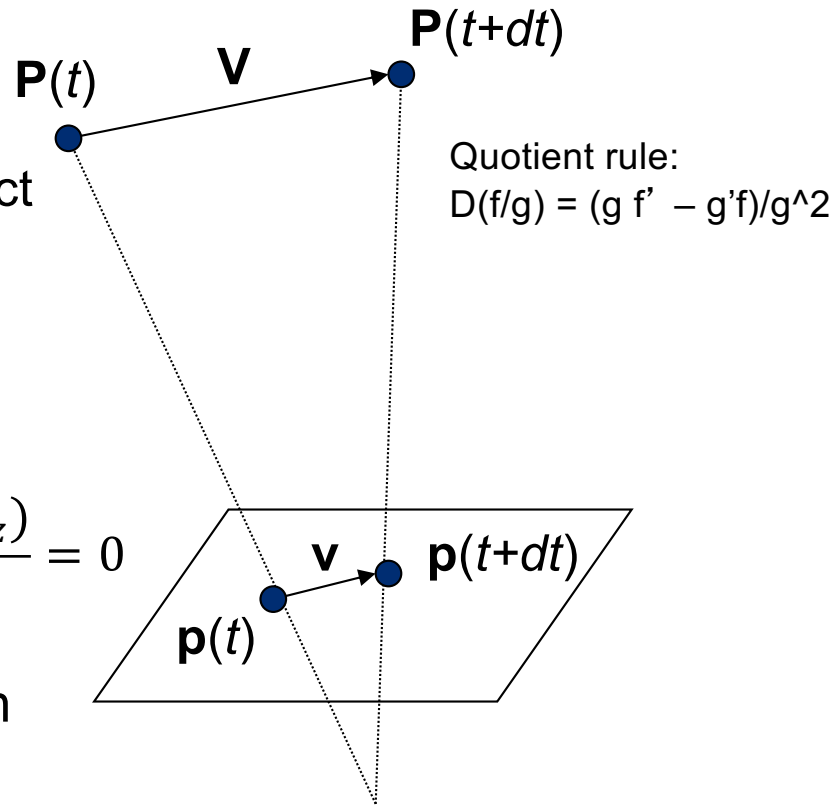
$$\mathbf{V} = (V_x, V_y, V_Z) \qquad \mathbf{p} = f\frac{\mathbf{P}}{Z}$$

To find image velocity **v**, differentiate **p** with respect to *t* (using quotient rule):

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

$$v_x = \frac{fV_x - V_z x}{Z} \qquad v_y = \frac{fV_y - V_z y}{Z} \qquad v_z = \frac{f(V_z - V_z)}{Z} = 0$$

Image motion is a function of both the 3D motion (**V**) and the depth of the 3D point (*Z*)



Slide adapted from K. Grauman.

Motion field and parallax

- Pure camera translation: **V** is constant everywhere

$$v_x = \frac{fV_x - V_z x}{Z}$$

$$v_y = \frac{fV_y - V_z y}{Z}$$

$$\mathbf{v} = \frac{V_z}{z}(\boldsymbol{p}_0 - \mathbf{p}),$$

$$\boldsymbol{p}_0 = (x_0, y_0) = \left(f\frac{V_x}{V_z}, f\frac{V_y}{V_z}, f\right) \; V_z \neq 0$$

(x_0, y_0)	focus of expansion/contraction in the image
$\mathbf{p}_0$	is the projection of the direction of translation
$\frac{z}{V_z}$	Is the "time to collision"

Slide adapted from K. Grauman.

Motion field and parallax

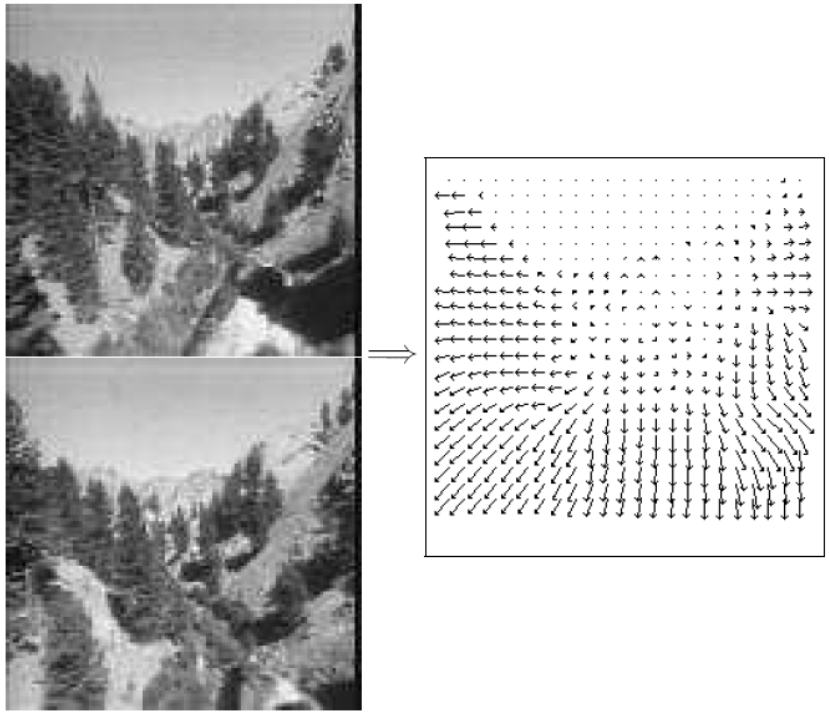


Figure 1.2: Two images taken from a helicopter flying through a canyon and the computed optical flow field.

- Length of flow vectors inversely proportional to depth Z of 3d point

$$v_x = \frac{fV_x - V_z x}{Z}$$

$$v_y = \frac{fV_y - V_z y}{Z}$$

points closer to the camera move more quickly across the image plane

Figure from Michael Black, Ph.D. Thesis
Slide adapted from K. Grauman.

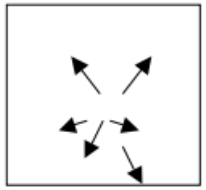
Motion field and parallax

- Pure translation: **V** is constant everywhere

$$\mathbf{v} = \frac{V_z}{z}(\boldsymbol{p}_0 - \mathbf{p}),$$

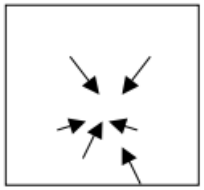
$$\boldsymbol{p}_0 = (x_0, y_0) = \left(f\frac{V_x}{V_z}, f\frac{V_y}{V_z}, f\right)$$

- V_z is nonzero:
  - Every motion vector points toward (or away from) $\mathbf{p}_0$, the vanishing point of the translation direction
  - Length inversely proportional to time to collision



$V_z < 0$

camera moving towards object



$V_z > 0$

camera moving away from object

Slide adapted from K. Grauman.

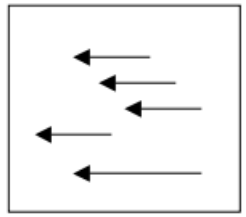
Motion field and parallax

- Pure translation: **V** is constant everywhere

$$v_x = \frac{fV_x - V_z x}{Z} \qquad v_y = \frac{fV_y - V_z y}{Z}$$

$$\mathbf{v} = \frac{f}{z} V$$

- 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



Slide adapted from K. Grauman.

General Visual Motion

Let us assume there is one rigid object and the camera moves with translational velocity $\mathbf{T}$ and rotational velocity $\boldsymbol{\omega}$

For a given point $\mathbf{P} = [X, Y, Z]^T$ on the object, we have

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

where

$$\mathbf{V} = -\mathbf{T} - \boldsymbol{\omega} \times \mathbf{P}, \quad V_z = -T_z - \omega_x Y + \omega_y X$$

The Result:

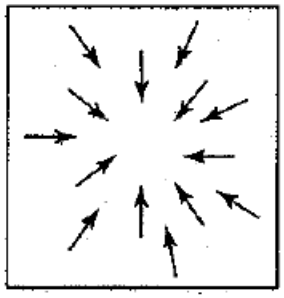
$$\frac{du}{dt} = (T_z u - f\, T_x)/Z + f\omega_y - \omega_z v - \omega_x uv/f + \omega_y u^2/f$$

$$\frac{dv}{dt} = (T_z v - f\, T_y)/Z - f\omega_x + \omega_z u + \omega_y uv/f - \omega_x v^2/f$$

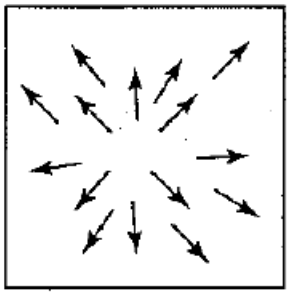
Motion due to translation: depends on depth

Motion due to rotation: independent of depth

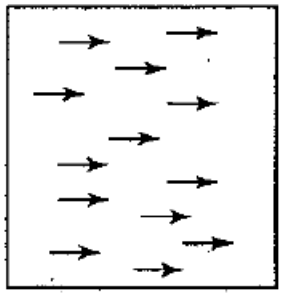
Motion field + camera motion



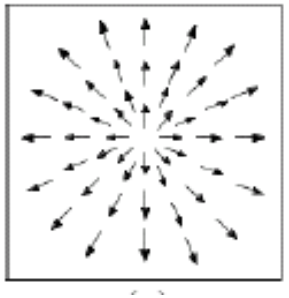
Zoom out



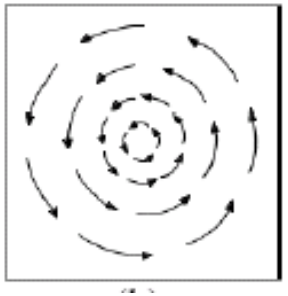
Zoom in



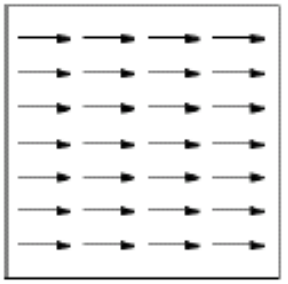
Pan right to left



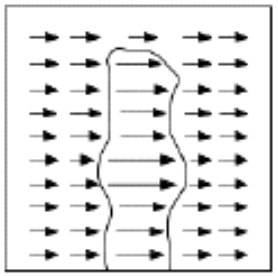
Forward motion



Rotation



Rotation about Y



Closer objects appear to move faster!!

Slide adapted from Savarese and Grauman.

Optical flow

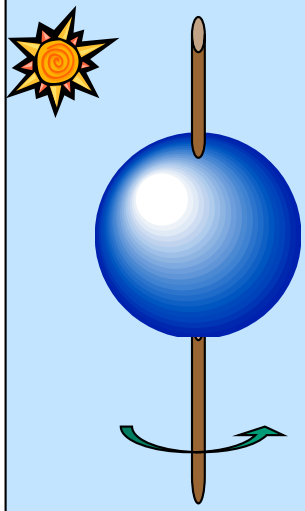
- Definition: optical flow is the ***apparent* motion** of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field

Where did each pixel in image 1 go to in image 2

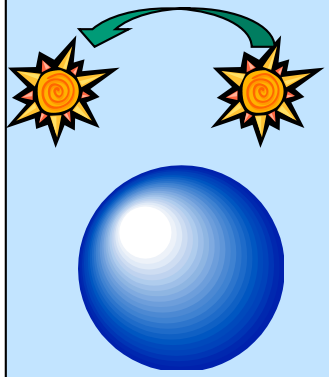
Slide adapted from K. Grauman, Zelnik-Manor, Savarese.

Is Optical Flow = Motion Field?

The screen is stationary yet displays motion



Homogeneous objects generate zero optical flow.



Fixed sphere. Changing light source.

Non-rigid texture motion

Slide adapted from Savarese.

Summary

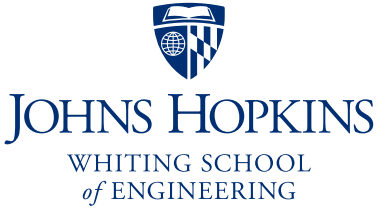
Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

- Essential concepts in this lecture:
  - The motion field
  - Optical flow and the difference between motion and optical flow
  - Geometric properties of the motion field

Johns Hopkins Engineering

Computer Vision

Motion and Optical Flow



Motion and Optical Flow

Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

Topics:

- Calculating Optical Flow

Motion estimation techniques

- 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)

- 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

Slide adapted from K. Grauman.

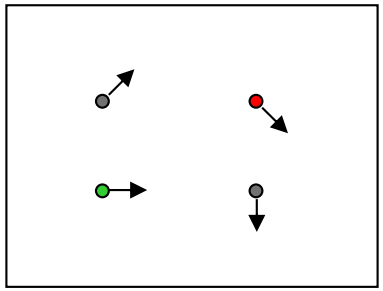
The Optical Flow Field

- Goal:
  Find for each pixel a velocity vector $\vec{u} = (u, v)$ which says:
  - How quickly is the pixel moving across the image
  - In which direction it is moving

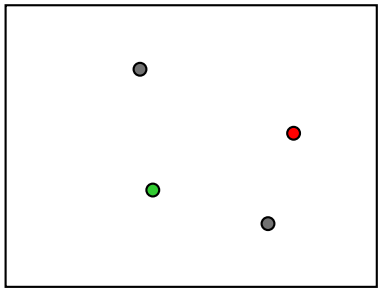
- Very long literature (continues today); we will talk about the fundamentals

Slide adapted from Savarese.

Estimating optical flow



$I(x,y,t-1)$



$I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field between them.
- Key assumptions
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

Slide adapted from K. Grauman.

Brightness constancy

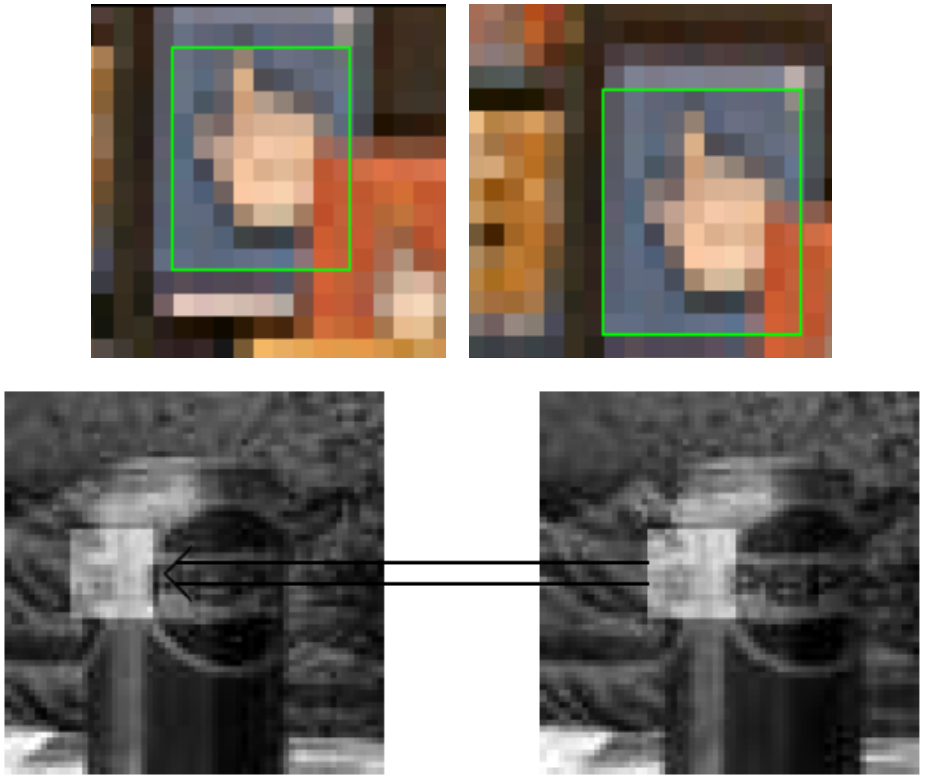
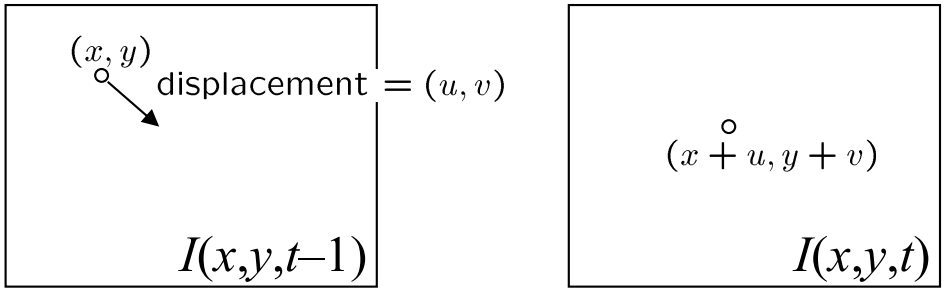


Figure 1.5: Data conservation assumption. The highlighted region in the right image looks roughly the same as the region in the left image, despite the fact that it has moved.

Figure by Michael Black
Slide adapted from K. Grauman, T. Darrell, R. Szeliski.

The brightness constancy constraint



- Brightness Constancy Equation: $I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$

Can be written as: shorthand: $I_x = \frac{\partial I}{\partial x}$

$$I(x, y, t-1) \approx I(x, y, t) + I_x \cdot u(x, y) + I_y \cdot v(x, y)$$

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

Slide adapted from K. Grauman.

The brightness constancy constraint

$$I_x \cdot u + I_y \cdot v + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation, two unknowns
- Intuitively, what does this constraint mean?

$$\nabla I \cdot (u, v) + I_t = 0$$

- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown

Slide adapted from K. Grauman.

The brightness constancy constraint

$$I_x \cdot u + I_y \cdot v + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation, two unknowns

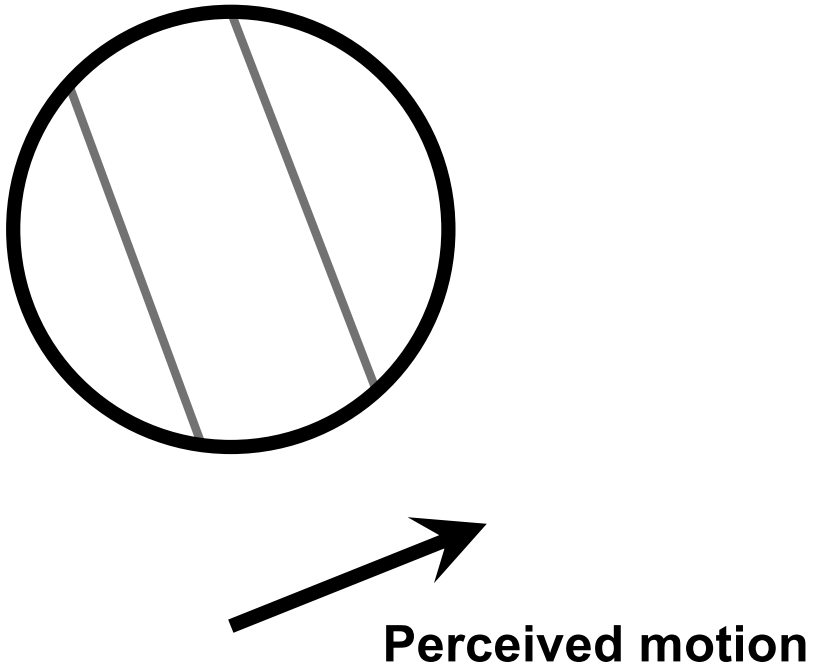
$$\nabla I \cdot (u, v) + I_t = 0$$

- Intuitively, what does this constraint mean?
- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown
- If (*u*, *v*) satisfies the equation, so does (*u+u'*, *v+v'*) if

$$\nabla I \cdot (u', v') = 0$$

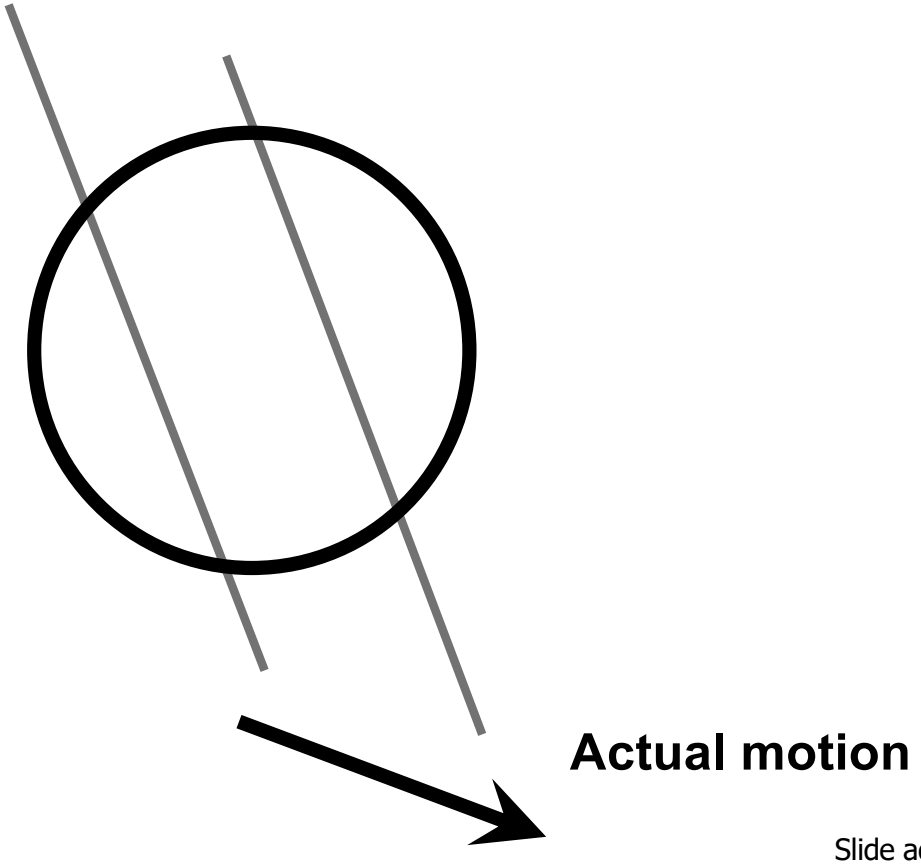
Slide adapted from K. Grauman.

The aperture problem



Slide adapted from K. Grauman.

The aperture problem



Slide adapted from K. Grauman.

The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Slide adapted from K. Grauman.

The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Slide adapted from K. Grauman.

The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Slide adapted from K. Grauman.

Lucas-Kanade: Solving the aperture problem (grayscale image)

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \ v]$$

- One approach is *regularization*
  - *Add an energy term preferring similar pixel in a neighborhood*

$$R(p_i) = \sum_{j \in N(i)} (u_i - u_j)^2 + (v_i - v_j)^2$$

- Solve a regularized least squares problem:

$$O(\boldsymbol{u}, \boldsymbol{v}) = \sum_i (I_t(p_i) + \nabla I(p_i) \cdot [u_i, v_i])^2 + R(p_i)$$

Slide adapted from Seitz, Grauman.

Lucas-Kanade: Solving the aperture problem (grayscale image)

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \ v]$$

  - If we use a 5x5 window, that gives us 25 equations per pixel

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

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Slide adapted from Seitz, Grauman.

Lucas-Kanade: Solving the aperture problem

Prob: we have more equations than unknowns

$$\underset{25\text{x}2}{A}\ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem over patches

- minimum least squares solution given by solution (in d) of: $\underset{2\text{x}2}{(A^T A)}\ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$

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

$A^T A$ $\qquad$ $A^T b$

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

Slide adapted from Seitz, Grauman.

Conditions for solvability

Look Familiar?

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

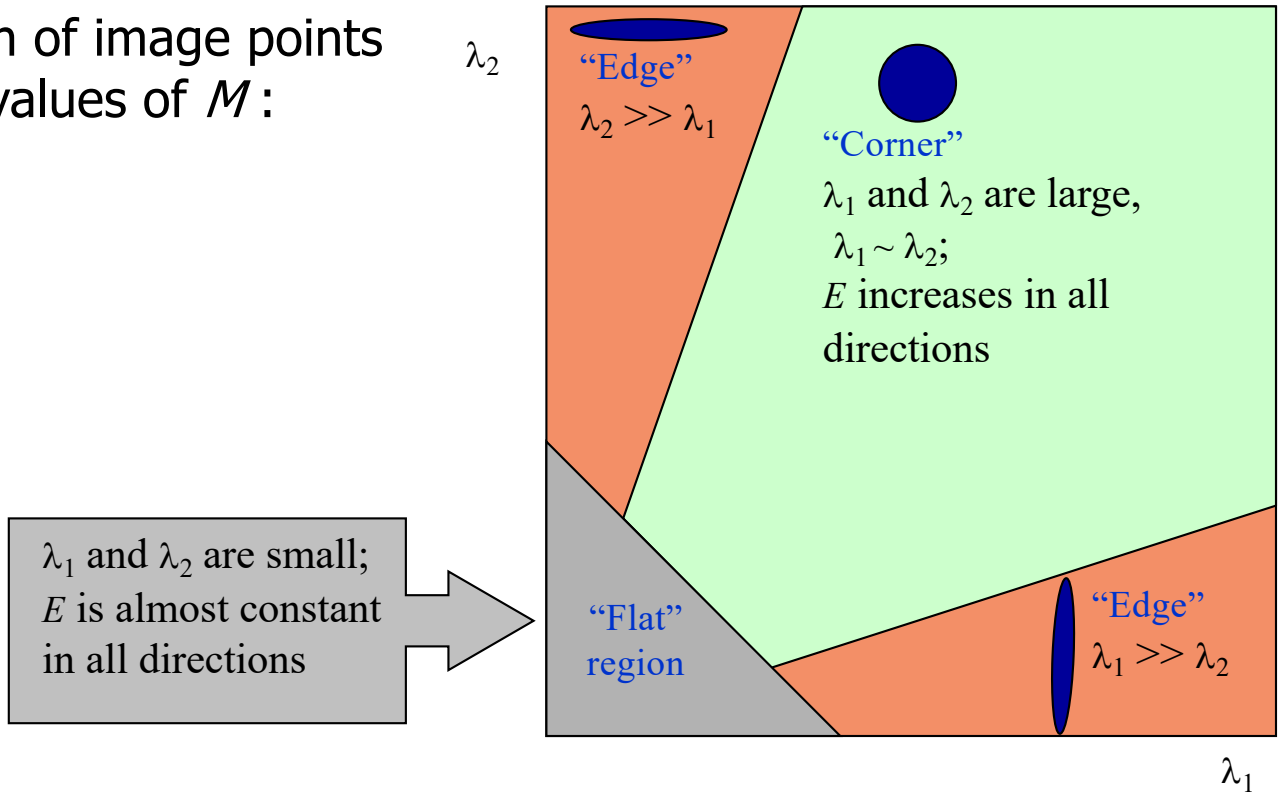
When is this solvable?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ should not be too small
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be too small
- $\mathbf{A^T A}$ should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Slide adapted from Seitz, Grauman.

Conditions for solvability

Classification of image points using eigenvalues of M :



Source: Kokkinos, Saverese.

Edge

- gradients very large or very small
- large λ_1, small λ_2

Slide adapted from Seitz, Grauman.

Low-texture region

- gradients have small magnitude
- small λ_1, small λ_2

Slide adapted from Seitz, Grauman.

High-texture region

- gradients are different, large magnitudes
- large λ_1, large λ_2

Slide adapted from Seitz, Grauman.

Can we measure optical flow reliability?

- Can we measure "quality" of optical flow in regions from just a single image?

- High Quality / Good features to track:
  - Harris corners (guarantee small error sensitivity)

- Poor Quality / Bad features to track:
  - Image points when either λ_1 or λ_2 (or both) is small (i.e., edges or uniform textured regions)

Note we could use regularization to deal with this!

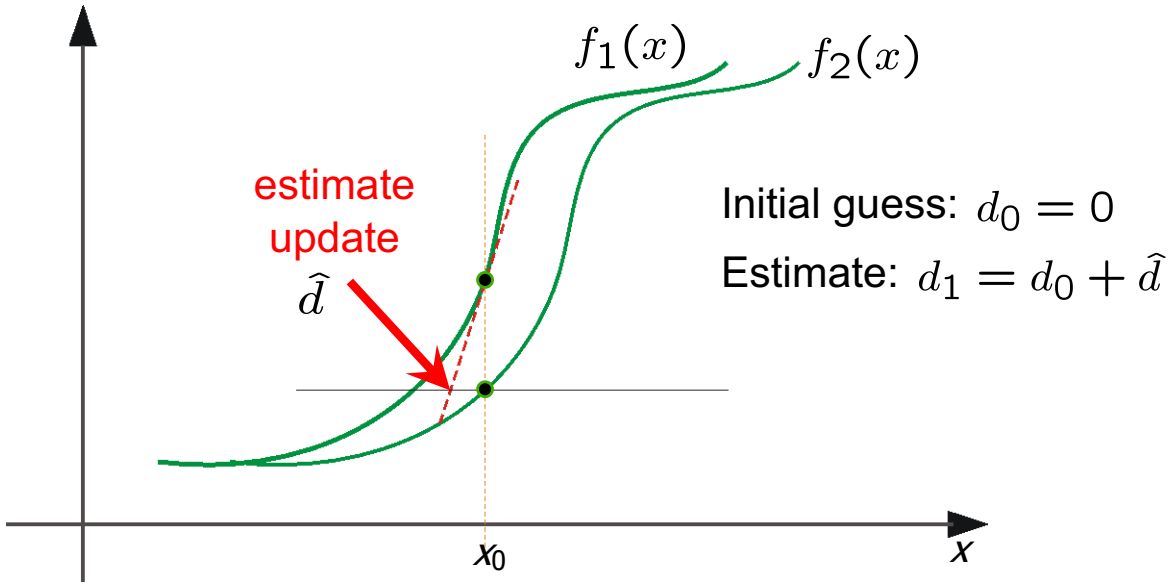
Slide adapted from Savarese.

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

Slide adapted from Szeliski.

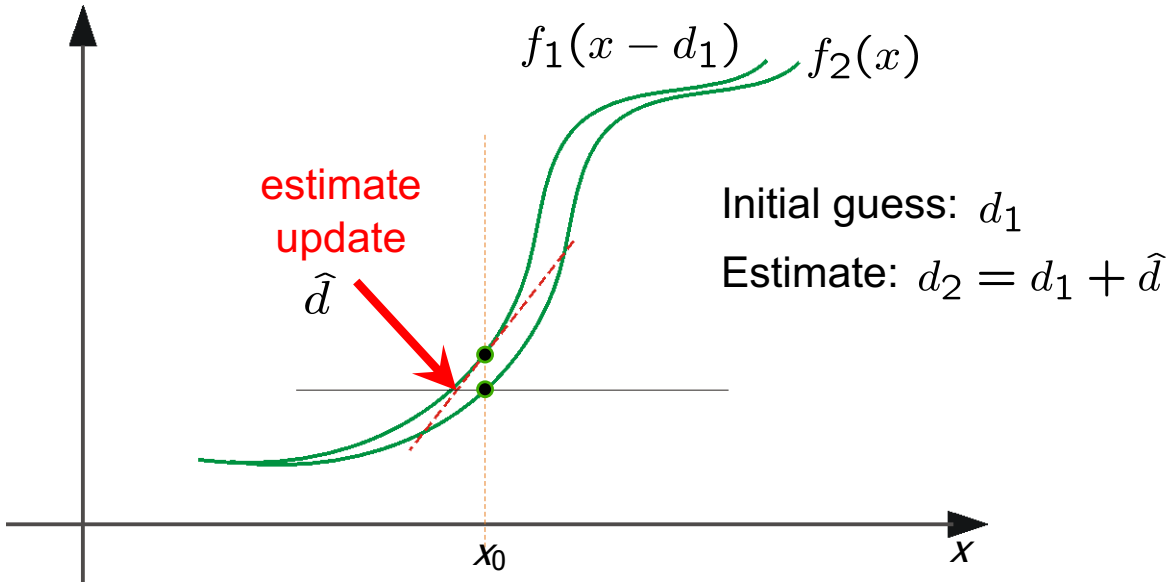
Optical Flow: Iterative Estimation



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

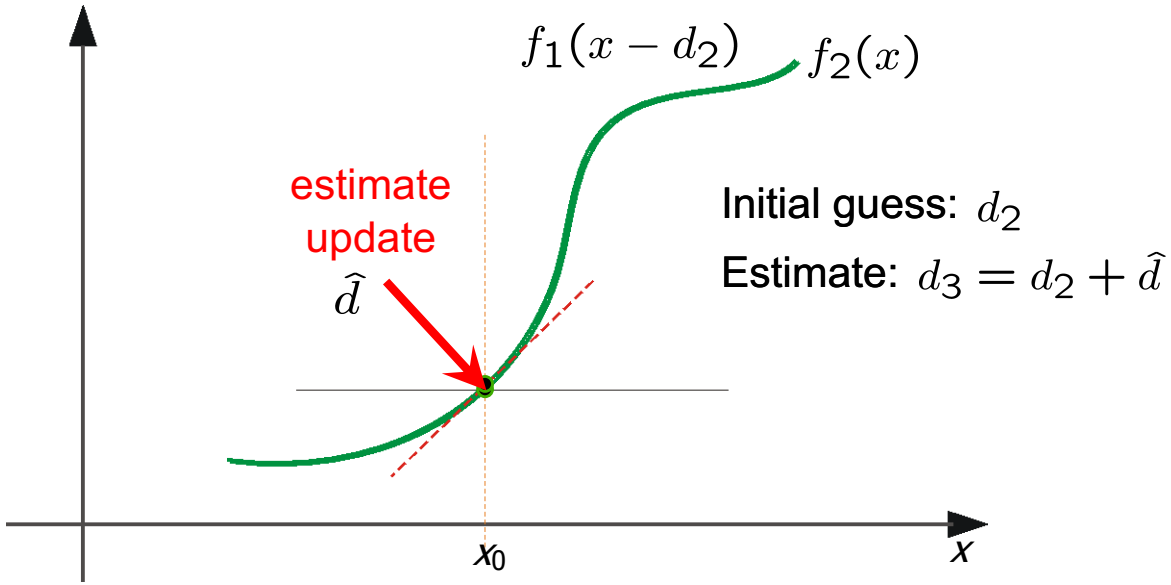
Slide adapted from Szeliski.

Optical Flow: Iterative Estimation



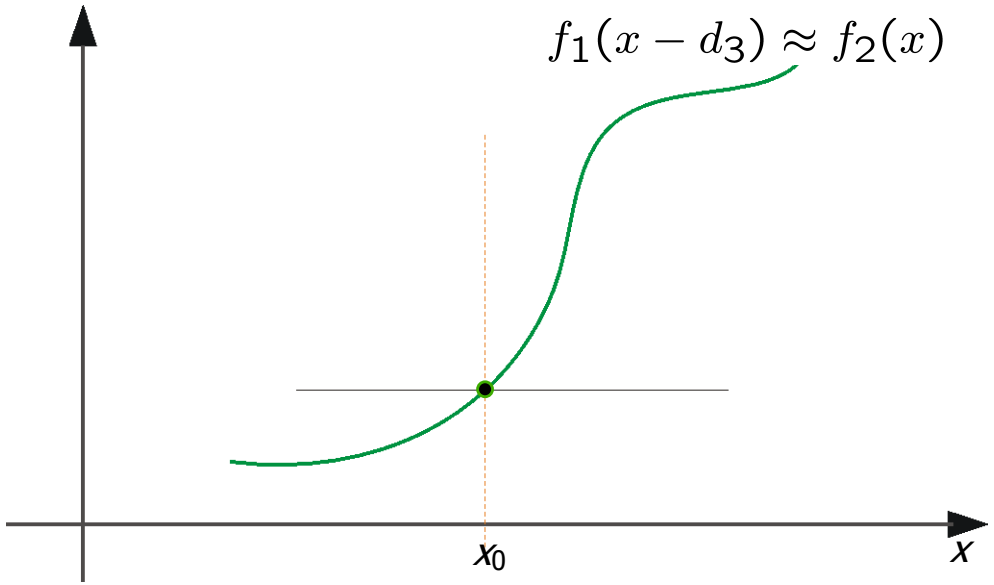
Slide adapted from Szeliski.

Optical Flow: Iterative Estimation



Slide adapted from Szeliski.

Optical Flow: Iterative Estimation

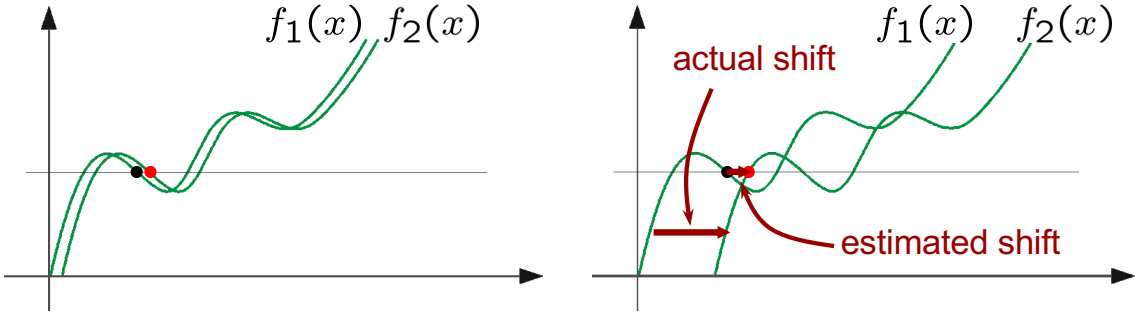


Slide adapted from Szeliski.

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?



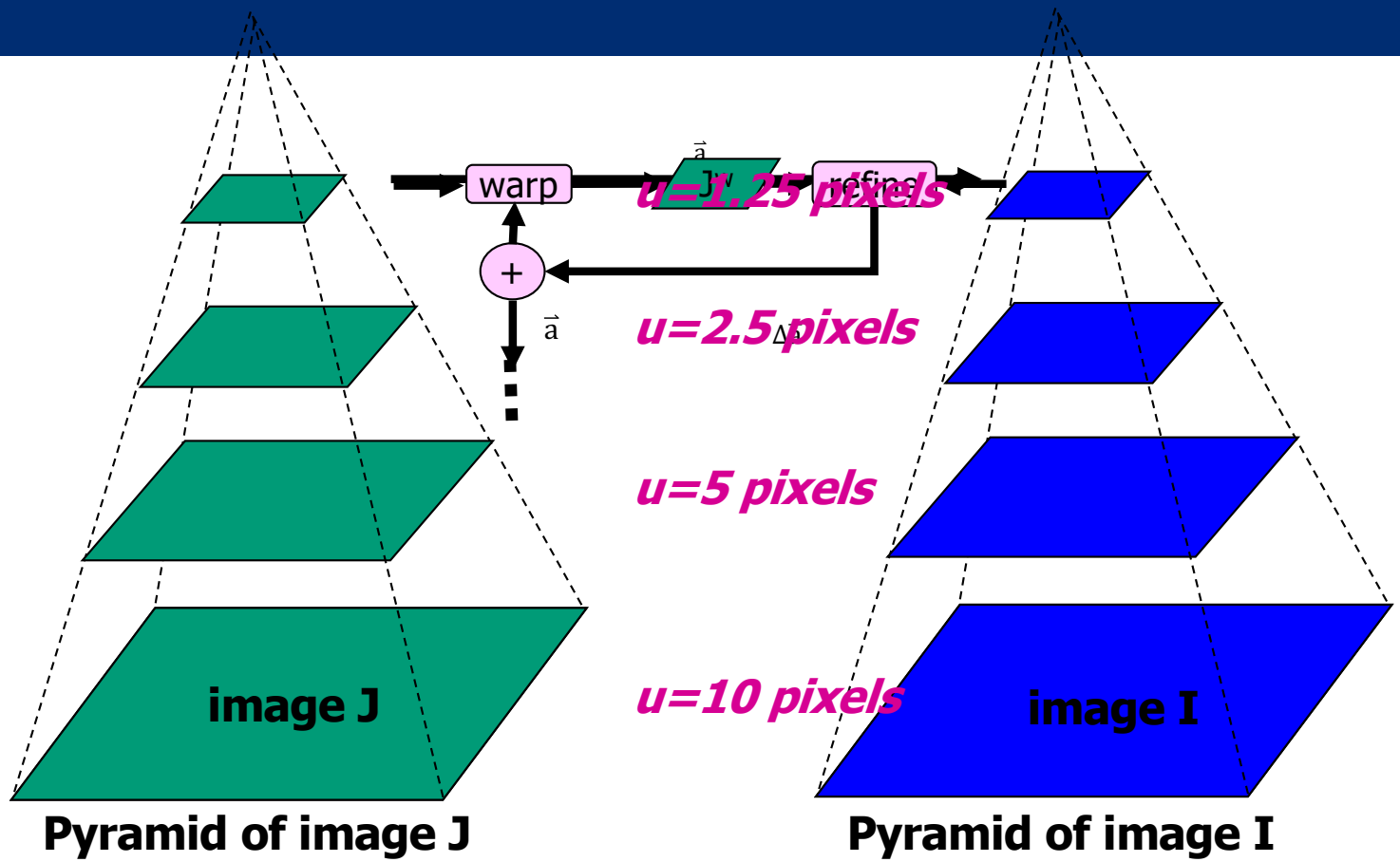
nearest match is correct (no aliasing)

nearest match is incorrect (aliasing)

To overcome aliasing: coarse-to-fine estimation.

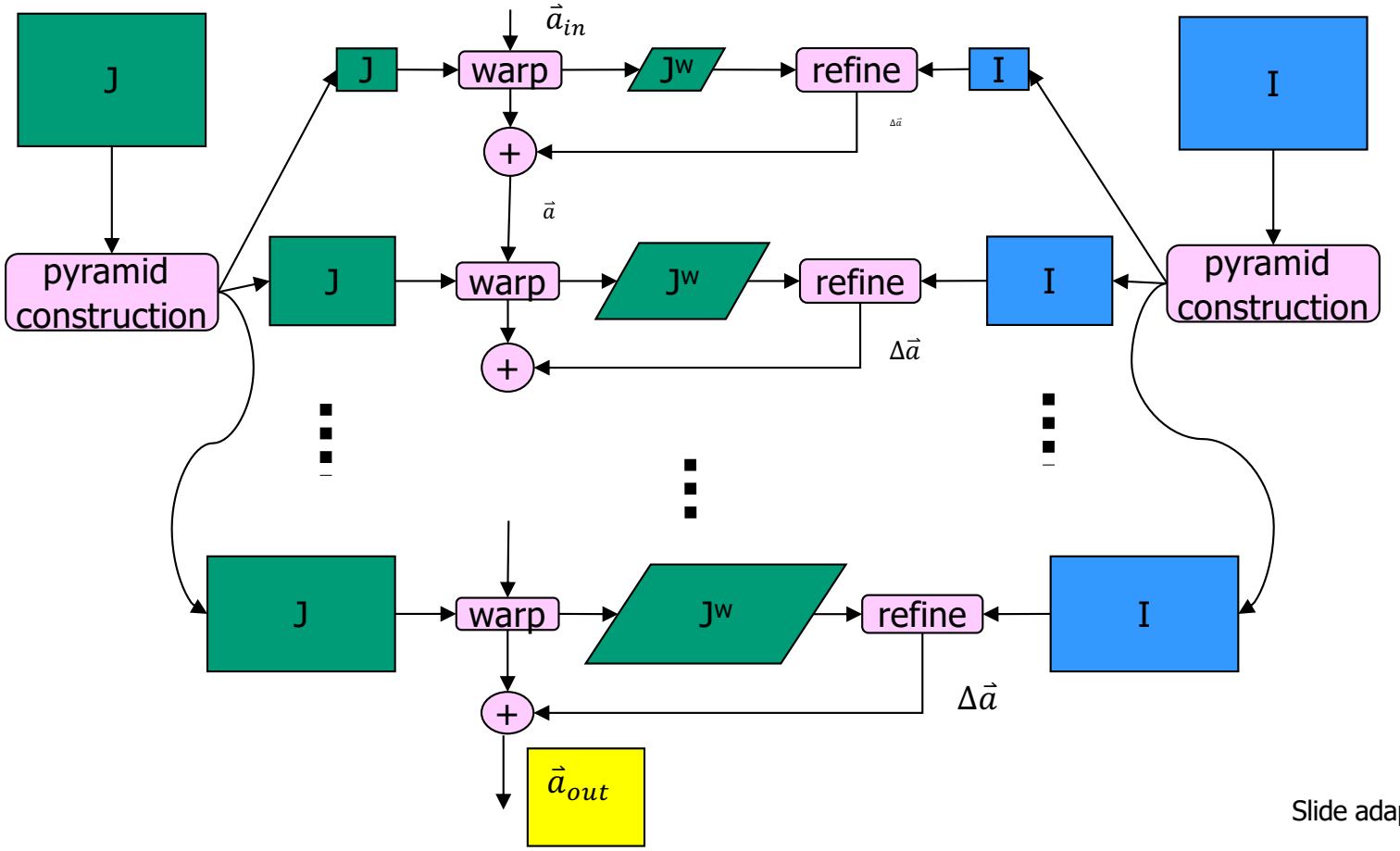
Slide adapted from Szeliski.

Coarse-to-Fine Estimation



Slide adapted from Szeliski.

Coarse-to-Fine Estimation



Slide adapted from Szeliski.

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) though coarse to fine helps
- 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*

Slide adapted from Szeliski.

Summary

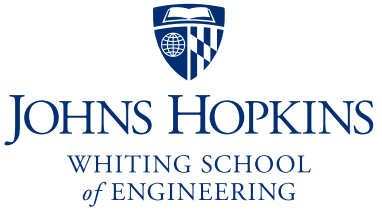
Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

- Essential concepts in this lecture:
  - The brightness constancy constraint
  - Patch-based methods to extract optical flow
  - Coarse to fine estimation

Johns Hopkins Engineering

Computer Vision

Motion and Optical Flow



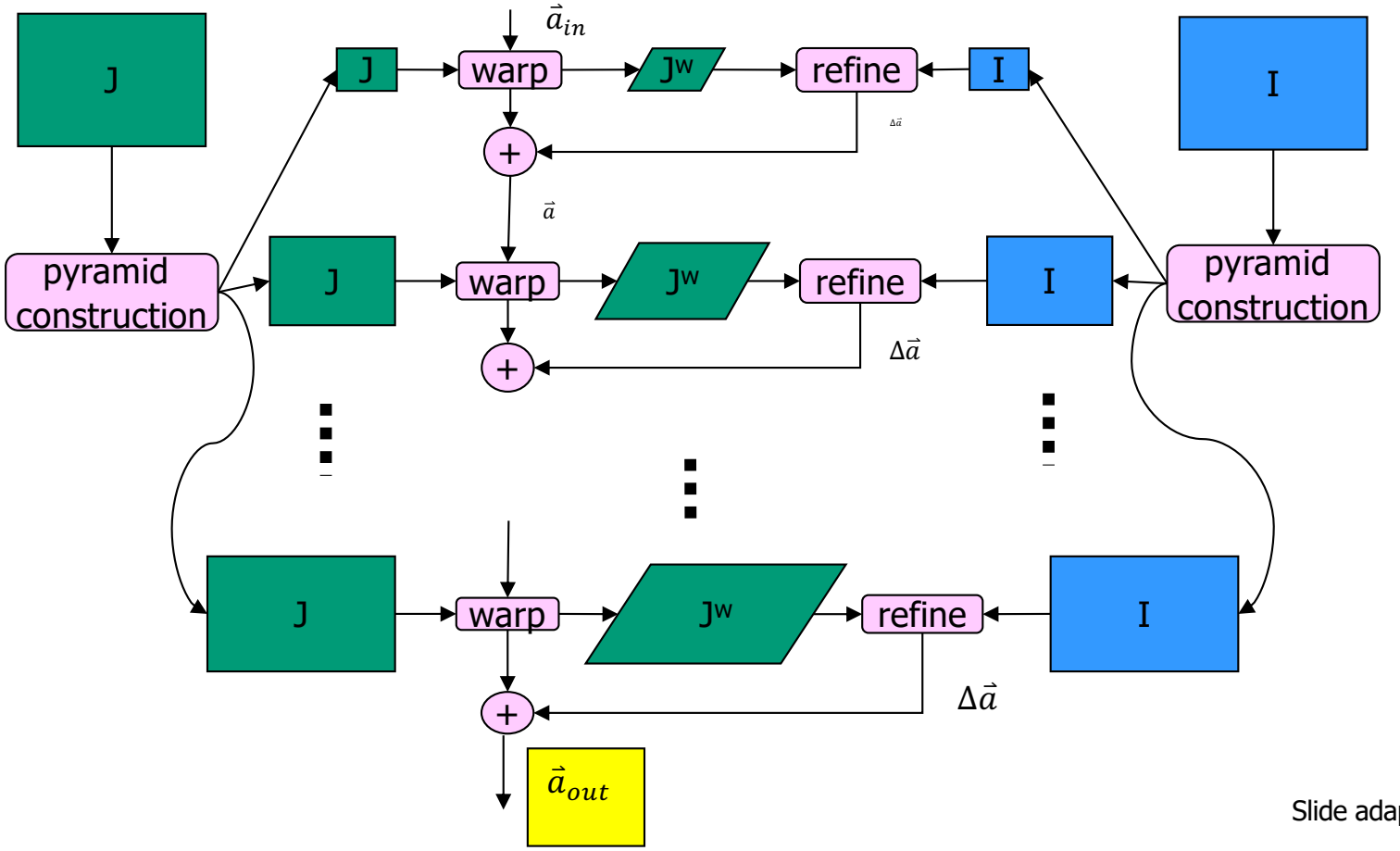
Motion and Optical Flow

Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

Topics:

- Tracking

Coarse-to-Fine Estimation



Slide adapted from Szeliski.

From Optical Flow to Tracking

- To represent an area of an image of size 2h+1, 2w+1 we define

$$R_i(d) = vec\big(I(\boldsymbol{p} + \boldsymbol{d}, t_i)\big), \qquad p = (u, v);\ u \in [-w, w]; v \in [-h, h]$$

this means arrange the pixels as an N=(2h+1)*(2w+1) vector

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image

From Optical Flow to Tracking

- To represent an area of an image of size 2h+1, 2w+1 we define

$$R_i(d) = vec\big(I(\boldsymbol{p} + \boldsymbol{d}, t_i)\big), \qquad p = (u, v);\ u \in [-w, w]; v \in [-h, h]$$

this means arrange the pixels as an N=(2h+1)*(2w+1) vector

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Naïve Tracking:**
  - Define $\Delta\, d_i = d_i - d_{i-1}$ so $d_k = d_0 + \sum_{i=1}^{k} \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing inter-frame difference:

$$O(\Delta\, d_i) = \left|\left|R_i(d_i) - R_{i-1}(d_i)\right|\right|^2$$

From Optical Flow to Tracking

- To represent an area of an image of size 2h+1, 2w+1 we define

$$R_i(d) = vec\big(I(\boldsymbol{p} + \boldsymbol{d}, t_i)\big), \qquad p = (u, v);\ u \in [-w, w]; v \in [-h, h]$$

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Naïve Tracking:**
  - Define $\Delta\, d_i = d_i - d_{i-1}$ so $d_k = d_0 + \sum_{i=1}^{k} \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing inter-frame difference:

Recall

$$\begin{bmatrix} I_x(\mathrm{p}_1) & I_y(\mathrm{p}_1) \\ I_x(\mathrm{p}_2) & I_y(\mathrm{p}_2) \\ \vdots & \vdots \\ I_x(\mathrm{p}_{25}) & I_y(\mathrm{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathrm{p}_1) \\ I_t(\mathrm{p}_2) \\ \vdots \\ I_t(\mathrm{p}_{25}) \end{bmatrix}$$

From Optical Flow to Tracking

- To represent an area of an image of size 2h+1, 2w+1 we define

$$R_i(d) = vec\big(I(\boldsymbol{p} + \boldsymbol{d}, t_i)\big), \qquad p = (u, v);\ u \in [-w, w]; v \in [-h, h]$$

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Naïve Tracking:**
  - Define $\Delta\, d_i = d_i - d_{i-1}$ so $d_k = d_0 + \sum_{i=1}^{k} \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing inter-frame difference:

This approximates the spatial derivative

This approximates the temporal derivative

$$\begin{bmatrix} R_{i,x}(p_1 + d_{i-1}) & R_{i,y}(p_1 + d_{i-1}) \\ R_{i,x}(p_2 + d_{i-1}) & R_{i,y}(p_2 + d_{i-1}) \\ \vdots & \vdots \\ R_{i,x}(p_N + d_{i-1}) & R_{i,y}(p_N + d_{i-1}) \end{bmatrix} \Delta\, d_i = -(R_i(d_{i-1}) - R_{i-1}(d_{i-1}))$$

From Optical Flow to Tracking

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Naïve Tracking:**
  - Define $\Delta\, d_i = d_i - d_{i-1}$ so $d_k = d_0 + \sum_{i=1}^{k} \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing inter-frame difference:

$$\begin{bmatrix} R_{i,x}(p_1 + d_{i-1}) & R_{i,y}(p_1 + d_{i-1}) \\ R_{i,x}(p_2 + d_{i-1}) & R_{i,y}(p_2 + d_{i-1}) \\ \vdots & \vdots \\ R_{i,x}(p_N + d_{i-1}) & R_{i,y}(p_N + d_{i-1}) \end{bmatrix} \Delta\, d_i = -(R_i(d_{i-1}) - R_{i-1}(d_{i-1}))$$

  - Problem: displacements may not be integers
  - Solution: Use sampling with interpolation

From Optical Flow to Tracking

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Naïve Tracking:**
  - Define $\Delta\, d_i = d_i - d_{i-1}$ so $d_k = d_0 + \sum_{i=1}^{k} \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing inter-frame difference:

$$\begin{bmatrix} R_{i,x}(p_1 + d_{i-1}) & R_{i,y}(p_1 + d_{i-1}) \\ R_{i,x}(p_2 + d_{i-1}) & R_{i,y}(p_2 + d_{i-1}) \\ \vdots & \vdots \\ R_{i,x}(p_N + d_{i-1}) & R_{i,y}(p_N + d_{i-1}) \end{bmatrix} \Delta\, d_i = -(R_i(d_{i-1}) - R_{i-1}(d_{i-1}))$$

  - Problem: displacements may be in error so we "slip" off our original target
  - Solution: Reference back to the *original* target

From Optical Flow to Tracking

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Less Naïve Tracking:**
  - Define $d_i = d_{i-1} + \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing *residual difference back to reference*:

$$O(\Delta\, d_i\,) = \left|\left|R_i(d_{i-1} + \Delta\, d_i) - R_0(d_0)\right|\right|^2$$

  - Instead of computing interframe difference, we compute a *correction* of our current estimate of where the target is.
  - Note we could *predict* where we expect the target to be at time i instead of using our last estimate

From Optical Flow to Tracking

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Less Naïve Tracking:**
  - Define $d_i = d_{i-1} + \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing *residual different back to reference*:

$$\begin{bmatrix} R_{0,x}(p_1) & R_{0,y}(p_1) \\ R_{0,x}(p_2) & R_{0,y}(p_2) \\ \vdots & \vdots \\ R_{0,x}(p_N) & R_{0,y}(p_N) \end{bmatrix} \Delta\, d_i = -(R_i(d_{i-1}) - R_0(d_0))$$

This still approximates the temporal derivative

Notice this spatial derivatives of original target which are constant over time

From Optical Flow to Tracking

- Select a target region $\mathrm{R}^* = R_0(d_0)$
- Our goal is to compute a series of locations $d_0, d_1, d_2, \ldots$ image by image
- **Less Naïve Tracking:**
  - Define $d_i = d_{i-1} + \Delta\, d_i$
  - Solve for $\Delta\, d_i$ using optical flow by minimizing *residual different back to reference*:

$$\begin{bmatrix} R_{0,x}(p_1) & R_{0,y}(p_1) \\ R_{0,x}(p_2) & R_{0,y}(p_2) \\ \vdots & \vdots \\ R_{0,x}(p_N) & R_{0,y}(p_N) \end{bmatrix} \Delta\, d_i = -(R_i(d_{i-1}) - R_0(d_0))$$

This still approximates the temporal derivative

  - Problem: now image constancy is a problem
  - Solution: incorporate illumination normalizations (brightness, contrast)

From Translation to More ….

- Define $R_i(\mu) = vec\big(I(\boldsymbol{f}(\boldsymbol{\mu}, \boldsymbol{p}), t_i)\big),\ p = (u, v);\ u \in [-w, w]; v \in [-h, h]$
- Select a target region $\mathrm{R}^* = R_0(\mu_0)$
- Our goal is to compute a series of configurations $\mu_0, \mu_1, \mu_2, \ldots$ image by image

$$f(\mu, p) = A(\mu_{1:4})p + \mu_{5:6}$$

$\mu_0 \quad \ldots \quad \mu_{t_1} \quad \ldots \quad \mu_{t_2} \quad \ldots \quad \mu_{t_3} \quad \ldots$

Hager, Belhumeur, TPAMI, 1998

From Translation to More ….

- Define $R_i(\mu) = vec\big(I(\boldsymbol{f}(\boldsymbol{\mu}, \boldsymbol{p}), t_i)\big),\ p = (u, v);\ u \in [-w, w]; v \in [-h, h]$
- Select a target region $\mathrm{R}^* = R_0(\mu_0)$
- Our goal is to compute a series of configurations $\mu_0, \mu_1, \mu_2, \ldots$ image by image
  - Define $\mu_i = \mu_{i-1} + \Delta\,\mu_i$
  - Solve for $\Delta\,\mu_i$ using optical flow by minimizing *residual different back to reference*:

$$\begin{bmatrix} R_{0,\mu_1}(p_1) & R_{0,\mu_2}(p_1) & \ldots R_{0,\mu_n}(p_1) \\ R_{0,\mu_1}(p_2) & R_{0,\mu_2}(p_1) & \cdots R_{0,\mu_n}(p_1) \\ \vdots & \vdots & \ddots \quad \vdots \\ R_{0,\mu_1}(p_N) & R_{0,\mu_2}(p_1) & \cdots R_{0,\mu_n}(p_1) \end{bmatrix} \Delta\,\mu_i = -(R_i(\mu_{i-1}) - R_0(\mu_0))$$

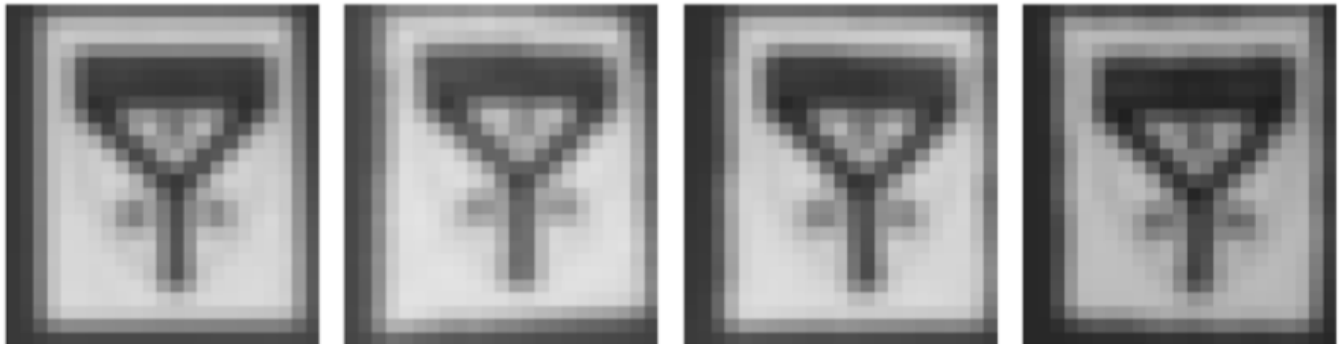
This still approximates the temporal derivative

  - Problem: now image constancy is a problem
  - Solution: incorporate illumination normalizations (brightness, contrast)

From Translation to More ….

- Define $R_i(\mu) = vec\big(I(\boldsymbol{f}(\boldsymbol{\mu}, \boldsymbol{p}), t_i)\big),\ p = (u, v);\ u \in [-w, w]; v \in [-h, h]$
- Select a target region $\mathrm{R}^* = R_0(\mu_0)$
- Our goal is to compute a series of configurations $\mu_0, \mu_1, \mu_2, \ldots$ image by image

$$f(\mu, p) = A(\mu_{1:4})p + \mu_{5:6}$$



Inverse warp of image over time

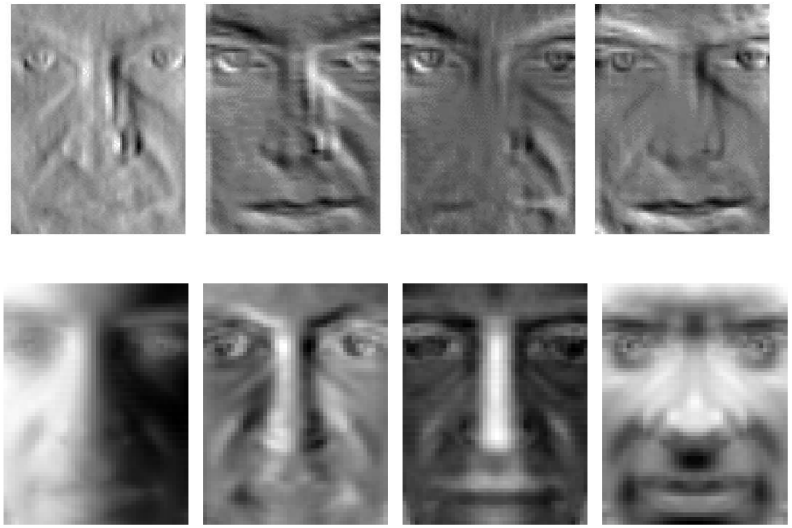
Hager, Belhumeur, TPAMI, 1998

Even More Tracking

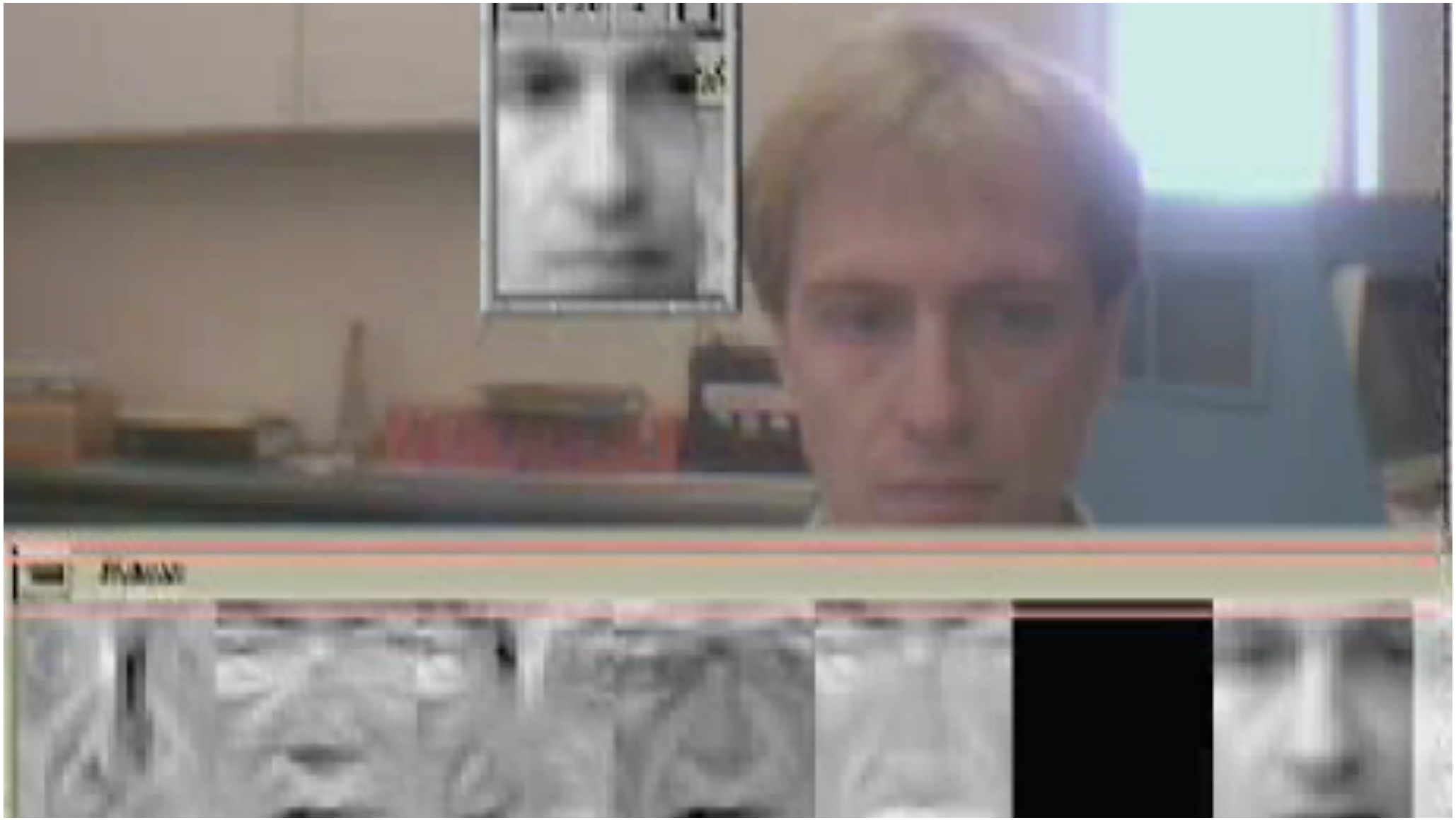
- More generally, we can write both an arbitrary transformation f and illumination basis B

$$R_i(\mu) = vec\big(I(\boldsymbol{f}(\boldsymbol{\mu}, \boldsymbol{p}), t_i)\big) + \lambda B,\ p = (u, v);\ u \in [-w, w]; v \in [-h, h]$$

- We can linearize around μ just as before and jointly solve for both motion and illumination change



Hager, Gregory D., and Peter N. Belhumeur. "Efficient region tracking with parametric models of geometry and illumination." *TPAMI* 10 (1998): 1025-1039.



Summary

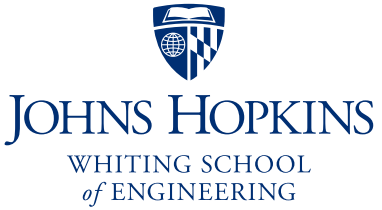
Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

- Essential concepts in this lecture:
  - Tracking as a continuous process of reference back to a base frame
  - Extending translation motion to other motion models
  - Incorporating illumination compensation

Johns Hopkins Engineering

Computer Vision

Motion and Optical Flow



Motion and Optical Flow

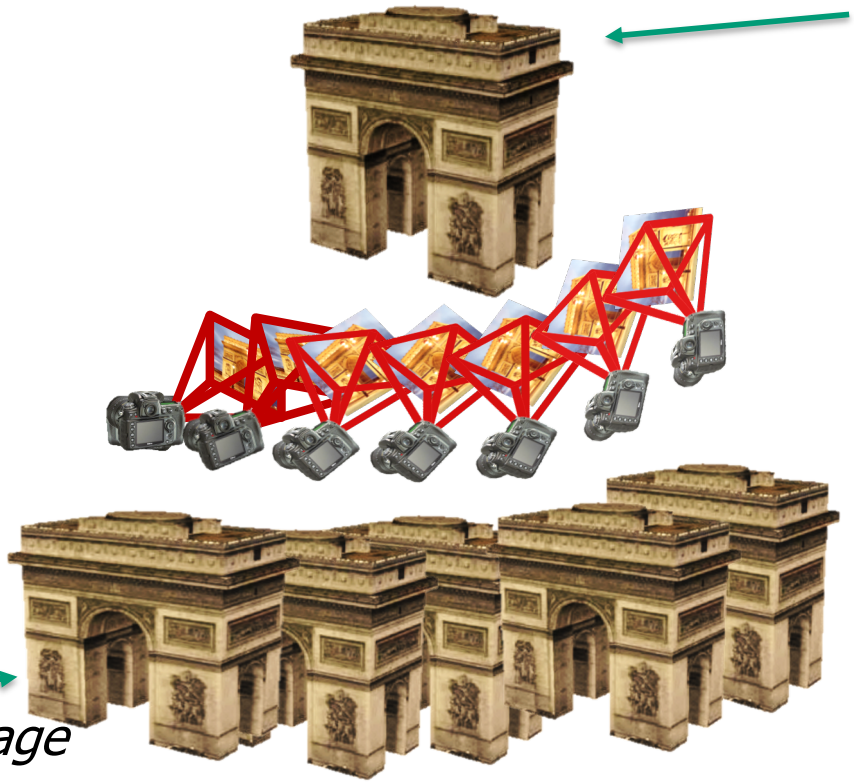
Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

Topics:

- Applications of Motion Estimation

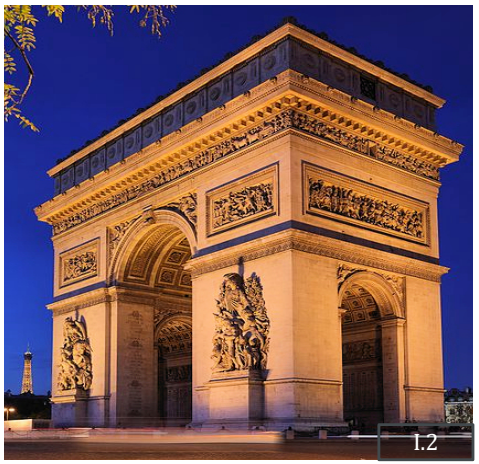
From Two to … Many Images (In Sequence)

Left Image



Structure (3D points P)

Motion (R, t) for each image



Right Image

Structure Estimation

Orthogonal projection

$$p_{i,j} = R_i(P_j - t_i); R_i \quad R_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$ (we'll omit scaling which doesn't matter)

Note by defining $t_i = \sum_j P_{i,j}/n$ we can simplify to

$$p_{i,j} = R_i(P_j - \sum_j \frac{P_{i,j}}{n})$$

Create a 2N x 3 matrix R from the R_i's and a 3 x n shape matrix S from the points P_j

W = R S describes all projections in all images

SVD: W = U D V^t --- define A = U $D^{1/2}$; S = $D^{1/2}$ V^t

Find a matrix Q that orthogonalizes A and compute R* = A Q and S* = Q^{-1} S

Tomasi, Carlo, and Takeo Kanade. "Shape and motion from image streams under orthography: a factorization method." *IJCV* 9, no. 2 (1992): 137-154.

Structure Estimation

Tomasi, Carlo, and Takeo Kanade. "Shape and motion from image streams under orthography: a factorization method." *International journal of computer vision* 9, no. 2 (1992): 137-154.

Structure Estimation Extensions

- Projective structure and motion

  Triggs, Bill. "Factorization methods for projective structure and motion." In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 845-851. IEEE, 1996.

- Multiple moving objects

  Vidal, R. and Ma, Y., 2006. A unified algebraic approach to 2-D and 3-D motion segmentation and estimation. *Journal of Mathematical Imaging and Vision*, *25*(3), pp.403-421.

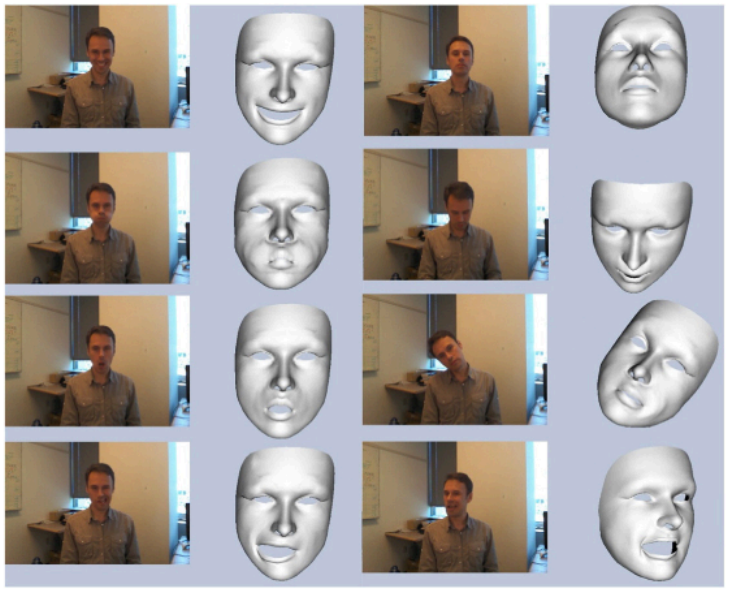
- Omnidirectional motion

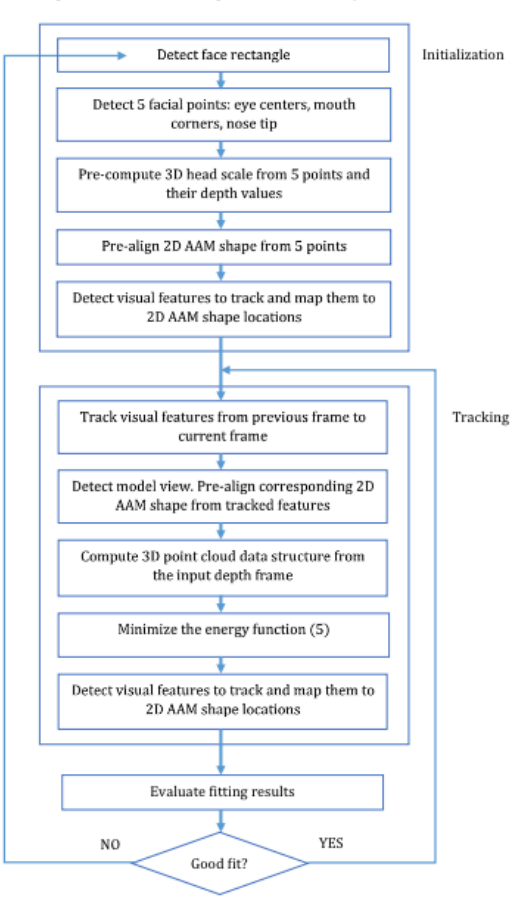
  Corke, P., Strelow, D., & Singh, S. Omnidirectional visual odometry for a planetary rover. In IROS, 2004.

Structure Estimation Extensions

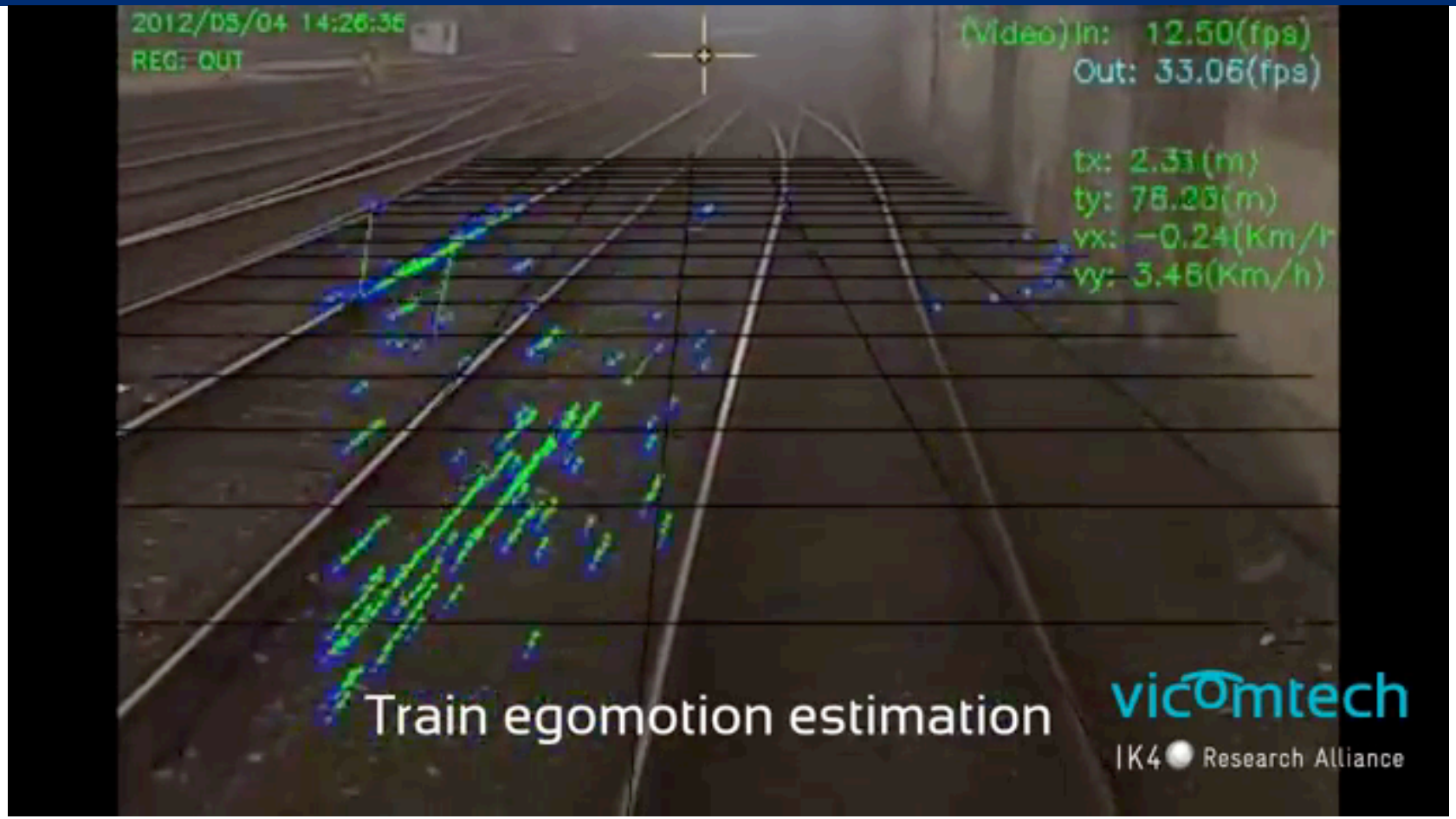
- Nonrigid motions: Active Appearance Models

Smolyanskiy, N., Huitema, C., Liang, L. and Anderson, S.E., 2014. Real-time 3D face tracking based on active appearance model constrained by depth data. *Image and Vision Computing*, *32*(11), pp.860-869.



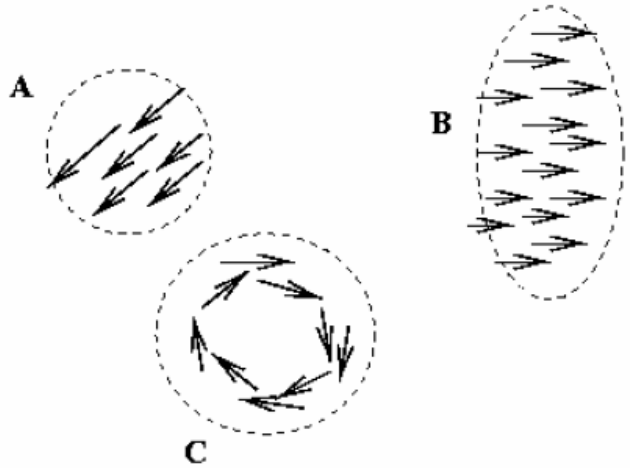


Egomotion Estimation



Applications To Segmentation

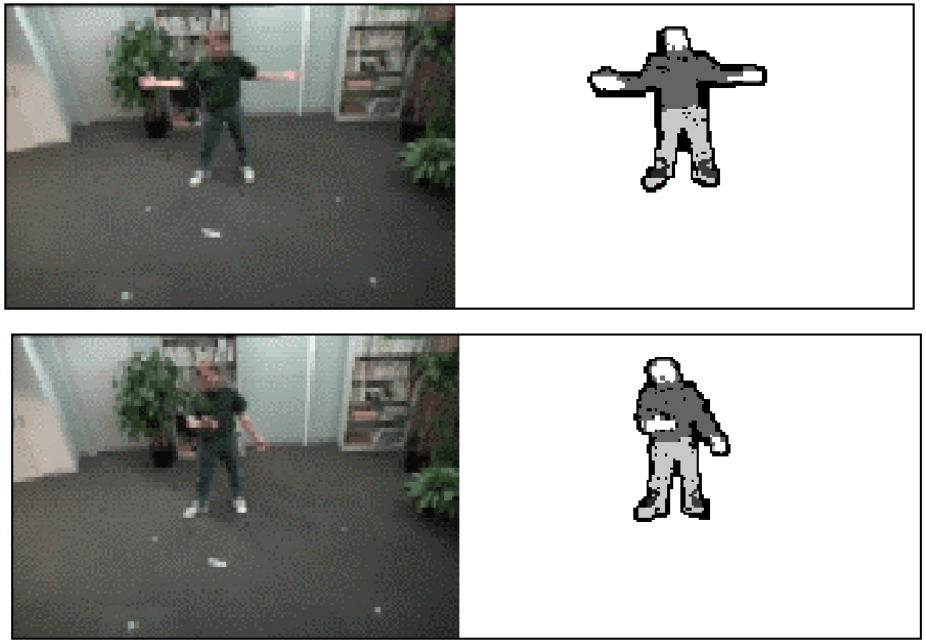
- Background subtraction
- Shot boundary detection
- Motion segmentation
  - Segment the video into multiple *coherently* moving objects



Slide adapted from K. Grauman.

Applications to Segmentation

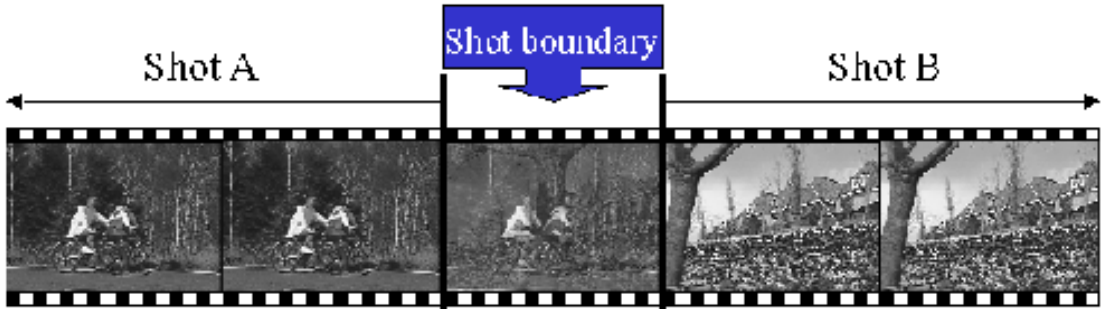
- Background subtraction
  - A static camera is observing a scene
  - Goal: separate the static *background* from the moving *foreground*



Slide adapted from K. Grauman.

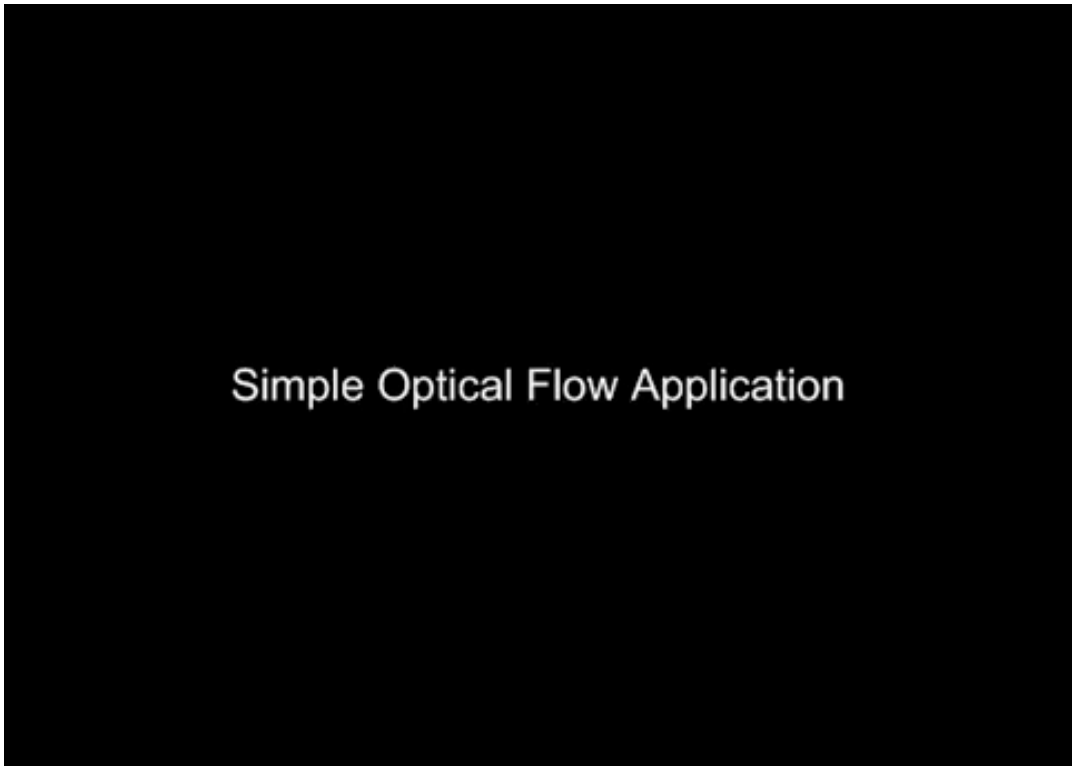
Applications to Segmentation

- 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



Slide adapted from K. Grauman.

Optical Flow for Games!



https://www.youtube.com/watch?v=E7h4OaTtCzY

Motion Paint: an example use of optical flow

- Use optical flow to track brush strokes, in order to animate them to follow underlying scene motion.

http://www.fxguide.com/article333.html

Motion Paint: an example use of optical flow

https://www.youtube.com/watch?v=S5S_ABFcF_4

Summary

Motion Analysis: Extracting information about movement and geometry from a time-series of images sampled with small motion between them.

- Essential concepts in this lecture:
  - Using time-series information to compute structure
  - Directions for extending structure estimation
  - Applications of optical flow

JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING