ECE5554 – Computer Vision
Lecture 5a – Motion Tracking

Creed Jones, PhD

ece BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

VIRGINIA TECH

Course update

- HW2 is due TONIGHT at 11:59 PM
- HW3 is posted; due next Wednesday, July 27

Today's Objectives

- Motion Estimation – the concept
- Lucas-Kanade motion estimation
  - Optical Flow constraint
  - Gradient constraint
  - Solution
- Kanade-Lucas-Tomasi keypoint tracking
  - Ideal features for tracking
  - The KLT method

INTRODUCTION TO MOTION ESTIMATION

(THANKS TO DR. A. L. ABBOTT FOR SEVERAL OF THESE SLIDES)

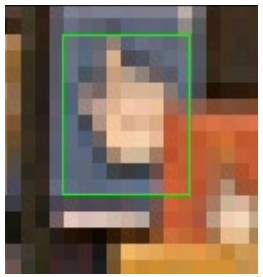
Classes of Techniques for Motion Estimation

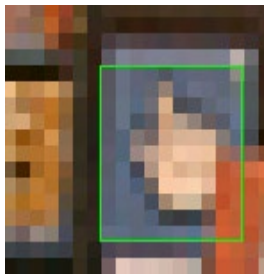
- ***Feature-based methods***
  - Extract visual features (corners, textured areas) and track them
  - Sparse motion fields, but possibly robust tracking
  - Suitable especially when image motion is large (10s of pixels)
- ***Direct-methods***
  - Directly recover image motion from spatio-temporal image brightness variations
  - Global motion parameters directly recovered without an intermediate feature motion calculation
  - Dense motion fields, but more sensitive to appearance variations
  - Suitable for video and when image motion is small (< 10 pixels)

Template matching (revisited)

- How do we determine correspondences? *Block Matching*
  - We described NGC: Normalized grayscale correlation
  - Another approach was *SSD* (sum squared differences)

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

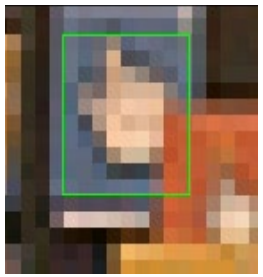




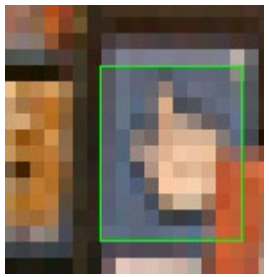
Template matching (revisited)

- How do we determine correspondences? *Block Matching*
  - We described NGC: Normalized grayscale correlation
  - Another approach was *SSD* (sum squared differences)

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



I



J

Finding possible motion (offsets) between images is complex; we need to employ some reasonability constraints

- We cannot assume that all points move the same
  - In general, the whole image does not experience a common displacement
  - Affine transformation may take place
  - Warping may take place

- We can rely on some *constraints:*
  - Image brightness does not change much from image to image
    - The Brightness Constraint
  - Gradients in the image are related to determination of motion, directionally
    - The Gradient Constraint

The Brightness Constraint requires that successive images (*I* and *J)* have nearly the same pattern of intensities – or *brightness* (allowing for motion of blocks)

- Brightness Constancy Equation:

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

Or, equivalently, minimize :

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

Linearizing (assuming small *(u,v)*) using Taylor series expansion, and using I_x to represent $\frac{\partial I}{\partial x}$:

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

Gradient Constraint (or the Optical Flow Constraint) says that the component of motion in the gradient direction is determined, but the component of the flow parallel to an edge is unknown

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

Minimizing:

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

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

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

In general $I_x, I_y \neq 0$

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

Patch Translation [Lucas-Kanade]

Assume a single velocity for all pixels within an image patch

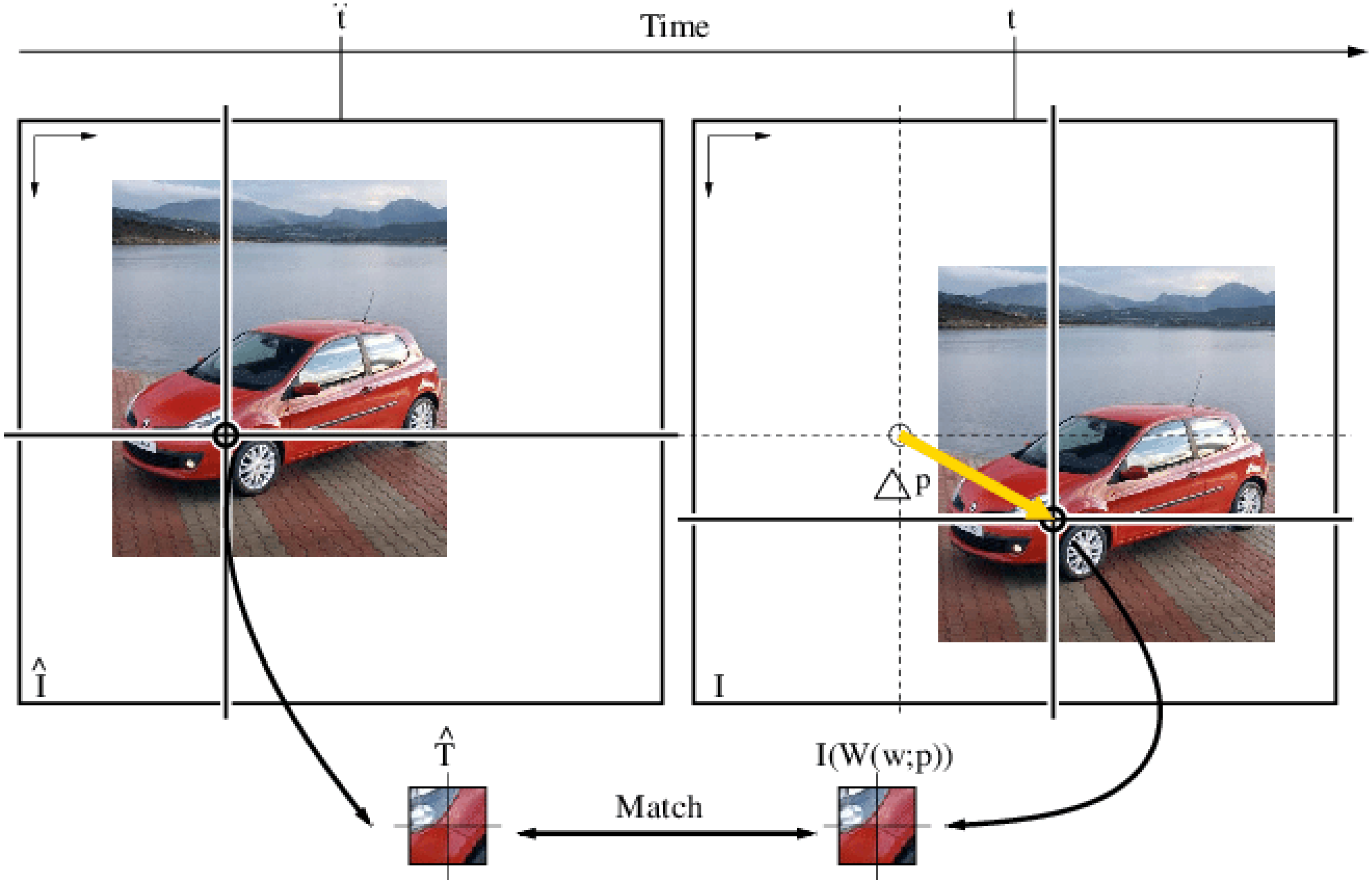
$$E(u,v) = \sum_{x,y \in \Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t \right)^2$$

Minimizing

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

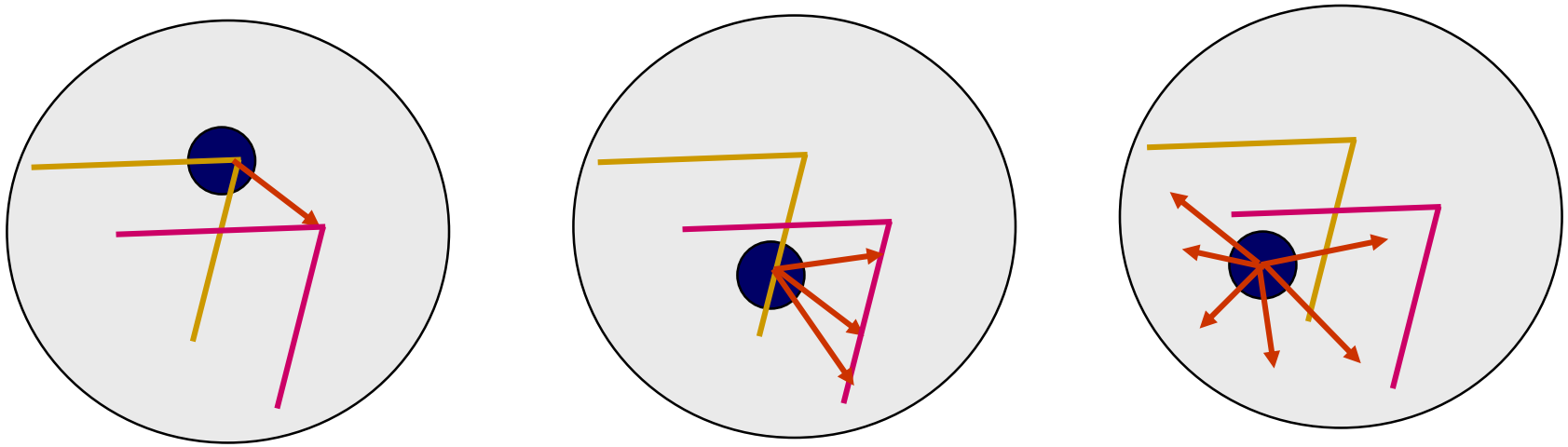
$$\left(\sum \nabla I \, \nabla I^T \right) \vec{U} = -\sum \nabla I I_t$$

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


t
Time
t
Δp
$\hat{I}$
I
$\hat{T}$
I(W(w;p))
Match

Local Patch Analysis

- How *certain* are the motion estimates?



The Aperture Problem

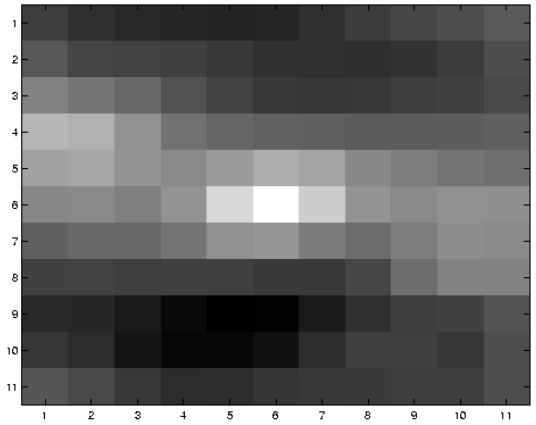
Let $M = \sum (\nabla I)(\nabla I)^T$ and $b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

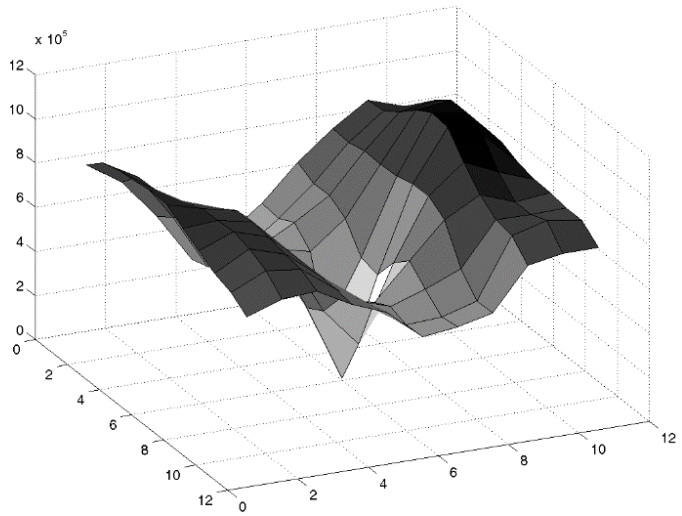
- Algorithm: At each pixel compute U by solving $MU = b$

- *M* is singular if all gradient vectors point in the same direction
  - e.g., along an edge
  - of course, trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only *normal flow* is available (aperture problem)

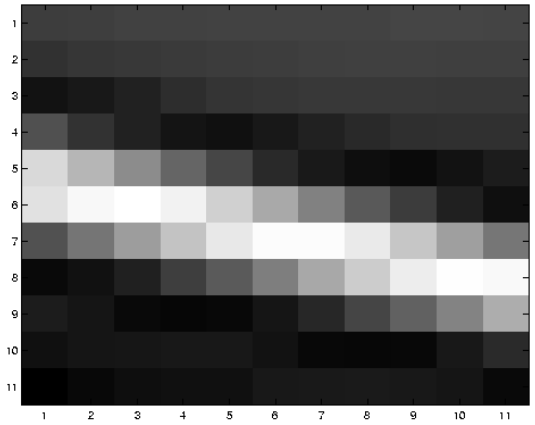
- Corners and textured areas are OK

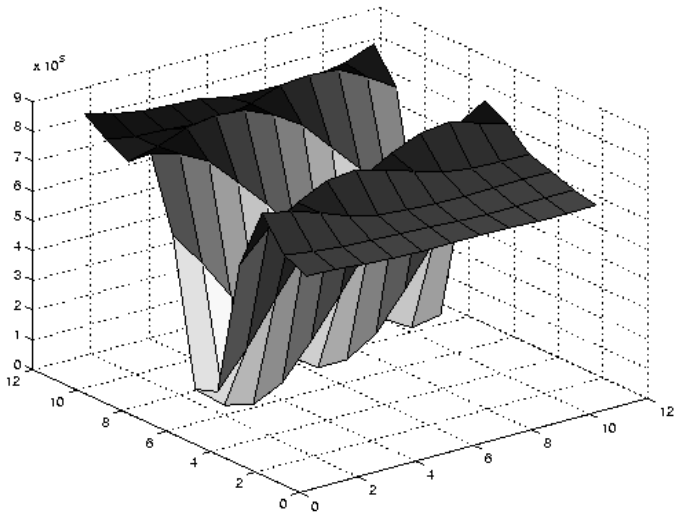
SSD Surface – Textured area



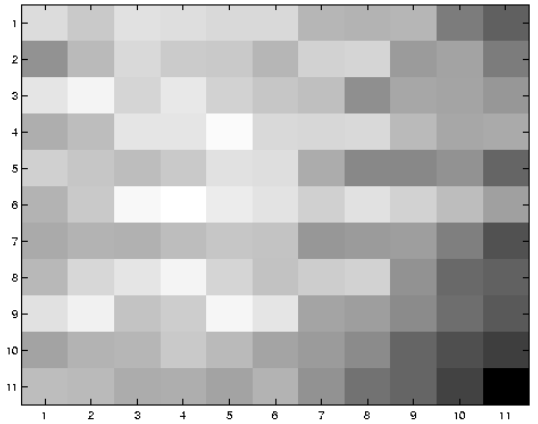


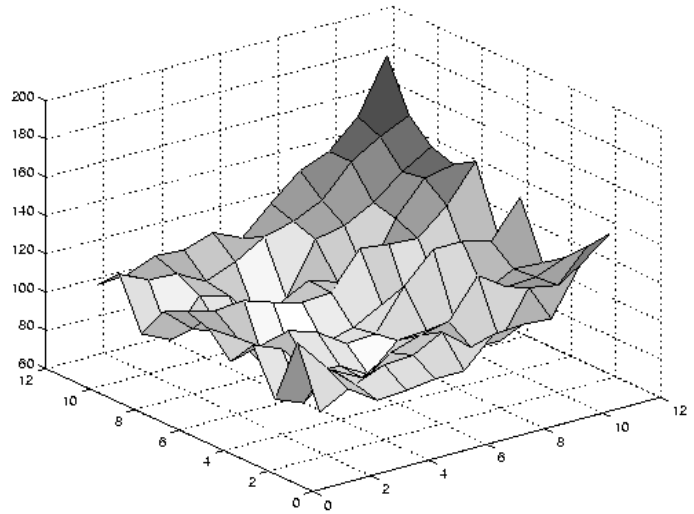
SSD Surface -- Edge





SSD – homogeneous area

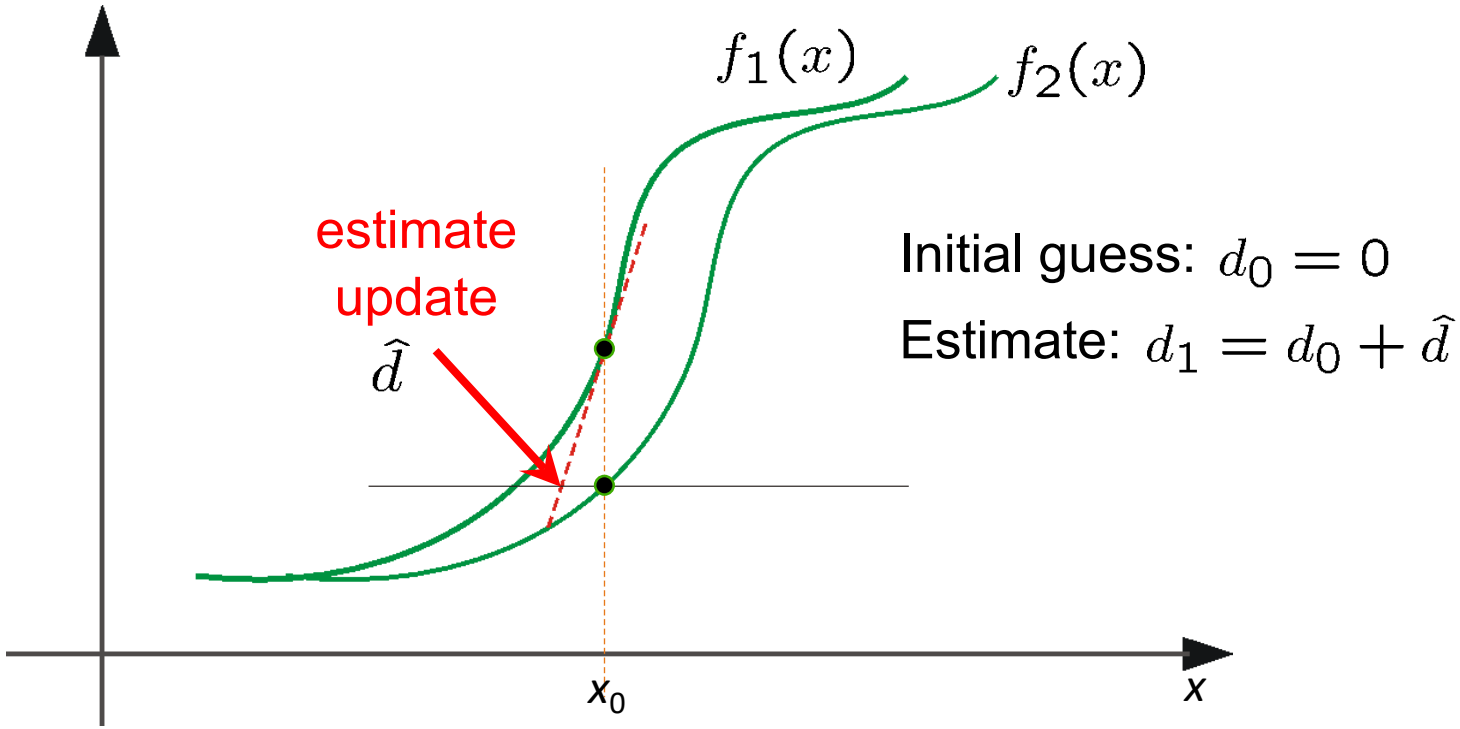




Iterative Refinement

- Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation
- Warp one image toward the other using the estimated flow field
  (easier said than done)
- Refine estimate by repeating the process

Optical Flow: Iterative Estimation (in 1D)

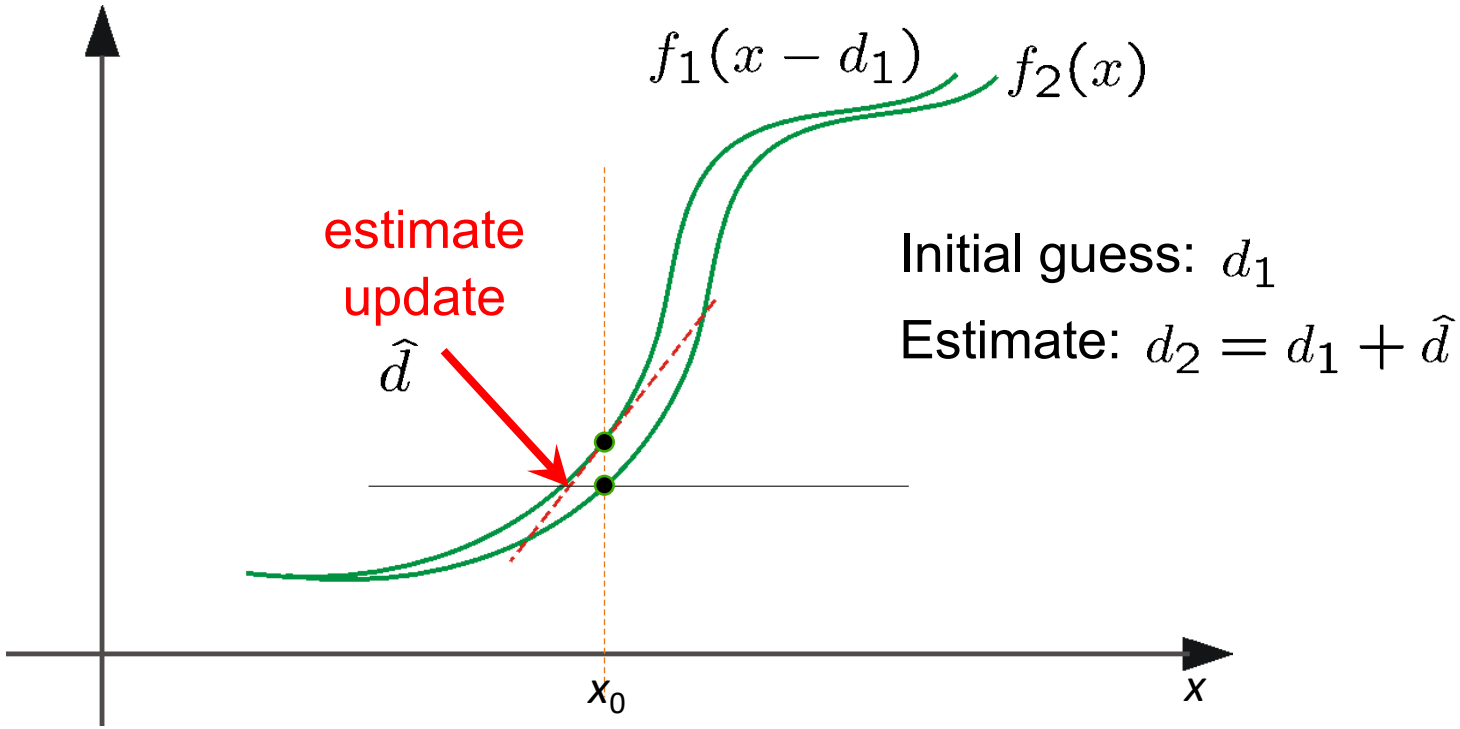


Initial guess: $d_0 = 0$

Estimate: $d_1 = d_0 + \hat{d}$

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

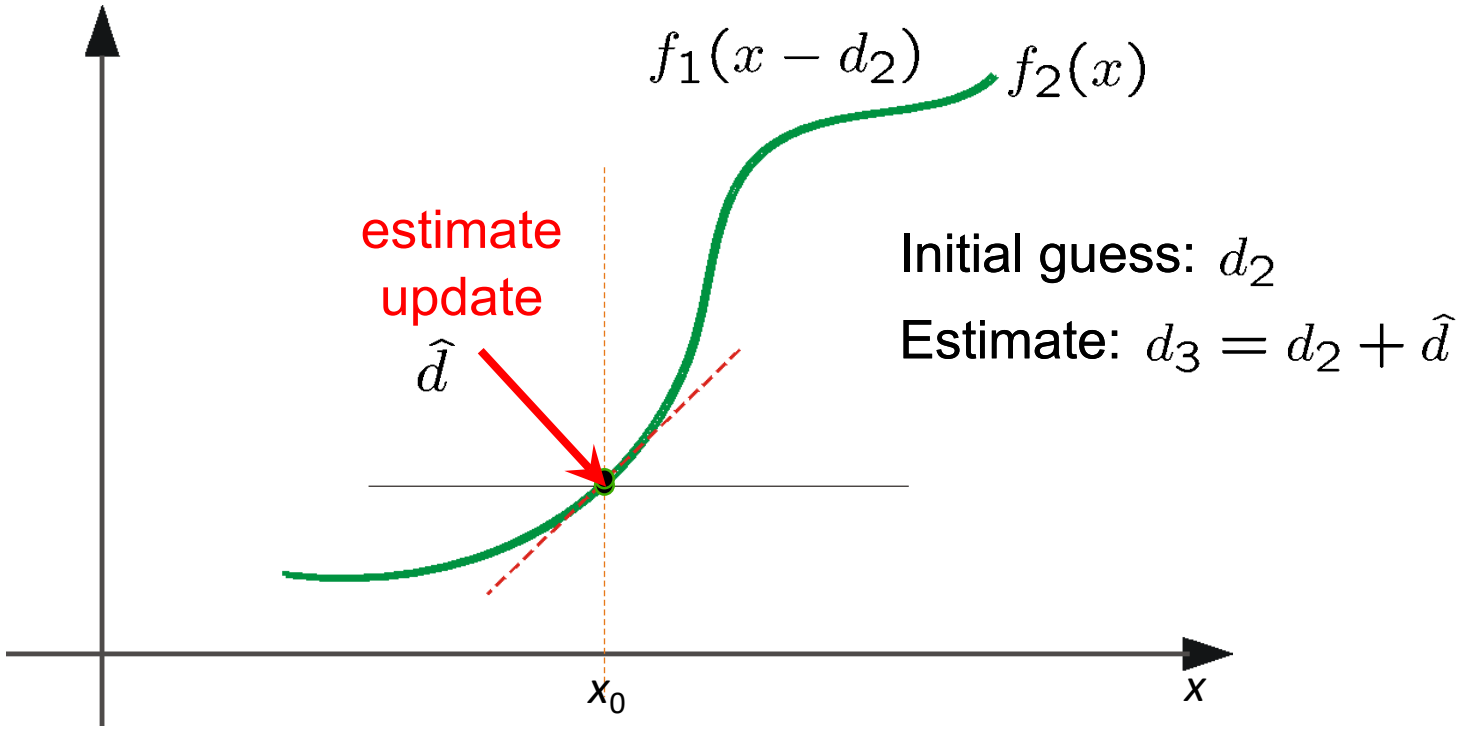
Optical Flow: Iterative Estimation (in 1D)



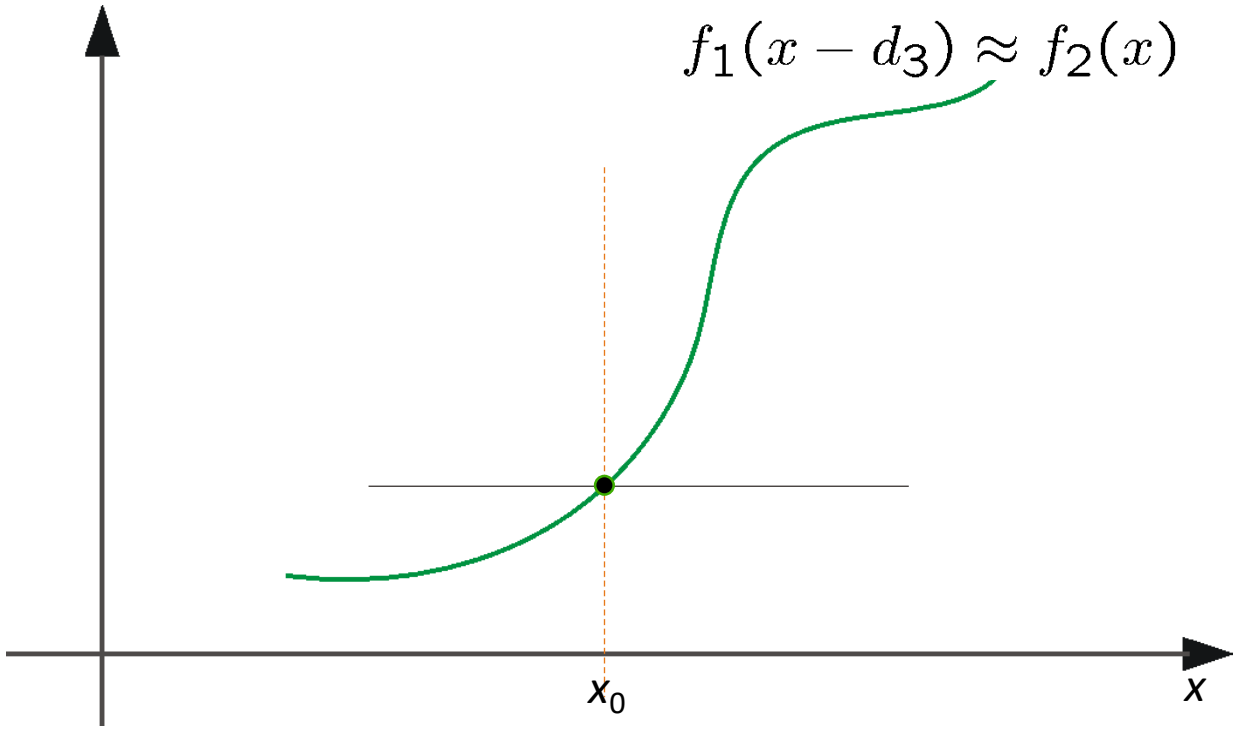
Initial guess: d_1

Estimate: $d_2 = d_1 + \hat{d}$

Optical Flow: Iterative Estimation (in 1D)



Optical Flow: Iterative Estimation (in 1D)



Optical Flow: Iterative Estimation

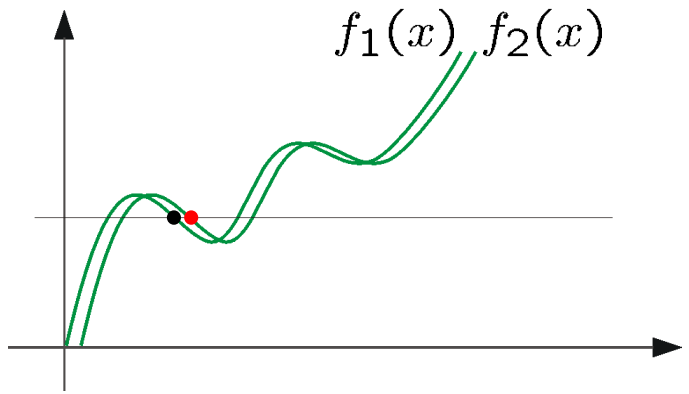
- Assumptions:
  - Displacement from image to image is not large
  - Local warping of the image is very small

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

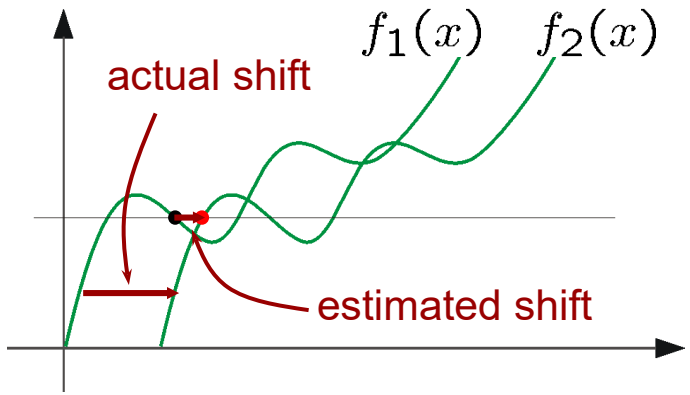
Optical Flow: Aliasing

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

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



nearest match is correct (no aliasing)



nearest match is incorrect (aliasing)

To overcome aliasing: coarse-to-fine estimation.

The Lucas-Kanade image alignment method:

The error function, SSD (Sum of Squared Differences):	$\sum_x \left[I\left(W(x,p)\right) - T(x) \right]^2$

The Lucas-Kanade image alignment method:

The error function, SSD (Sum of Squared Differences):	$\sum_x \left[I\left(W(x,p)\right) - T(x)\right]^2$
Incremental update:	$\sum_x \left[I\left(W(x,p+\Delta p)\right) - T(x)\right]^2$

The Lucas-Kanade image alignment method:

The error function, SSD (Sum of Squared Differences):	$\sum_x \left[I\left(W(x,p)\right) - T(x)\right]^2$
Incremental update:	$\sum_x \left[I\left(W(x,p+\Delta p)\right) - T(x)\right]^2$
Linearize:	$\sum_x \left[I\left(W(x,p)\right) + \nabla I \frac{\partial W}{\partial p}\Delta p - T(x)\right]^2$

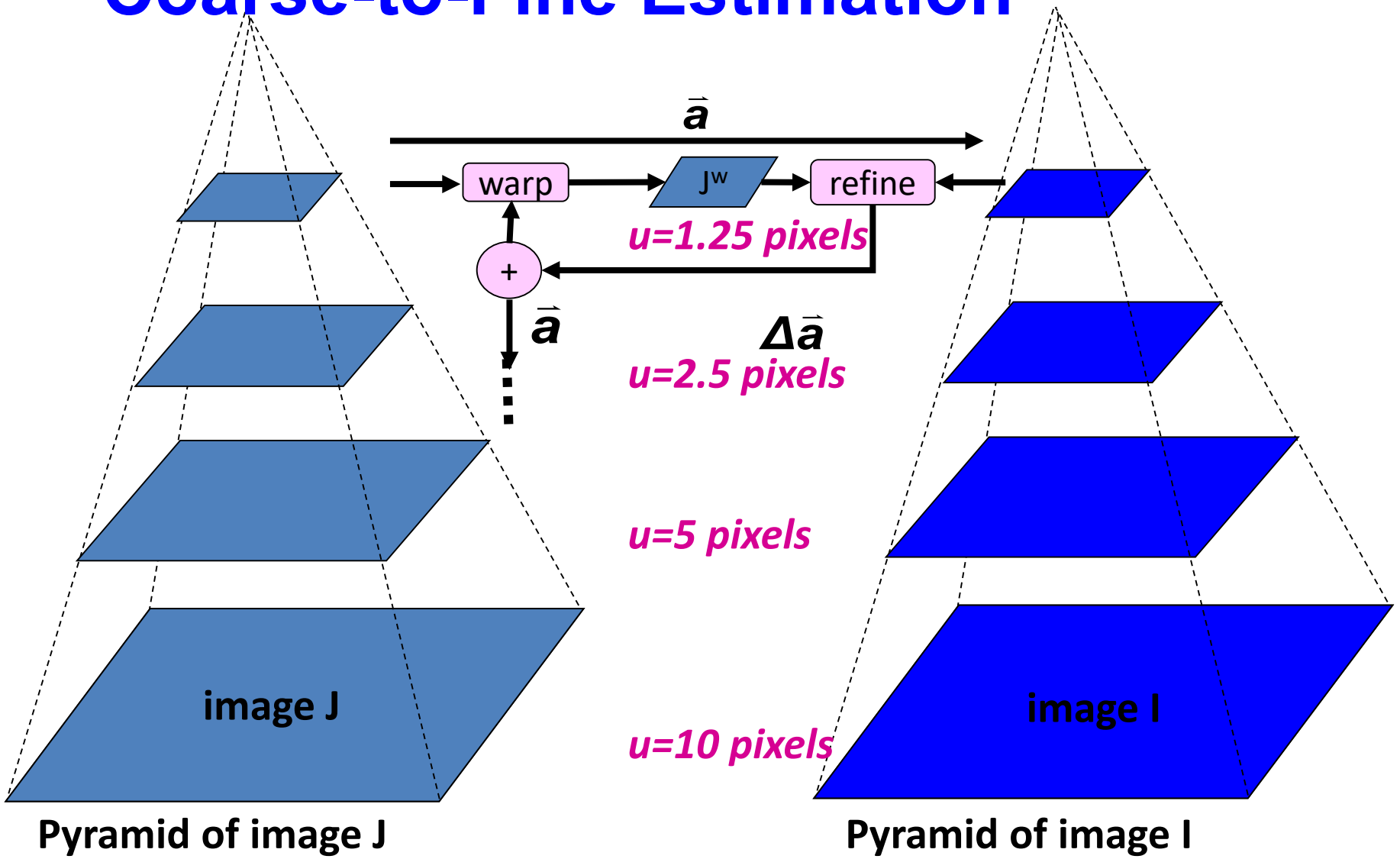
The Lucas-Kanade image alignment method:

The error function, SSD (Sum of Squared Differences):	$\sum_x \left[I\left(W(x,p)\right) - T(x)\right]^2$
Incremental update:	$\sum_x \left[I\left(W(x,p+\Delta p)\right) - T(x)\right]^2$
Linearize:	$\sum_x \left[I\left(W(x,p)\right) + \nabla I \frac{\partial W}{\partial p}\Delta p - T(x)\right]^2$
Gradient update:	$\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[T(x) - I\left(W(x,p)\right)\right]$ $H = \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial p}\right]$

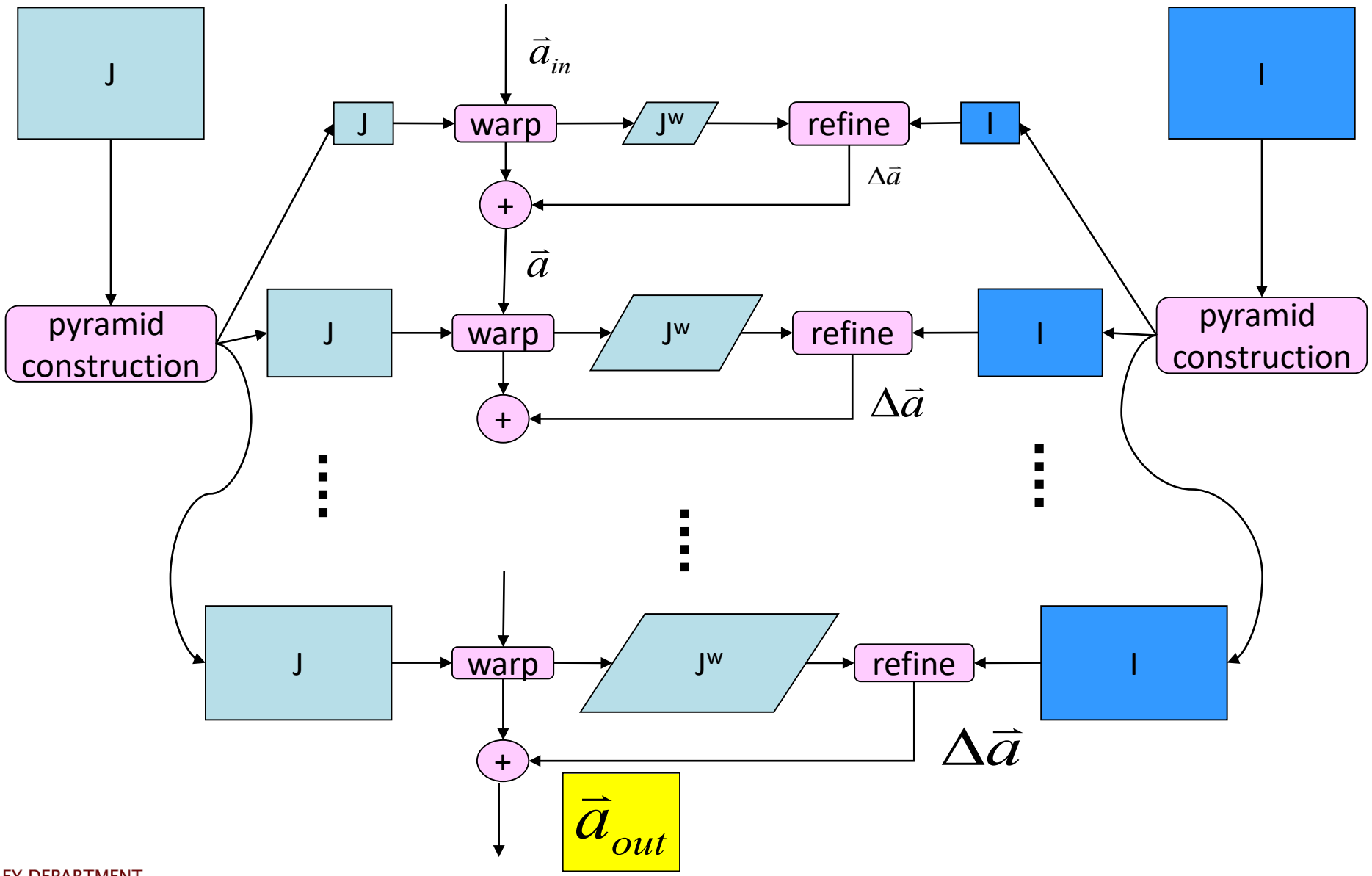
The Lucas-Kanade image alignment method:

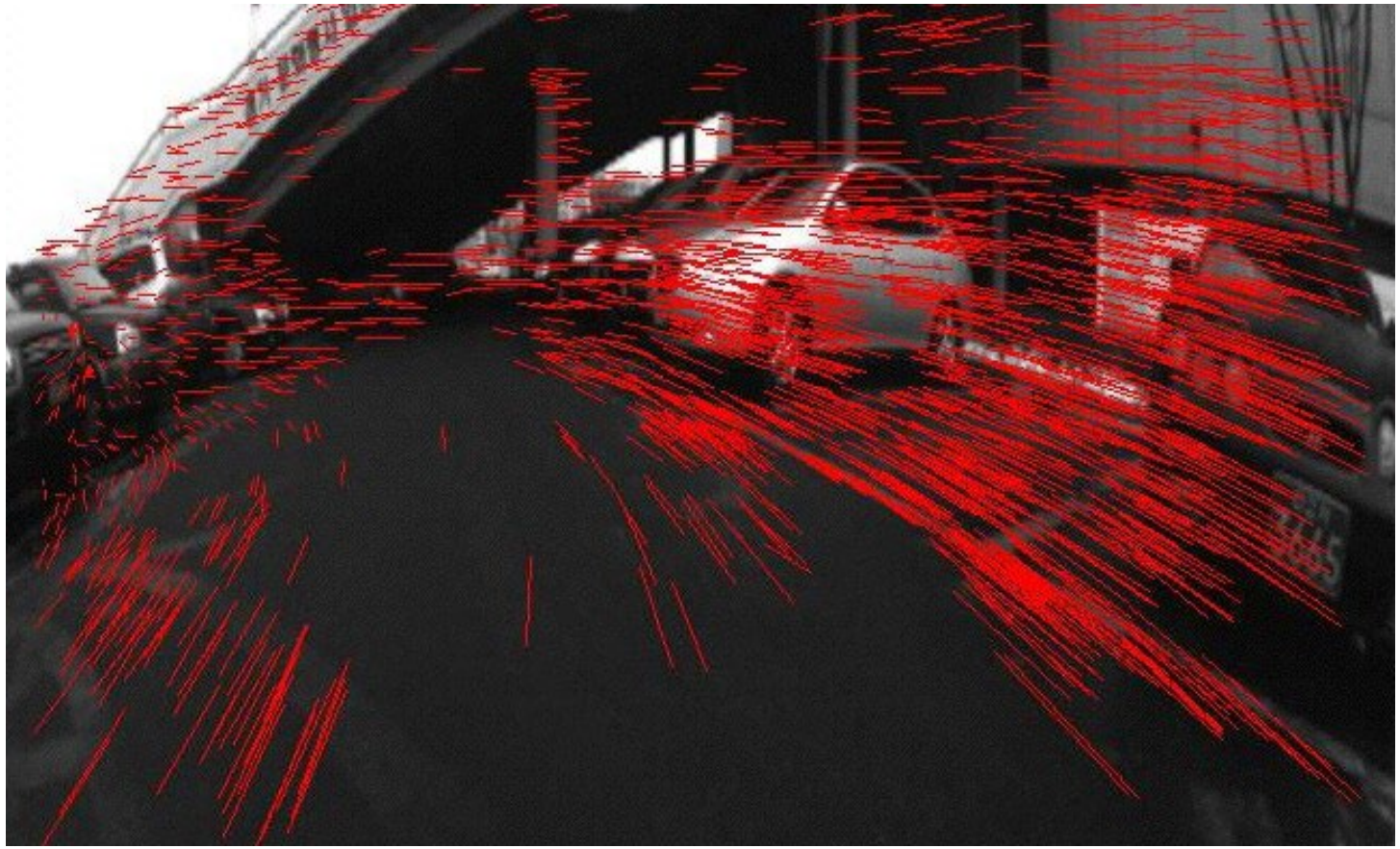
The error function, SSD (Sum of Squared Differences):	$\sum_x \left[I\left(W(x,p)\right) - T(x)\right]^2$
Incremental update:	$\sum_x \left[I\left(W(x,p+\Delta p)\right) - T(x)\right]^2$
Linearize:	$\sum_x \left[I\left(W(x,p)\right) + \nabla I \frac{\partial W}{\partial p}\Delta p - T(x)\right]^2$
Gradient update:	$\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[T(x) - I\left(W(x,p)\right)\right]$ $H = \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial p}\right]$
Update:	$p \leftarrow p + \Delta p$

Coarse-to-Fine Estimation



Coarse-to-Fine Estimation





Kanade-Lucas-Tomasi (KLT) Tracker

Many of these slides originated with Kris Kitani of Carnegie Mellon University

Feature-based tracking

- Up to now, we've been aligning entire images but we can also track just small image regions too!
- How should we select features?
- How should we track them from frame to frame?

Kanade-Lucas-Tomasi (KLT) Tracker

An Iterative Image Registration Technique with an Application to Stereo Vision.

1981

Detection and Tracking of Feature Points.

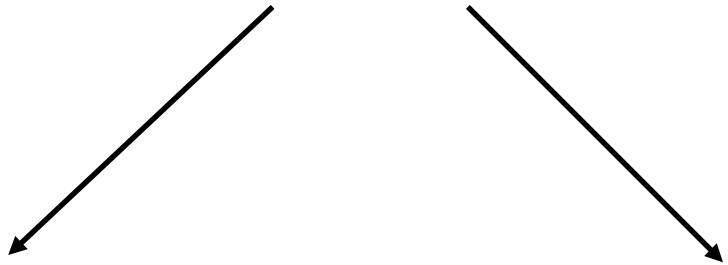
1991

The original KLT algorithm

Good Features to Track.

1994

Kanade-Lucas-Tomasi



How should we select features?

Tomasi-Kanade

Method for choosing the best feature (image patch) for tracking

How should we track them from frame to frame?

Lucas-Kanade

Method for aligning (tracking) an image patch

What are good features for tracking?

What are good features for tracking?

Intuitively, we want to avoid smooth regions and edges. But is there a more *principled* way to define good features?

What are good features for tracking?

Can be derived from the tracking algorithm

'A feature is good if it can be tracked well'

Recall the Lucas-Kanade image alignment method:

The error function, SSD (Sum of Squared Differences):	$\sum_x \left[I\left(W(x,p)\right) - T(x)\right]^2$
Incremental update:	$\sum_x \left[I\left(W(x,p+\Delta p)\right) - T(x)\right]^2$
Linearize:	$\sum_x \left[I\left(W(x,p)\right) + \nabla I \frac{\partial W}{\partial p}\Delta p - T(x)\right]^2$
Gradient update:	$\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[T(x) - I\left(W(x,p)\right)\right]$ $H = \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial p}\right]$
Update:	$p \leftarrow p + \Delta p$

Stability of gradient descent iterations depends on …

$$\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[T(x) - I\big(W(x,p)\big)\right]$$

The inversion of the Hessian matrix:

$$H = \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial p}\right]$$

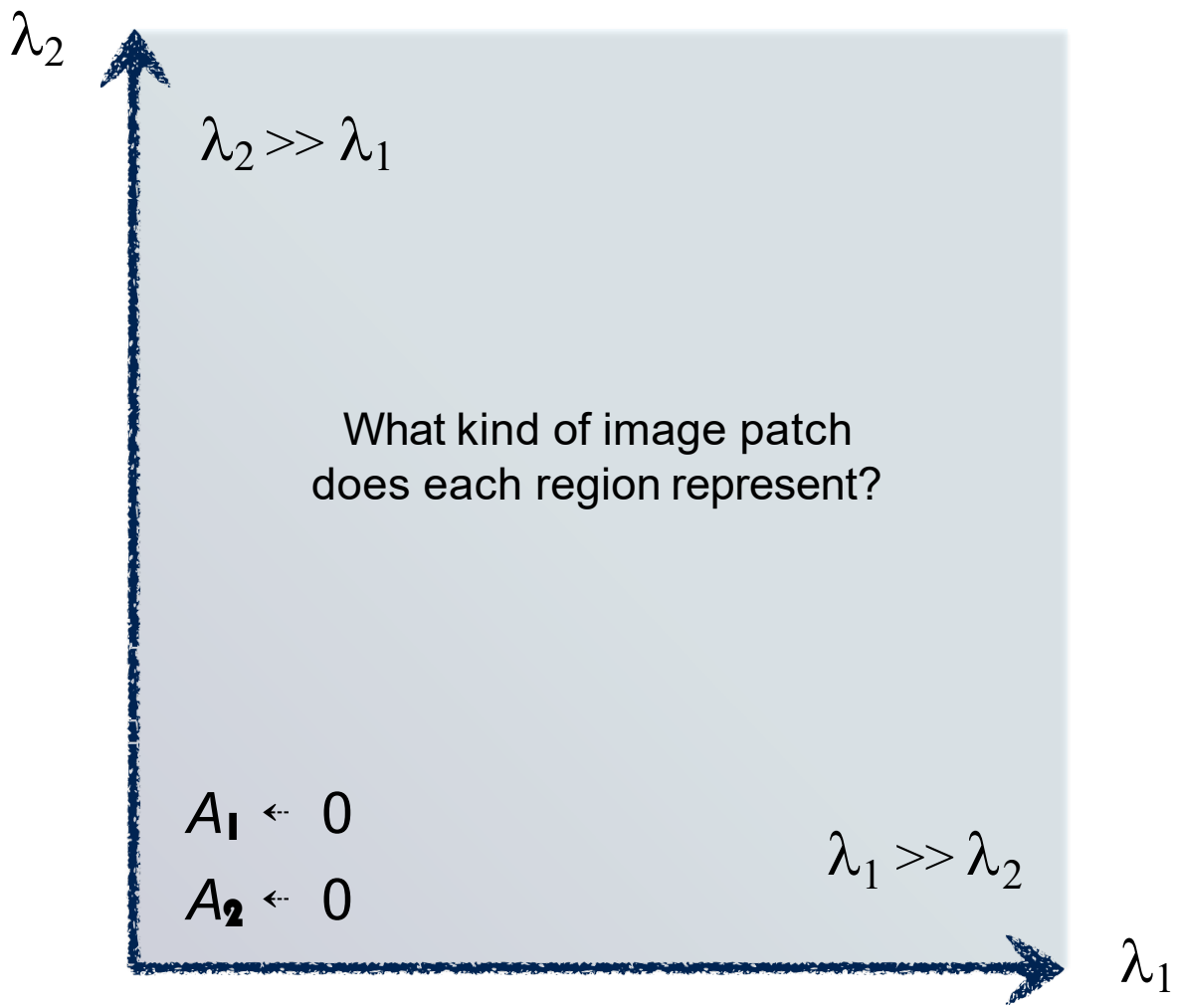
This inversion will fail, and the KLT update will be meaningless, when H is singular. What would cause this to happen?

When will the Hessian matrix $H = \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial p}\right]$ be singular?

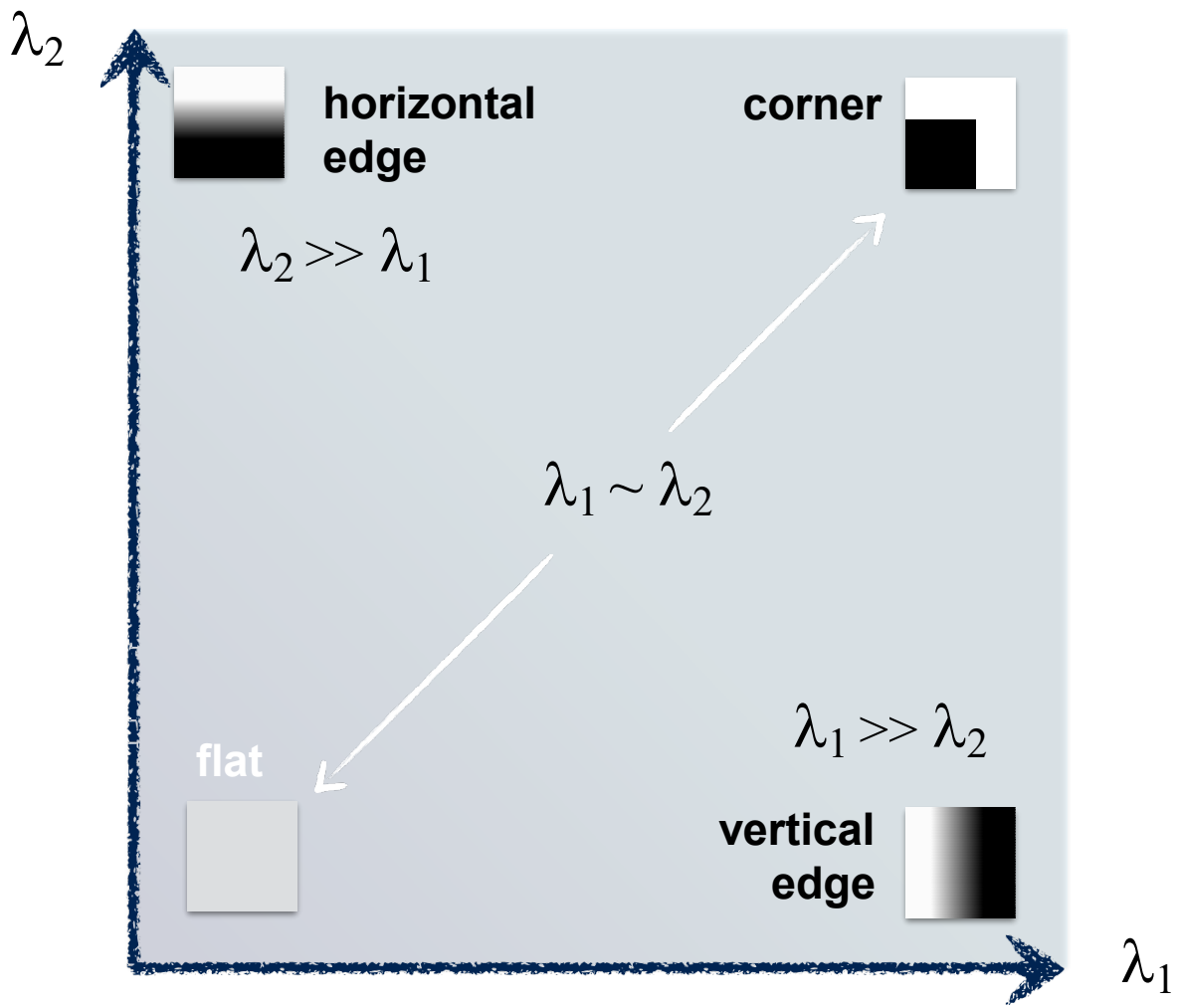
- When the gradient is only in one direction – along an edge
- Consider this case: $W(\boldsymbol{x}, p) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}; \frac{\partial W}{\partial p} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- The Hessian is: $H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial W}{\partial p}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial p}\right] = \sum_{\boldsymbol{x}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$H = \begin{bmatrix} \sum_{\boldsymbol{x}} I_x I_x & \sum_{\boldsymbol{x}} I_x I_y \\ \sum_{\boldsymbol{x}} I_y I_x & \sum_{\boldsymbol{x}} I_y I_y \end{bmatrix}$$

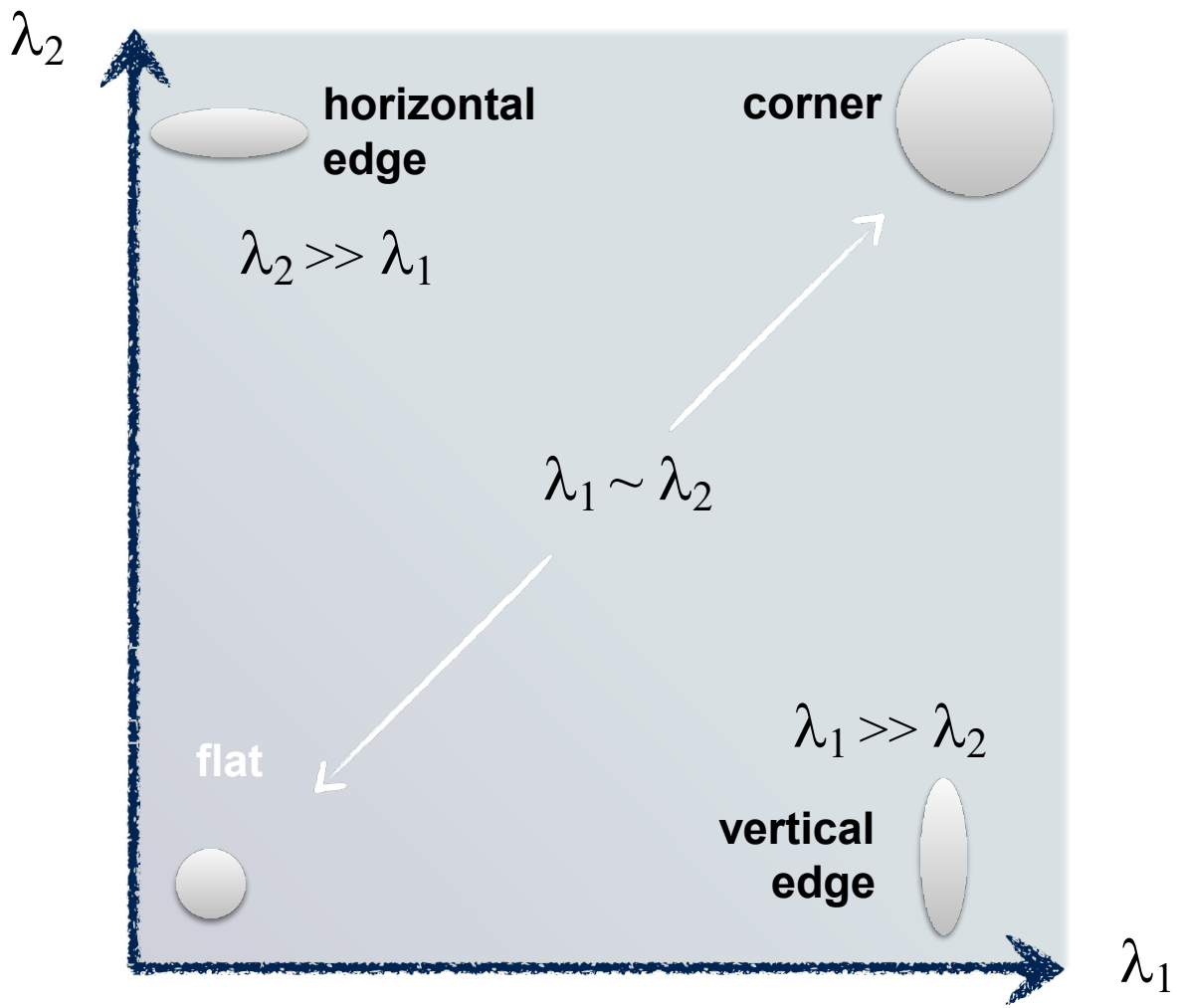
interpreting eigenvalues



interpreting eigenvalues



interpreting eigenvalues



What are good features for tracking?

What are good features for tracking?

$$min(\lambda_1, \lambda_2) > \lambda_{crit}$$

Locations with minimum eigenvalue greater than some “lowest” value represent keypoints of gradient change in more than one direction (like Shi-Tomasi corner detection)

The KLT algorithm for tracking keypoints

1. Find areas satisfying $min(\lambda_1, \lambda_2) > \lambda_{crit}$ (corners)

2. For each corner compute displacement to next image in the sequence, using the Lucas-Kanade method

3. Store displacement of each corner, update corner position
    1. Take note of points that leave the image

4. (optional) Add more corner points every M frames using step 1

5. Repeat 2 & 3 (and maybe 4) for each new image

We have found trajectories for each corner point

Mathematics of the KLT method

1. Find keypoints in image 1 using Shi-Tomasi: points where $min(\lambda_1, \lambda_2) > \lambda_{crit}$

2. For each image *t*:

   For each keypoint, located at (x,y) in image *t*:

   a) Initialize the location in image *t+1* as (x', y') = (x, y); use interpolation!

   b) Compute (u,v) by solving $\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$, where $I_t = I(x', y', t+1) - I(x, y, t)$

   c) Update (x', y') as $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix}$

   d) Recalculate I_t; use interpolation!

   e) Repeat steps b, c and d until there is little change in (x', y') – until (u,v) are small

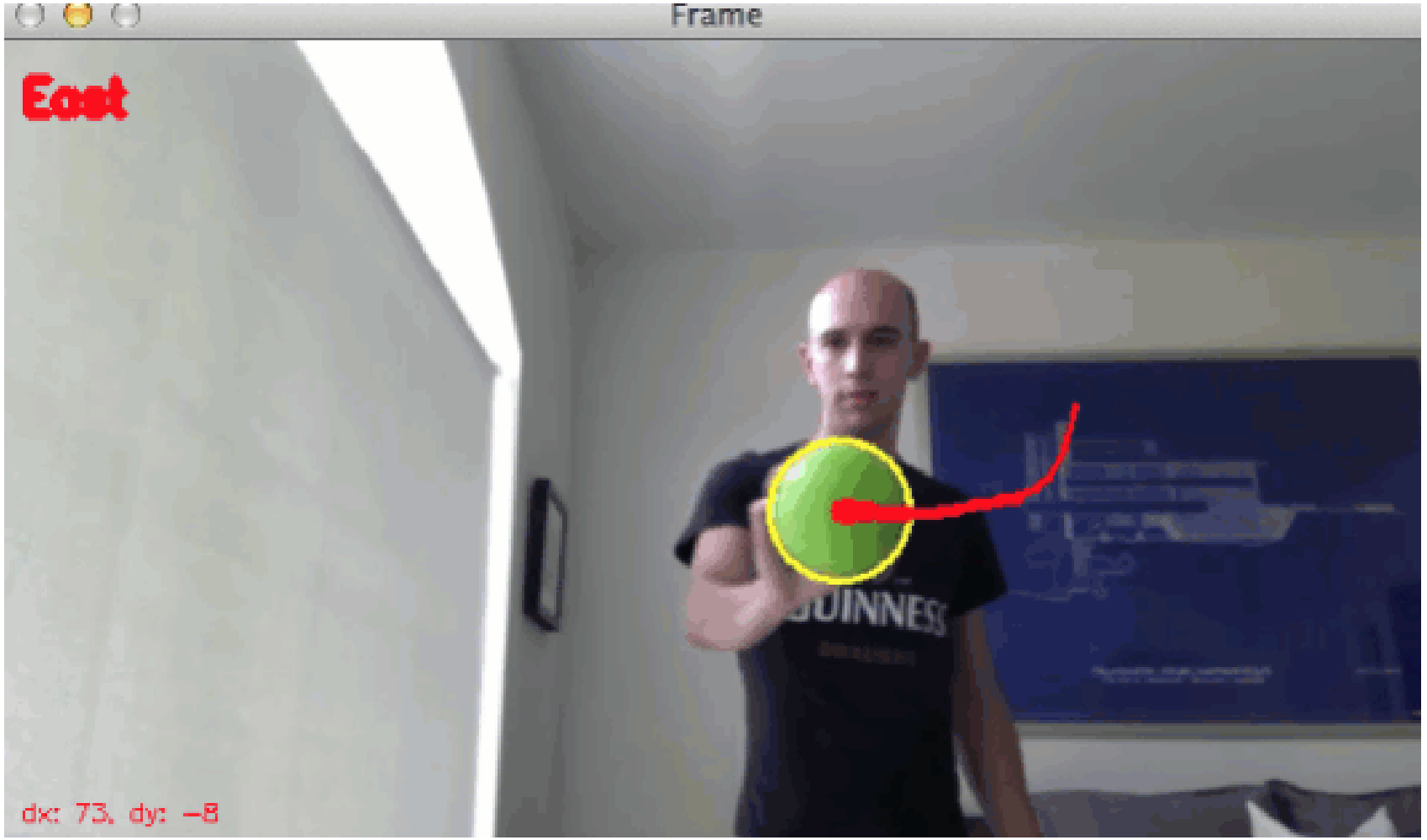
3. Store displacement of each corner, update corner position (x, y)
   Take note of points that leave the image

4. (optional) Add more corner points every M frames using step 1

5. Repeat 2 & 3 (and maybe 4) for each new image

This formulation of KLT assumes only translation between successive images

- The degree of rotation should be small
- More elaborate forms of image-to-image transformation can be handled
  - Rotation
  - Scale
  - Affine
- The Hessian matrix is larger in those cases
  - See https://www.crcv.ucf.edu/wp-content/uploads/2019/03/Lecture-10-KLT.pdf



Adrian Rosebrock, https://www.pyimagesearch.com/2015/09/21/opencv-track-object-movement/

Today's Objectives

- Motion Estimation – the concept
- Lucas-Kanade motion estimation
  - Optical Flow constraint
  - Gradient constraint
  - Solution
- Kanade-Lucas-Tomasi keypoint tracking
  - Ideal features for tracking
  - The KLT method