Optical Flow & Tracking

brightness constancy, motion estimation, image alignment, & tracking

CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Yannis Gkioulekas (CMU 16-385)

TCIG
Toronto Computational Imaging Group

Announcements

- HW 3 due on Nov 20

Course Overview

Overview

- review
  - camera models & calibration
  - two-view geometry
  - structure from motion
  - NeRF
- optical flow
  - brightness constancy
  - Lucas-Kanade (constant flow)
  - Horn-Schunk (smooth flow)
- motion magnification
- image alignment & tracking

The pinhole camera matrix

More compactly, we can write the pinhole camera matrix as:

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}]$$

where

$$\boldsymbol{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \qquad \boldsymbol{t} = -\boldsymbol{R}\widetilde{\boldsymbol{C}} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

2D Euclidean transform

3D rotation

3D translation

intrinsic parameters

extrinsic parameters

Geometric camera calibration

Given a set of matched points

$$\{\mathbf{X}_i, \boldsymbol{x}_i\}$$

point in 3D space — point in the image

and camera model

$$\boldsymbol{x} = \boldsymbol{f}(\mathbf{X}; \boldsymbol{p}) = \mathbf{P}\mathbf{X}$$

projection model — parameters — Camera matrix

Find the (pose) estimate of

$$\mathbf{P}$$

We'll use a **perspective** camera model for pose estimation

How can we make these relations linear?

$$x' = \frac{\boldsymbol{p}_1^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}} \qquad y' = \frac{\boldsymbol{p}_2^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}}$$

Make them linear with algebraic manipulation…

$$\boldsymbol{p}_2^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} y' = 0$$

$$\boldsymbol{p}_1^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} x' = 0$$

Now we can setup a system of linear equations
with multiple point correspondences

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}|-\mathbf{R}\mathbf{c}] \\ &= [\mathbf{M}|-\mathbf{M}\mathbf{c}] \end{aligned}$$

Find the camera center **C**

$$\mathbf{P}\mathbf{c} = \mathbf{0}$$

SVD of P!

***c** is the Eigenvector corresponding to smallest Eigenvalue*

Find intrinsic **K** and rotation **R**

$$\mathbf{M} = \mathbf{K}\mathbf{R}$$

QR decomposition

Triangulation

Given a set of (noisy) matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

and camera matrices

$$\mathbf{P}, \mathbf{P}'$$

Estimate the 3D point

$$\mathbf{X}$$

Using the fact that the cross product should be zero

$$\mathbf{x} \times \mathbf{P}\boldsymbol{X} = \mathbf{0}$$

$$\begin{bmatrix} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \\ x\boldsymbol{p}_2^\top \boldsymbol{X} - y\boldsymbol{p}_1^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Third line is a linear combination of the first and second lines.
(x times the first line plus y times the second line)

One 2D to 3D point correspondence give you 2 equations

Concatenate the 2D points from both images

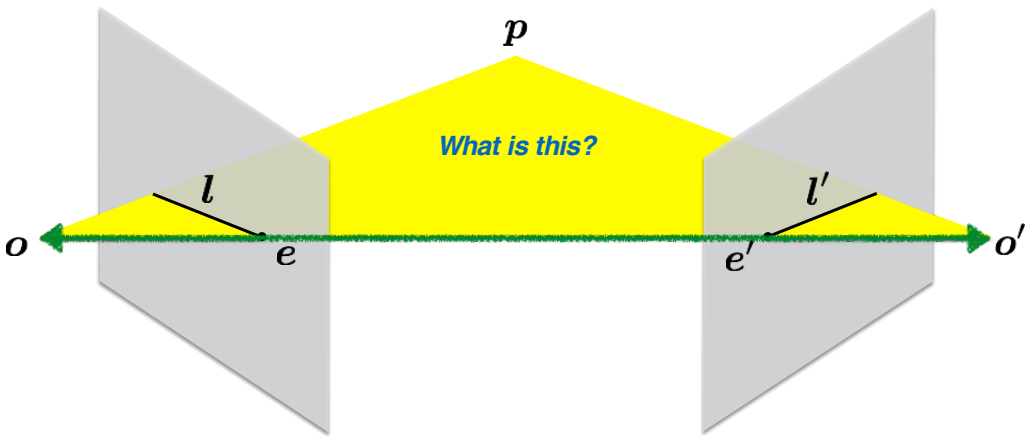
$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{A}\boldsymbol{X} = \mathbf{0}$$

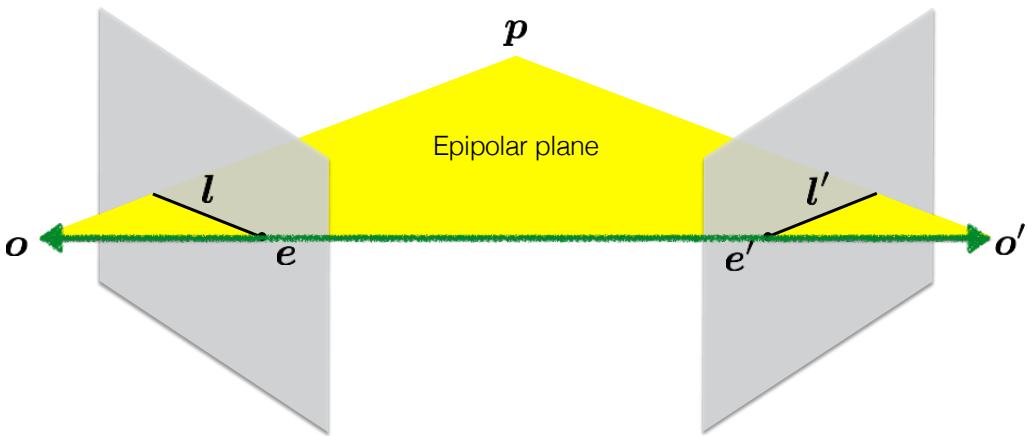
How do we solve homogeneous linear system?

S V D !

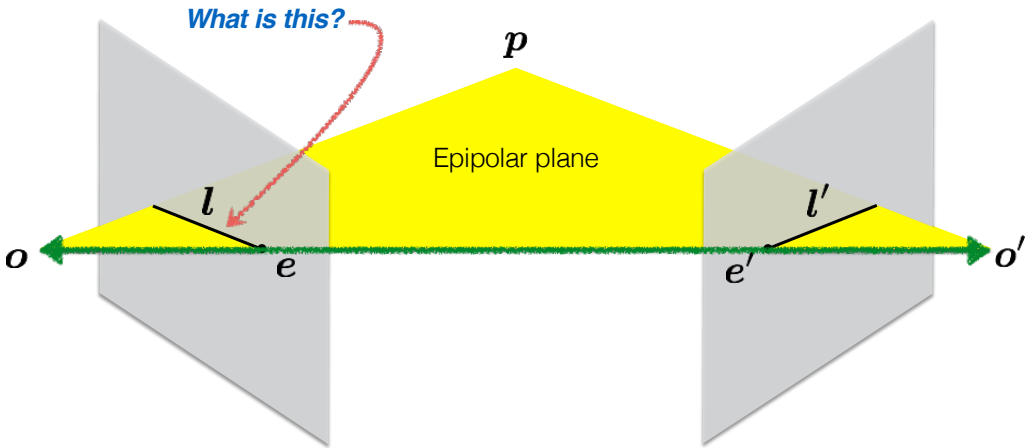
Quiz



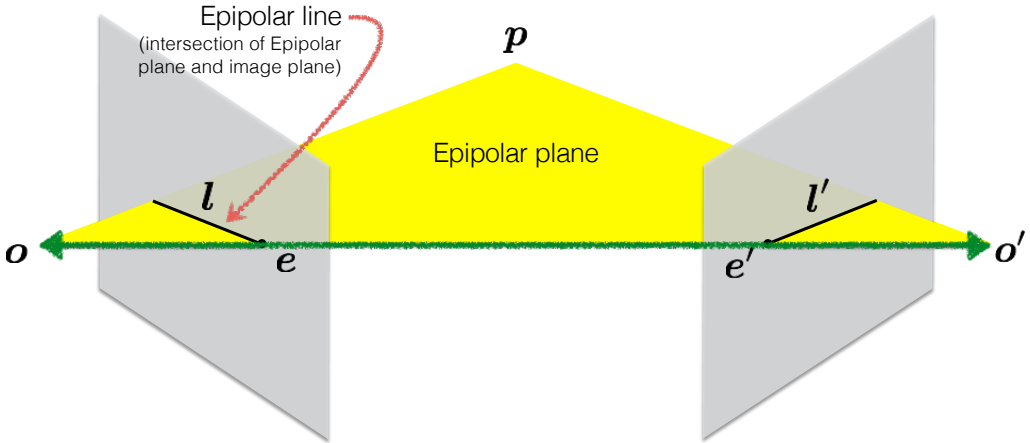
Quiz



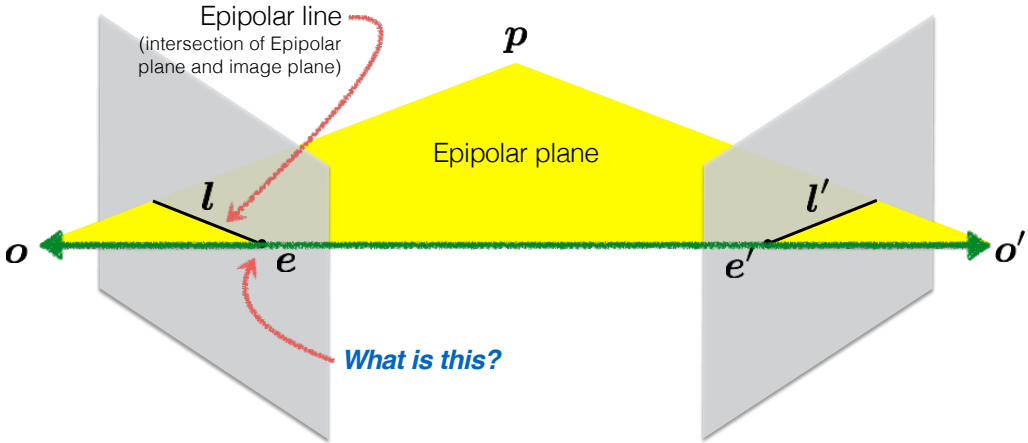
Quiz



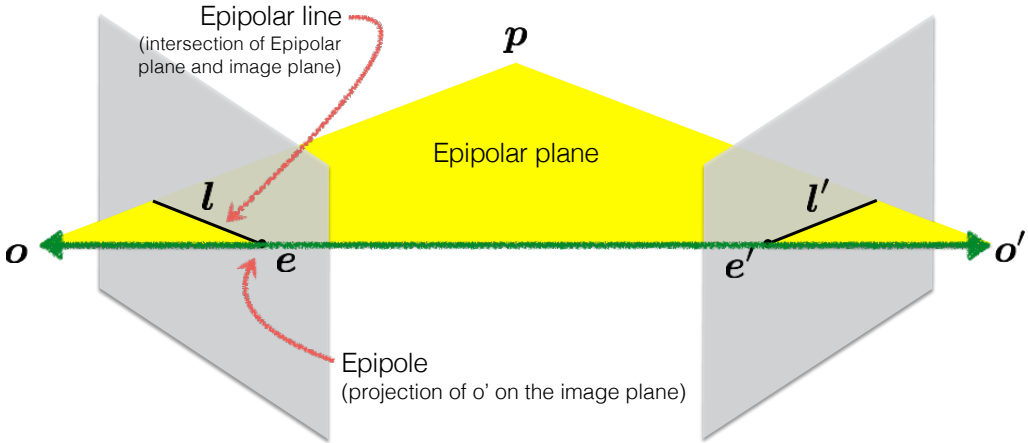
Quiz


Epipolar line
(intersection of Epipolar plane and image plane)
p
Epipolar plane
l
l′
o
e
e′
o′

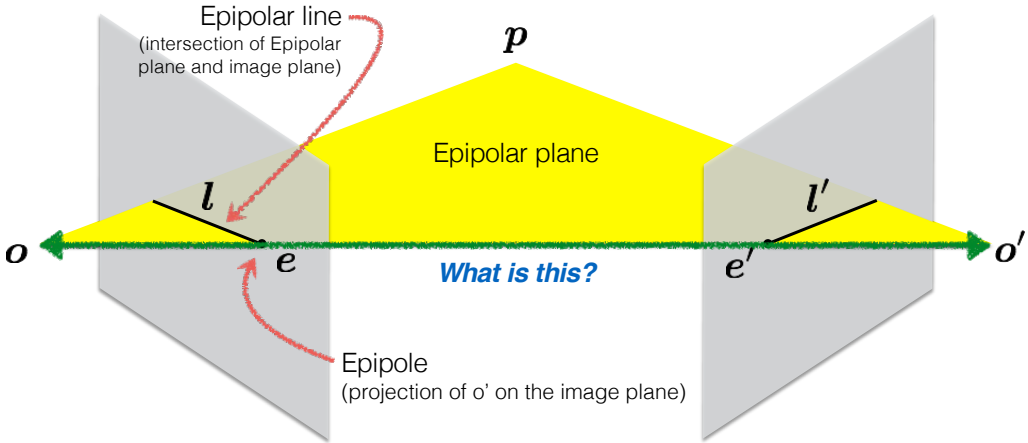
Quiz


Epipolar line
(intersection of Epipolar plane and image plane)
p
Epipolar plane
l
l′
o
e
e′
o′
What is this?

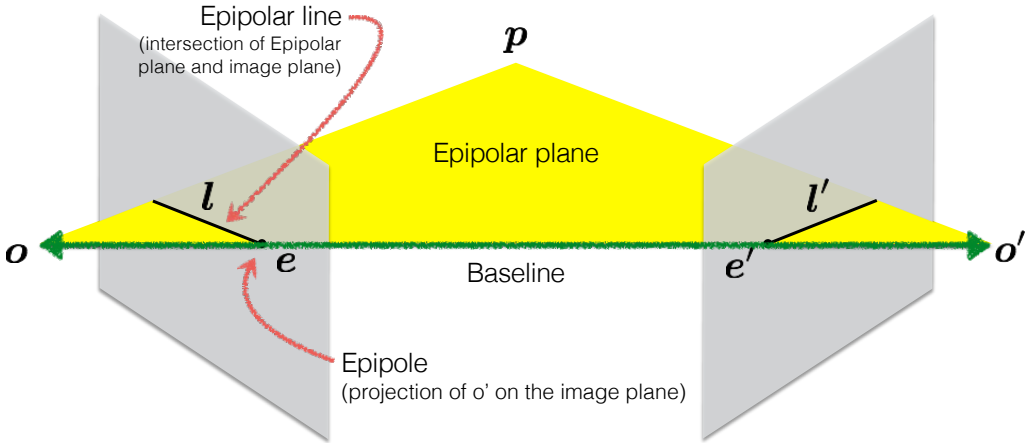
Quiz


Epipolar line
(intersection of Epipolar plane and image plane)
p
Epipolar plane
l
l′
o
e
e′
o′
Epipole
(projection of o' on the image plane)

Quiz

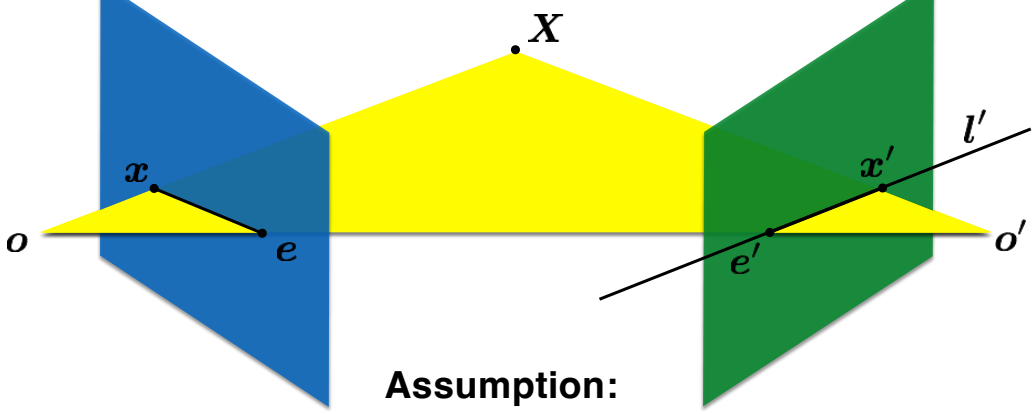

Epipolar line
(intersection of Epipolar plane and image plane)
p
Epipolar plane
l
l′
o
e
What is this?
e′
o′
Epipole
(projection of o' on the image plane)

Quiz


Epipolar line
(intersection of Epipolar plane and image plane)
p
Epipolar plane
l
l′
o
e
Baseline
e′
o′
Epipole
(projection of o' on the image plane)

Given a point in one image,
multiplying by the **essential matrix** will tell us
the **epipolar line** in the second view.

$$\mathbf{E}\boldsymbol{x} = \boldsymbol{l}'$$



Assumption:

2D points expressed in camera coordinate system (i.e., intrinsic matrices are identities)

properties of the E matrix

Longuet-Higgins equation $\boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x}=0$

Epipolar lines $\boldsymbol{x}^{\top}\boldsymbol{l}=0$ $\quad$ $\boldsymbol{x}'^{\top}\boldsymbol{l}'=0$

$\boldsymbol{l}'=\mathbf{E}\boldsymbol{x}$ $\quad$ $\boldsymbol{l}=\mathbf{E}^{T}\boldsymbol{x}'$

Epipoles $\boldsymbol{e}'^{\top}\mathbf{E}=\mathbf{0}$ $\quad$ $\mathbf{E}\boldsymbol{e}=\mathbf{0}$

(2D points expressed in camera coordinate system)

properties of the F/E matrix

Longuet-Higgins equation	$\boldsymbol{x}'^{\top}\mathbf{F}\boldsymbol{x} = 0$	
Epipolar lines	$\boldsymbol{x}^{\top}\boldsymbol{l} = 0$ $\boldsymbol{l}' = \mathbf{F}\boldsymbol{x}$	$\boldsymbol{x}'^{\top}\boldsymbol{l}' = 0$ $\boldsymbol{l} = \mathbf{F}^{T}\boldsymbol{x}'$
Epipoles	$\boldsymbol{e}'^{\top}\mathbf{F} = \mathbf{0}$	$\mathbf{F}\boldsymbol{e} = \mathbf{0}$

(points in **image** coordinates)

Eight-Point Algorithm

Assume you have ***M*** matched *image* points

$$\{\boldsymbol{x}_m, \boldsymbol{x}'_m\} \qquad m = 1, \ldots, M$$

Each correspondence should satisfy

$$\boldsymbol{x}'^{\top}_m \mathbf{F} \boldsymbol{x}_m = 0$$

*How would you solve for the 3 x 3 **F** matrix?*

Eight-Point Algorithm

0. (Normalize points)
1. Construct the M x 9 matrix **A**
2. Find the SVD of **A**
3. Entries of **F** are the elements of column of **V** corresponding to the least singular value
4. (Enforce rank 2 constraint on F)
5. (Un-normalize F)

Stereo Rectification

1. **Rotate** the right camera by **R**
   (aligns camera coordinate system orientation only)

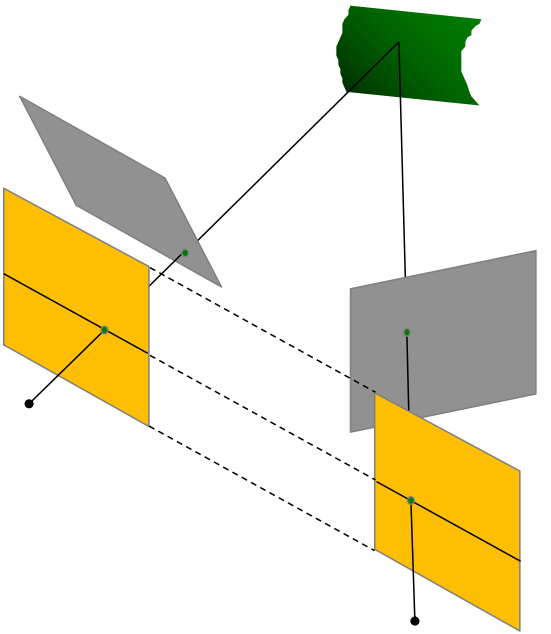
2. Rotate (**rectify**) the left camera so that the epipole is at infinity

3. Rotate (**rectify**) the right camera so that the epipole is at infinity

4. Adjust the **scale**

What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers

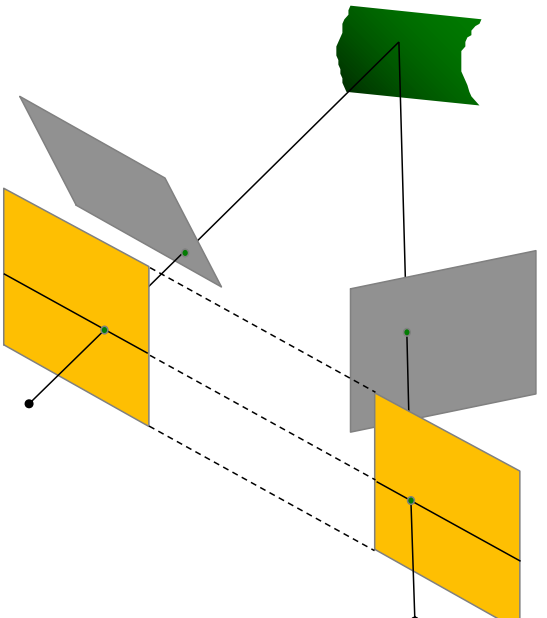


How can you do this?

What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers

Need two homographies (3x3 transform), one for each input image reprojection



C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision.Computer Vision and Pattern Recognition, 1999.

Reconstruction

(2 view structure from motion)

Given a set of matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

Estimate the camera matrices

$$\mathbf{P}, \mathbf{P}'$$ ← ‘motion’ (of the cameras)

Estimate the 3D point

$$\mathbf{X}$$ ← ‘structure’

Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\, \mathbf{I} \mid \mathbf{0} \,]$ and $\mathbf{P'} = [\, [\mathbf{e'}_\times]\mathbf{F} \mid \mathbf{e'} \,]$

3. For each point correspondence, compute the point **X** in 3D space (triangulation)
   Triangulation with $x = \mathbf{P}\,X$ and $x' = \mathbf{P'}\,X$

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \ldots, m, \; j = 1, \ldots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix}$$

- We have 2*mn* knowns — *n* measured 2D point locations in each of the *m* images
- 8*m* + 3*n* - 12 unknowns — 8 affine transformations for each of the m images and *n* 3D points minus 12 dof for affine ambiguity

Algorithm summary

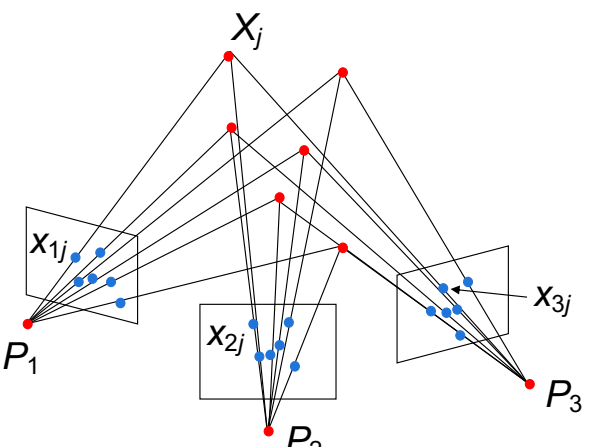
- Given: *m* images and *n* features $\mathbf{x}_{ij}$
- For each image *i, c*enter the feature coordinates
- Construct a $2m \times n$ measurement matrix **D**:
  - Column *j* contains the projection of point *j* in all views
  - Row *i* contains one coordinate of the projections of all the *n* points in image *i*
- Factorize **D**:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^{\mathbf{T}}$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of **U**
  - Create $\mathbf{V}_3$ by taking the first 3 columns of **V**
  - Create $\mathbf{W}_3$ by taking the upper left 3×3 block of **W**
- Create the motion and shape matrices:
  - $\mathbf{M} = \mathbf{U}_3\mathbf{W}_3^{1/2}$ and $\mathbf{S} = \mathbf{W}_3^{1/2}\,\mathbf{V}_3^{T}$ (**or** $\mathbf{M} = \mathbf{U}_3$ and $\mathbf{S} = \mathbf{W}_3\mathbf{V}_3^{T}$)
- Eliminate affine ambiguity

Projective structure from motion

- Given: *m* images of *n* fixed 3D points

$$z_{ij}\,\mathbf{x}_{ij} = \mathbf{P}_i\,\mathbf{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* correspondences $\mathbf{x}_{ij}$



Sequential structure from motion

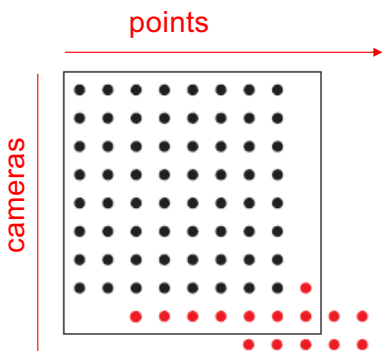
•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
- Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*

•Refine structure and motion: bundle adjustment



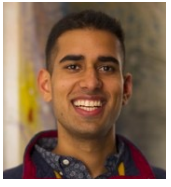
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020

Ben Mildenhall*

UC Berkeley

Pratul Srinivasan*



UC Berkeley

Google

Matt Tancik*

UC Berkeley

Jon Barron

Google Research

Google

Ravi Ramamoorthi

UC San Diego

UC San Diego

Ren Ng

UC Berkeley

Volume rendering estimation: integrating color along a ray

Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

final rendered color along ray

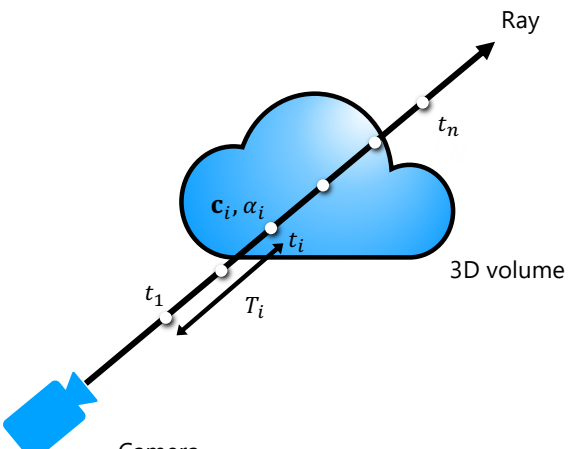
weights

colors

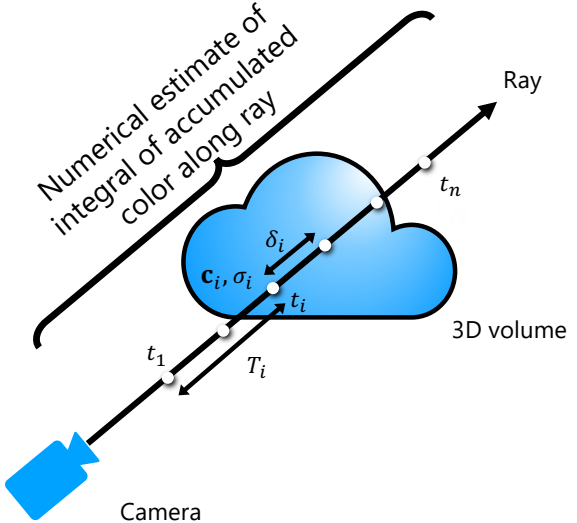
How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

Computing the color for a set of rays through the pixels of an image yields a rendered image



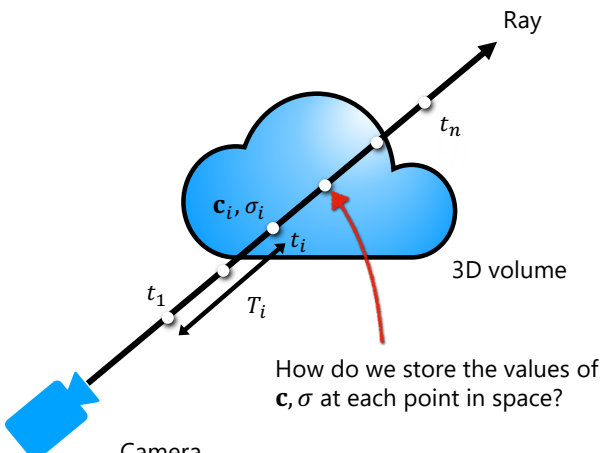
Volume rendering estimation: integrating color along a ray



$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$

Slight modification: α is not directly stored in the volume, but instead is derived from a stored volume density sigma (σ) that is multiplied by the distance between samples delta (δ):

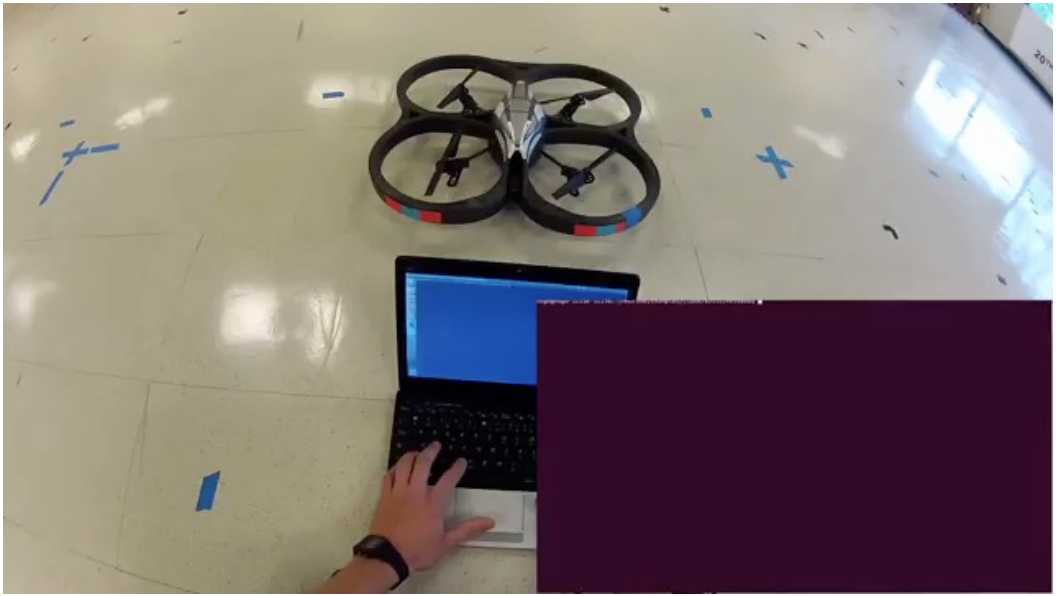
Volume rendering estimation: integrating color along a ray



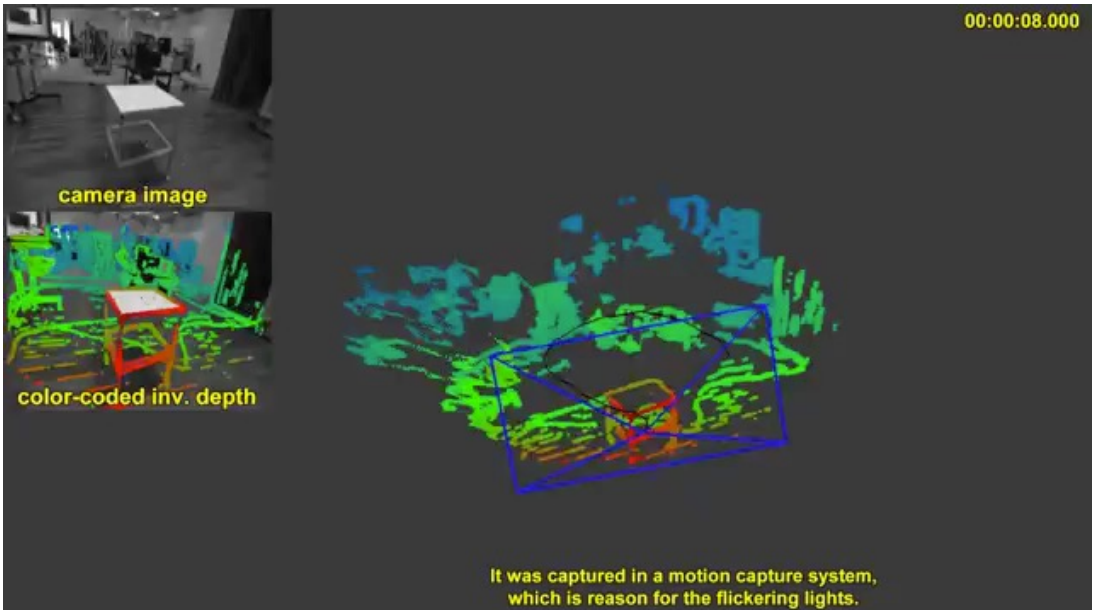
Computing the color for a set of rays through the pixels of an image yields a rendered image

Optical flow

Optical flow used for feature tracking on a drone



optical flow used for motion estimation in visual odometry



Optical Flow

Problem Definition

Given two consecutive image frames,
estimate the motion of each pixel

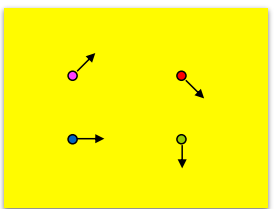
Assumptions

Brightness constancy

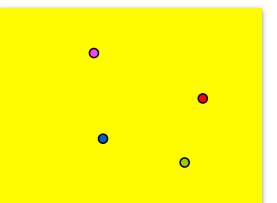
Small motion

Optical Flow

(Problem definition)



$I(x, y, t)$



$I(x, y, t')$

Estimate the motion
(flow) between these
two consecutive images

How is this different from estimating a 2D transform?

Key Assumptions

(unique to optical flow)

Color Constancy

(Brightness constancy for intensity images)

Implication: allows for pixel to pixel comparison
(not image features)

Key Assumptions

(unique to optical flow)

Color Constancy

(Brightness constancy for intensity images)

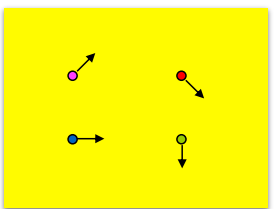
Implication: allows for pixel to pixel comparison
(not image features)

Small Motion

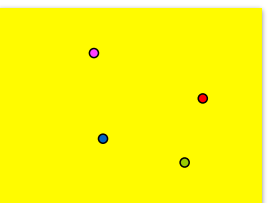
(pixels only move a little bit)

Implication: linearization of the brightness
constancy constraint

Approach



$I(x, y, t)$



$I(x, y, t')$

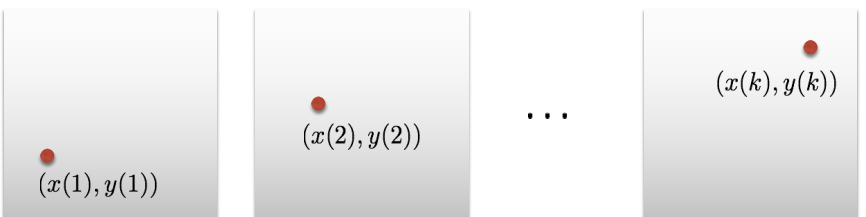
Look for nearby pixels with the same color

(small motion) (color constancy)

Assumption 1

Brightness constancy

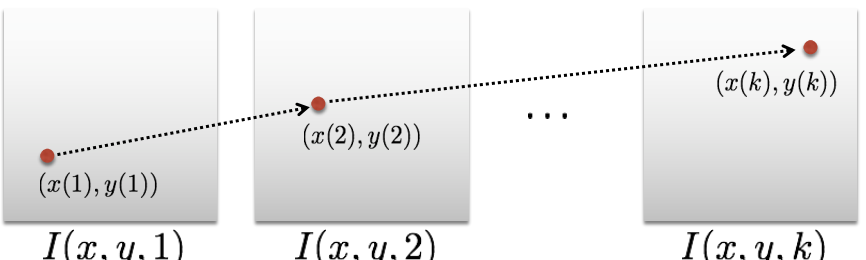
Scene point moving through image sequence



Assumption 1

Brightness constancy

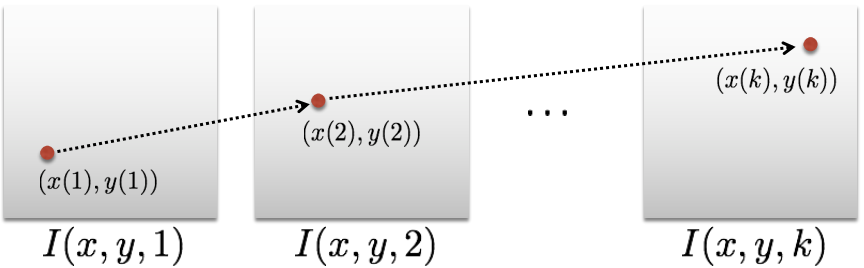
Scene point moving through image sequence



Assumption 1

Brightness constancy

Scene point moving through image sequence

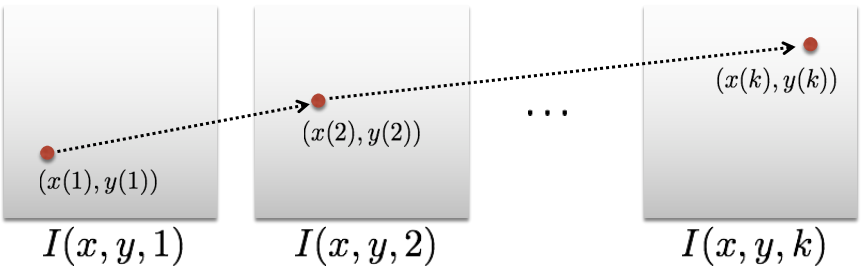


Assumption:Brightness of the point will remain the same

Assumption 1

Brightness constancy

Scene point moving through image sequence



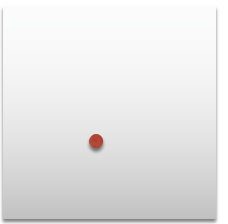
Assumption:Brightness of the point will remain the same

$$I(x(t), y(t), t) = C$$

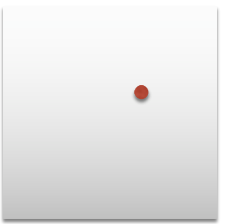
constant

Assumption 2

Small motion



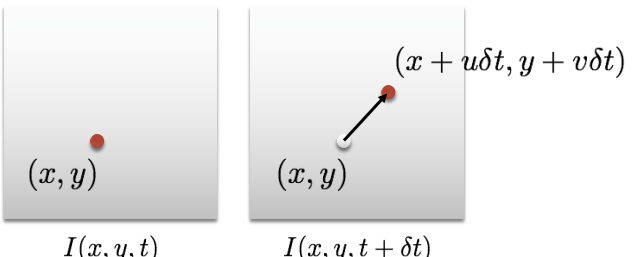
$I(x, y, t)$



$I(x, y, t + \delta t)$

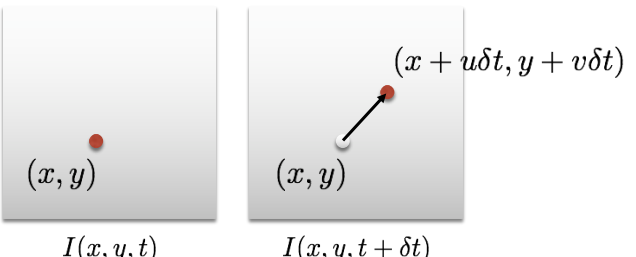
Assumption 2

Small motion



Assumption 2

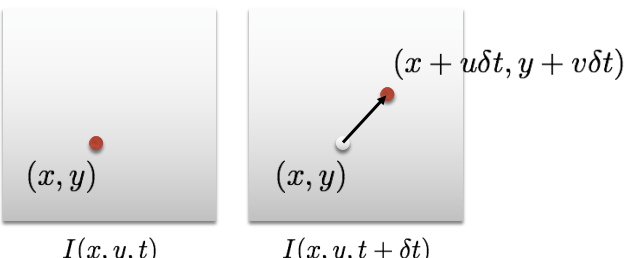
Small motion



Optical flow (velocities): (u, v) Displacement: $(\delta x, \delta y) = (u\delta t, v\delta t)$

Assumption 2

Small motion



Optical flow (velocities): (u, v) Displacement: $(\delta x, \delta y) = (u\delta t, v\delta t)$

For a ***really small space-time step***…

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

… the brightness between two consecutive image frames is the same

These assumptions yield the …

Brightness Constancy Equation

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

total derivative

partial derivative

Equation is not obvious. Where does this come from?

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

For small space-time step, brightness of a point is the same

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

For small space-time step, brightness of a point is the same

Insight:
If the time step is really small,
we can ***linearize*** the intensity function

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Multivariable Taylor Series Expansion

(First order approximation, two variables)

$$f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b)$$

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Multivariable Taylor Series Expansion

(First order approximation, two variables)

$$f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b)$$

$$I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = I(x, y, t)$$

assuming small motion

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Multivariable Taylor Series Expansion

(First order approximation, two variables)

$$f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b)$$

partial derivative

$$\underset{\text{fixed point}}{I(x, y, t)} + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = I(x, y, t)$$

assuming small motion

cancel terms

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Multivariable Taylor Series Expansion

(First order approximation, two variables)

$$f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b)$$

$$I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = I(x, y, t)$$ assuming small motion

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$ cancel terms

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Multivariable Taylor Series Expansion

(First order approximation, two variables)

$$f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b)$$

$$I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = I(x, y, t)$$

assuming small motion

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

divide by δt

take limit $\delta t \to 0$

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Multivariable Taylor Series Expansion

(First order approximation, two variables)

$$f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b)$$

$$I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = I(x, y, t)$$

assuming small motion

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

divide by δt

take limit $\delta t \to 0$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Multivariable Taylor Series Expansion

(First order approximation, two variables)

$$f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b)$$

$$I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = I(x, y, t)$$

assuming small motion

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

divide by δt

take limit $\delta t \to 0$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Brightness Constancy Equation

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Brightness Constancy Equation

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Brightness Constancy Equation

$$\underset{\text{(x-flow)}}{I_x u} + \underset{\text{(y-flow)}}{I_y v} + I_t = 0$$

shorthand notation

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Brightness Constancy Equation

$$\underset{\text{(x-flow)}}{I_x}u + \underset{\text{(y-flow)}}{I_y}v + I_t = 0$$

shorthand notation

$$\underset{(1 \times 2)}{\nabla I^\top} \underset{(2 \times 1)}{\boldsymbol{v}} + I_t = 0$$

vector form

(putting the math aside for a second…)

What do the terms of the
brightness constancy equation represent?

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

(putting the math aside for a second…)

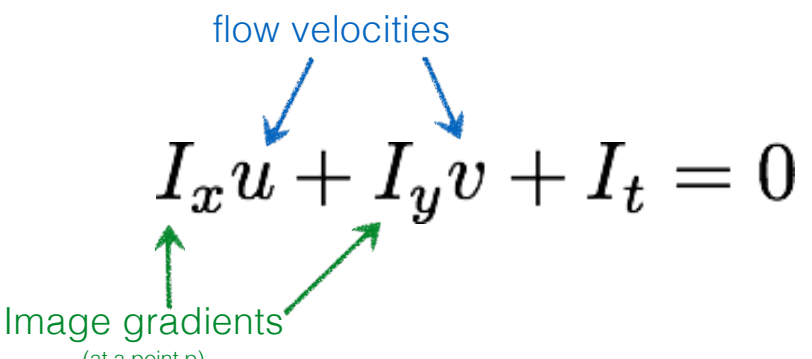
What do the term of the
brightness constancy equation represent?

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

Image gradients
(at a point p)

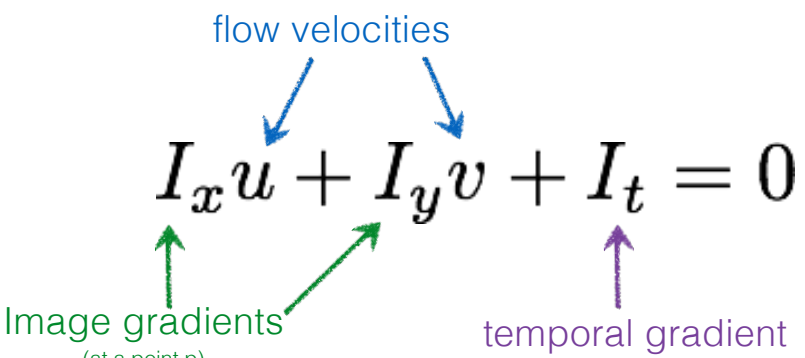
(putting the math aside for a second…)

What do the term of the
brightness constancy equation represent?



(putting the math aside for a second…)

What do the term of the
brightness constancy equation represent?



How do you compute these terms?

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

How do you compute …

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative

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

How do you compute …

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative

Forward difference
Sobel filter
Derivative-of-Gaussian filter
…

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

How do you compute …

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative

Forward difference
Sobel filter
Derivative-of-Gaussian filter
…

$$I_t = \frac{\partial I}{\partial t}$$

temporal derivative

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

How do you compute …

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative

Forward difference
Sobel filter
Derivative-of-Gaussian filter
…

$$I_t = \frac{\partial I}{\partial t}$$

temporal derivative

frame differencing

Frame differencing

t

1	1	1	1	1
1	1	1	1	1
1	10	10	10	10
1	10	10	10	10
1	10	10	10	10
1	10	10	10	10

-

$t+1$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	10	10	10
1	1	10	10	10
1	1	10	10	10

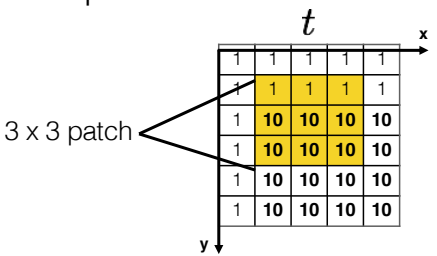
=

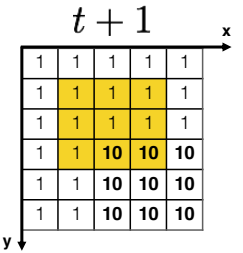
$$I_t = \frac{\partial I}{\partial t}$$

0	0	0	0	0
0	0	0	0	0
0	9	9	9	9
0	9	0	0	0
0	9	0	0	0
0	9	0	0	0

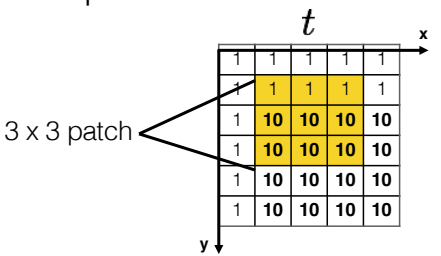
(example of a forward difference)

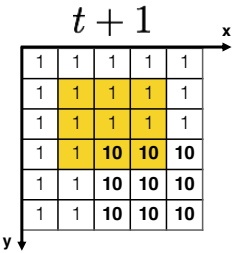
Example:


t
x
y
3 x 3 patch
1 1 1 1 1
1 1 1 1 1
1 10 10 10 10
1 10 10 10 10
1 10 10 10 10
1 10 10 10 10

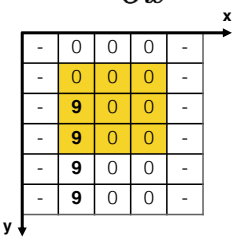

$t+1$
x
y
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 10 10 10
1 1 10 10 10
1 1 10 10 10

Example:



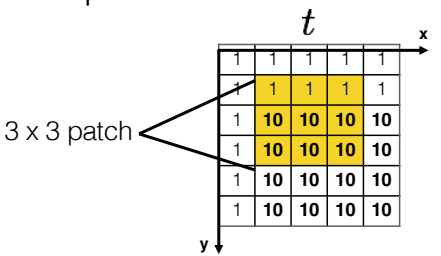


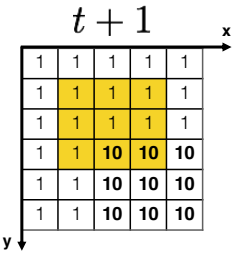
$$I_x = \frac{\partial I}{\partial x}$$



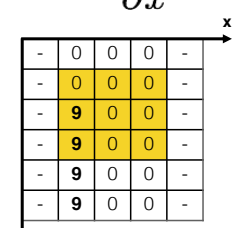
-1 0 1

Example:



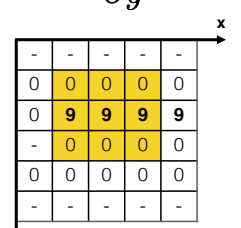


$$I_x = \frac{\partial I}{\partial x}$$



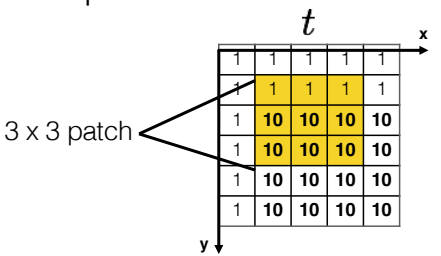
-1 0 1

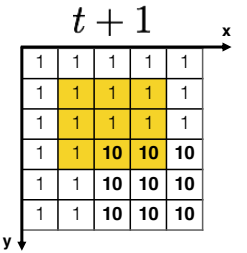
$$I_y = \frac{\partial I}{\partial y}$$



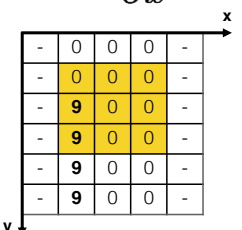
-1
0
1

Example:



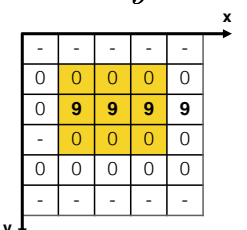


$$I_x = \frac{\partial I}{\partial x}$$



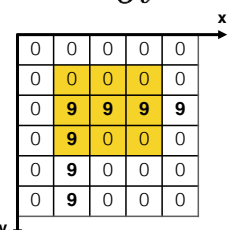
-1 0 1

$$I_y = \frac{\partial I}{\partial y}$$



-1
0
1

$$I_t = \frac{\partial I}{\partial t}$$



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

How do you compute …

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative

Forward difference
Sobel filter
Derivative-of-Gaussian filter
…

$$u = \frac{dx}{dt} \quad v = \frac{dy}{dt}$$

optical flow

How do you compute this?

$$I_t = \frac{\partial I}{\partial t}$$

temporal derivative

frame differencing

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

How do you compute …

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative

Forward difference
Sobel filter
Derivative-of-Gaussian filter
…

$$u = \frac{dx}{dt} \quad v = \frac{dy}{dt}$$

optical flow

We need to solve for this!
(this is the unknown in the optical flow problem)

$$I_t = \frac{\partial I}{\partial t}$$

temporal derivative

frame differencing

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

How do you compute …

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative

Forward difference
Sobel filter
Derivative-of-Gaussian filter
…

$$u = \frac{dx}{dt} \quad v = \frac{dy}{dt}$$

optical flow

(u, v)
Solution lies on a line

Cannot be found uniquely
with a single constraint

$$I_t = \frac{\partial I}{\partial t}$$

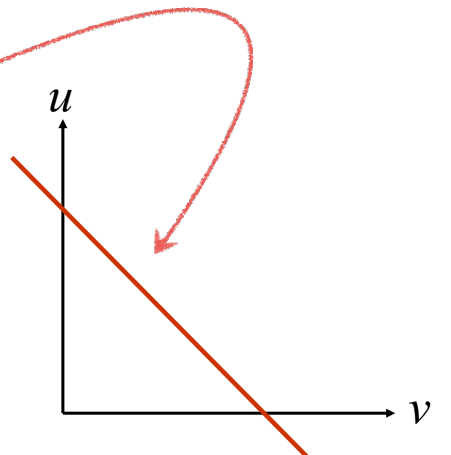
temporal derivative

frame differencing

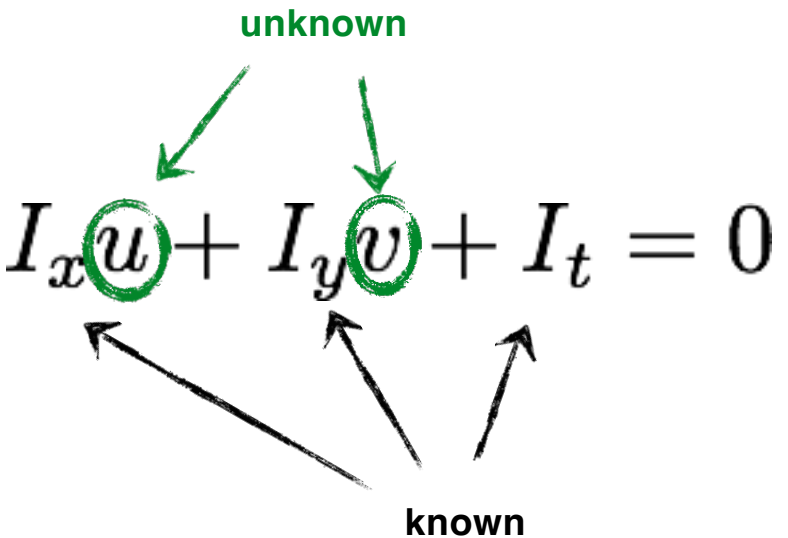
Solution lies on a straight line

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

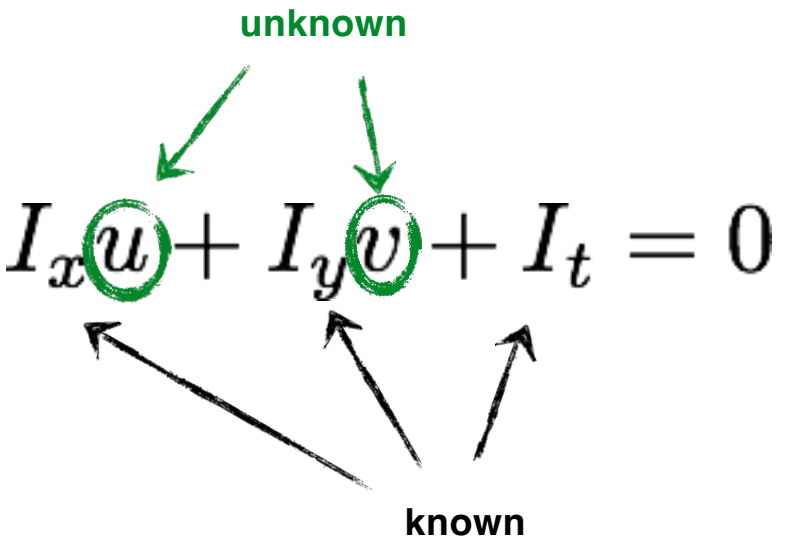
many combinations of u and v will satisfy the equality



The solution cannot be determined uniquely with a single constraint (a single pixel)



We need at least ____ equations to solve for 2 unknowns.



Where do we get more equations (constraints)?

Horn-Schunck
Optical Flow (1981)

brightness constancy

small motion

‘smooth’ flow
(flow can vary from pixel to pixel)

global method
(dense)

Lucas-Kanade
Optical Flow (1981)

method of differences

‘constant’ flow
(flow is constant for all pixels)

local method
(sparse)

Constant flow

Where do we get more equations (constraints)?

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

Assume that the surrounding patch (say 5x5) has
‘**constant flow**’

Assumptions:

Flow is locally smooth

Neighboring pixels have same displacement

Using a 5 x 5 image patch, gives us  equations

Assumptions:

Flow is locally smooth

Neighboring pixels have same displacement

Using a 5 x 5 image patch, gives us 25 equations

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

Equivalent to solving:

$$\underbrace{\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}}_{A^\top A} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\hat{x}} = \underbrace{-\begin{bmatrix} \sum_{p \in P} I_x I_t \\ \sum_{p \in P} I_y I_t \end{bmatrix}}_{A^\top b}$$

where the summation is over each pixel ***p*** in patch ***P***

$$x = (A^\top A)^{-1} A^\top b$$

Equivalent to solving:

$$\underbrace{A^\top A}_{\left[\begin{array}{cc} \sum_{p\in P} I_x I_x & \sum_{p\in P} I_x I_y \\ \sum_{p\in P} I_y I_x & \sum_{p\in P} I_y I_y \end{array}\right]} \underbrace{\hat{x}}_{\left[\begin{array}{c} u \\ v \end{array}\right]} = - \underbrace{A^\top b}_{\left[\begin{array}{c} \sum_{p\in P} I_x I_t \\ \sum_{p\in P} I_y I_t \end{array}\right]}$$

where the summation is over each pixel ***p*** in patch ***P***

Sometimes called 'Lucas-Kanade Optical Flow'
(can be interpreted to be a special case of the LK method with a translational warp model)

When is this solvable?

$$A^\top A\hat{x} = A^\top b$$

When is this solvable?

$$A^\top A\hat{x} = A^\top b$$

$A^\top A$ should be invertible

$A^\top A$ should not be too small

λ_1 and λ_2 should not be too small

$A^\top A$ should be well conditioned

λ_1/λ_2 should not be too large (λ_1=larger eigenvalue)

Where have you seen this before?

$$A^\top A = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

Where have you seen this before?

$$A^\top A = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

Harris Corner Detector!

Where have you seen this before?

$$A^\top A = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

Harris Corner Detector!

What are the implications?

Implications

- Corners are when λ1, λ2 are big; this is also when Lucas-Kanade optical flow works best
- Corners are regions with two different directions of gradient (at least)
- Corners are good places to compute flow!

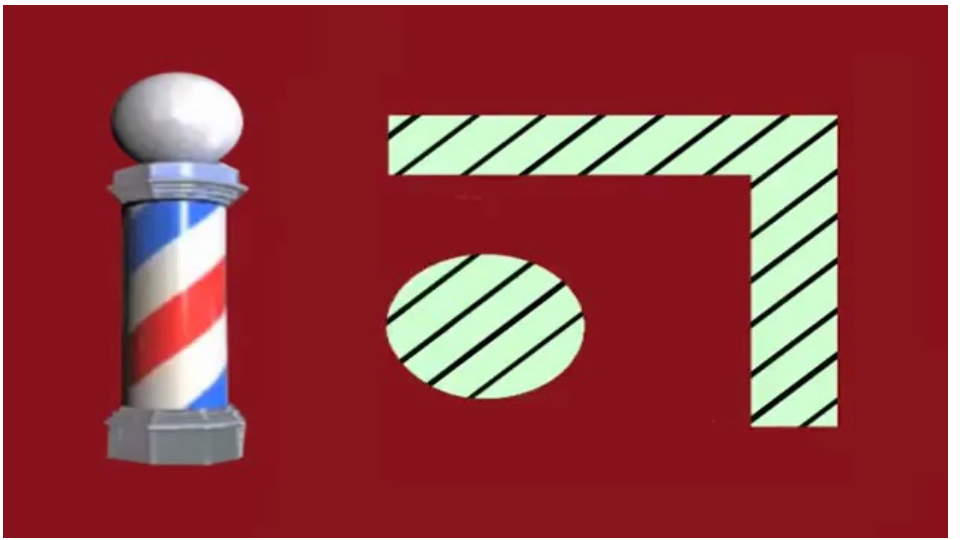
What happens when you have no ‘corners’?

You want to compute optical flow.
What happens if the image patch contains only a line?

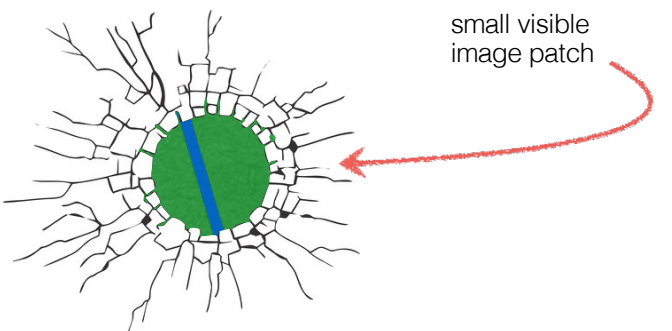
Barber’s pole illusion

Barber’s pole illusion

Barber’s pole illusion

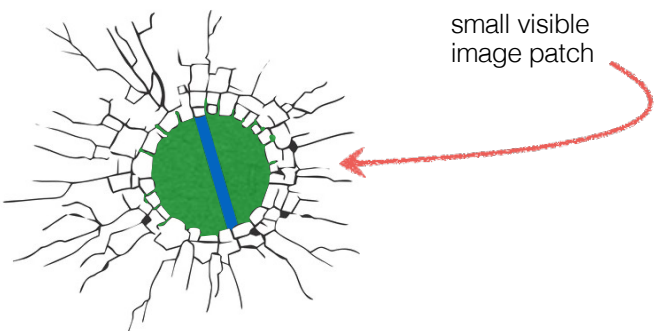


Aperture Problem



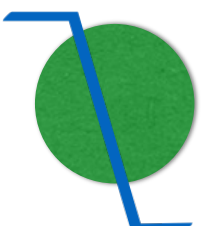
In which direction is the line moving?

Aperture Problem

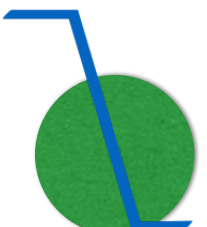


In which direction is the line moving?

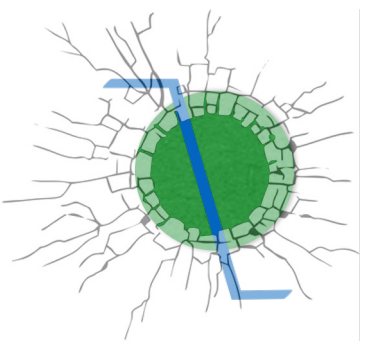
Aperture Problem



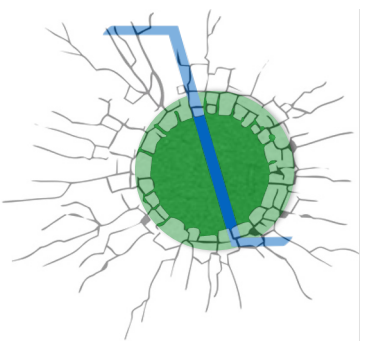
Aperture Problem

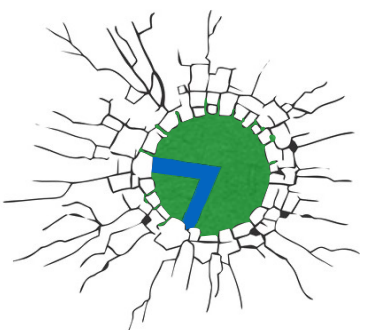


Aperture Problem

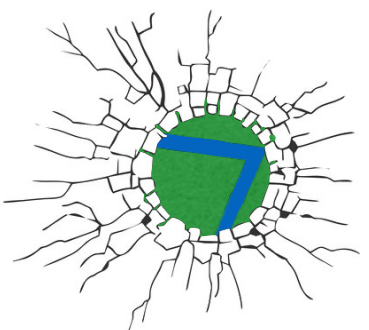


Aperture Problem

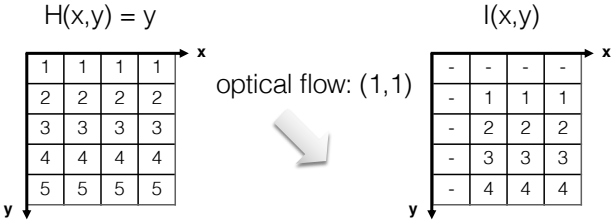




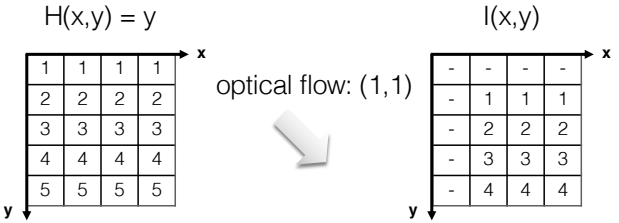
Want patches with different gradients to
the avoid aperture problem



Want patches with different gradients to
the avoid aperture problem


H(x,y) = y
x
y
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
5 5 5 5
optical flow: (1,1)
I(x,y)
x
y
- - - -
- 1 1 1
- 2 2 2
- 3 3 3
- 4 4 4

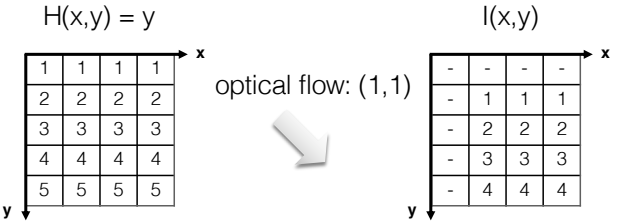
$$I_x u + I_y v + I_t = 0$$



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

Compute gradients

$$I_x(3,3) = 0$$
$$I_y(3,3) = 1$$
$$I_t(3,3) = I(3,3) - H(3,3) = -1$$



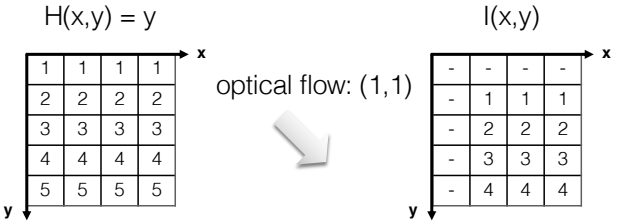
$$I_x u + I_y v + I_t = 0$$

Compute gradients

$$I_x(3,3) = 0$$
$$I_y(3,3) = 1$$
$$I_t(3,3) = I(3,3) - H(3,3) = -1$$

Solution:

$$v = 1$$



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

Compute gradients

$$I_x(3,3) = 0$$
$$I_y(3,3) = 1$$
$$I_t(3,3) = I(3,3) - H(3,3) = -1$$

Solution:

$$v = 1$$

We recover the v of the optical flow but not the u.

This is the aperture problem.

Horn-Schunck optical flow

Horn-Schunck
Optical Flow (1981)

brightness constancy

small motion

‘smooth’ flow
(flow can vary from pixel to pixel)

global method
(dense)

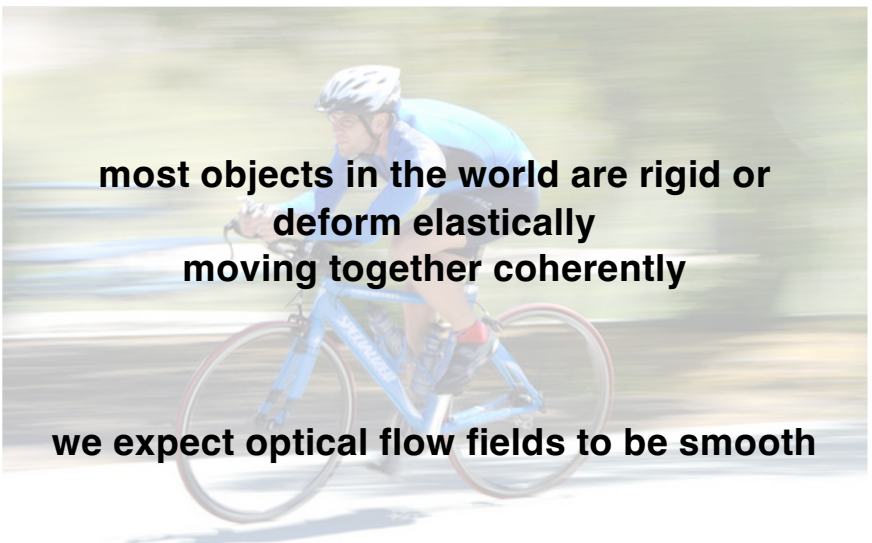
Lucas-Kanade
Optical Flow (1981)

method of differences

‘constant’ flow
(flow is constant for all pixels)

local method
(sparse)

Smoothness


most objects in the world are rigid or
deform elastically
moving together coherently
we expect optical flow fields to be smooth

Key idea

(of Horn-Schunck optical flow)

Enforce

brightness constancy

Enforce

smooth flow field

to compute optical flow

Key idea

(of Horn-Schunck optical flow)

Enforce

brightness constancy

Enforce

smooth flow field

to compute optical flow

Enforce
brightness constancy

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

For every pixel,

$$\min_{u,v} \left[I_x u_{ij} + I_y v_{ij} + I_t \right]^2$$

Enforce
brightness constancy

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

For every pixel,

$$\min_{u,v} \left[I_x u_{ij} + I_y v_{ij} + I_t \right]^2$$

lazy notation for $I_x(i,j)$

Key idea

(of Horn-Schunck optical flow)

Enforce

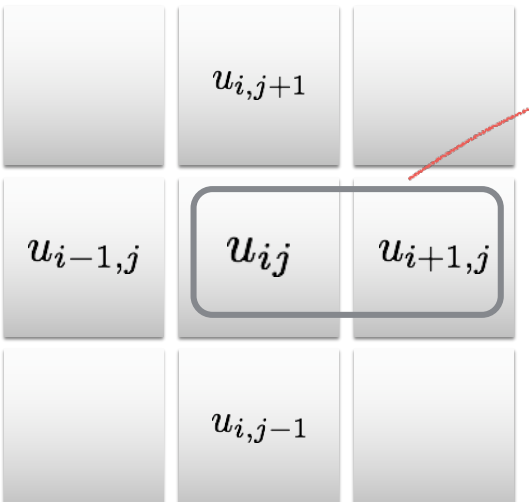
brightness constancy

Enforce

smooth flow field

to compute optical flow

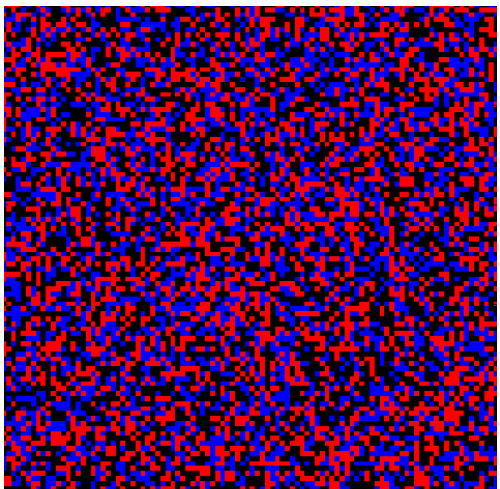
Enforce **smooth flow field**

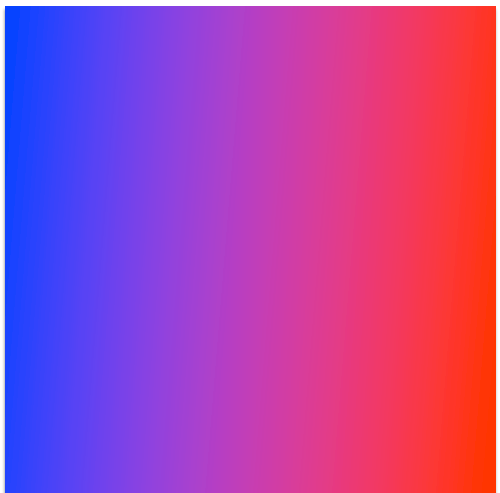


u-component of flow

$$\min_{\boldsymbol{u}} (u_{i,j} - u_{i+1,j})^2$$

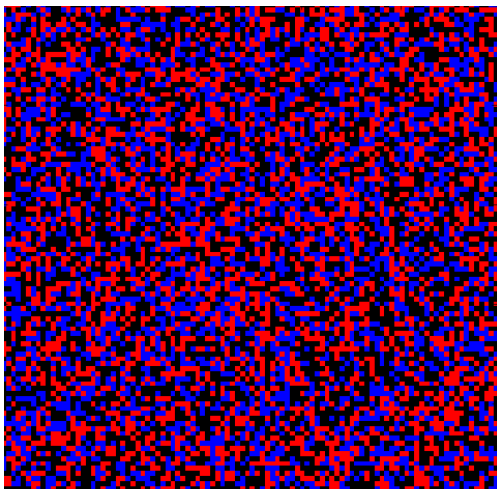
Which flow field optimizes the objective? $\min_{\boldsymbol{u}}(u_{i,j} - u_{i+1,j})^2$





$$\sum_{ij}(u_{ij} - u_{i+1,j})^2 \quad ? \quad \sum_{ij}(u_{ij} - u_{i+1,j})^2$$

Which flow field optimizes the objective? $\min_{\boldsymbol{u}}(u_{i,j} - u_{i+1,j})^2$



big

small

Key idea

(of Horn-Schunck optical flow)

Enforce

brightness constancy

Enforce

smooth flow field

to compute optical flow

bringing it all together…

Horn-Schunck optical flow

$$\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}$$

smoothness

brightness constancy

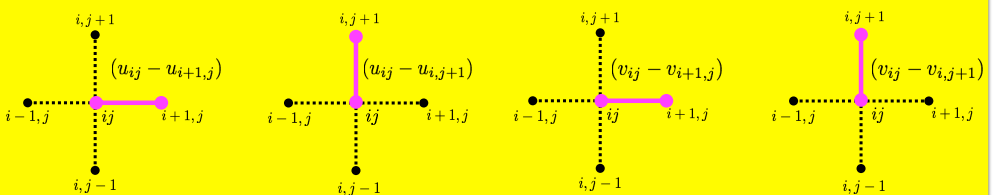
weight

HS optical flow objective function

Brightness constancy $$E_d(i,j) = \left[I_x u_{ij} + I_y v_{ij} + I_t \right]^2$$

Smoothness

$$E_s(i,j) = \frac{1}{4}\left[(u_{ij} - u_{i+1,j})^2 + (u_{ij} - u_{i,j+1})^2 + (v_{ij} - v_{i+1,j})^2 + (v_{ij} - v_{i,j+1})^2 \right]$$



How do we solve this minimization problem?

$$\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}$$

How do we solve this minimization problem?

$$\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}$$

Compute partial derivative, derive update equations

(gradient decent!)

Compute the partial derivatives of this huge sum!

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

smoothness term

brightness constancy

Compute the partial derivatives of this huge sum!

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

it's not so bad…

$$\frac{\partial E}{\partial u_{kl}}=$$

how many u terms depend on k and l?

Compute the partial derivatives of this huge sum!

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

it's not so bad…

$$\frac{\partial E}{\partial u_{kl}}=$$

how many u terms depend on k and l?

FOUR from smoothness

ONE from brightness constancy

Compute the partial derivatives of this huge sum!

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

it's not so bad…

$$\frac{\partial E}{\partial u_{kl}} = 2(u_{kl}-\bar{u}_{kl})+2\lambda(I_x u_{kl}+I_y v_{kl}+I_t)I_x$$

how many u terms depend on k and l?

FOUR from smoothness

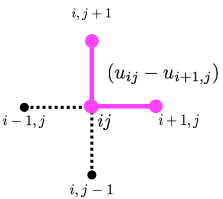
ONE from brightness constancy

Compute the partial derivatives of this huge sum!

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

$(u_{ij}^2-2u_{ij}u_{i+1,j}+u_{i+1,j}^2)$ $\quad$ $(u_{ij}^2-2u_{ij}u_{i,j+1}+u_{i,j+1}^2)$

(variable will appear four times in sum)

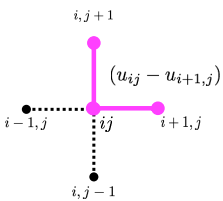


Compute the partial derivatives of this huge sum!

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

$(u_{ij}^2-2u_{ij}u_{i+1,j}+u_{i+1,j}^2)$ $\quad$ $(u_{ij}^2-2u_{ij}u_{i,j+1}+u_{i,j+1}^2)$

(variable will appear four times in sum)



$$\frac{\partial E}{\partial u_{kl}}=2(u_{kl}-\bar{u}_{kl})+2\lambda(I_x u_{kl}+I_y v_{kl}+I_t)I_x$$

$$\frac{\partial E}{\partial v_{kl}}=2(v_{kl}-\bar{v}_{kl})+2\lambda(I_x u_{kl}+I_y v_{kl}+I_t)I_y$$

short hand for local average $\quad \bar{u}_{ij}=\frac{1}{4}\left\{u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}\right\}$

$$\frac{\partial E}{\partial u_{kl}} = 2(u_{kl} - \bar{u}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_x$$

$$\frac{\partial E}{\partial v_{kl}} = 2(v_{kl} - \bar{v}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_y$$

Where are the extrema of E?

$$\frac{\partial E}{\partial u_{kl}} = 2(u_{kl} - \bar{u}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_x$$

$$\frac{\partial E}{\partial v_{kl}} = 2(v_{kl} - \bar{v}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_y$$

Where are the extrema of E?

(set derivatives to zero and solve for unknowns u and v)

$$(1 + \lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1 + \lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

$$\frac{\partial E}{\partial u_{kl}} = 2(u_{kl} - \bar{u}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_x$$

$$\frac{\partial E}{\partial v_{kl}} = 2(v_{kl} - \bar{v}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_y$$

Where are the extrema of E?

(set derivatives to zero and solve for unknowns u and v)

$$(1 + \lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1 + \lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

this is a linear system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ *how do you solve this?*

ok, take a step back, why are we doing all this math?

We are solving for the optical flow (u,v) given two constraints

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

smoothness

brightness constancy

We need the math to minimize this (back to the math)

Partial derivatives of Horn-Schunck objective function E:

$$\frac{\partial E}{\partial u_{kl}} = 2(u_{kl} - \bar{u}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_x$$

$$\frac{\partial E}{\partial v_{kl}} = 2(v_{kl} - \bar{v}_{kl}) + 2\lambda(I_x u_{kl} + I_y v_{kl} + I_t)I_y$$

Where are the extrema of E?

(set derivatives to zero and solve for unknowns u and v)

$$(1 + \lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1 + \lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$ *how do you solve this?*

$$(1+\lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1+\lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

Recall $\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b} = \dfrac{\operatorname{adj}\mathbf{A}}{\det \mathbf{A}}\boldsymbol{b}$

$$(1+\lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1+\lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

Recall $\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b} = \dfrac{\operatorname{adj}\mathbf{A}}{\det \mathbf{A}}\boldsymbol{b}$

$$A = \begin{bmatrix} 1+\lambda I_x^2 & \lambda I_x I_y \\ \lambda I_x I_y & 1+\lambda I_y^2 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \bar{u}_{kl} - \lambda I_x I_t \\ \bar{v}_{kl} - \lambda I_y I_t \end{bmatrix}.$$

$$(1+\lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1+\lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

Recall $\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b} = \dfrac{\operatorname{adj}\mathbf{A}}{\det \mathbf{A}}\boldsymbol{b}$

$$A = \begin{bmatrix} 1+\lambda I_x^2 & \lambda I_x I_y \\ \lambda I_x I_y & 1+\lambda I_y^2 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \bar{u}_{kl} - \lambda I_x I_t \\ \bar{v}_{kl} - \lambda I_y I_t \end{bmatrix}.$$

$$(1+\lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1+\lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

Recall $\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b} = \dfrac{\operatorname{adj}\mathbf{A}}{\det \mathbf{A}}\boldsymbol{b}$

$$A = \begin{bmatrix} 1+\lambda I_x^2 & \lambda I_x I_y \\ \lambda I_x I_y & 1+\lambda I_y^2 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \bar{u}_{kl} - \lambda I_x I_t \\ \bar{v}_{kl} - \lambda I_y I_t \end{bmatrix}.$$

$$\operatorname{adj} A = \begin{bmatrix} 1+\lambda I_y^2 & -\lambda I_x I_y \\ -\lambda I_x I_y & 1+\lambda I_x^2 \end{bmatrix}, \qquad \det A = (1+\lambda I_x^2)(1+\lambda I_y^2) - (\lambda I_x I_y)^2 = 1+\lambda(I_x^2+I_y^2).$$

$$(1+\lambda I_x^2)u_{kl} + \lambda I_x I_y v_{kl} = \bar{u}_{kl} - \lambda I_x I_t$$

$$\lambda I_x I_y u_{kl} + (1+\lambda I_y^2)v_{kl} = \bar{v}_{kl} - \lambda I_y I_t$$

Recall $\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b} = \dfrac{\mathrm{adj}\mathbf{A}}{\det \mathbf{A}}\boldsymbol{b}$

Same as the linear system:

$$\underset{\text{(det A)}}{\{1+\lambda(I_x^2+I_y^2)\}}u_{kl} = (1+\lambda I_x^2)\bar{u}_{kl} - \lambda I_x I_y \bar{v}_{kl} - \lambda I_x I_t$$

$$\underset{\text{(det A)}}{\{1+\lambda(I_x^2+I_y^2)\}}v_{kl} = (1+\lambda I_y^2)\bar{v}_{kl} - \lambda I_x I_y \bar{u}_{kl} - \lambda I_y I_t$$

$$\{1+\lambda(I_x^2+I_y^2)\}u_{kl} = (1+\lambda I_x^2)\bar{u}_{kl} - \lambda I_x I_y \bar{v}_{kl} - \lambda I_x I_t$$

$$\{1+\lambda(I_x^2+I_y^2)\}v_{kl} = (1+\lambda I_y^2)\bar{v}_{kl} - \lambda I_x I_y \bar{u}_{kl} - \lambda I_y I_t$$

Rearrange to get update equations:

$$\underset{\text{new value}}{\hat{u}_{kl}} = \underset{\text{old average}}{\bar{u}_{kl}} - \frac{I_x\bar{u}_{kl} + I_y\bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x$$

$$\hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x\bar{u}_{kl} + I_y\bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

Recall: $\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}$

When lambda is small (lambda inverse is big)…

$$\hat{u}_{kl} = \bar{u}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x$$

new value — old average

$$\hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

Recall: $\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}$

When lambda is small (lambda inverse is big)…

$$\hat{u}_{kl} = \bar{u}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x$$

new value — old average

goes to zero

$$\hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

goes to zero

Recall: $\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}$

When lambda is small (lambda inverse is big)…

$$\hat{u}_{kl} = \bar{u}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x$$

new value — old average

goes to zero

$$\hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

goes to zero

…we only care about smoothness.

ok, take a step back, why did we do all this math?

We are solving for the optical flow (u,v) given two constraints

$$\sum_{ij}\left\{\frac{1}{4}\left[(u_{ij}-u_{i+1,j})^2+(u_{ij}-u_{i,j+1})^2+(v_{ij}-v_{i+1,j})^2+(v_{ij}-v_{i,j+1})^2\right]+\lambda\left[I_x u_{ij}+I_y v_{ij}+I_t\right]^2\right\}$$

smoothness

brightness constancy

We needed the math to minimize this (now to the algorithm)

Horn-Schunck Optical Flow Algorithm

1. Precompute image gradients $I_y \quad I_x$
2. Precompute temporal gradients I_t
3. Initialize flow field $\boldsymbol{u} = \mathbf{0}$ $\boldsymbol{v} = \mathbf{0}$
4. While not converged

   Compute flow field updates for each pixel:

$$\hat{u}_{kl} = \bar{u}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x \qquad \hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

Just 8 lines of code!

Motion magnification using optical flow

How would you achieve this effect?

original

motion-magnified

- Compute optical flow from frame to frame.
- Magnify optical flow velocities.
- Appropriately warp image intensities.

How would you achieve this effect?

naïvely motion-magnified

motion-magnified

- Compute optical flow from frame to frame.
- Magnify optical flow velocities.
- Appropriately warp image intensities.

In practice, many additional steps are required for a good result.

Some more examples

Some more examples

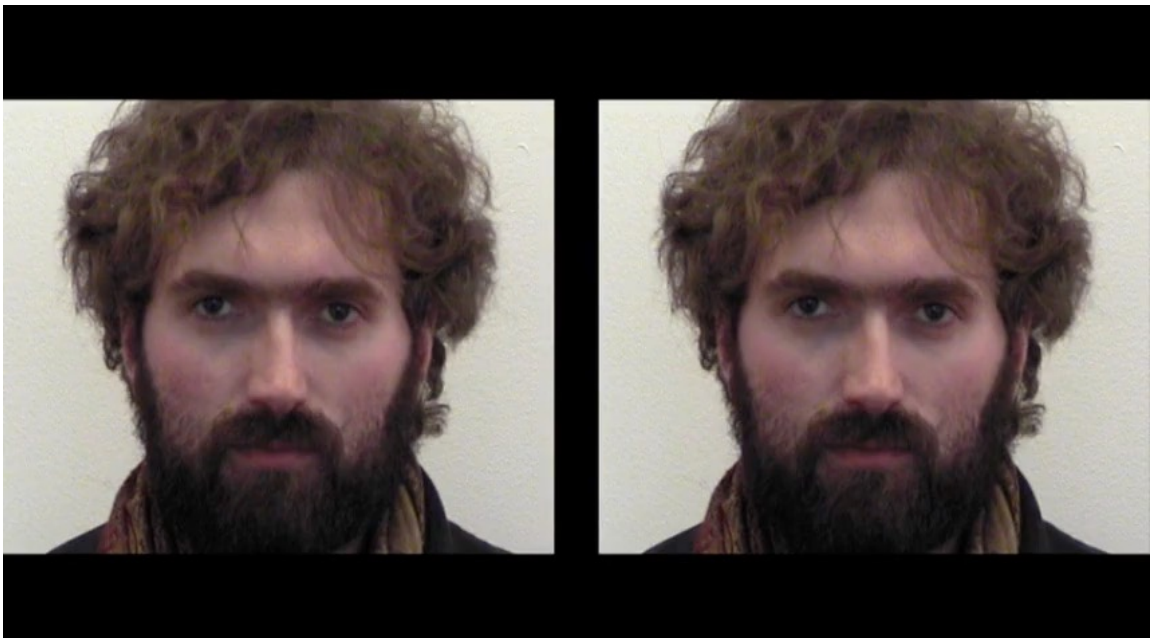
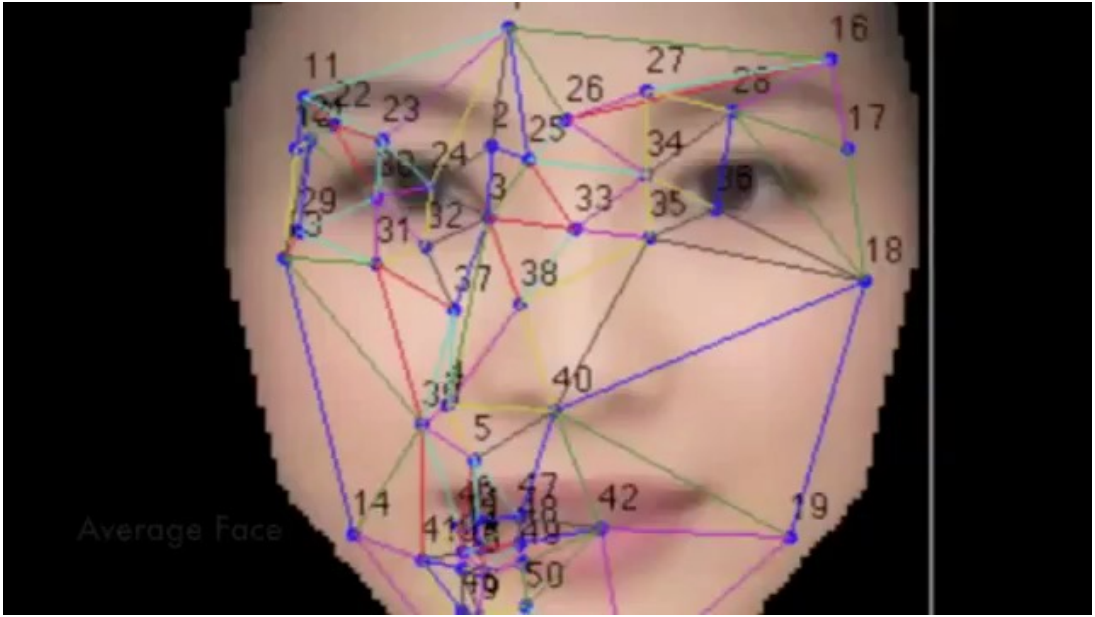
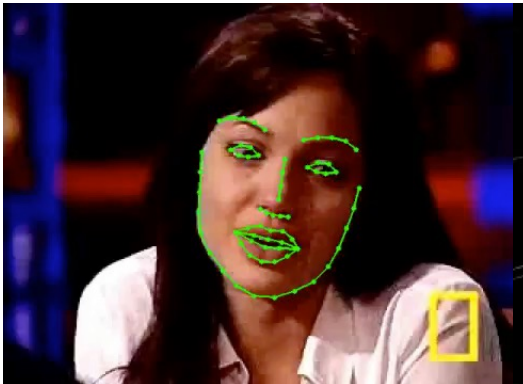
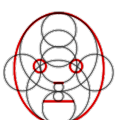


Image alignment


Average Face





IntraFace

http://www.humansensing.cs.cmu.edu/intraface/


31 fps

How can I find  in the image?

Idea #1: Template Matching

Slow, combinatory, global solution

Idea #2: Pyramid Template Matching

Faster, combinatory, locally optimal

Idea #3: Model refinement

(when you have a good initial solution)

Fastest, locally optimal

Some notation before we get into the math…

2D image transformation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$$

2D image coordinate

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Parameters of the transformation

$$\boldsymbol{p} = \{p_1, \ldots, p_N\}$$

Warped image

$$I(\boldsymbol{x}') = I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$$

Pixel value at a coordinate

Translation

Affine

Some notation before we get into the math…

2D image transformation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$$

2D image coordinate

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Parameters of the transformation

$$\boldsymbol{p} = \{p_1, \ldots, p_N\}$$

Warped image

$$I(\boldsymbol{x}') = I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$$

Pixel value at a coordinate

Translation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

transform coordinate

Affine

Some notation before we get into the math…

2D image transformation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$$

2D image coordinate

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Parameters of the transformation

$$\boldsymbol{p} = \{p_1, \ldots, p_N\}$$

Warped image

$$I(\boldsymbol{x}') = I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$$

Pixel value at a coordinate

Translation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

transform coordinate

Affine

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

affine transform coordinate

can be written in matrix form when linear

affine warp matrix can also be 3x3 when last row is [0 0 1]

$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ takes a _____ as input and returns a _____

$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ is a function of ____ variables

$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ returns a _____ of dimension ___ x ___

$\boldsymbol{p} = \{p_1, \ldots, p_N\}$ where N is _____ for an affine model

$I(\boldsymbol{x}') = I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$ this warp changes pixel values?

Image alignment
(problem definition)

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

warped image

template image

Find the warp parameters **p** such that
the SSD is minimized

Find the warp parameters **p** such that
the SSD is minimized

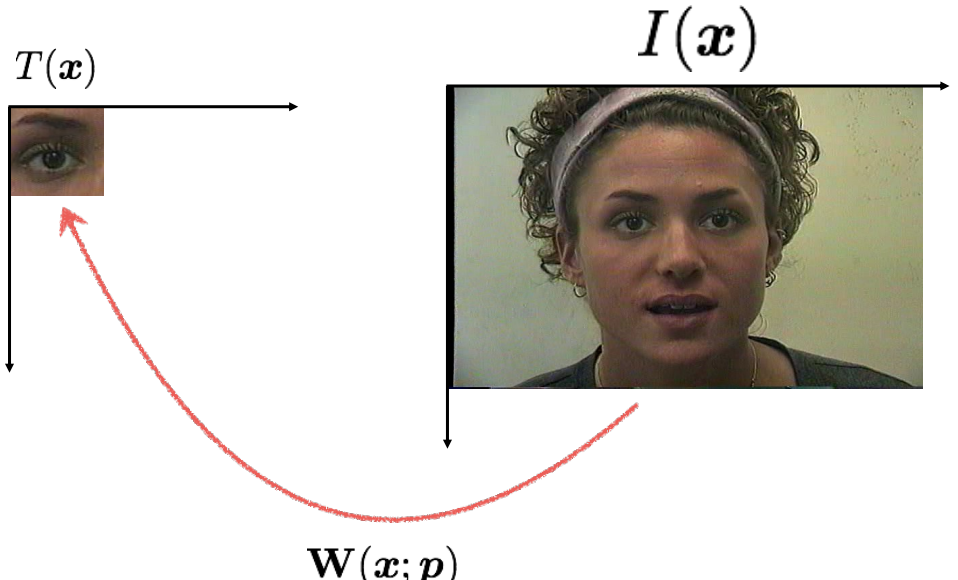


Image alignment
(problem definition)

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

warped image

template image

Find the warp parameters **p** such that
the SSD is minimized

How could you find a solution to this problem?

This is a non-linear (quadratic) function of a non-parametric function!

(Function $\boldsymbol{I}$ is non-parametric)

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

Hard to optimize

What can you do to make it easier to solve?

This is a non-linear (quadratic) function of a non-parametric function!

(Function $\boldsymbol{I}$ is non-parametric)

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

Hard to optimize

What can you do to make it easier to solve?

assume good initialization,
linearized objective and update incrementally

Lucas-Kanade alignment

(pretty strong assumption)

If you have a good initial guess $\boldsymbol{p}$...

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

can be written as ...

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

(a small incremental adjustment)
(this is what we are solving for now)

This is **still** a non-linear (quadratic) function of a non-parametric function!

(Function $\boldsymbol{I}$ is non-parametric)

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p} + \Delta \boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

*How can we linearize the function $\boldsymbol{I}$ for a really small perturbation of **p**?*

This is **still** a non-linear (quadratic) function of a non-parametric function!

(Function $\boldsymbol{I}$ is non-parametric)

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p} + \Delta \boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

*How can we linearize the function $\boldsymbol{I}$ for a really small perturbation of **p**?*

Taylor series approximation!

$$\sum_{\boldsymbol{x}} [I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) - T(\boldsymbol{x})]^2$$

Multivariable Taylor Series Expansion
(First order approximation)

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

Multivariable Taylor Series Expansion
(First order approximation)

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

Recall: $\boldsymbol{x'} = \mathbf{W}(\boldsymbol{x};\boldsymbol{p})$

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \frac{\partial I(\mathbf{W}(\mathbf{x};\mathbf{p}))}{\partial \mathbf{p}} \Delta\mathbf{p}$$

$$= I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \frac{\partial I(\mathbf{W}(\mathbf{x};\mathbf{p}))}{\partial \mathbf{x}'} \frac{\partial \mathbf{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p} \quad \text{(chain rule)}$$

$$= I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \quad \text{(short-hand notation)}$$

$$\sum_{\boldsymbol{x}} [I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) - T(\boldsymbol{x})]^2$$

Multivariable Taylor Series Expansion
(First order approximation)

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

Linear approximation

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

What are the unknowns here?

$$\sum_{\boldsymbol{x}} [I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) - T(\boldsymbol{x})]^2$$

Multivariable Taylor Series Expansion
(First order approximation)

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

Linear approximation

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Now, the function is a linear function of the unknowns

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

$\boldsymbol{x}$ is a _____ of dimension ___ x ___

output of $\mathbf{W}$ is a _____ of dimension ___ x ___

$\boldsymbol{p}$ is a _____ of dimension ___ x ___

$I(\cdot)$ is a function of _____ variables

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

$\boldsymbol{x}$ is a vector of dimension 2 x 1

output of $\mathbf{W}$ is a _____ of dimension ___ x ___

$\boldsymbol{p}$ is a _____ of dimension ___ x ___

$I(\cdot)$ is a function of _____ variables

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

$\boldsymbol{x}$ is a vector of dimension 2 x 1

output of $\mathbf{W}$ is a vector of dimension 2 x 1

$\boldsymbol{p}$ is a _____ of dimension ___ x ___

$I(\cdot)$ is a function of _____ variables

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

$\boldsymbol{x}$ is a vector of dimension 2 x 1

output of $\mathbf{W}$ is a vector of dimension 2 x 1

$\boldsymbol{p}$ is a vector of dimension 6 x 1

$I(\cdot)$ is a function of _____ variables

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

$\boldsymbol{x}$ is a vector of dimension 2 x 1

output of $\mathbf{W}$ is a vector of dimension 2 x 1

$\boldsymbol{p}$ is a vector of dimension 6 x 1

$I(\cdot)$ is a function of 2 variables

The Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$

(A matrix of partial derivatives)

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} W_x(x, y) \\ W_y(x, y) \end{bmatrix}$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}$$

Rate of change of the warp

The Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$

(A matrix of partial derivatives)

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} W_x(x,y) \\ W_y(x,y) \end{bmatrix}$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}$$

Rate of change of the warp

Affine transform

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} p_1 x + p_3 y + p_5 \\ p_2 x + p_4 y + p_6 \end{bmatrix}$$

$$\frac{\partial W_x}{\partial p_1} = x \qquad \frac{\partial W_x}{\partial p_2} = 0 \qquad \cdots$$

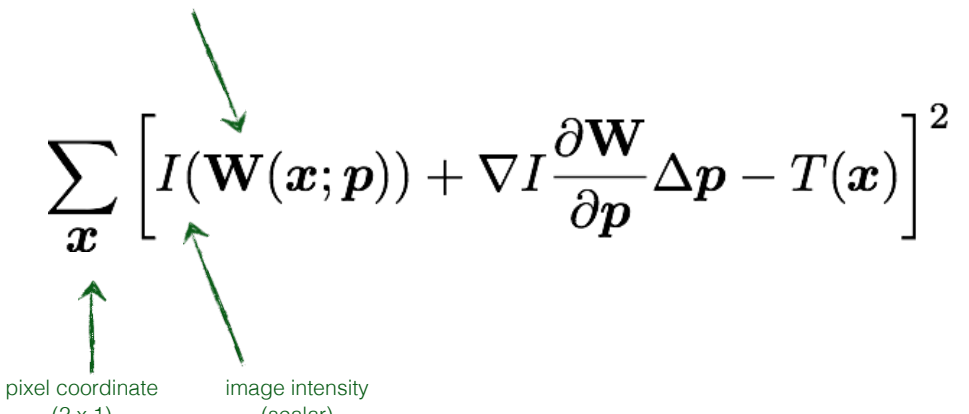
$$\frac{\partial W_y}{\partial p_1} = 0 \qquad \cdots$$

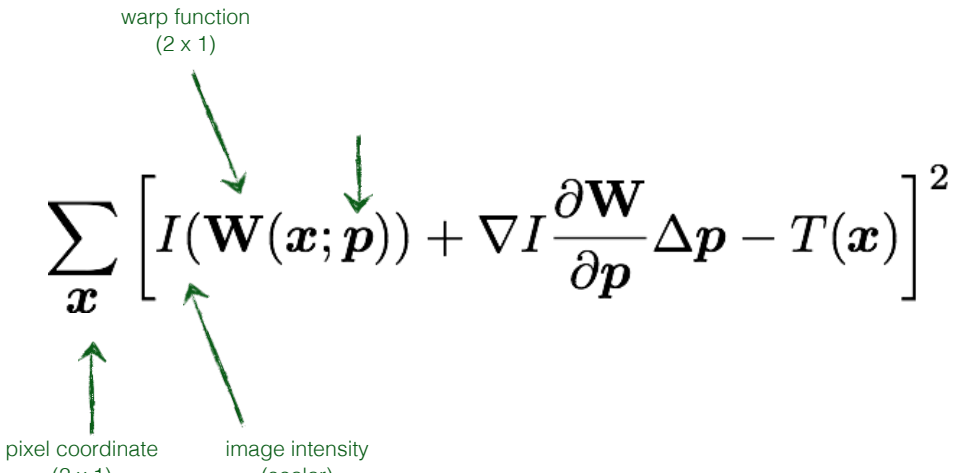
$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

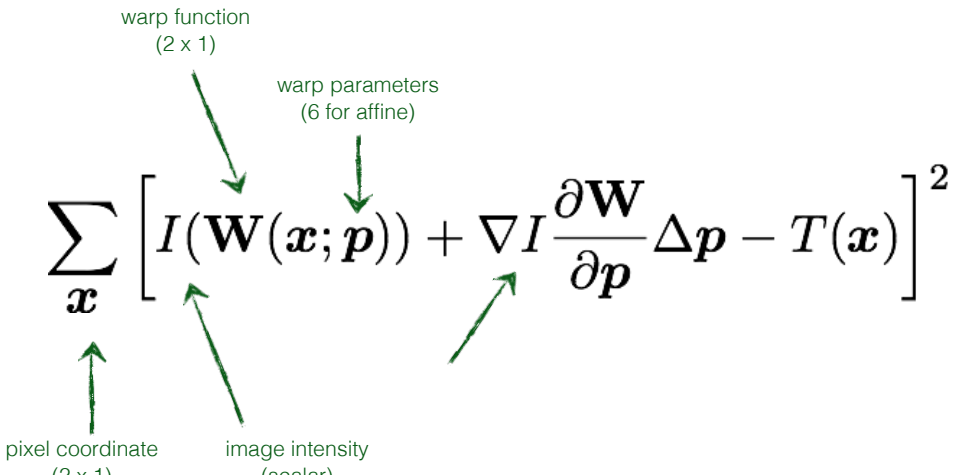
$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

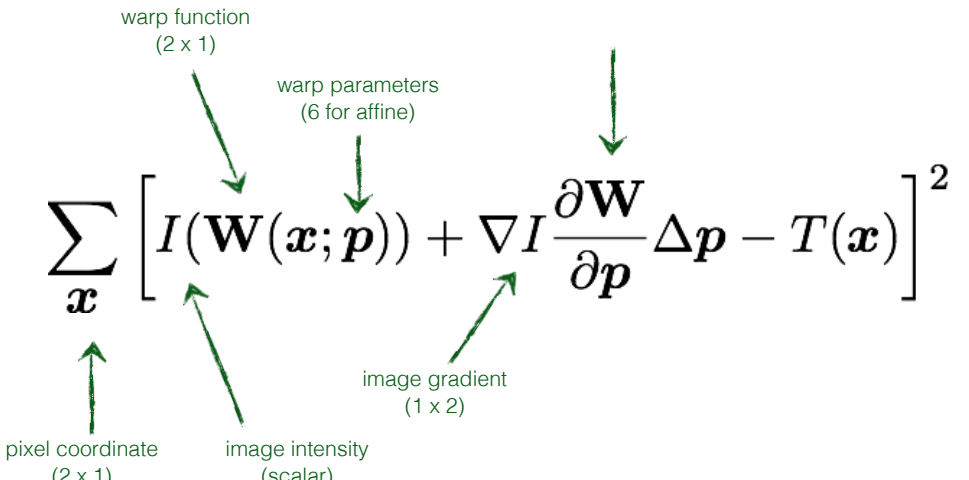
$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

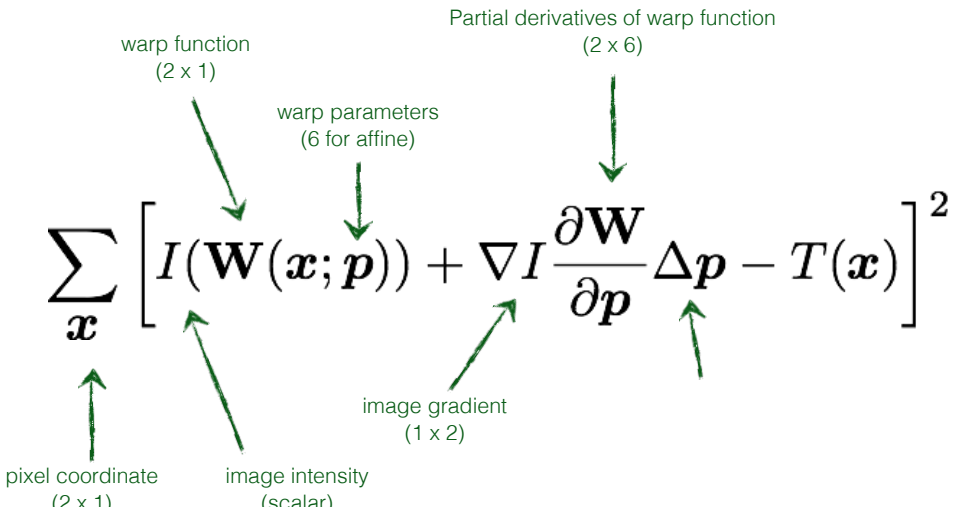
pixel coordinate
(2 x 1)

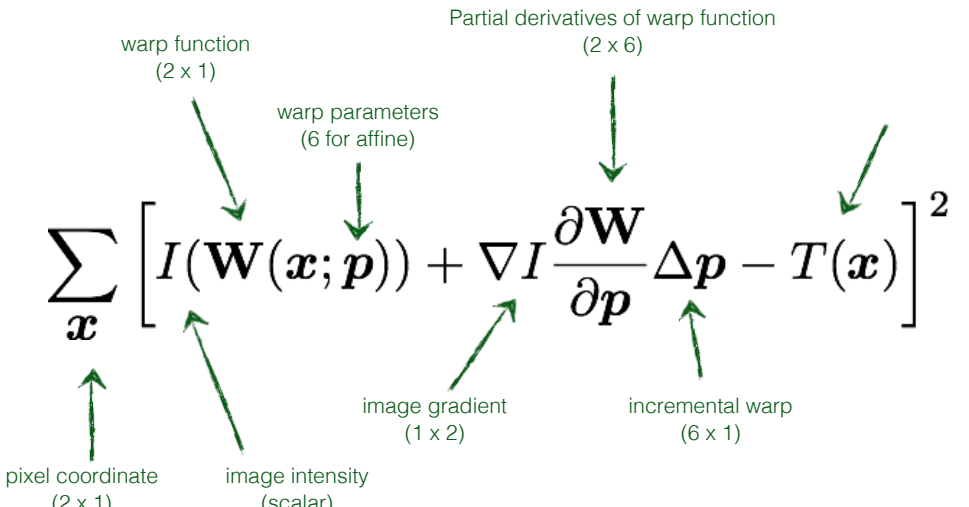

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$
pixel coordinate
(2 x 1)
image intensity
(scalar)

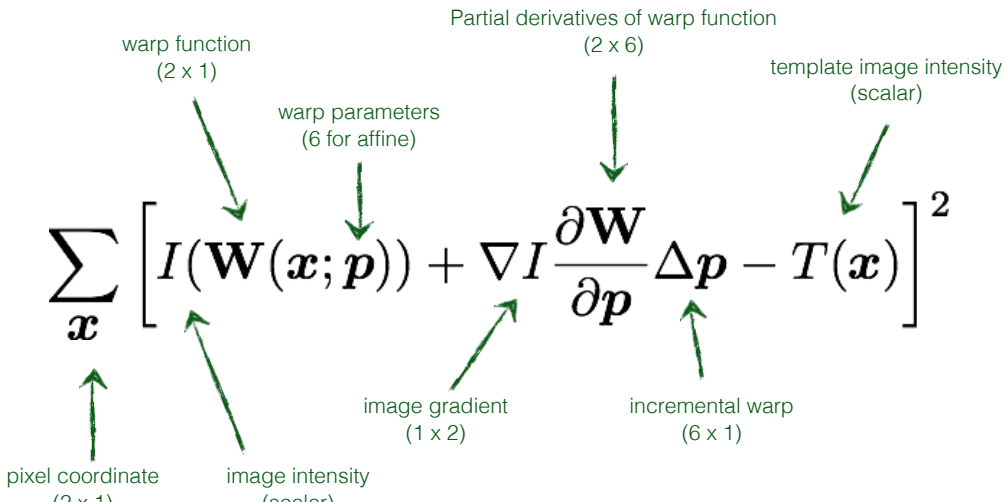

warp function
(2 x 1)
$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$
pixel coordinate
(2 x 1)
image intensity
(scalar)


warp function
(2 x 1)
warp parameters
(6 for affine)
$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$
pixel coordinate
(2 x 1)
image intensity
(scalar)


warp function
(2 x 1)
warp parameters
(6 for affine)
$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$
image gradient
(1 x 2)
pixel coordinate
(2 x 1)
image intensity
(scalar)


warp function
(2 x 1)
warp parameters
(6 for affine)
Partial derivatives of warp function
(2 x 6)
$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$
image gradient
(1 x 2)
pixel coordinate
(2 x 1)
image intensity
(scalar)


warp function
(2 x 1)
warp parameters
(6 for affine)
Partial derivatives of warp function
(2 x 6)
$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$
pixel coordinate
(2 x 1)
image intensity
(scalar)
image gradient
(1 x 2)
incremental warp
(6 x 1)



When you implement this, you will compute everything in parallel and store as matrix … don't loop over x!

Summary

(of Lucas-Kanade Image Alignment)

Problem:

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

warped image — template image

Difficult non-linear optimization problem

Strategy:

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

Assume known approximate solution
Solve for increment

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Taylor series approximation
Linearize

then solve for $\Delta\boldsymbol{p}$

OK, so how do we solve this?

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Another way to look at it…

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

(moving terms around)

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - \{T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))\} \right]^2$$

vector of constants

vector of variables

constant

Have you seen this form of optimization problem before?

Another way to look at it…

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - \{T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))\} \right]^2$$

constant variable constant

Looks like $\mathbf{Ax} - \mathbf{b}$

How do you solve this?

Least squares approximation

$$\hat{x} = \arg\min_{x} ||Ax - b||^2 \quad \text{is solved by} \quad x = (A^\top A)^{-1} A^\top b$$

Applied to our tasks:

$$\min_{\Delta \boldsymbol{p}} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - \{T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}))\} \right]^2$$

is optimized when

$$\Delta \boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}))]$$

after applying $x = (A^\top A)^{-1} A^\top b$

where $H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$

$A^\top A$

Solve:

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

warped image — template image

Difficult non-linear optimization problem

Strategy:

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

Assume known approximate solution
Solve for increment

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Taylor series approximation
Linearize

Solution:

$$\Delta\boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) \right]$$

Solution to least squares approximation

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$$

Hessian

This is called…

Gauss-Newton gradient decent non-linear optimization!

Lucas Kanade (Additive alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$
3. Compute gradient $\nabla I(\boldsymbol{x}')$ x'coordinates of the warped image (gradients of the warped image)
4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$
5. Compute Hessian H $H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$
6. Compute $\Delta \boldsymbol{p}$ $\Delta \boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$
7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

Just 8 lines of code!

Overview

- review
  - camera models & calibration
  - two-view geometry
  - structure from motion
  - NeRF
- optical flow
  - brightness constancy
  - Lucas-Kanade (constant flow)
  - Horn-Schunk (smooth flow)
- motion magnification
- image alignment & tracking

next time — tracking continued