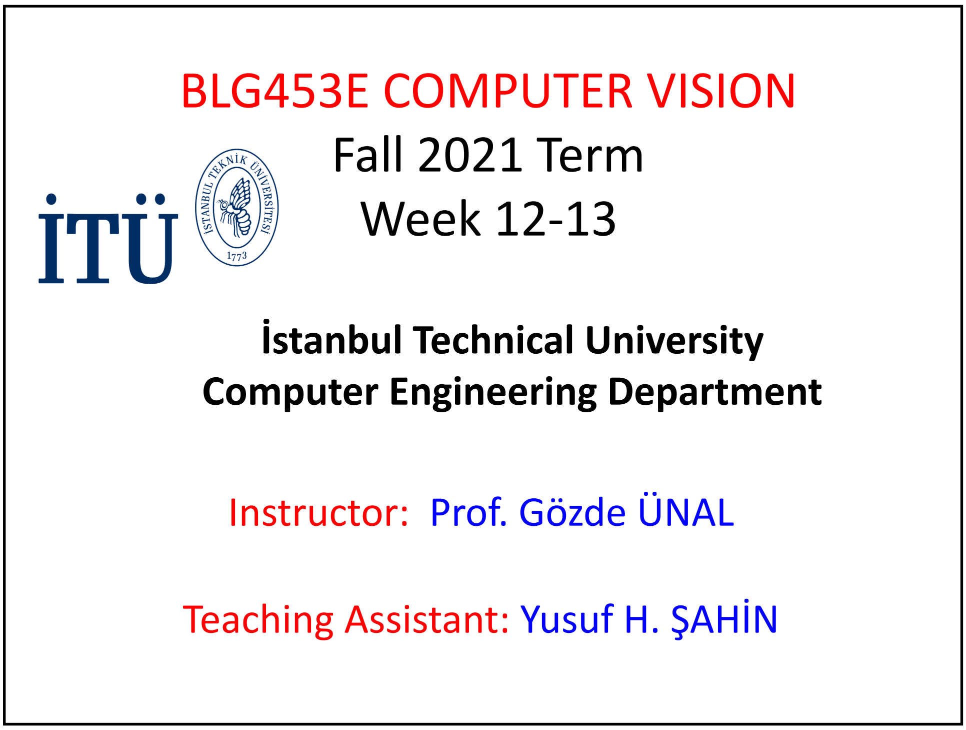


BLG453E COMPUTER VISION

Fall 2021 Term

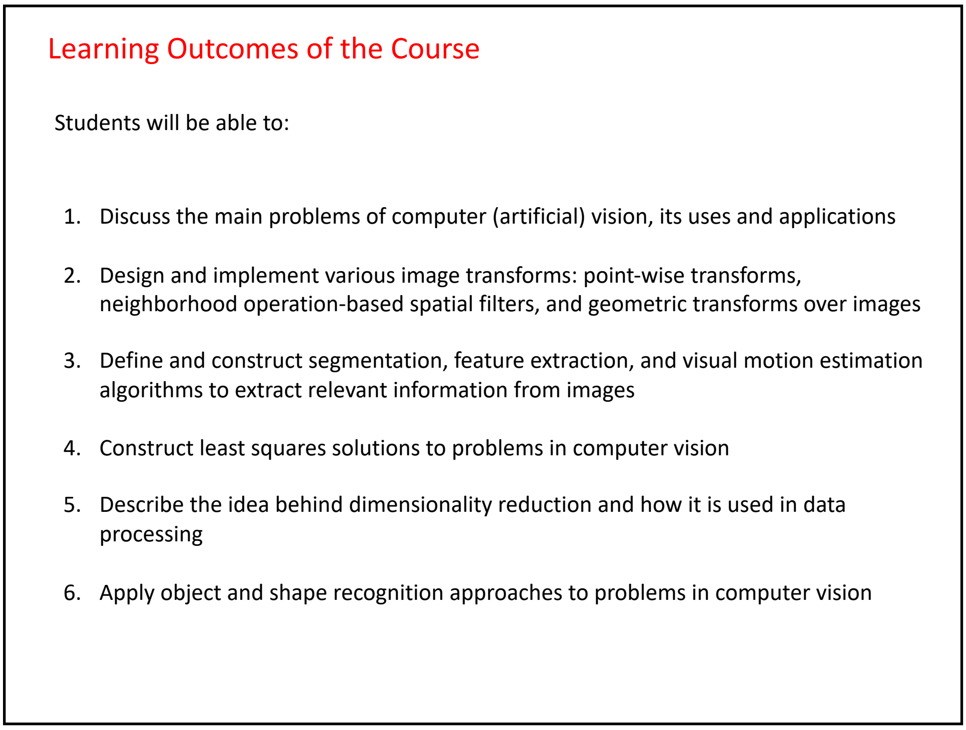
Week 12-13

İstanbul Technical University
Computer Engineering Department

Instructor: Prof. Gözde ÜNAL

Teaching Assistant: Yusuf H. ŞAHİN

1



Learning Outcomes of the Course

Students will be able to:

1. Discuss the main problems of computer (artificial) vision, its uses and applications
2. Design and implement various image transforms: point-wise transforms, neighborhood operation-based spatial filters, and geometric transforms over images
3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images
4. Construct least squares solutions to problems in computer vision
5. Describe the idea behind dimensionality reduction and how it is used in data processing
6. Apply object and shape recognition approaches to problems in computer vision

2

Week : Visual Motion Estimation

At the end of Week: Students will be able to:

LO 3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images

LO 4. Construct least squares solutions to problems in computer vision

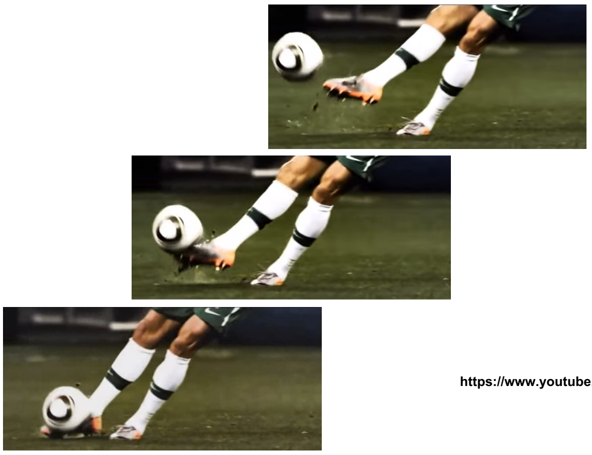
Overview of Today and Next Week:

1. Visual Motion Detection
2. Visual Motion (Optical Flow) Estimation

3

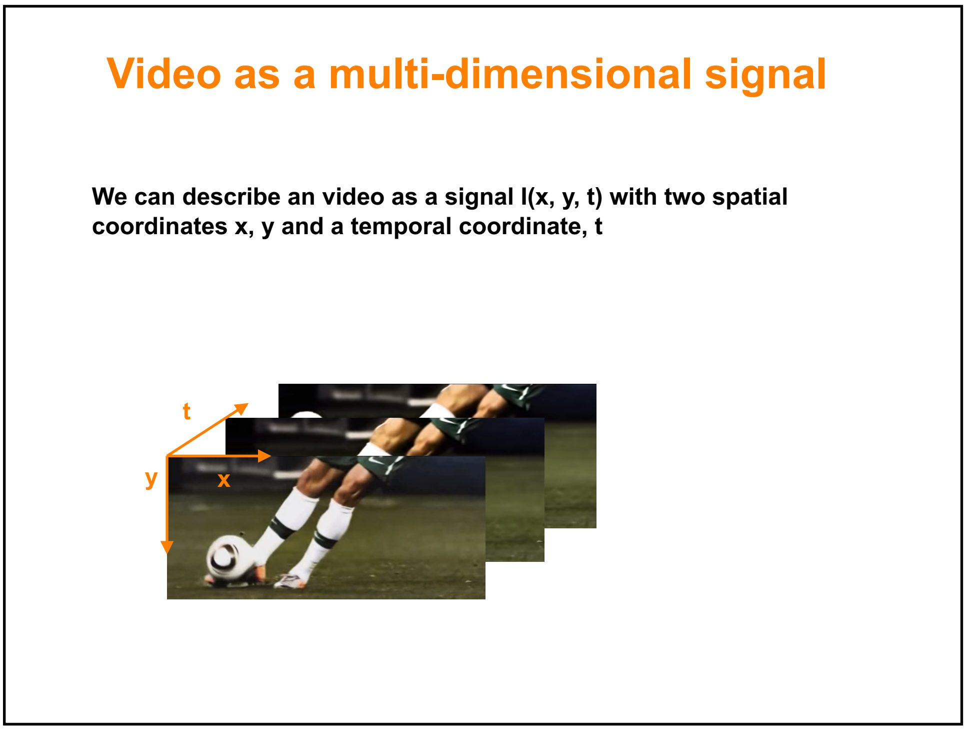
Video

Video is simply a series of images over time.



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

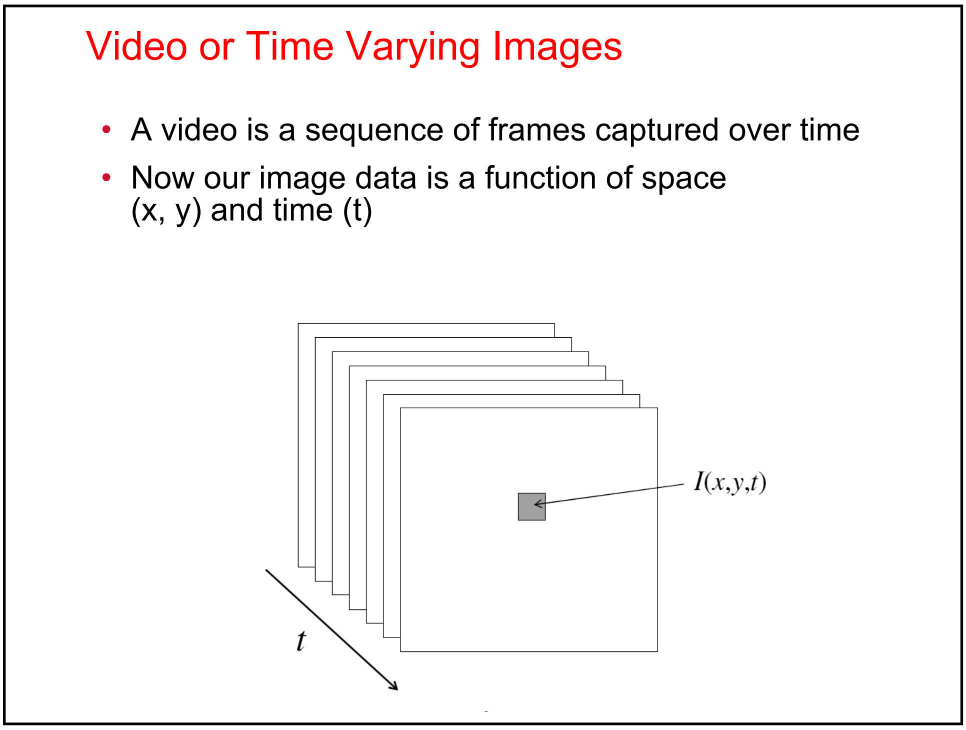
4



Video as a multi-dimensional signal

We can describe an video as a signal I(x, y, t) with two spatial coordinates x, y and a temporal coordinate, t

5



Video or Time Varying Images

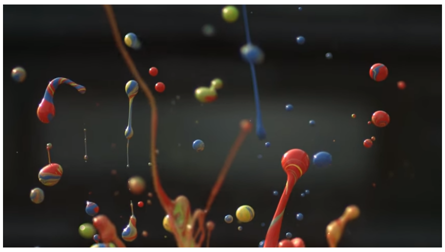
- A video is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and time (t)

6

Framerate

One of the defining characteristics of video is the framerate, usually measured in frames per second (FPS), or Hertz (Hz). Common framerates:

- 24 Hz: Traditional film
- 25 Hz: PAL format
- 50/60 Hz: Used in high end TVs. Supported by YouTube as well. Most modern cameras can record video at this rate.
- Higher framerates possible (depending on hardware):



Min: 3:04-3:40

Slow mo guys

https://www.youtube.com/watch?v=5WKU7gG_ApU

7

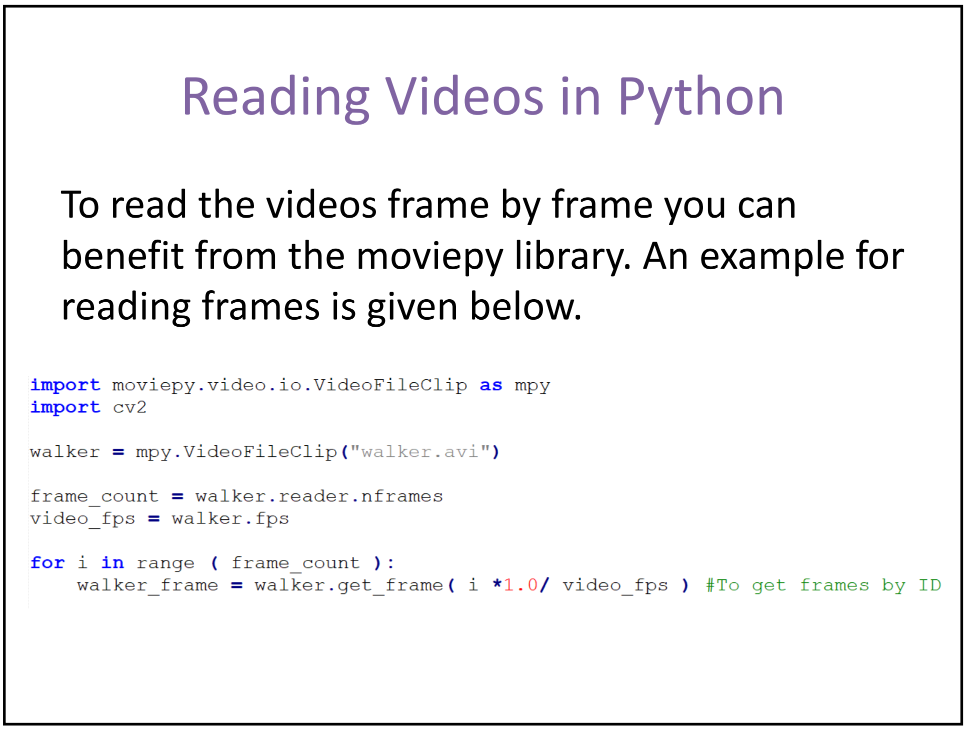
Dynamic Scene Analysis

- Motion: A powerful cue used by humans and animals to extract objects or regions of interest from a background

- Motion in imaging: arises from relative displacement between the sensing system and the scene being viewed

Motion can be used for Scene Segmentation; Tracking Objects; for Extracting Higher-level knowledge from the scene, …

8



Reading Videos in Python

To read the videos frame by frame you can benefit from the moviepy library. An example for reading frames is given below.

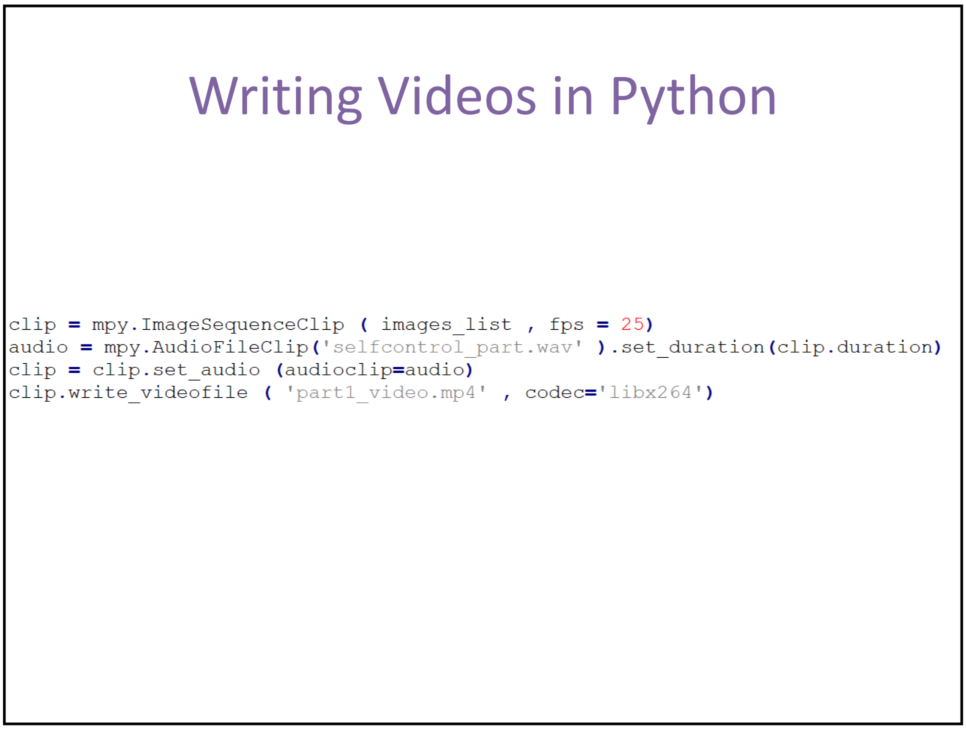
```
import moviepy.video.io.VideoFileClip as mpy
import cv2

walker = mpy.VideoFileClip("walker.avi")

frame_count = walker.reader.nframes
video_fps = walker.fps

for i in range ( frame_count ):
    walker_frame = walker.get_frame( i *1.0/ video_fps ) #To get frames by ID
```

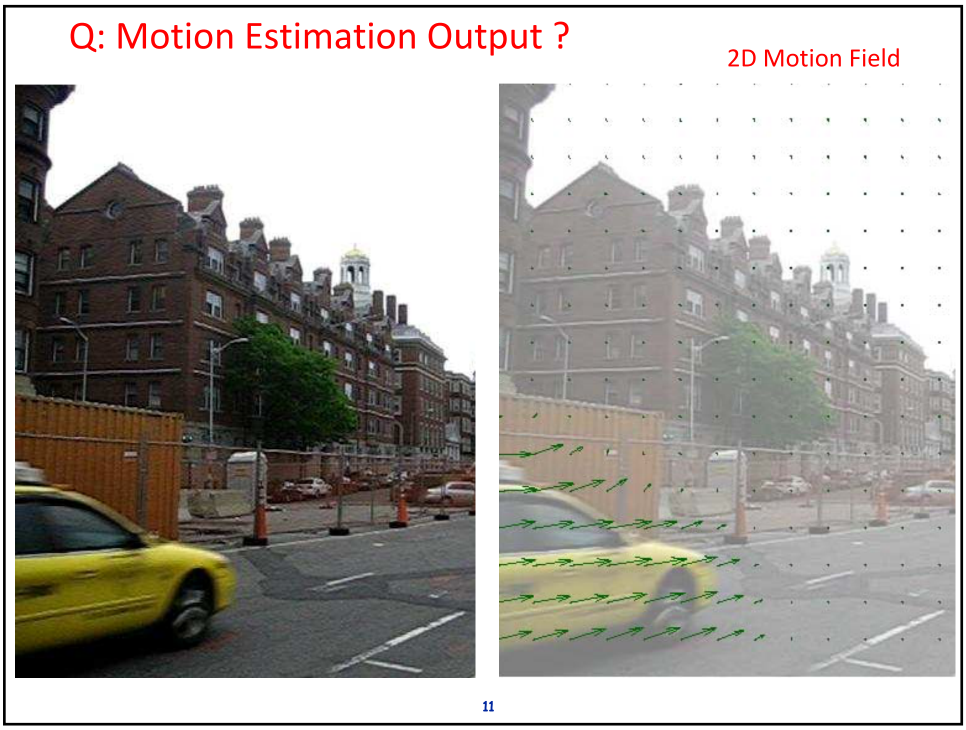
9



Writing Videos in Python

```
clip = mpy.ImageSequenceClip ( images_list , fps = 25)
audio = mpy.AudioFileClip('selfcontrol_part.wav' ).set_duration(clip.duration)
clip = clip.set_audio (audioclip=audio)
clip.write_videofile ( 'part1_video.mp4' , codec='libx264')
```

10

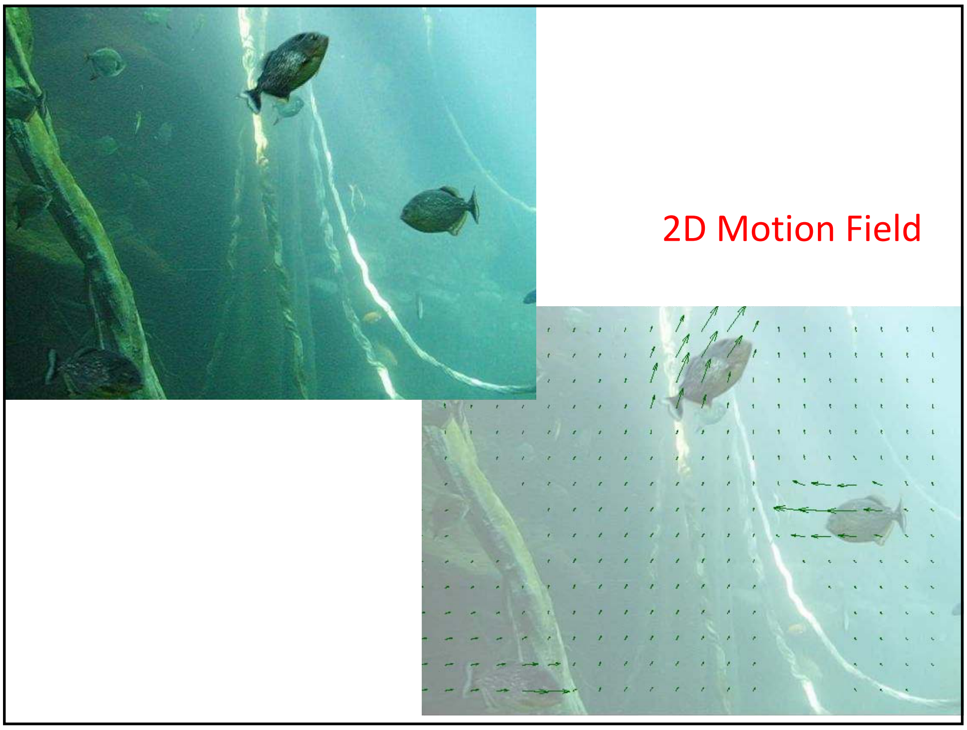


Q: Motion Estimation Output ?

2D Motion Field

11

11

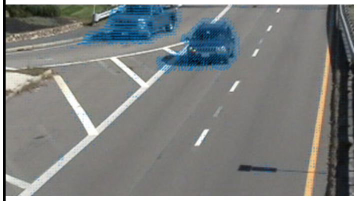


2D Motion Field

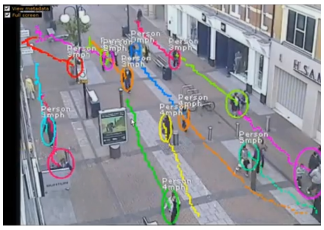
12

Video analysis

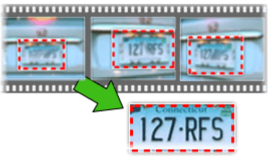
One can apply computer vision techniques to video to solve a variety of problems



Motion estimation



Tracking
https://www.youtube.com/watch?v=d-RCKfVjFI4



Super-resolution

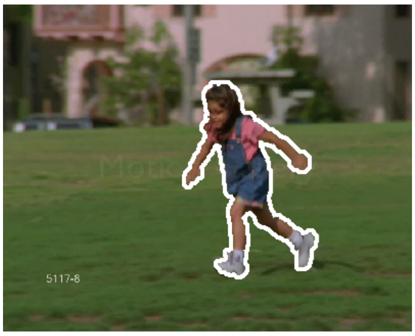
Dr. Greg Slabaugh

13

Motion Segmentation

Input video

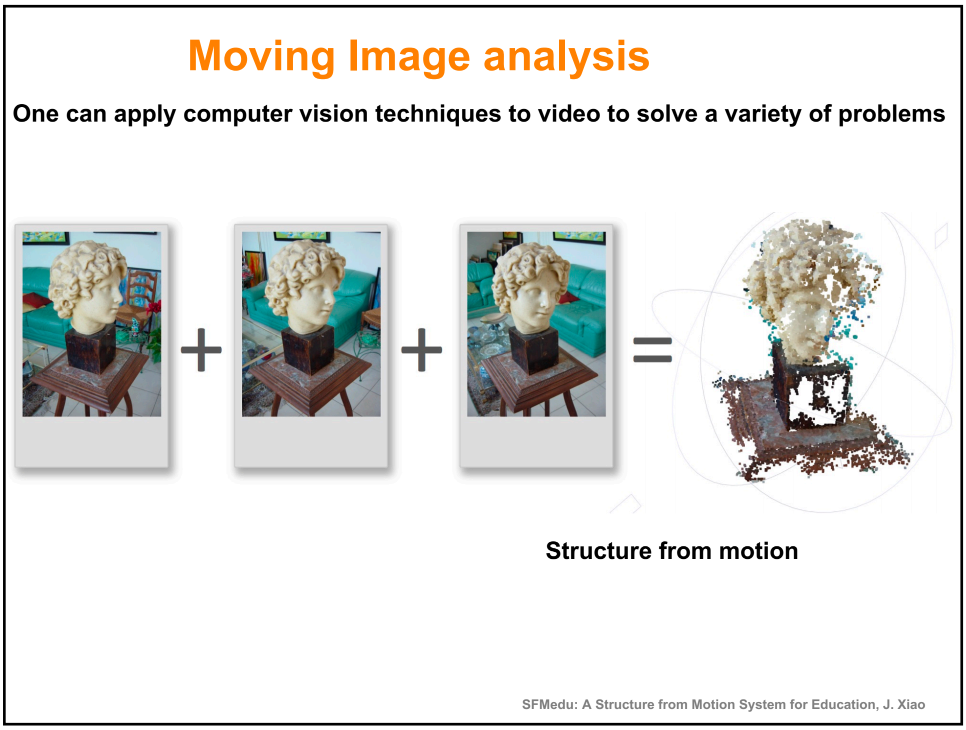
Output



- Automatic video object cut-out
- Offline analysis with minimum user input
- Behavior analysis via post-processing

(Tsai, Flagg, Rehg) Gatech

14



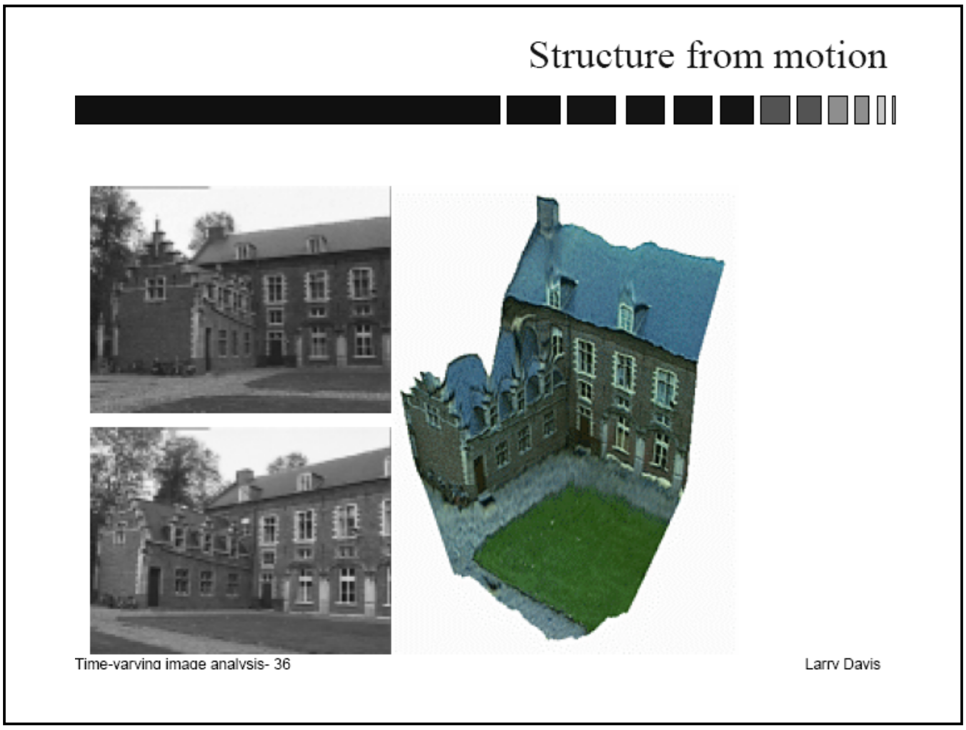
Moving Image analysis

One can apply computer vision techniques to video to solve a variety of problems

Structure from motion

SFMedu: A Structure from Motion System for Education, J. Xiao

15



Structure from motion

Time-varying image analysis- 36

Larry Davis

16

More Applications of Motion Estimation and Detection

* Visual Surveillance: stationary camera watches a workspace -find moving objects and alert an operator

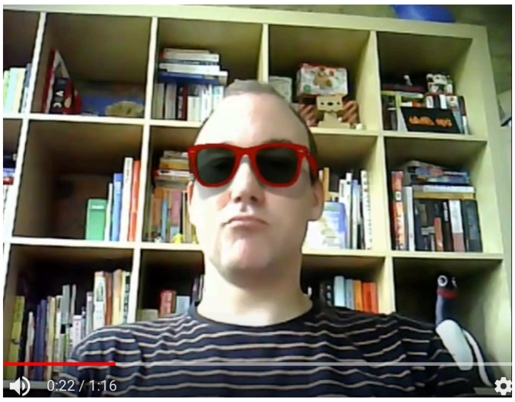
* Video Coding: use image motion to perform more efficient coding of images

* SLAM / Navigation: moving camera navigates a workspace

- camera moves through the world - estimate its trajectory » use this to control the movement of a robot through the world;
- » use this to remove unwanted jitter from image sequence – image stabilization and mosaicking

Augmenting a video

Place synthetic objects into a video stream by processing *each video frame*.



https://www.youtube.com/watch?v=Ag7H4YScqZs

How can we estimate motion from images?

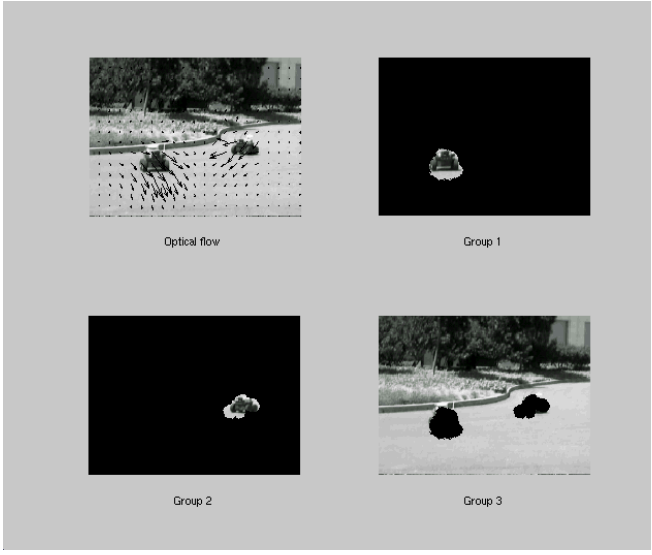
Flower Garden image sequence

http://i21www.ira.uka.de/image_sequences/

University of Karlsruhe: Image Sequences publicly available:

19

Motion Estimation and Motion Segmentation Example



Optical flow

Group 1

Group 2

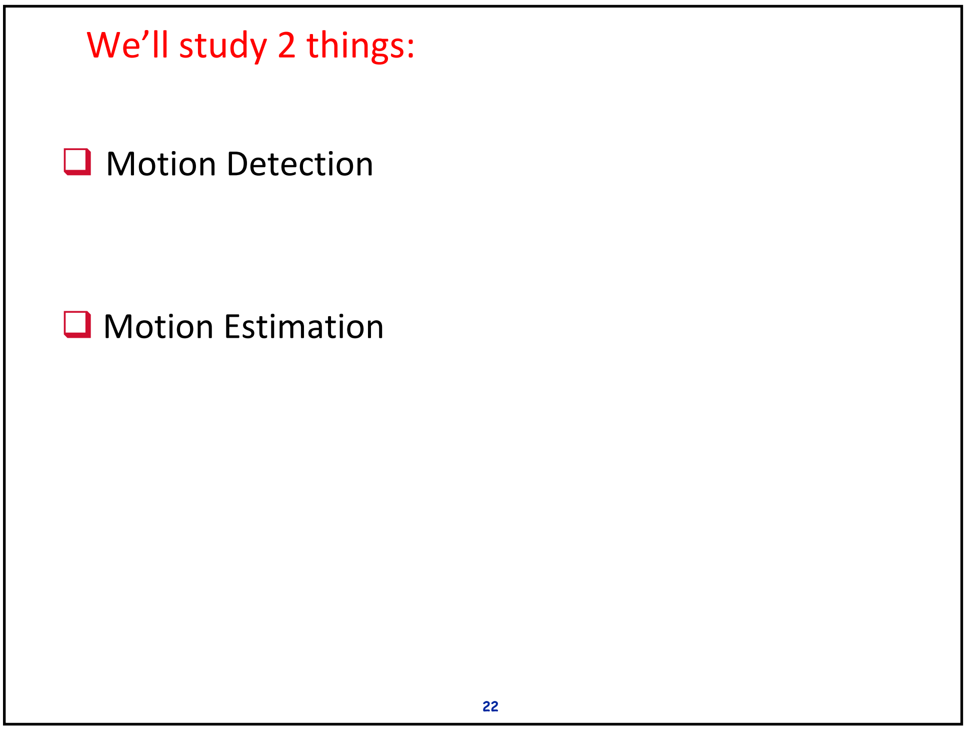
Group 3

Figure from book: An Invitation to 3D vision

20

MASKS © 2004

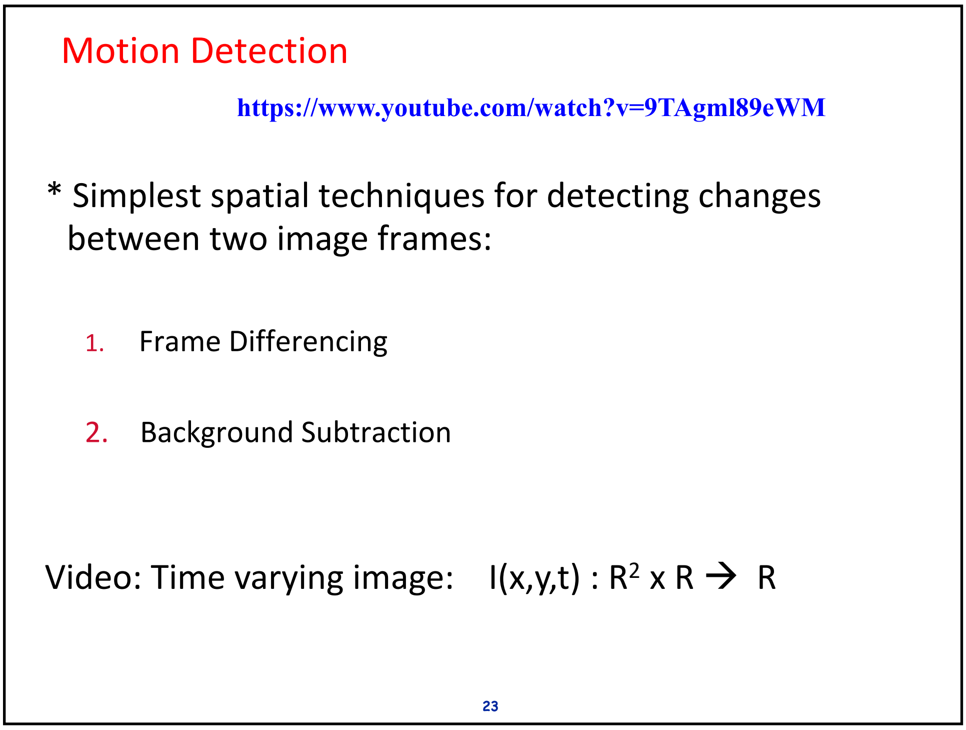
20



We'll study 2 things:

❑ Motion Detection

❑ Motion Estimation



Motion Detection

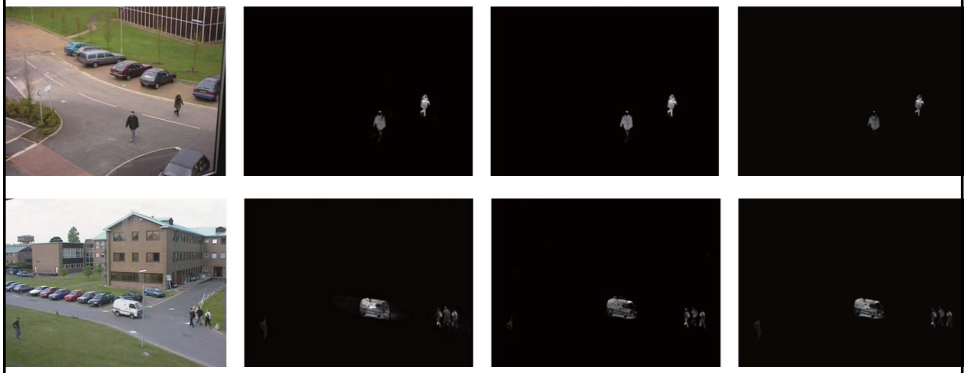
https://www.youtube.com/watch?v=9TAgml89eWM

* Simplest spatial techniques for detecting changes between two image frames:

1. Frame Differencing
2. Background Subtraction

Video: Time varying image: I(x,y,t) : R^2 x R → R

Motion Detection based on Temporal Differencing



Method calculates the differences of adjacent frames, also a background difference, details of how those are combined are in the following paper:

Figures from the Article: Motion Saliency Detection based on Temporal Difference, Z. Wang, J. Xiong, Q. Zhang, J. Electronic Imaging, 24(3), 2015. http://electronicimaging.spiedigitallibrary.org/article.aspx?articleid=2342756

Frame Differencing: How to choose T?

$$d_{ij}(x,y)=\begin{cases}1 & if\left|I(x,y,t_i)-I(x,y,t_j)\right|>T\\ 0 & otherwise\end{cases}$$

* T must be chosen so that the probability of mistaking differences due to noise for real motion is small

A simple method to find T:

Acquire consecutive images of a static scene, with no illumination changes, and look at the histogram of the difference images

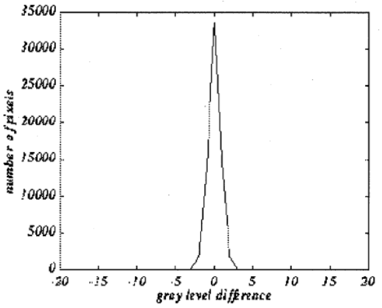


Figure 8.11 Two sample snapshots, taken at close instants, from an overnight, indoors surveillance sequence, and the histogram of the differences of the gray values between the two images.

* Assumption: difference is mainly due to camera noise: Histogram should look like a zero-mean Gaussian

* Choose the threshold T as a multiple of the standard deviation

Trucco, Verri, Book: "Introductory Techniques for 3D Computer Vision", Chapter 8.6 Motion Segmentation

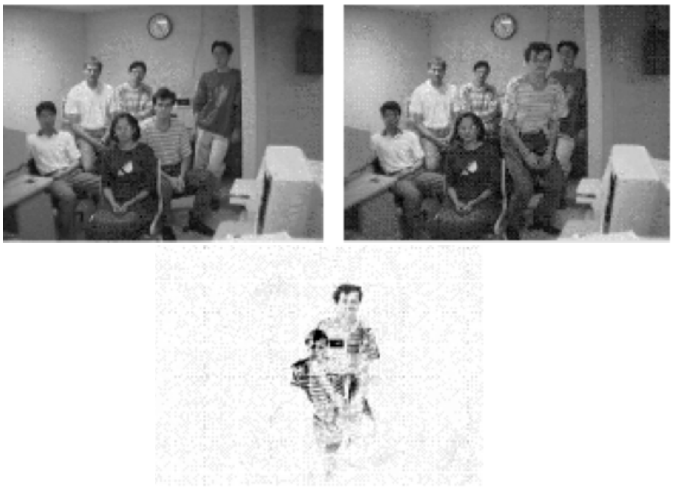
THQ: Simple Image Differencing

Simple image differencing tells/gives you which of the following?

i. If the image changed

ii. Where the image changed

iii. Motion vector



Images: Bryan Morse, CS650

Motion Detection: Frame Differencing

THQ: Limitations

Q: What are Limitations to this approach ?

Dynamic background, camera motion, shadows, changes in scene (weather, illumination), clutter

- ❑ Implicit assumptions:
  - ❑ Static camera or images are registered spatially
  - ❑ Illumination stays relatively constant

Non-zero-valued entries in difference image d_{ij} may also result from noise including camera noise

* Typically they are isolated points and can be removed by a connected comp. analysis

Background Subtraction to detect moving objects

https://www.youtube.com/watch?v=T-L9FoH3D9w

- If we know what the background looks like, it is easy to identify “interesting bits”
- Idea: use a temporal moving average to estimate background (or reference) image which has stationary components

  * Compare this frame with (i.e. subtract) subsequent frames including a moving object
  * Cancels out stationary elements
  * Large absolute values are “interesting pixels”

- Applications
  - Monitoring in an office
  - Tracking cars on a road
  - Surveillance

Check out OpenCV tutorial: http://docs.opencv.org/trunk/d1/dc5/tutorial_background_subtraction.html

Background subtraction

- A classic computer vision problem is to detect change in a video. A simple way to achieve this is to use a background image. One can compute the difference between the image and background, and threshold the result.
- This identifies pixels whose colour has changed.

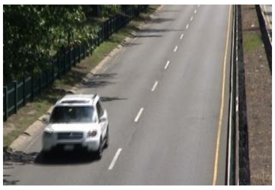
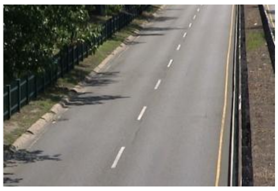
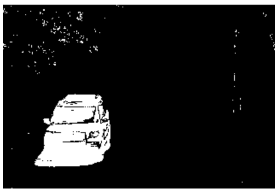


Image (I)

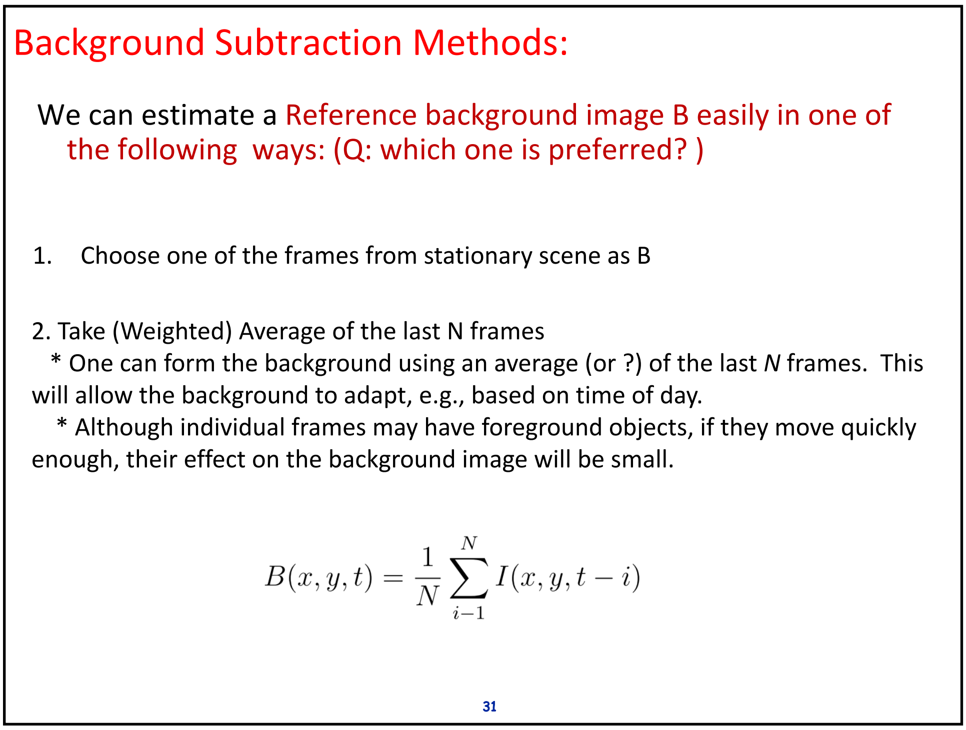

Background (B)


Thresholded difference (M)

$$F(x,y,t) = I(x,y,t) - B(x,y)$$

$$M(x,y,t) = \begin{cases} 1, & |F(x,y,t)| > T \\ 0, & \text{otherwise} \end{cases}$$

http://wordpress-jodoin.dmi.usherb.ca/dataset2014/



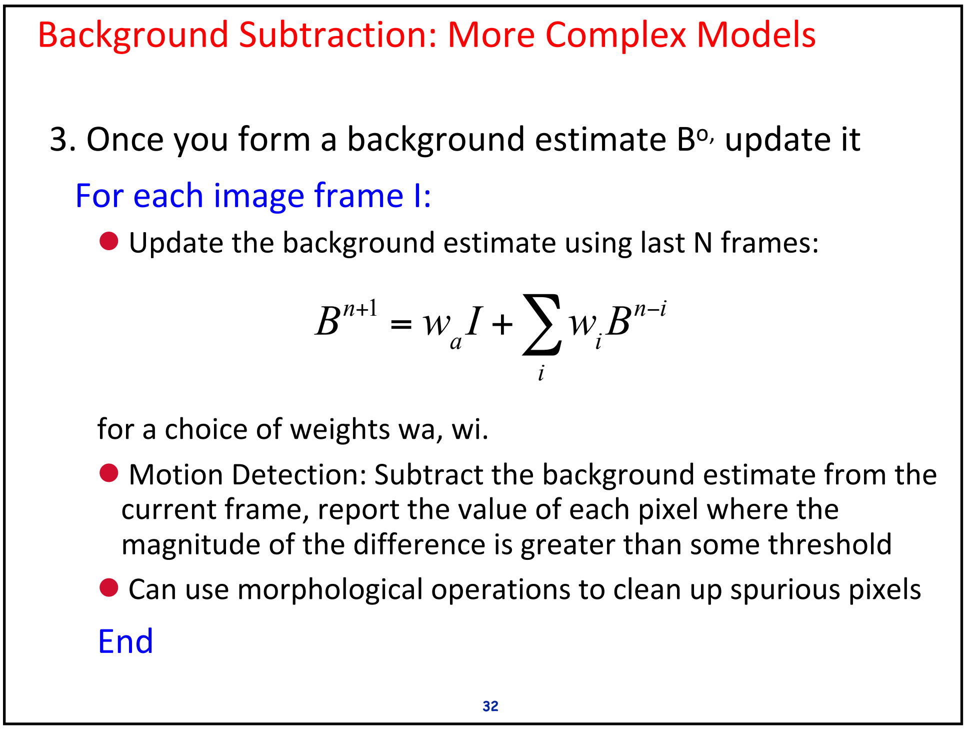
Background Subtraction Methods:

We can estimate a Reference background image B easily in one of the following ways: (Q: which one is preferred?)

1. Choose one of the frames from stationary scene as B

2. Take (Weighted) Average of the last N frames
* One can form the background using an average (or ?) of the last *N* frames. This will allow the background to adapt, e.g., based on time of day.
* Although individual frames may have foreground objects, if they move quickly enough, their effect on the background image will be small.

$$B(x,y,t) = \frac{1}{N}\sum_{i-1}^{N} I(x,y,t-i)$$



Background Subtraction: More Complex Models

3. Once you form a background estimate B^{o,} update it

For each image frame I:

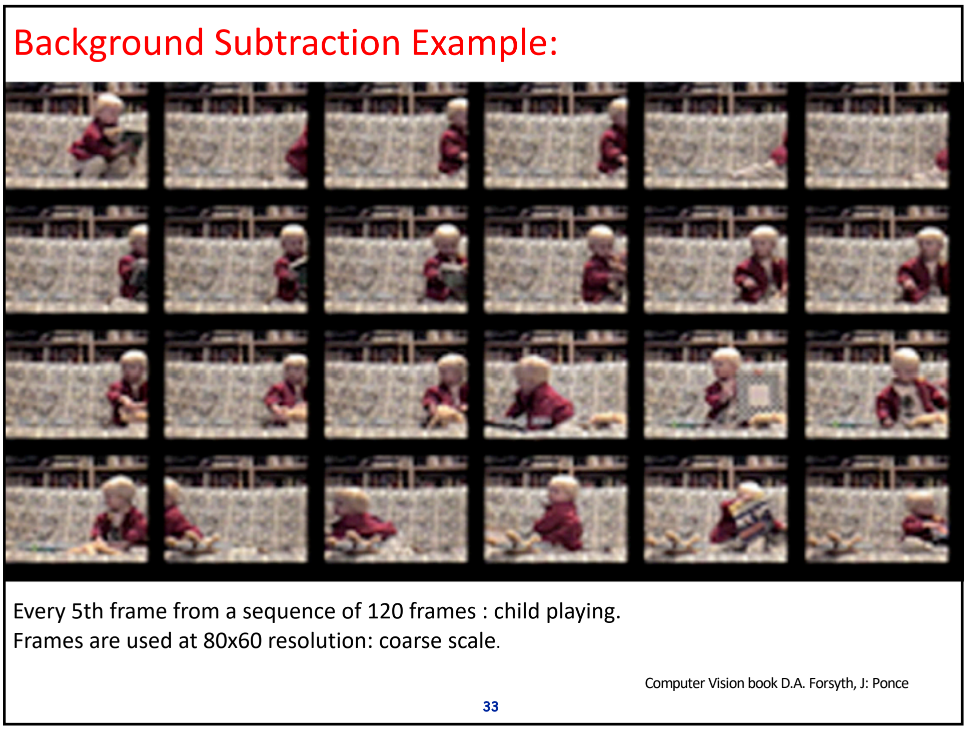
- Update the background estimate using last N frames:

$$B^{n+1} = w_a I + \sum_i w_i B^{n-i}$$

for a choice of weights wa, wi.

- Motion Detection: Subtract the background estimate from the current frame, report the value of each pixel where the magnitude of the difference is greater than some threshold
- Can use morphological operations to clean up spurious pixels

End



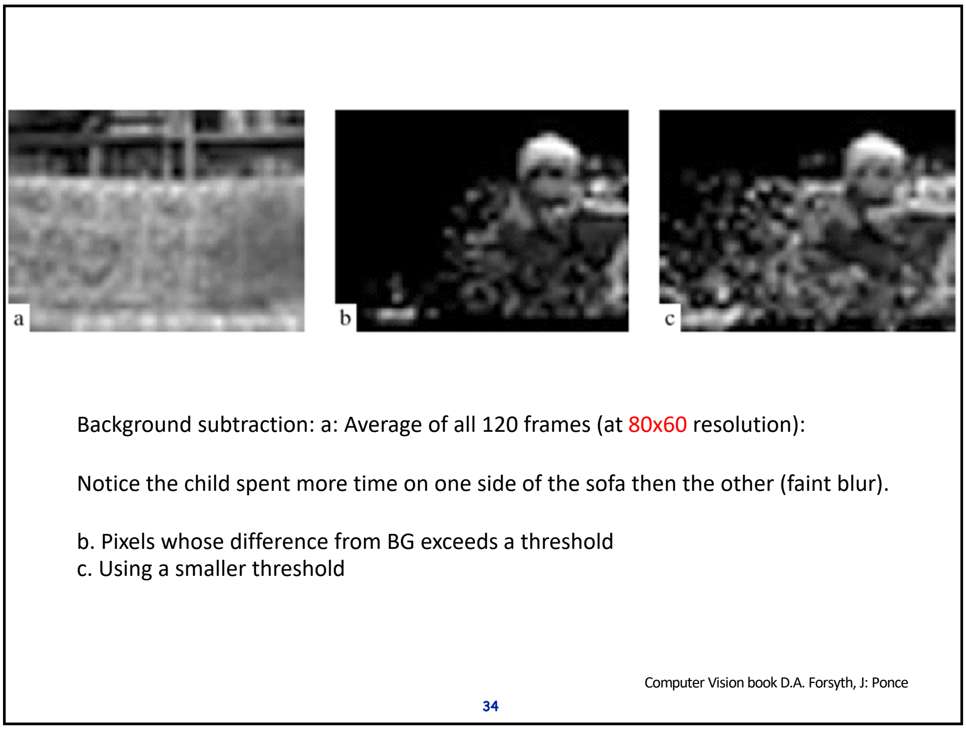
Background Subtraction Example:

Every 5th frame from a sequence of 120 frames : child playing.
Frames are used at 80x60 resolution: coarse scale.

Computer Vision book D.A. Forsyth, J: Ponce

33

33



Background subtraction: a: Average of all 120 frames (at 80x60 resolution):

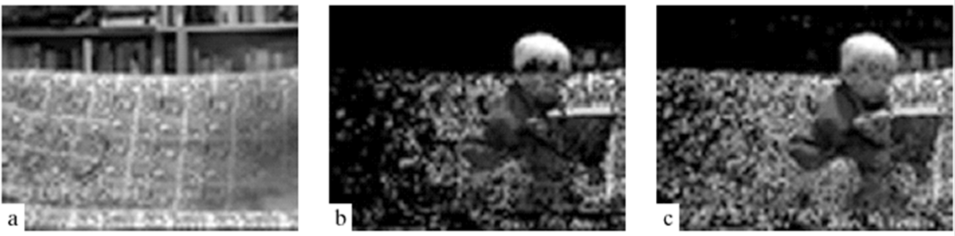
Notice the child spent more time on one side of the sofa then the other (faint blur).

b. Pixels whose difference from BG exceeds a threshold
c. Using a smaller threshold

Computer Vision book D.A. Forsyth, J: Ponce

34

34



Background subtraction: Same as the previous slide, but this time using 160x120 frames
Notice that the pattern on the sofa has been mistaken for the child,
and this has markedly increased at high resolution.

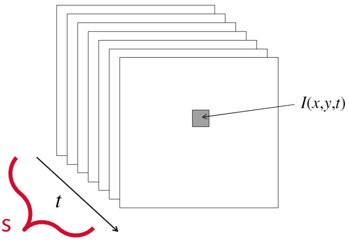
Why ? An important point here is that background subtraction works quite badly in the presence of high spatial frequencies, because when we estimate the background, we’re almost always going to smooth it.

Computer Vision book D.A. Forsyth, J: Ponce

Background Subtraction:

Idea: Relative to the consistent background, the moving objects are just temporal outliers, so: Use median filter across time

$$R(x,y) = \text{Median } \{I(x,y,t\_0), I(x,y,t\_1),...,I(x,y,t\_N)\}$$



Take the pixelwise median over last N frames
to create the background image

4-> Take result of temporal median filtering (preferred over a temporal average to prevent outliers)

Background Subtraction: More complex models

5. Per-pixel Gaussian fitting:

* For *each pixel*, one can determine the mean $\mu(x,y,t)$ and standard deviation $\sigma(x,y,t)$ of the intensity (or colour) based on the last *N* frames.

* A pixel can then be classified as foreground if its value lies outside some confidence interval of the mean.

$$M(x,y,t) = \begin{cases} 1, & \frac{|I(x,y,t)-\mu(x,y,t)|}{\sigma(x,y,t)} > k \\ 0, & \text{otherwise} \end{cases}$$

Note: there are fast ways to incrementally update the mean and standard deviation with each new frame.

Q: Use a single Gaussian or multiple?

Using Gaussian Mixture Models (GMMs): This combines K (e.g. K= 5) Gaussians to model the intensity at a pixel through time. A pixel is compared to the Gaussians, and the best matching Gaussian adapts

Background Subtraction: Cumulative Differences

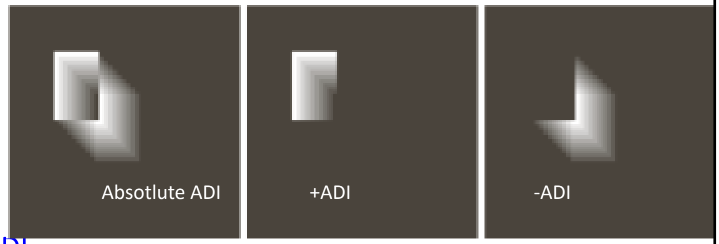
- ❑If one can obtain a Reference image, say R(x,y), then can look at Accumulative Difference Image (ADI):

$$A_k(x,y) = \begin{cases} A_{k-1}(x,y)+1 & if\,|R(x,y)-I(x,y,k)| > T \\ A_{k-1}(x,y) & otherwise \end{cases}$$

- ❑The entry at a pixel at frame k then corresponds to # times its intensity was different from the reference image

Background Subtraction: Cumulative Differences

Suppose a rectangular object is moving in South-East direction:



+ADI: Positive ADI

$$P_k(x,y)=\begin{cases} P_{k-1}(x,y)+1 & if\left(R(x,y)-I(x,y,k)\right)>T \\ P_{k-1}(x,y) & otherwise \end{cases}$$

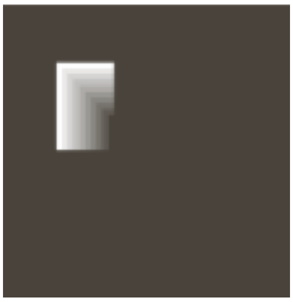
-ADI: Negative ADI

$$N_k(x,y)=\begin{cases} N_{k-1}(x,y)+1 & if\left(I(x,y,k)-R(x,y)\right)>T \\ N_{k-1}(x,y) & otherwise \end{cases}$$

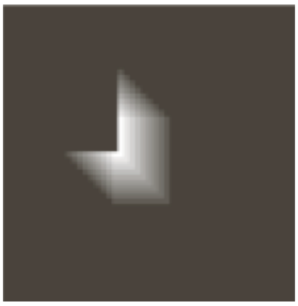
Digital Image Processing, Gonzalez & Woods

THQ: Cumulative differences in background modeling

+ADI



-ADI



Match

+ADI or –ADI gives you a way to estimate initial position and size of a moving object?

+ADI or –ADI gives you the direction of the moving object?

Background Subtraction: Cumulative

❑ Considering positive and negative ADIs:

+ADI: +ADI: gives size of moving object, its initial position

$$P_k(x,y)=\begin{cases}P_{k-1}(x,y)+1 & if\big(R(x,y)-I(x,y,k)\big)>T\\ P_{k-1}(x,y) & otherwise\end{cases}$$

-ADI: gives the direction

- ADI:

$$N_k(x,y)=\begin{cases}N_{k-1}(x,y)+1 & if\big(I(x,y,k)-R(x,y)\big)>T\\ N_{k-1}(x,y) & otherwise\end{cases}$$

❑ Absolute ADI has both

Cumulative Difference Images

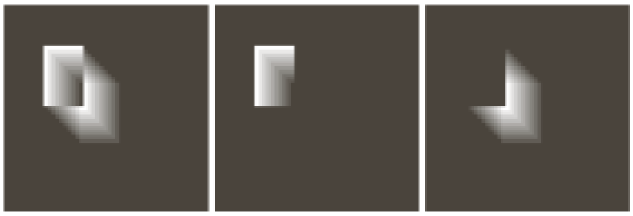


FIGURE 10.59 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.

*** +ADI has no change anymore = the object has completely displaced w.r.t. its initial position

Digital Image Processing, Gonzalez & Woods

Background Removal

- ❑ Establishing a Reference Image in practice needed when there are multiple moving objects in busy scenes
- ❑ Goal: Remove the principal moving objects in the scene
- ❑ Note that frequent update is required
- ❑ Idea: Use ADIs

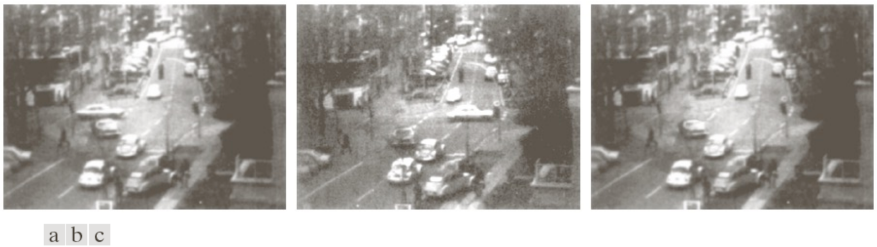
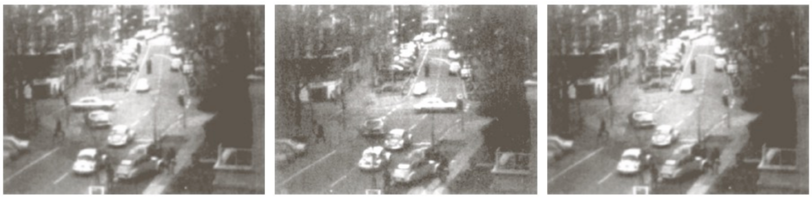


FIGURE 10.60 Building a static reference image. (a) and (b) Two frames in a sequence. (c) Eastbound automobile subtracted from (a) and the background restored from the corresponding area in (b). (Jain and Jain.)

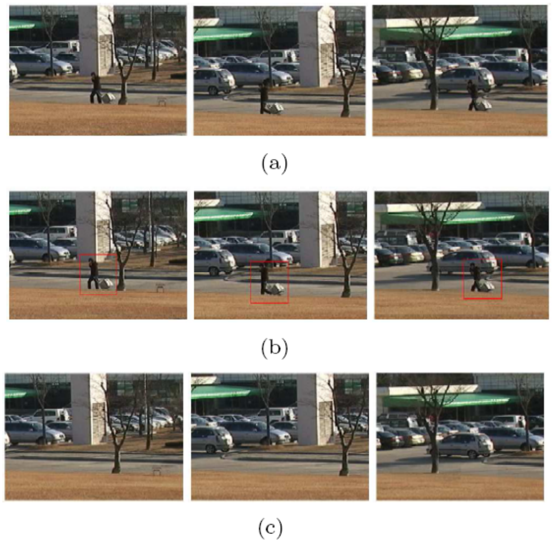
Background Subtraction: Remove moving objects

E.g. Can monitor the changes in the +ADI, e.g. When it stopped changing, it is possible to locate the original position of the moving object



- ❑ When a moving object is completely displaced w.r.t. its original position (i.e. the white car above), the background in the later frame (middle image above) can be duplicated in the 1st frame and one can obtain the reference image on the right.

- ❑ Can do this for all moving objects in the scene
- ❑ Can you see local processing involved?


(a)

(b)

(c)

Figure 5: Removal of a moving person (a) Original frames (10,40,70) (b) Tracked object window (c) After background completion

Moving object removal and , Background Completion in a Video Sequence, Park et al, ACIVS 2006

Motion Detection/Frame Differencing Challenges

* Noise in images can give high differences where there is no motion

» compare neighborhoods rather than points

» use connected components &/ morphological operations to clean spurious responses

* As objects move, their homogeneous interiors may not result in changing image intensities over short time periods

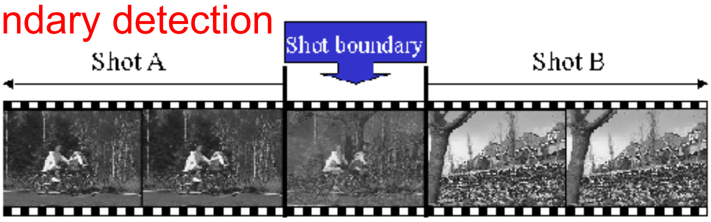
» motion detected only at boundaries

» requires subsequent grouping of moving pixels into objects

Motion Detection/Bg Subtraction (Summary)

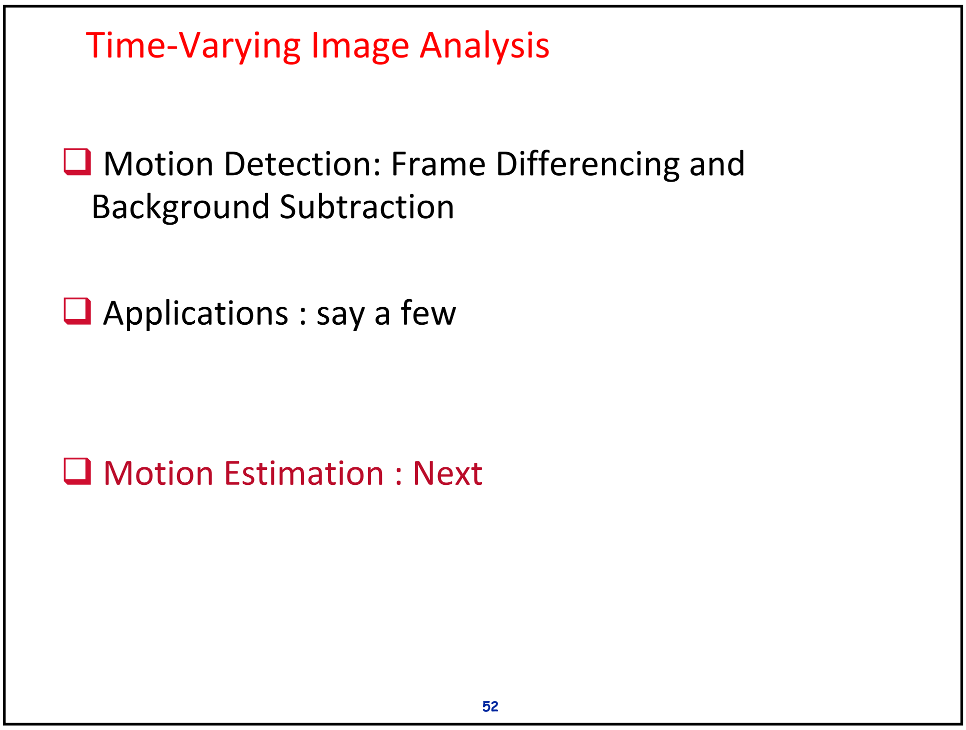
- ❑ create an image of the stationary background by averaging a long sequence
  - » for any pixel, most measurements will be from the background
  - » computing the median measurements, e.g., at each pixel, will with high probability assign that pixel the true background intensity – fixed threshold on differencing used to find “foreground” pixels
    - » can also compute a distribution of background pixels by fitting a mixture of Gaussians to set of intensities and assuming large population is the background - adaptive thresholding to find foreground pixels
- ❑ difference a frame from the known background frame
  - » even for interior points of homogeneous objects, likely to detect a difference
    - » this will also detect objects that are stationary but different from the background
  - » typical algorithm used in surveillance systems
- ❑ Motion detection algorithms such as these only work if the camera is stationary and objects are moving against a fixed background

Shot boundary detection



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

```
Simple idea:
For each frame
    Compute the distance between
    the current frame and the
    previous one
        Pixel-by-pixel differences (or
        Differences of color
        histograms or Block
        comparison)
    If the distance is greater than
    some threshold, classify the frame
    as a shot boundary
```

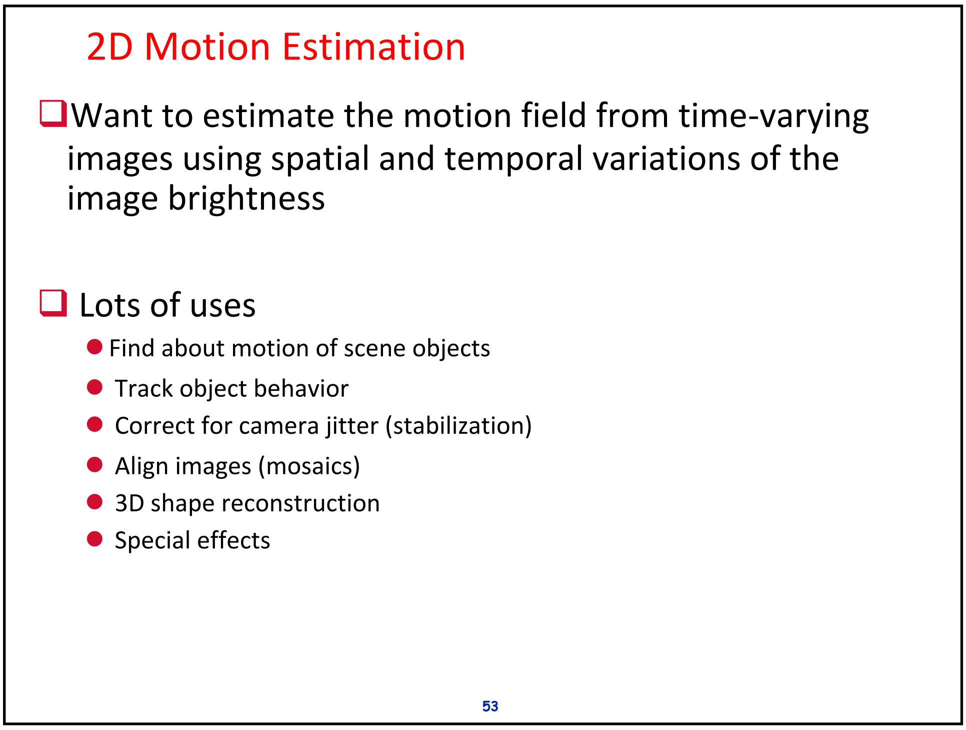


Time-Varying Image Analysis

- ❑ Motion Detection: Frame Differencing and Background Subtraction
- ❑ Applications : say a few
- ❑ Motion Estimation : Next

52

52

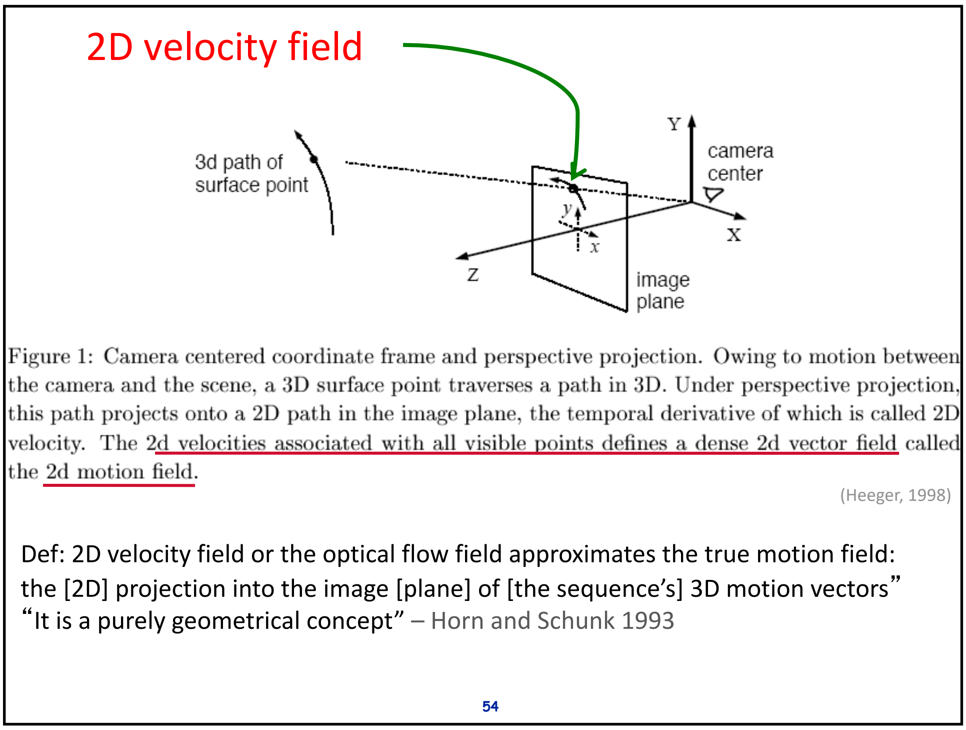


2D Motion Estimation

- ❑ Want to estimate the motion field from time-varying images using spatial and temporal variations of the image brightness
- ❑ Lots of uses
  - Find about motion of scene objects
  - Track object behavior
  - Correct for camera jitter (stabilization)
  - Align images (mosaics)
  - 3D shape reconstruction
  - Special effects

53

53



2D velocity field

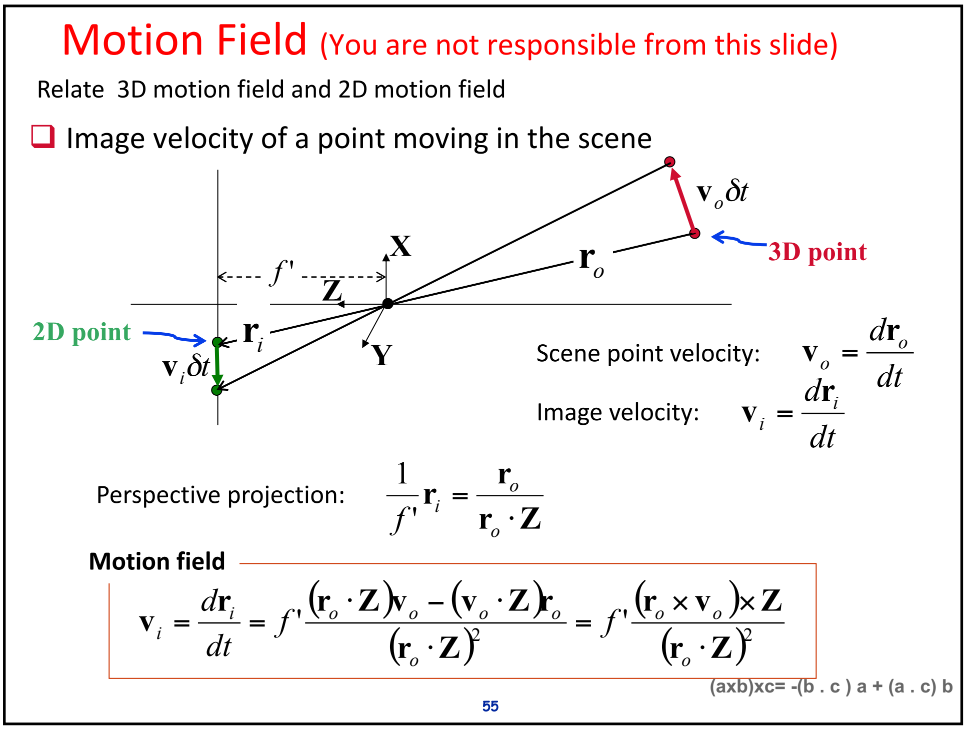
Figure 1: Camera centered coordinate frame and perspective projection. Owing to motion between the camera and the scene, a 3D surface point traverses a path in 3D. Under perspective projection, this path projects onto a 2D path in the image plane, the temporal derivative of which is called 2D velocity. The 2d velocities associated with all visible points defines a dense 2d vector field called the 2d motion field.

(Heeger, 1998)

Def: 2D velocity field or the optical flow field approximates the true motion field: the [2D] projection into the image [plane] of [the sequence's] 3D motion vectors"
"It is a purely geometrical concept" – Horn and Schunk 1993

54

54



Motion Field (You are not responsible from this slide)

Relate 3D motion field and 2D motion field

❑ Image velocity of a point moving in the scene

Scene point velocity: $\mathbf{v}_o = \frac{d\mathbf{r}_o}{dt}$

Image velocity: $\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt}$

Perspective projection: $\frac{1}{f'}\mathbf{r}_i = \frac{\mathbf{r}_o}{\mathbf{r}_o \cdot \mathbf{Z}}$

Motion field

$$\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt} = f' \frac{(\mathbf{r}_o \cdot \mathbf{Z})\mathbf{v}_o - (\mathbf{v}_o \cdot \mathbf{Z})\mathbf{r}_o}{(\mathbf{r}_o \cdot \mathbf{Z})^2} = f' \frac{(\mathbf{r}_o \times \mathbf{v}_o) \times \mathbf{Z}}{(\mathbf{r}_o \cdot \mathbf{Z})^2}$$

(axb)xc= -(b . c) a + (a . c) b

55

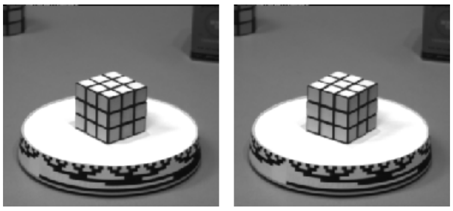
55

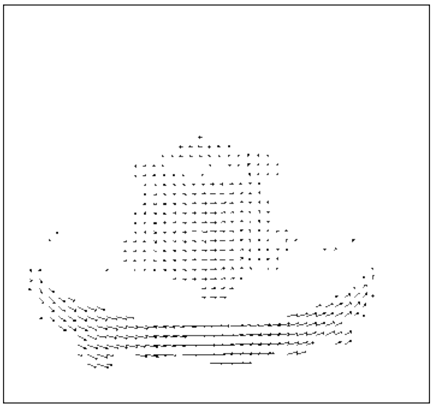
What is the next best thing to 2D True Motion Field: Optical Flow Field

Optical Flow = Motion of brightness pattern in the image

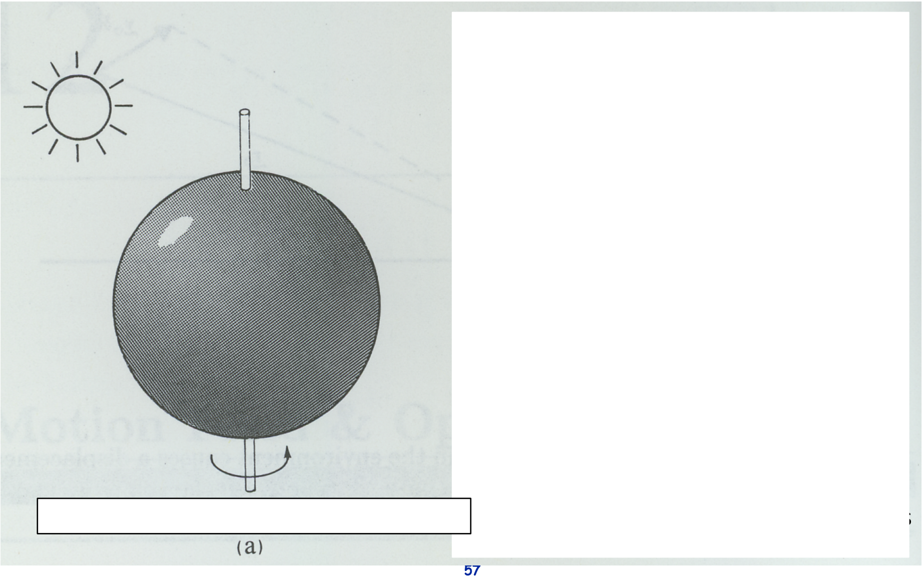
We want to estimate the apparent motion of the image brightness pattern, i.e. the optical flow field

Ideally Optical flow = True Motion field



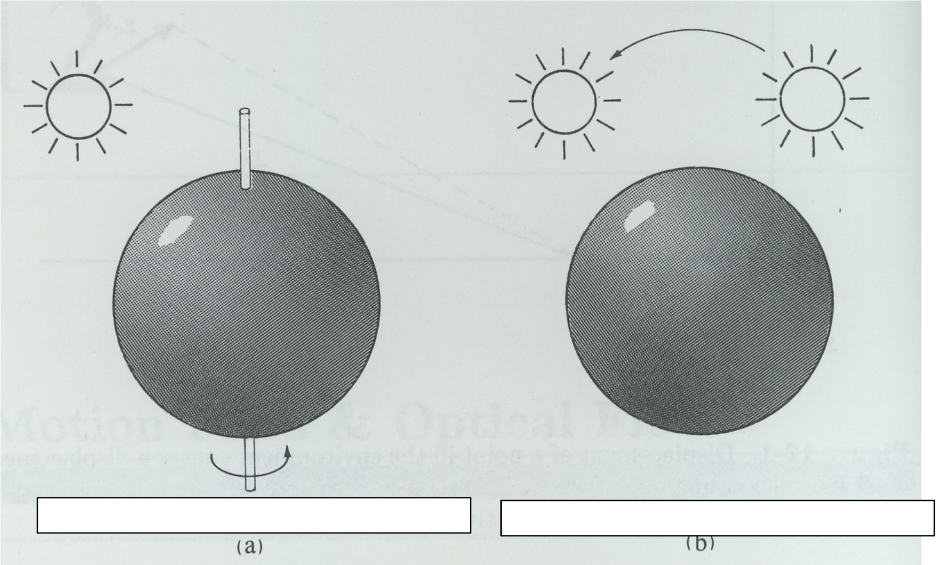


Optical Flow ≠ Motion Field



(a)

Optical Flow ≠ Motion Field

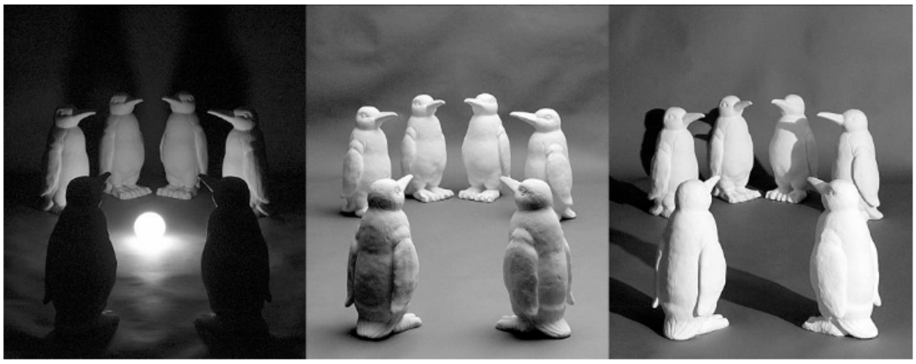


58

58

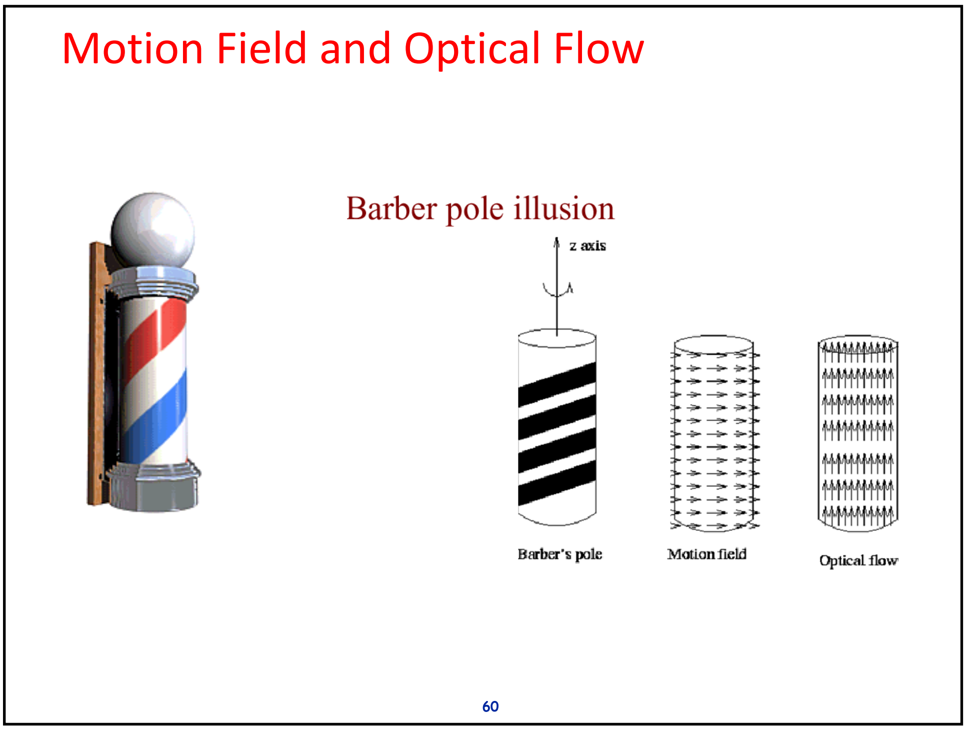
Optical Flow ≠ Motion Field

E.g. No motion field but shading changes



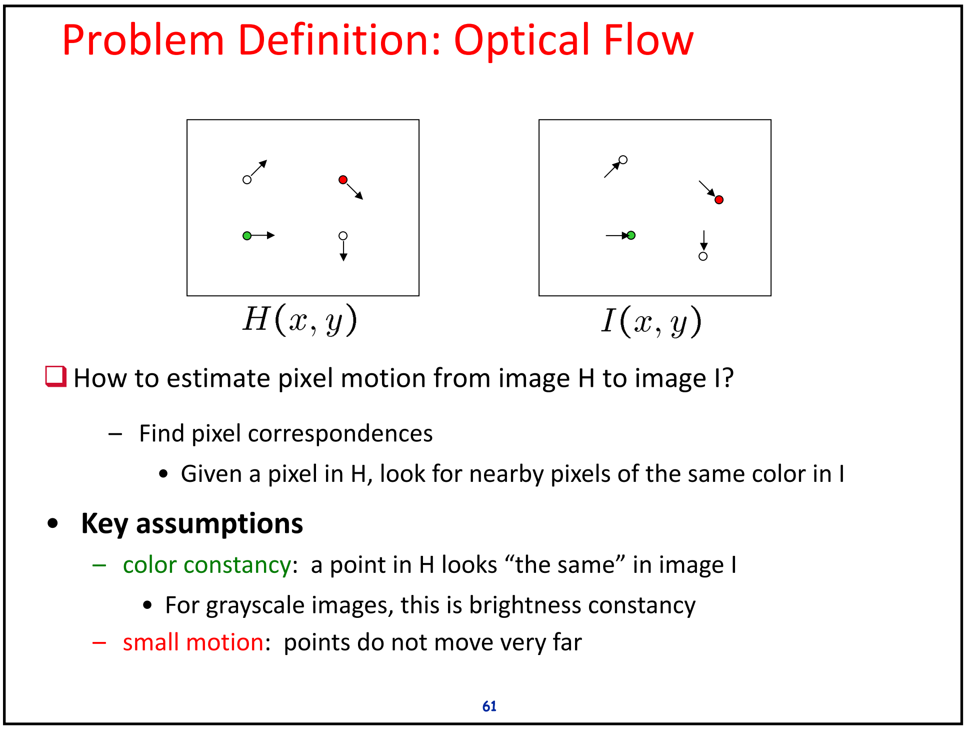
59

59



Motion Field and Optical Flow

Barber pole illusion



Problem Definition: Optical Flow

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

- Find pixel correspondences
  - Given a pixel in H, look for nearby pixels of the same color in I

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

Notation: Optical Flow

Time-varying image function is represented by: I(x,y,t)

2D optical flow = (u,v) = V

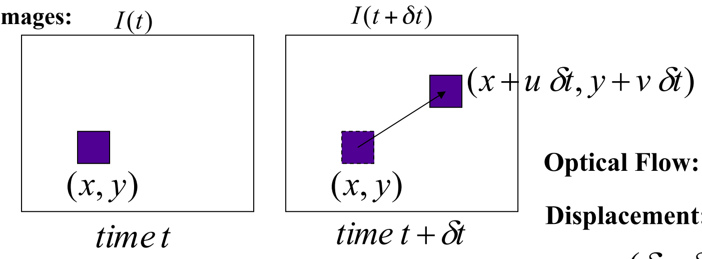
Partial derivatives denoted by subscripts: e.g. $I_x = \partial I / \partial x$

Nabla ∇ denotes the gradient operator:

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

Optical Flow Constraint Equation

Images:



Optical Flow: Velocities (u,v)

Displacement:

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

- Assume brightness of patch remains same in both images:

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

- Assume small motion: (Taylor expansion of LHS upto first order)

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

Optical Flow (OF) Constraint Equation

Divide this by δt

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

Inserting:

$u = \frac{\delta x}{\delta t}$ $v = \frac{\delta y}{\delta t}$ ⇨

$$\boxed{u \frac{\partial I}{\partial x} + v \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0}$$

OF Constraint Equation

Equivalently Assumption: stationarity of image brightness I over time

$\boxed{\frac{dI(x(t), y(t), t)}{dt} = 0}$ leads to $\frac{dx}{dt}\frac{\partial I}{\partial x} + \frac{dy}{dt}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$

OF Constraint Equation $\boxed{I_x\, u + I_y\, v + I_t = 0}$

64

64

Optical Flow Constraint Equation

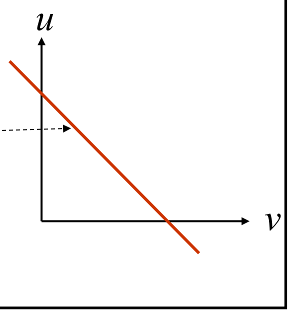
Optical Flow Constraint Equation

$$\boxed{I_x\, u + I_y\, v + I_t = 0}$$

We can compute I_x, I_y, I_t using gradient operators!

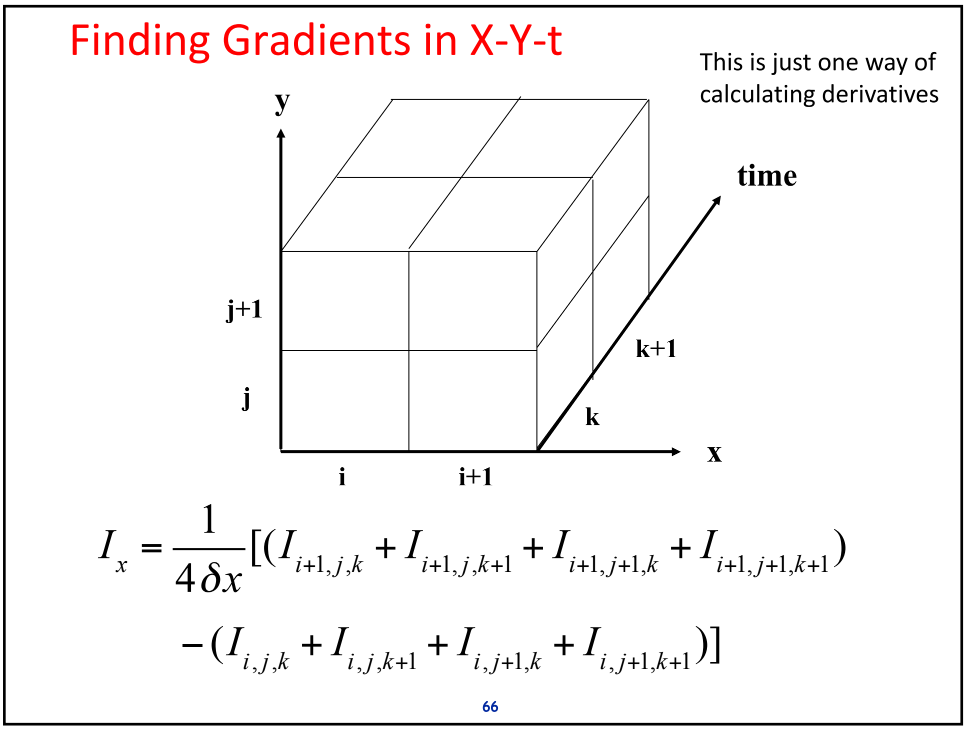
But: (u,v) cannot be found uniquely with this constraint, why?

Note: The constraint equation is a line in variables (u,v)



65

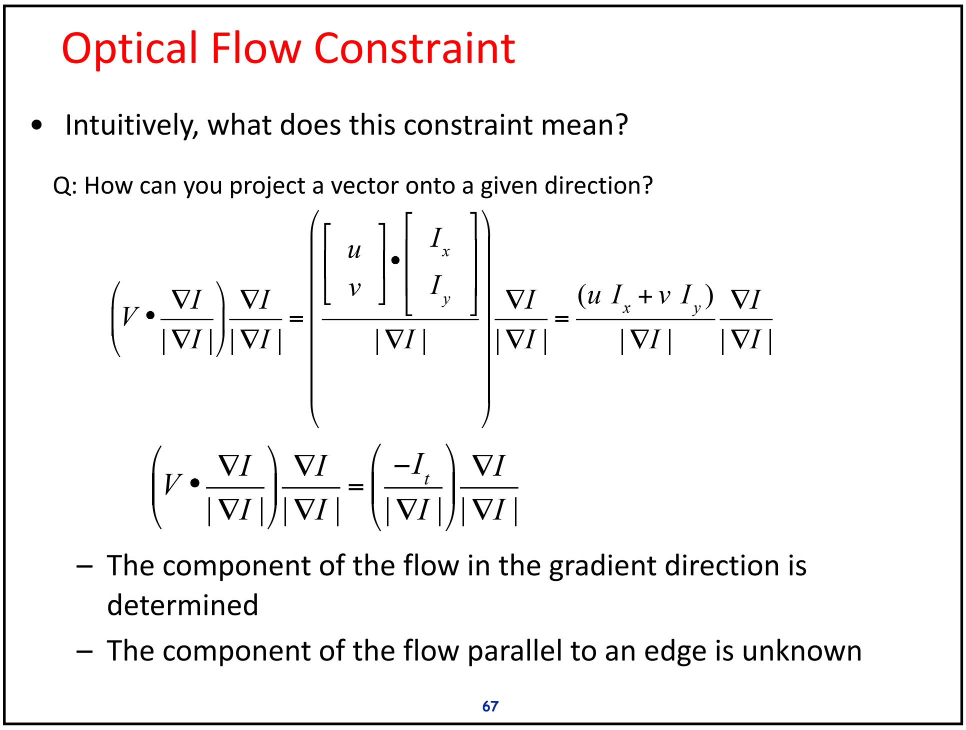
65



Finding Gradients in X-Y-t

This is just one way of calculating derivatives

$$I_x = \frac{1}{4\delta x}[(I_{i+1,j,k} + I_{i+1,j,k+1} + I_{i+1,j+1,k} + I_{i+1,j+1,k+1})$$
$$-(I_{i,j,k} + I_{i,j,k+1} + I_{i,j+1,k} + I_{i,j+1,k+1})]$$



Optical Flow Constraint

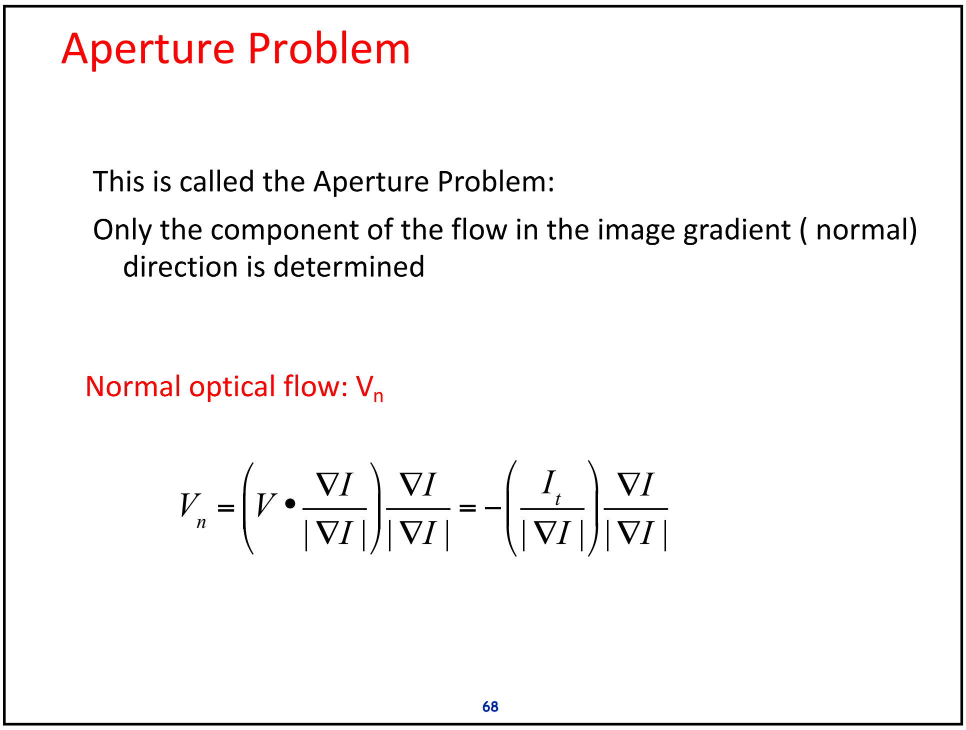
- Intuitively, what does this constraint mean?

Q: How can you project a vector onto a given direction?

$$\left(V \bullet \frac{\nabla I}{|\nabla I|}\right)\frac{\nabla I}{|\nabla I|} = \left(\frac{\begin{bmatrix} u \\ v \end{bmatrix} \bullet \begin{bmatrix} I_x \\ I_y \end{bmatrix}}{|\nabla I|}\right)\frac{\nabla I}{|\nabla I|} = \frac{(u\ I_x + v\ I_y)}{|\nabla I|}\frac{\nabla I}{|\nabla I|}$$

$$\left(V \bullet \frac{\nabla I}{|\nabla I|}\right)\frac{\nabla I}{|\nabla I|} = \left(\frac{-I_t}{|\nabla I|}\right)\frac{\nabla I}{|\nabla I|}$$

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



Aperture Problem

This is called the Aperture Problem:

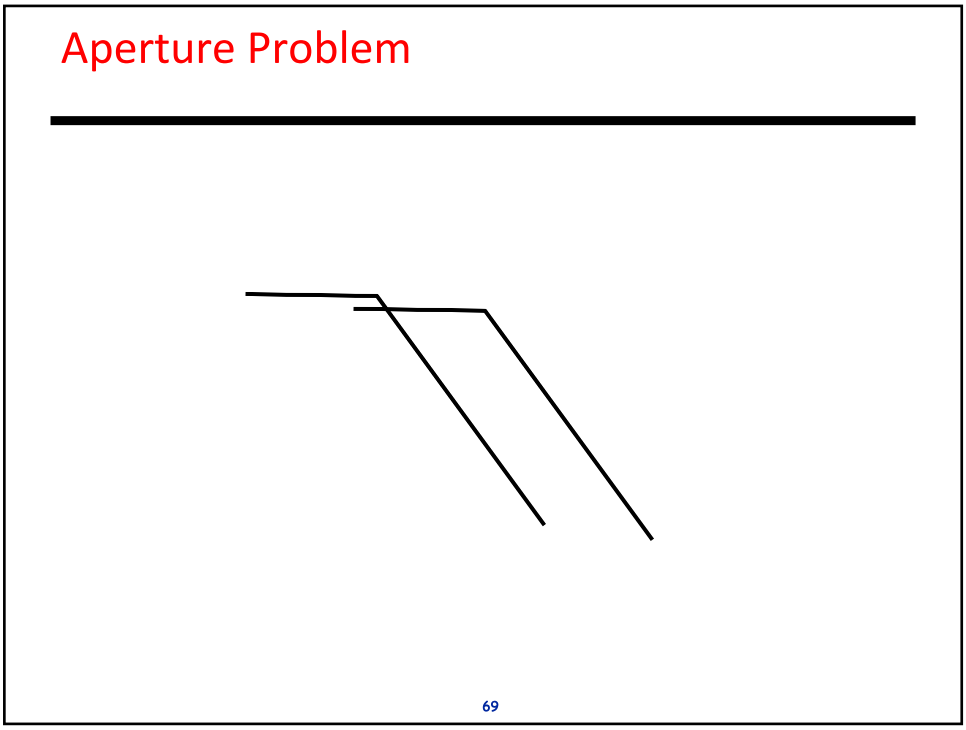
Only the component of the flow in the image gradient (normal) direction is determined

Normal optical flow: V_n

$$V_n = \left(V \bullet \frac{\nabla I}{|\nabla I|} \right) \frac{\nabla I}{|\nabla I|} = -\left(\frac{I_t}{|\nabla I|} \right) \frac{\nabla I}{|\nabla I|}$$

68

68

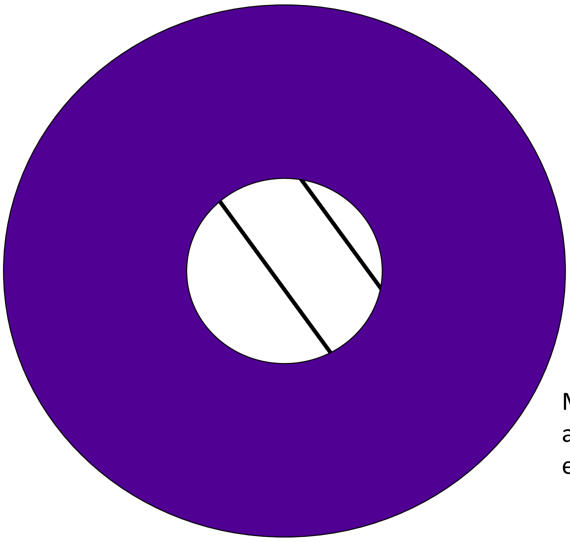


Aperture Problem

69

69

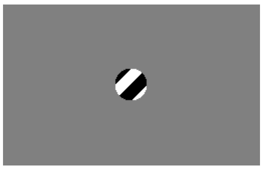
Aperture Problem



Motion viewed along just an edge is ambigous

70

Aperture Problem



The grating appears to be moving down and to the right, perpendicular to the orientation of the bars. But it could be moving in many other directions, such as only down, or only to the right. It is impossible to determine unless the ends of the bars become visible in the aperture. (Picture from Wiki)

71

Computing Optical Flow

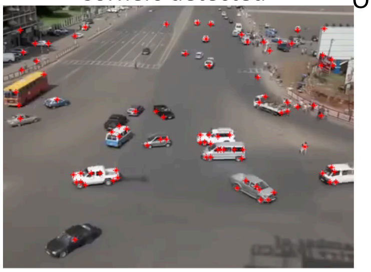
$$I_t(p) + \nabla I(p) \cdot [\ u(p) \quad v(p)\ ] = 0$$

We want to compute optical flow (2D) vectors:

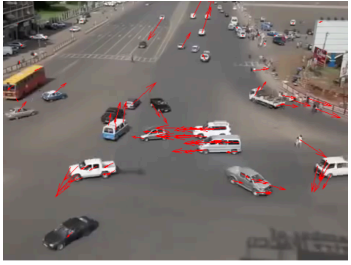
$I(t)$

$I(t+\delta t)$

Corners detected



Optical flow vectors over corner point



https://www.mathworks.com/matlabcentral/fileexchange/48745-lucas-kanade-tutorial-example-2?focused=3853985&tab=example

Computing Optical Flow

At a point (pixel coordinate) p in an image, for solving (u,v), we have:

$$I_t(p) + \nabla I(p) \cdot [\ u(p) \quad v(p)\ ] = 0$$

Q: How to get more equations for a pixel?

We need additional constraint(s) →

The Lucas-Kanade Method

- ❑How to get more equations for a pixel?
- ❑ Basic idea: impose additional constraints
- ❑ Most common is to assume that the flow field is smooth locally, but we'll study even a simpler assumption as shown below.

- ❑ Lucas-Kanade method assumes that flow field is locally constant:

i.e. pretend that the pixel's neighbors have the same (u,v), for all pi :

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

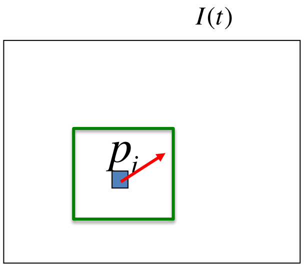
E.g. If we use a 5x5 window, that gives us 25 equations per pixel!

Motion Estimation: The Lucas-Kanade Method

Q: How to get more equations at each coordinate?

e.g. A common assumption is that the motion field is constant in a local neighborhood:

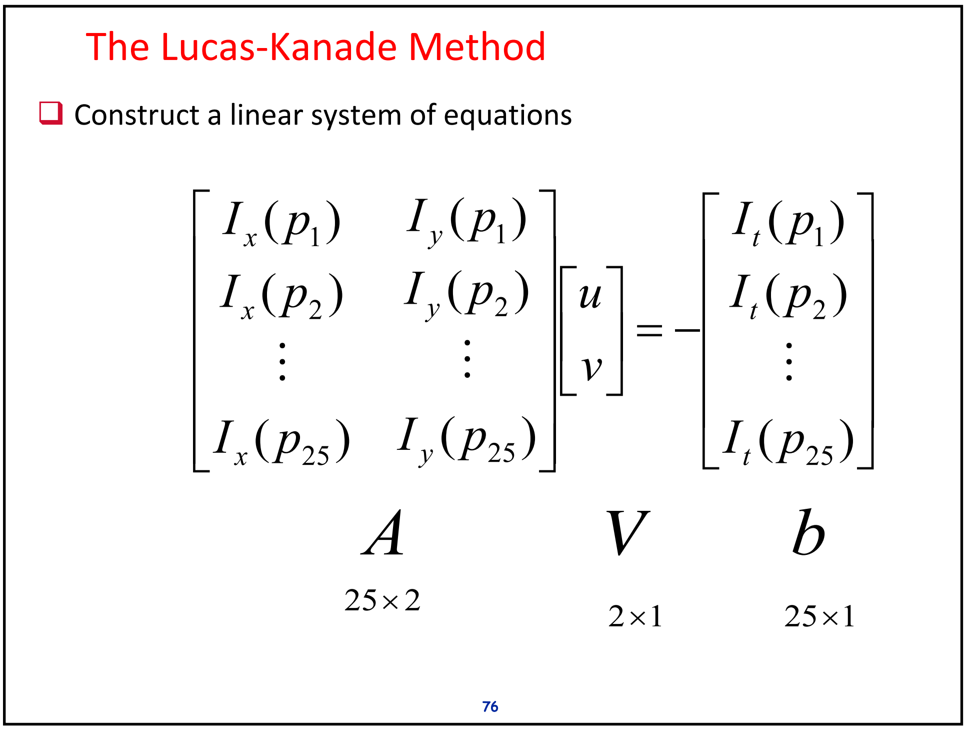
in the green box in this case (e.g. a 5x5 region)



Then, a set of spatial coordinates in that neighborhood satisfy:

$$\forall p_i :$$

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



The Lucas-Kanade Method

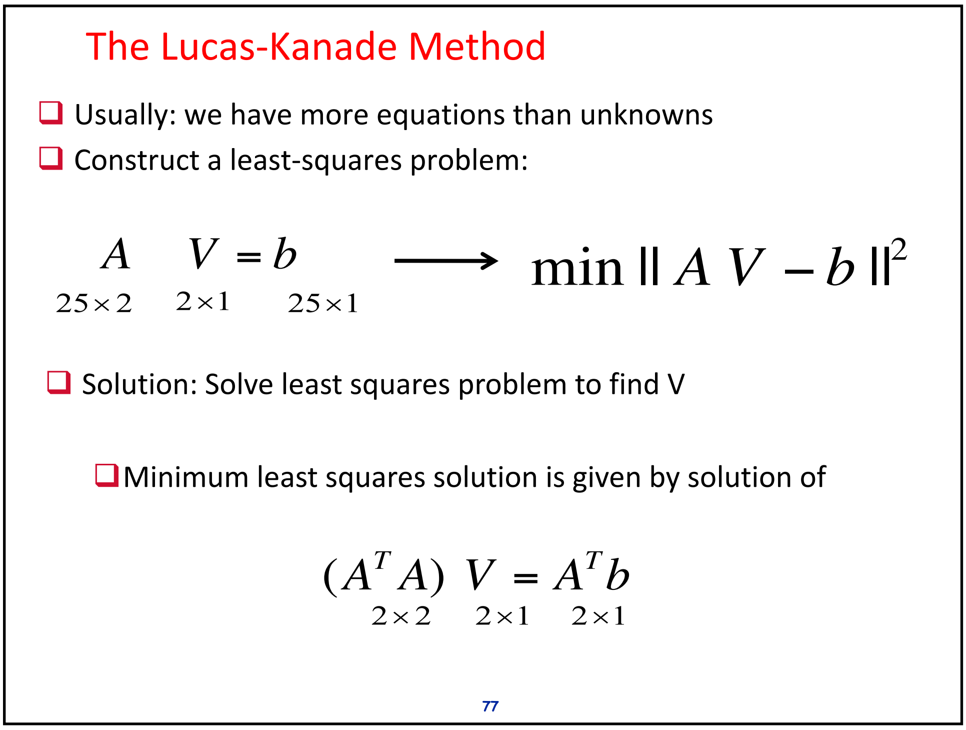
- ❑ Construct a linear system of equations

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

$$\underset{25\times 2}{A} \qquad \underset{2\times 1}{V} \qquad \underset{25\times 1}{b}$$

76

76



The Lucas-Kanade Method

- ❑ Usually: we have more equations than unknowns
- ❑ Construct a least-squares problem:

$$\underset{25\times 2}{A} \quad \underset{2\times 1}{V} = \underset{25\times 1}{b} \longrightarrow \min \| A\ V - b \|^2$$

- ❑ Solution: Solve least squares problem to find V
  - ❑ Minimum least squares solution is given by solution of

$$\underset{2\times 2}{(A^T A)} \ \underset{2\times 1}{V} = \underset{2\times 1}{A^T b}$$

77

77

The Lucas-Kanade Method

Optical Flow https://www.youtube.com/watch?v=JlLkkom6tWw

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

$$(A^T A) \qquad\qquad A^T b$$

- ❑ The summations are over all pixels in the KxK window
- ❑ 2D motion field solution:

$$\boxed{\begin{bmatrix} u \\ v \end{bmatrix} = -(A^T A)^{-1} A^T b} \; ★$$

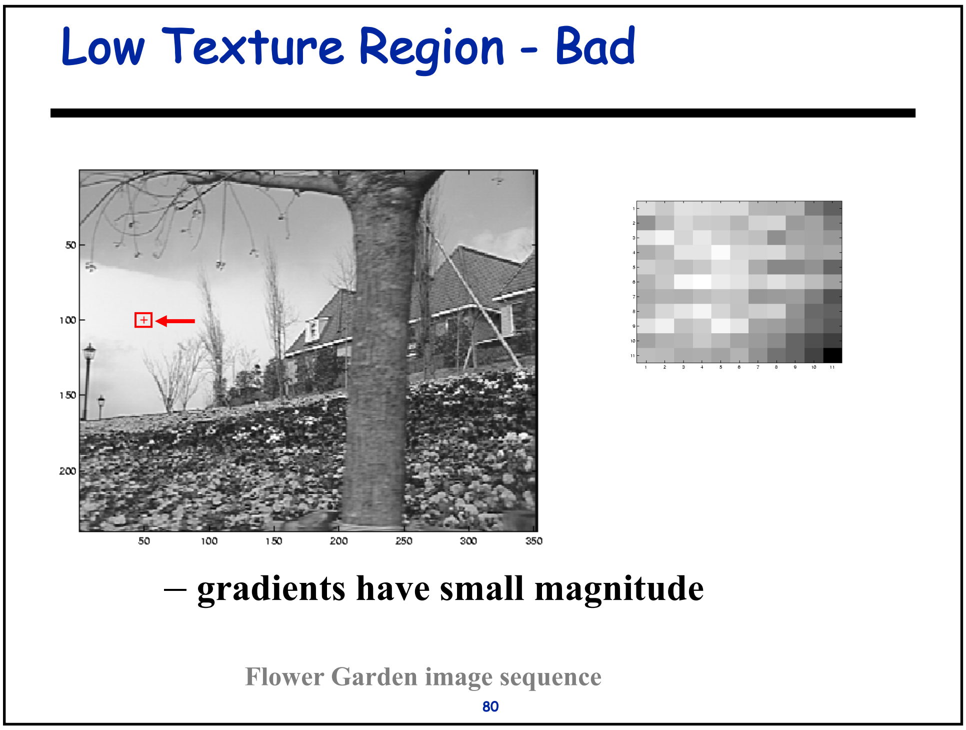
Conditions for Solvability

- ❑ Optimal (u,v) satisfies Lucas & Kanade equation

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

$$(A^T A) \qquad\qquad A^T b$$

- ❑ When is this solvable?
  - ❑ $A^T A$ should be invertible
  - ❑ $A^T A$ should not be too small due to noise
    - ❑ eigen values λ_1 and λ_2 of $A^T A$ should not be too small
  - ❑ $A^T A$ should be well-conditioned
    - ❑ λ_1/λ_2 should not be too large (λ_1 larger eigen value)



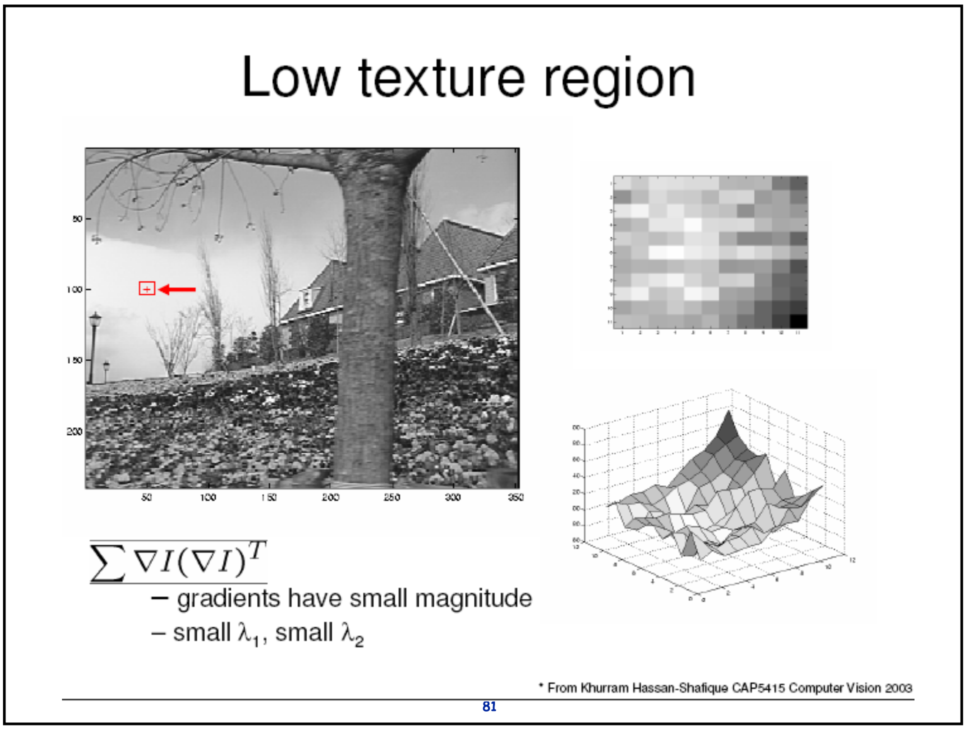
Low Texture Region - Bad

– gradients have small magnitude

Flower Garden image sequence

80

80



Low texture region

$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

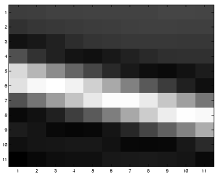
– small λ_1, small λ_2

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

81

81

Edges – so,so (aperture problem)

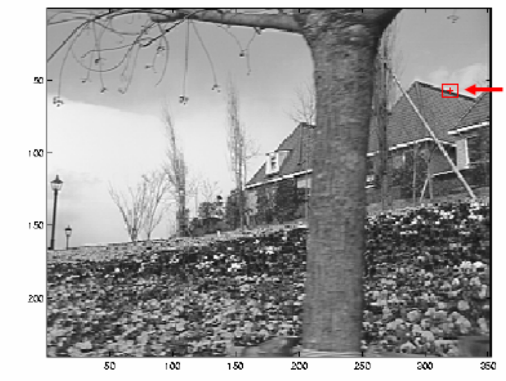


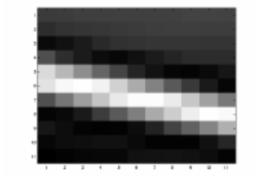
– large gradients, all in the same direction

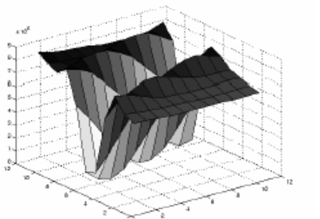
82

82

Edge







$\sum \nabla I(\nabla I)^T$

- large gradients, all the same
- large λ_1, small λ_2

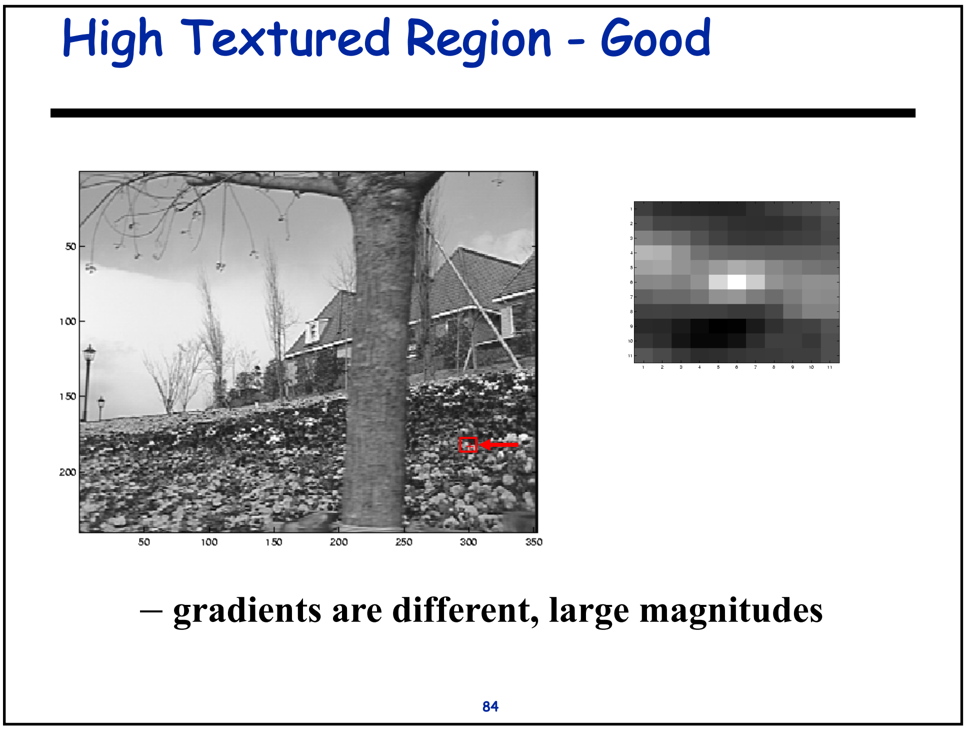
across edge direction

$$A^T A = \sum \nabla I \ \nabla I^T$$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

83

83

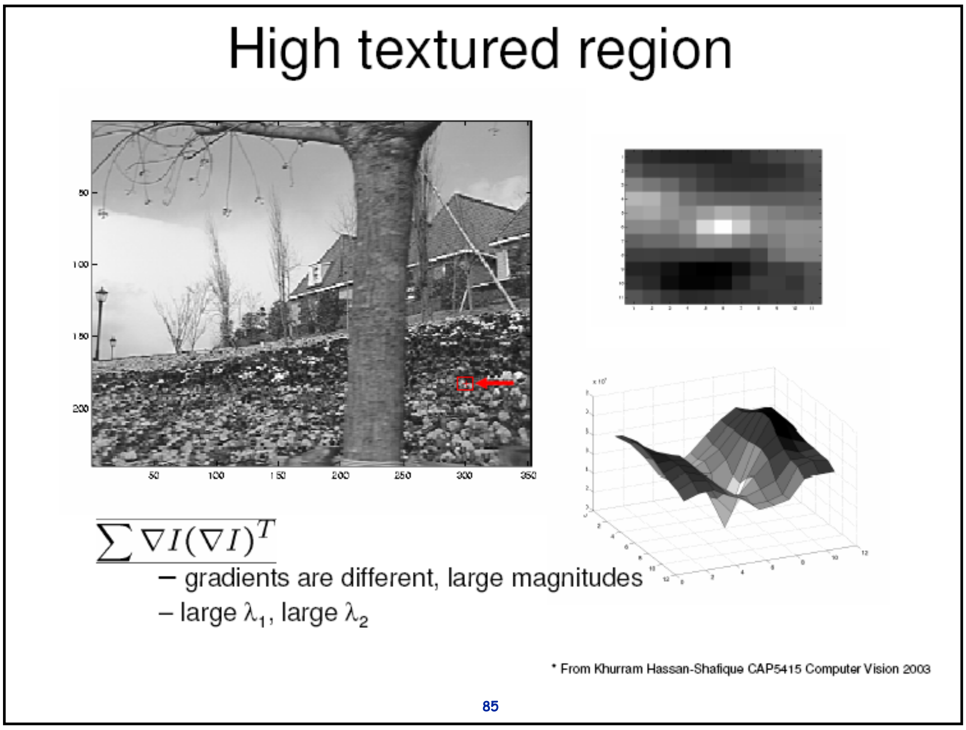


High Textured Region - Good

– **gradients are different, large magnitudes**

84

84



High textured region

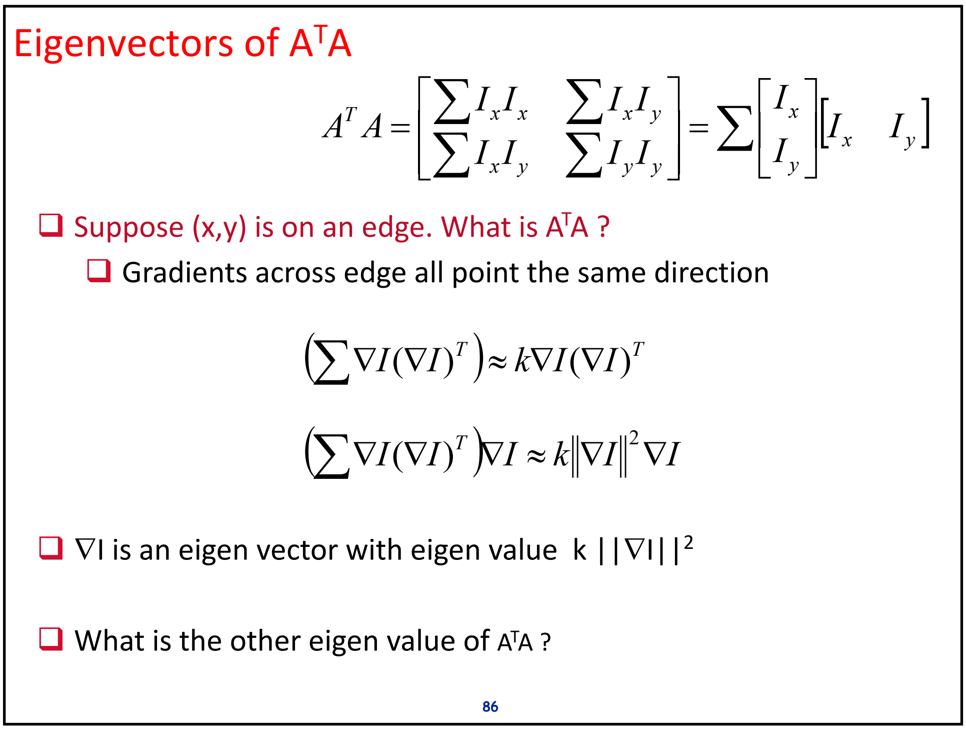
$\sum \nabla I (\nabla I)^T$

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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

85

85



Eigenvectors of $A^T A$

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix}$$

- ❑ Suppose (x,y) is on an edge. What is $A^T A$?
  - ❑ Gradients across edge all point the same direction

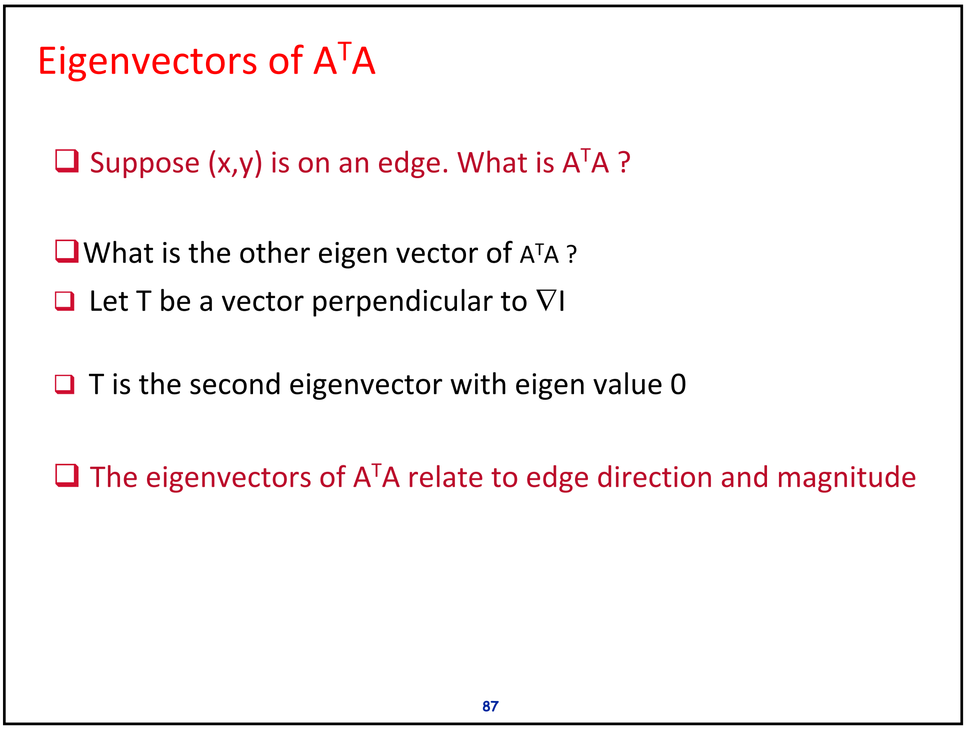
$$\left(\sum \nabla I (\nabla I)^T\right) \approx k \nabla I (\nabla I)^T$$

$$\left(\sum \nabla I (\nabla I)^T\right) \nabla I \approx k \|\nabla I\|^2 \nabla I$$

- ❑ ∇I is an eigen vector with eigen value k $||\nabla I||^2$
- ❑ What is the other eigen value of $A^T A$?

86

86

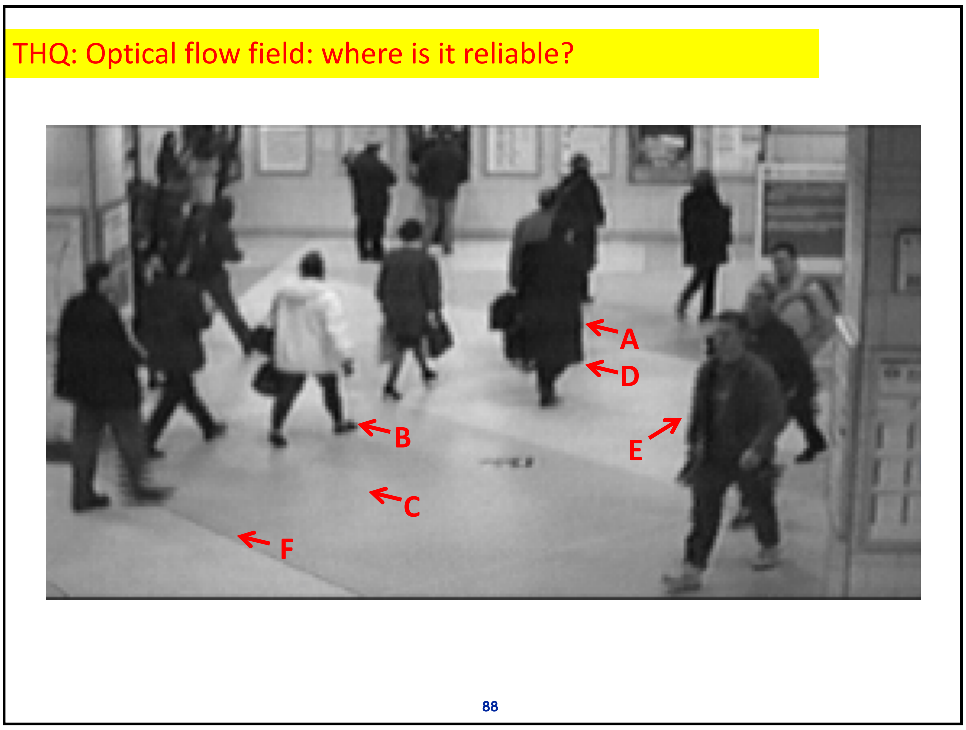


Eigenvectors of $A^T A$

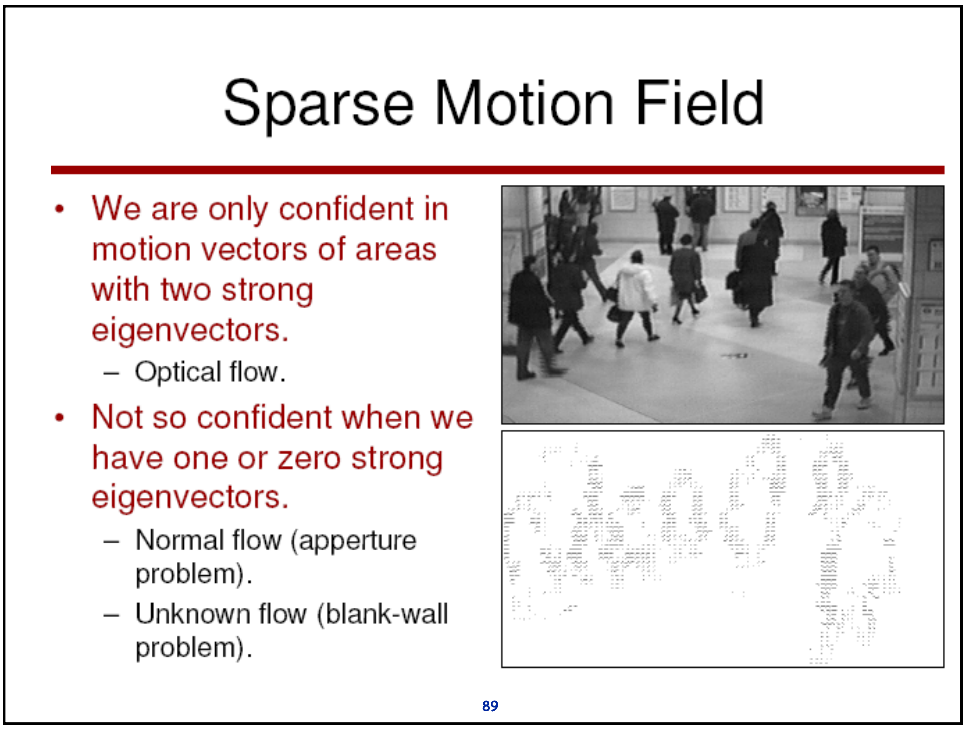
- ❑ Suppose (x,y) is on an edge. What is $A^T A$?
- ❑What is the other eigen vector of $A^T A$?
- ❑ Let T be a vector perpendicular to ∇I
- ❑ T is the second eigenvector with eigen value 0
- ❑ The eigenvectors of $A^T A$ relate to edge direction and magnitude

87

87



THQ: Optical flow field: where is it reliable?



Sparse Motion Field

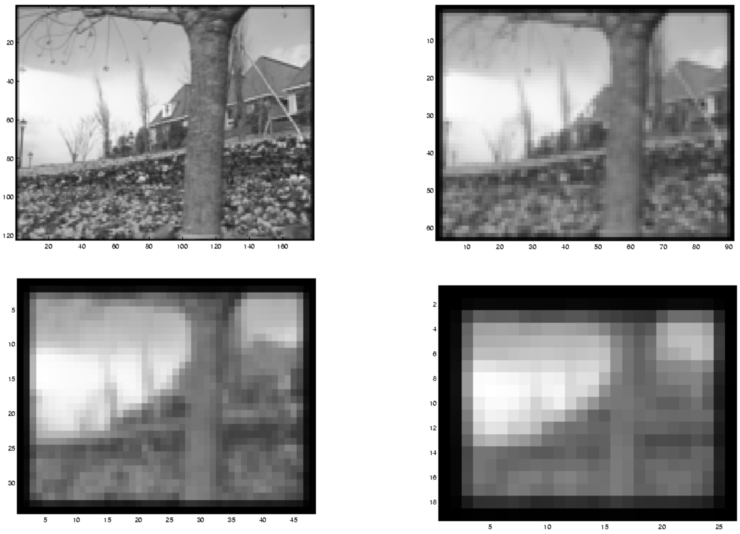
- We are only confident in motion vectors of areas with two strong eigenvectors.
  - Optical flow.
- Not so confident when we have one or zero strong eigenvectors.
  - Normal flow (apperture problem).
  - Unknown flow (blank-wall problem).

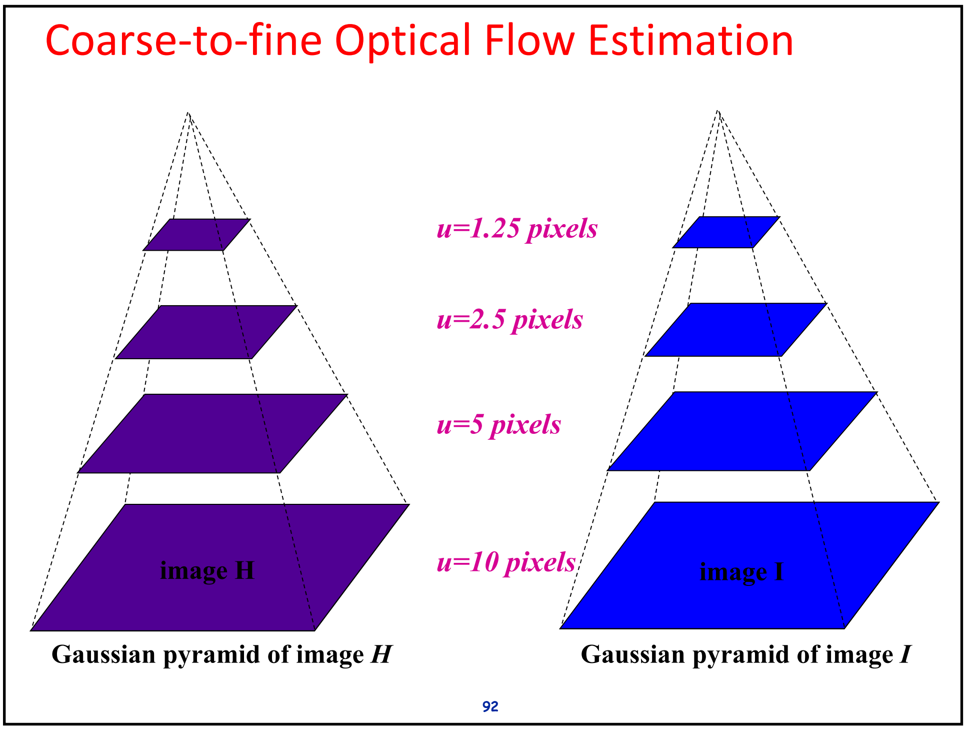
Revisiting the Small Motion Assumption

❑ Is this motion small enough?

- ● Probably not—it's much larger than one pixel (higher order terms dominate)
- ● How might we solve this problem?

Reduce the Resolution!

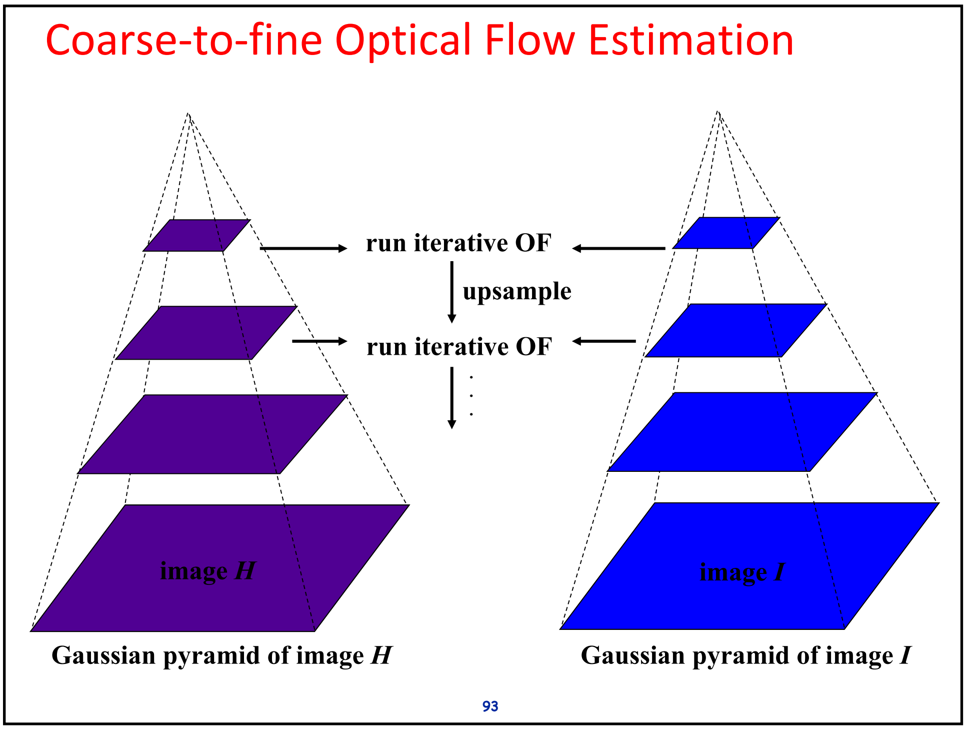




Coarse-to-fine Optical Flow Estimation

Gaussian pyramid of image *H*

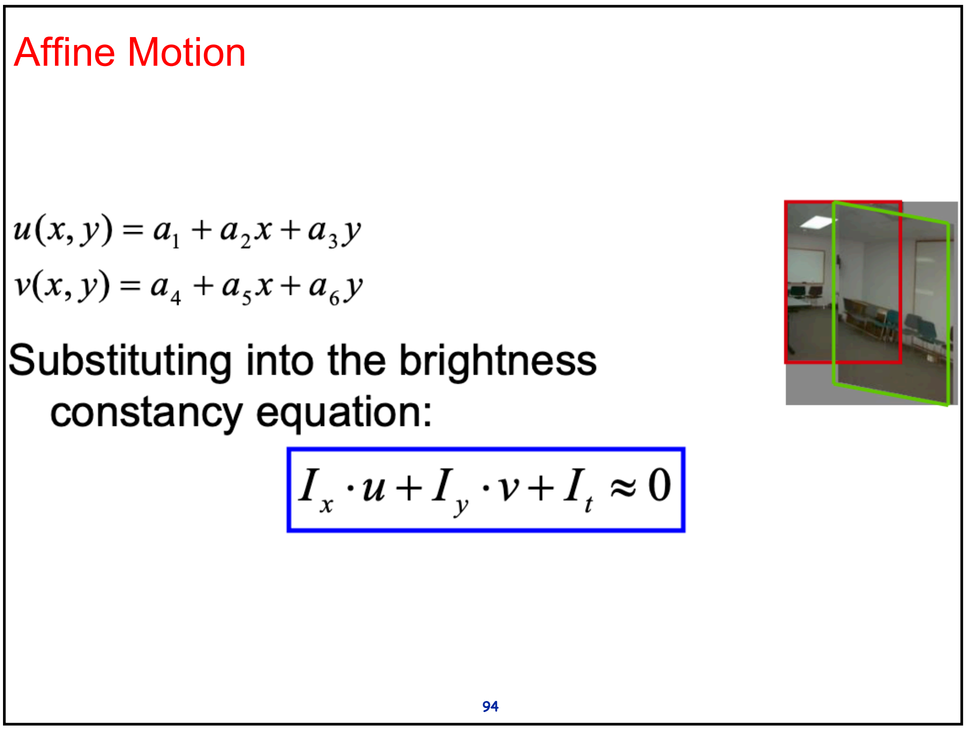
Gaussian pyramid of image *I*



Coarse-to-fine Optical Flow Estimation

Gaussian pyramid of image *H*

Gaussian pyramid of image *I*



Affine Motion

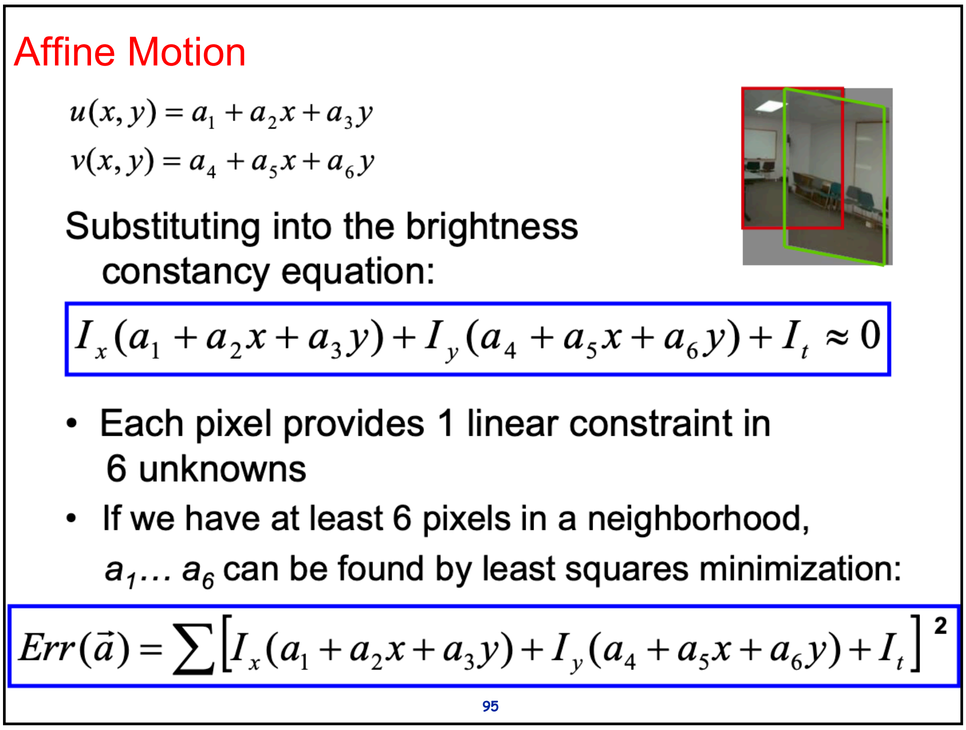
$$u(x,y) = a_1 + a_2 x + a_3 y$$
$$v(x,y) = a_4 + a_5 x + a_6 y$$

Substituting into the brightness constancy equation:

$$\boxed{I_x \cdot u + I_y \cdot v + I_t \approx 0}$$

94

94



Affine Motion

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

Substituting into the brightness constancy equation:

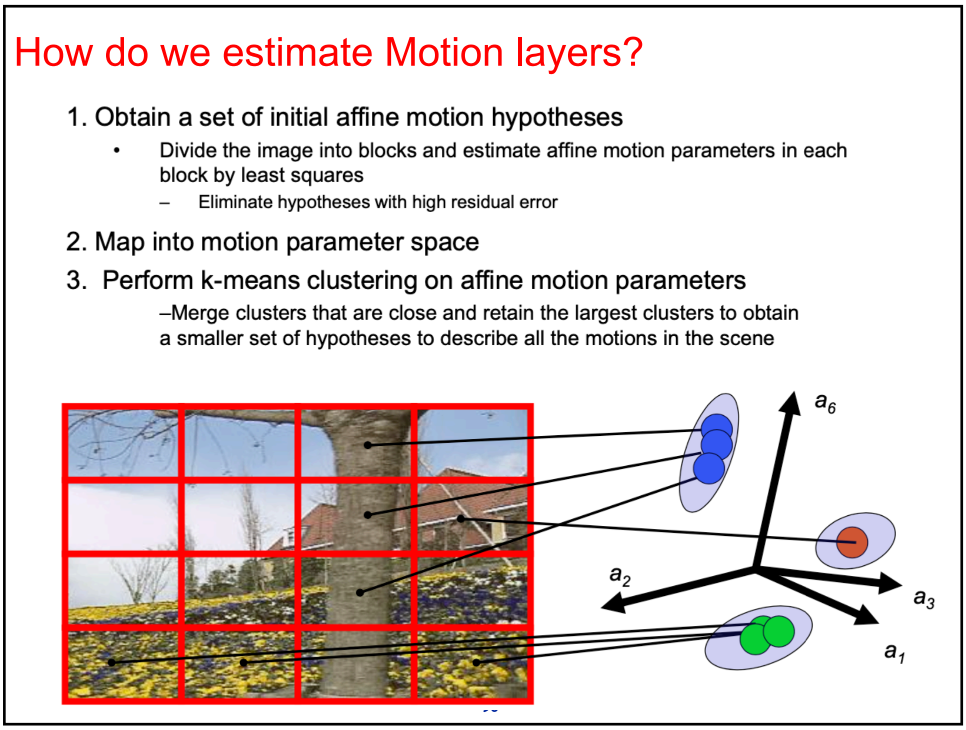
$$\boxed{I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \approx 0}$$

- Each pixel provides 1 linear constraint in 6 unknowns
- If we have at least 6 pixels in a neighborhood, $a_1 \ldots a_6$ can be found by least squares minimization:

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

95

95

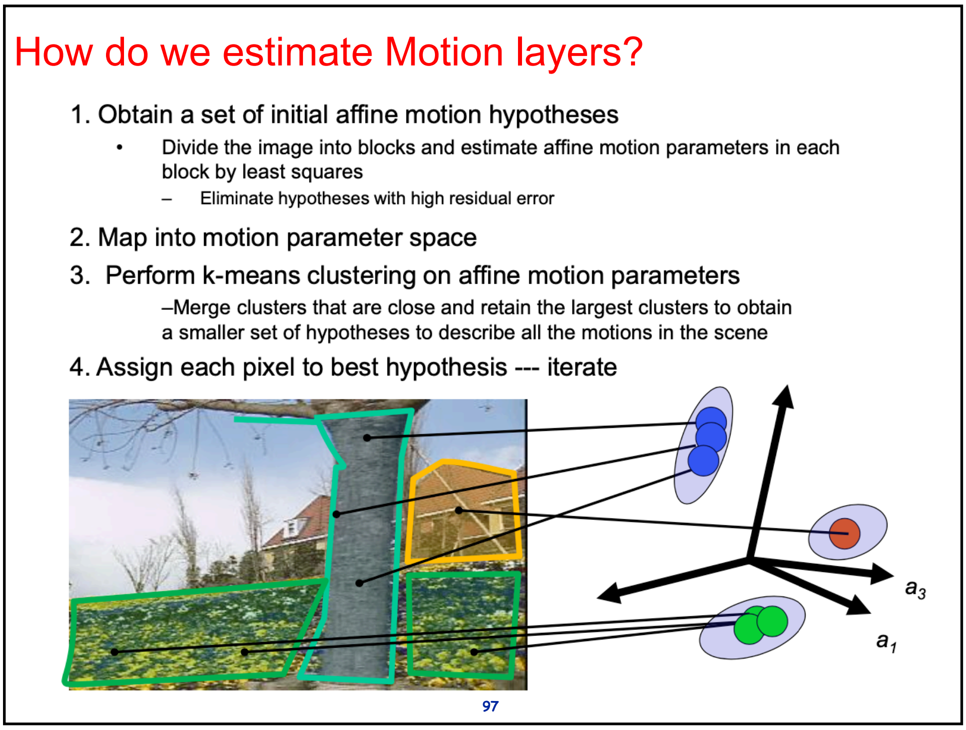


How do we estimate Motion layers?

1. Obtain a set of initial affine motion hypotheses
   - Divide the image into blocks and estimate affine motion parameters in each block by least squares
     - Eliminate hypotheses with high residual error
2. Map into motion parameter space
3. Perform k-means clustering on affine motion parameters

   –Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene

96

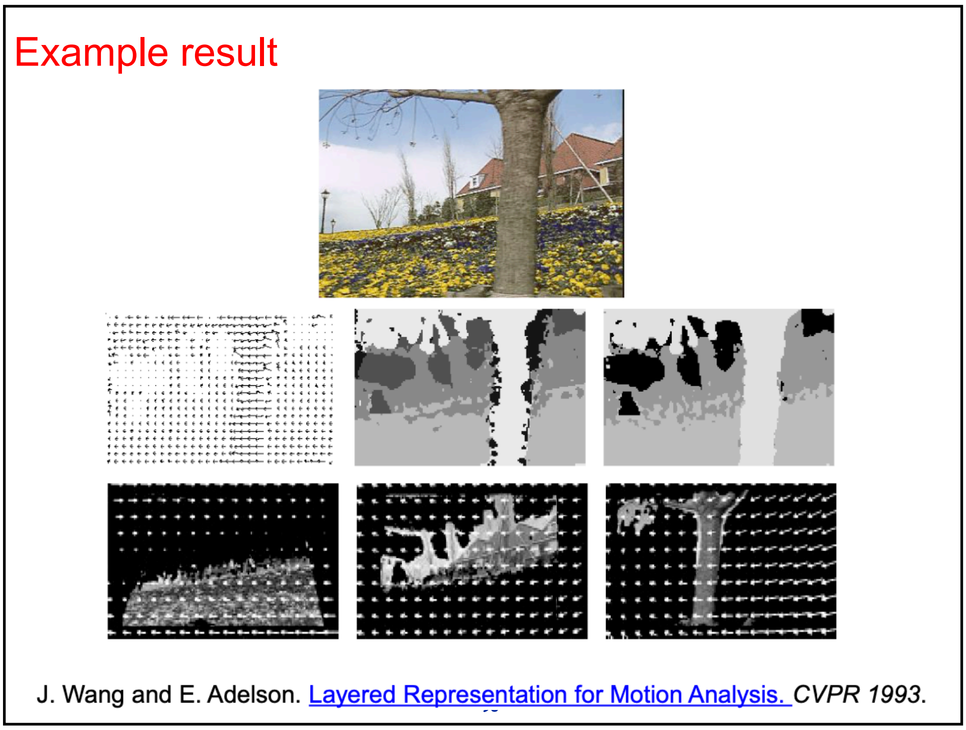


How do we estimate Motion layers?

1. Obtain a set of initial affine motion hypotheses
   - Divide the image into blocks and estimate affine motion parameters in each block by least squares
     - Eliminate hypotheses with high residual error
2. Map into motion parameter space
3. Perform k-means clustering on affine motion parameters

   –Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene
4. Assign each pixel to best hypothesis --- iterate

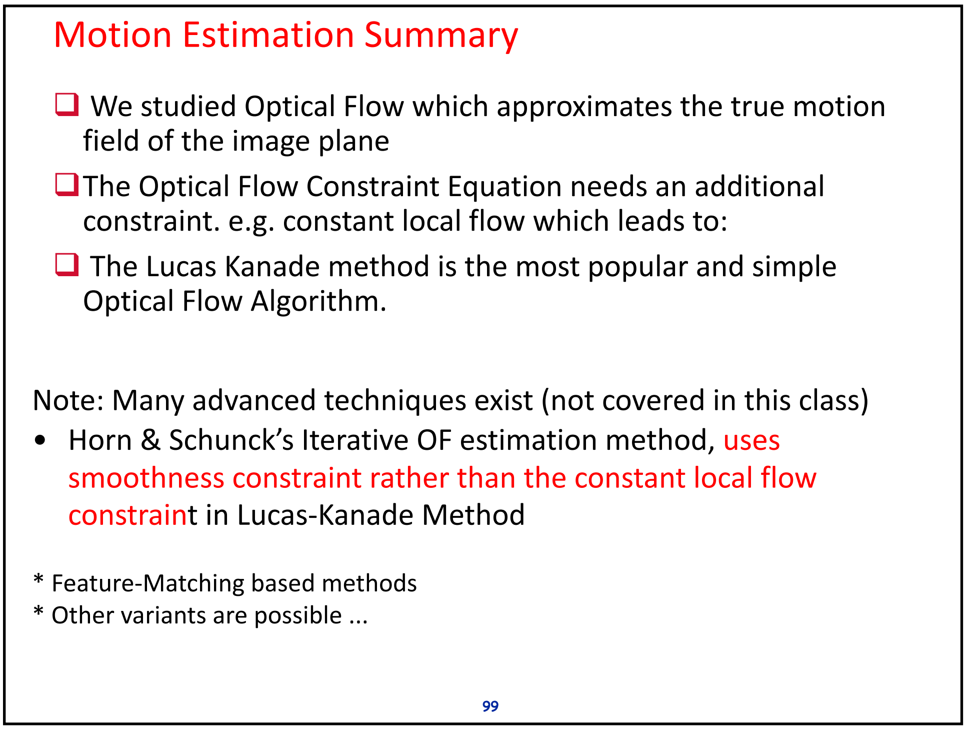
97



Example result

J. Wang and E. Adelson. Layered Representation for Motion Analysis. *CVPR 1993*.

98



Motion Estimation Summary

- ❑ We studied Optical Flow which approximates the true motion field of the image plane
- ❑The Optical Flow Constraint Equation needs an additional constraint. e.g. constant local flow which leads to:
- ❑ The Lucas Kanade method is the most popular and simple Optical Flow Algorithm.

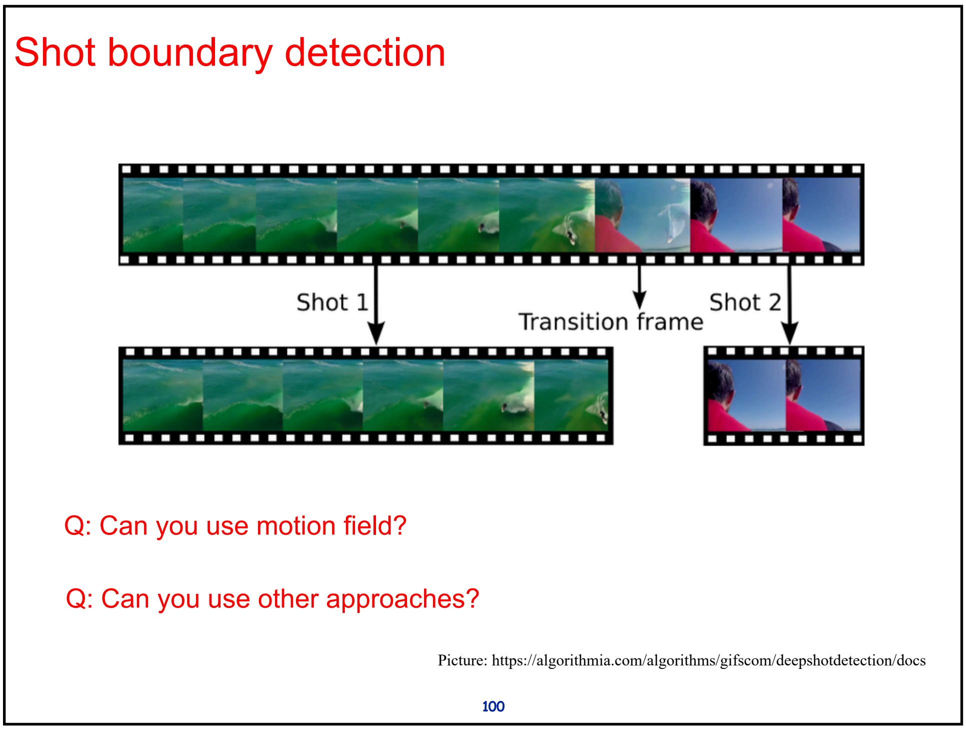
Note: Many advanced techniques exist (not covered in this class)

- Horn & Schunck’s Iterative OF estimation method, uses smoothness constraint rather than the constant local flow constraint in Lucas-Kanade Method

* Feature-Matching based methods
* Other variants are possible …

99

99

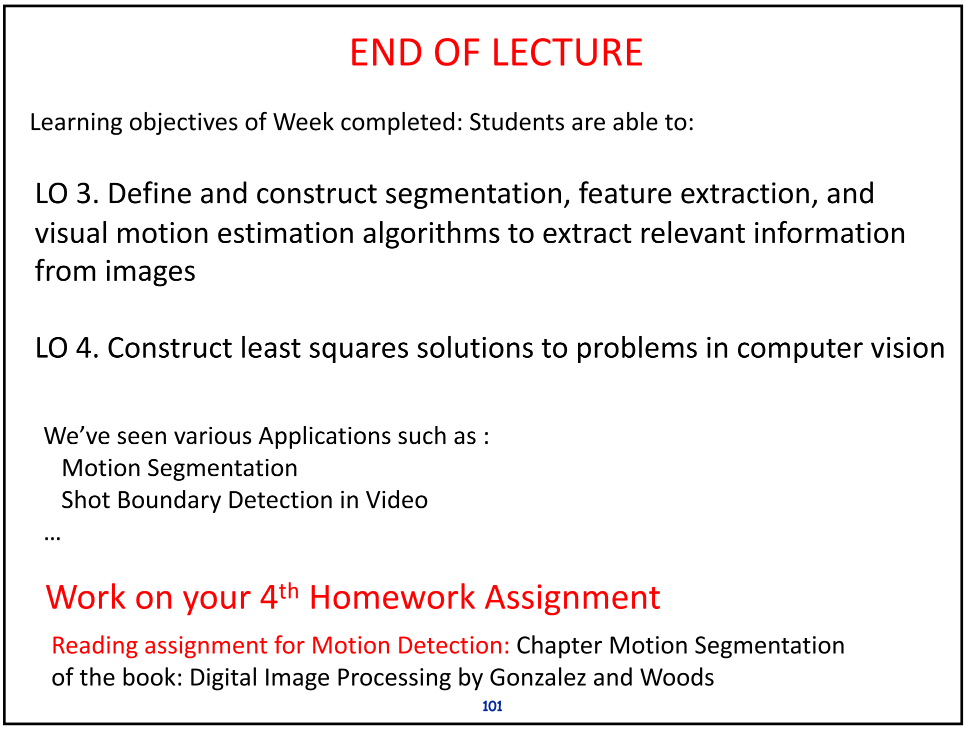


Shot boundary detection

Q: Can you use motion field?

Q: Can you use other approaches?

Picture: https://algorithmia.com/algorithms/gifscom/deepshotdetection/docs



END OF LECTURE

Learning objectives of Week completed: Students are able to:

LO 3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images

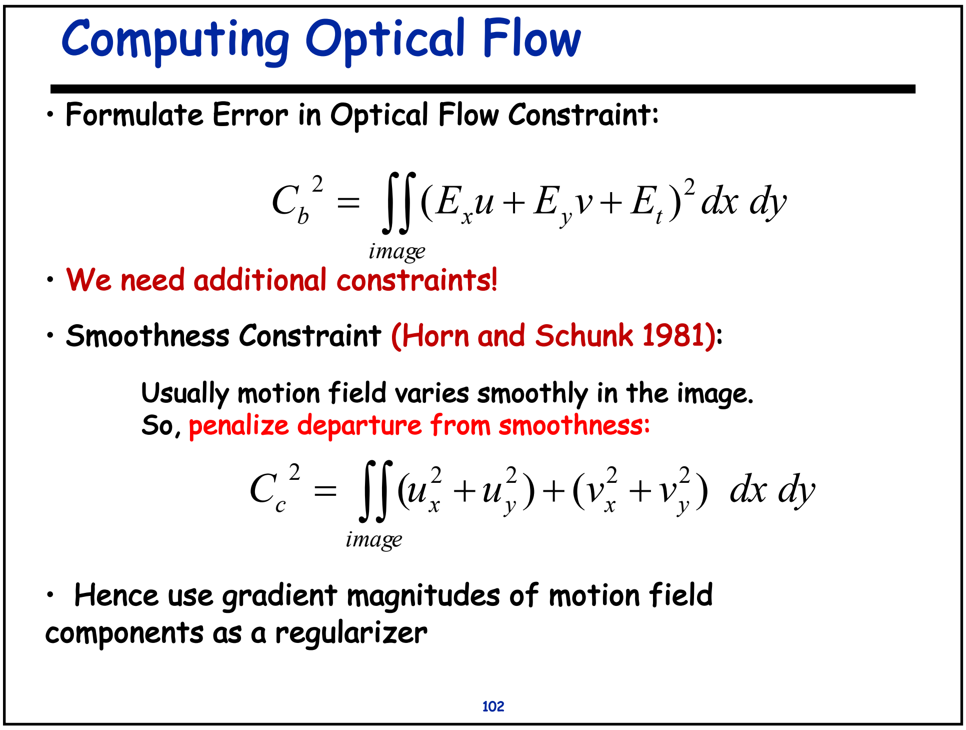
LO 4. Construct least squares solutions to problems in computer vision

We've seen various Applications such as :
- Motion Segmentation
- Shot Boundary Detection in Video

…

Work on your 4th Homework Assignment

Reading assignment for Motion Detection: Chapter Motion Segmentation of the book: Digital Image Processing by Gonzalez and Woods



Computing Optical Flow

- Formulate Error in Optical Flow Constraint:

$$C_b^{\ 2} = \iint_{image} (E_x u + E_y v + E_t)^2 \, dx \, dy$$

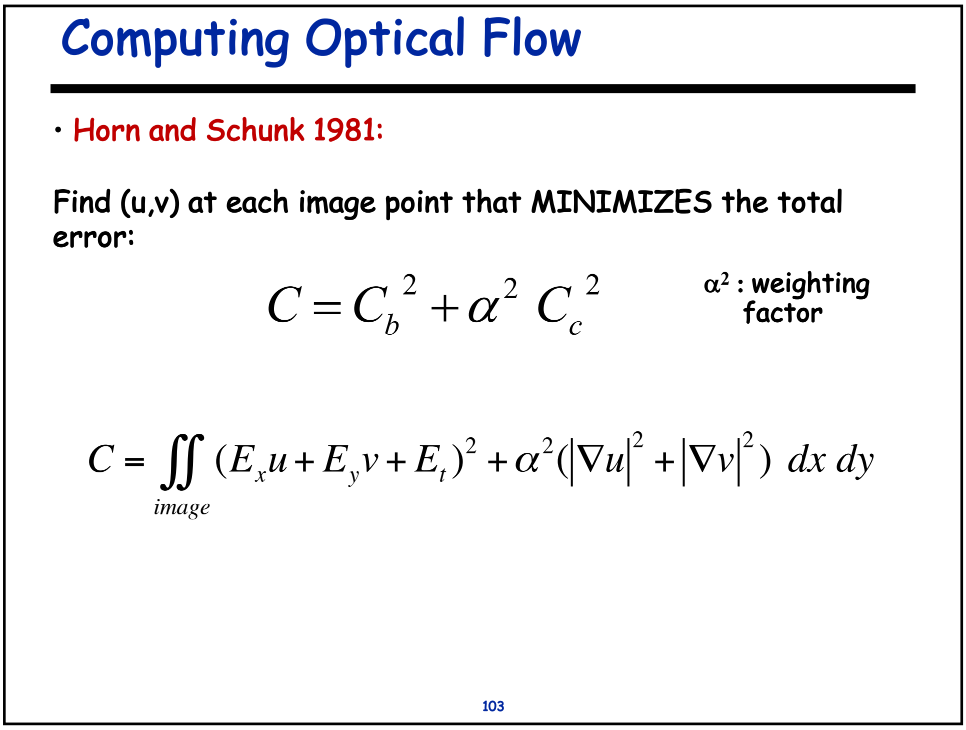
- We need additional constraints!
- Smoothness Constraint (Horn and Schunk 1981):

Usually motion field varies smoothly in the image.
So, penalize departure from smoothness:

$$C_c^{\ 2} = \iint_{image} (u_x^2 + u_y^2) + (v_x^2 + v_y^2) \ \ dx \, dy$$

- Hence use gradient magnitudes of motion field components as a regularizer

102



Computing Optical Flow

- Horn and Schunk 1981:

Find (u,v) at each image point that MINIMIZES the total error:

$$C = C_b^{\ 2} + \alpha^2 \ C_c^{\ 2}$$

α^2 : weighting factor

$$C = \iint_{image} (E_x u + E_y v + E_t)^2 + \alpha^2 (|\nabla u|^2 + |\nabla v|^2) \ dx \, dy$$

103

Computing Optical Flow (OF)

* Minimize the total error C using calculus of variations: Use Euler-Lagrange (E-L) equations to find the necessary condition for a minimizer of C

$$C = \iint_{image} (E_x u + E_y v + E_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2)\ dx\ dy$$

For an integrand with unknown u:

$$C(u) = \int_{\Omega} f(u, \nabla u, x)\,dx$$

Necessary condition that minimizes C is given by: →

$$f_u - div(f_{u'}) = 0$$

Note: The above E-L conditions are re-visited in advanced classes. Just know that it is possible to take a functional derivative to estimate an unknown function in a cost functional !

Horn & Schunk OF: Minimization

* Minimize the total error C using its Euler-Lagrange eqns:

$$E_x^2 u + E_x E_y v = \alpha^2 \nabla^2 u - E_x E_t,$$
$$E_x E_y u + E_y^2 v = \alpha^2 \nabla^2 v - E_y E_t.$$

Using the approximation of the Laplacian introduced in HS paper (next slide)

$$(\alpha^2 + E_x^2)u + E_x E_y v = (\alpha^2 \bar{u} - E_x E_t),$$
$$E_x E_y u + (\alpha^2 + E_y^2)v = (\alpha^2 \bar{v} - E_y E_t).$$

Horn & Schunk's Optical Flow

- Horn and Schunk,"Determining optical flow", Artificial Intelligence, 1981.

Laplacians of u and v are defined as

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad \text{and} \quad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}$$

Approximate the Laplacians of u and v as follows:

$$\nabla^2 u \approx (\bar{u}_{i,j,k} - u_{i,j,k}) \quad \text{and} \quad \nabla^2 v \approx (\bar{v}_{i,j,k} - v_{i,j,k}),$$

where the local averages $\bar{u}$ and $\bar{v}$ are defined as follows

$$\bar{u}_{i,j,k} = \tfrac{1}{6}\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\} + \tfrac{1}{12}\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\},$$

$$\bar{v}_{i,j,k} = \tfrac{1}{6}\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\} + \tfrac{1}{12}\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\}$$

1/12	1/6	1/12
1/6	-1	1/6
1/12	1/6	1/12

FIG. 3. The Laplacian is estimated by subtracting the value at a point from a weighted average of the values at neighboring points.

Horn & Schunk Optical Flow Algorithm

The determinant of the coefficient matrix equals $\alpha^2(\alpha^2 + E_x^2 + E_y^2)$. Solving for u and v we find that

$$(\alpha^2 + E_x^2 + E_y^2)u = +(\alpha^2 + E_y^2)\bar{u} - E_xE_y\bar{v} - E_xE_t,$$
$$(\alpha^2 + E_x^2 + E_y^2)v = -E_xE_y\bar{u} + (\alpha^2 + E_x^2)\bar{v} - E_yE_t.$$

These equations can be written in the alternate form

$$(\alpha^2 + E_x^2 + E_y^2)(u - \bar{u}) = -E_x[E_x\bar{u} + E_y\bar{v} + E_t],$$
$$(\alpha^2 + E_x^2 + E_y^2)(v - \bar{v}) = -E_y[E_x\bar{u} + E_y\bar{v} + E_t].$$

Horn & Schunk OF: Iterative Solution

- We now have a pair of equations for each point in the image. It would be very costly to solve these eqns simultaneously, as the corresponding matrix is sparse and very large.

- Iterative methods, such as the Gauss-Seidel method can be used.
- We can compute a new set of velocity estimates (u^{n+1}, v^{n+1}) from the estimated derivatives and the average of the previous velocity estimates (u^n, v^n) by:

$$u^{n+1} = \bar{u}^n - E_x[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2),$$
$$v^{n+1} = \bar{v}^n - E_y[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2).$$

Finding Optical Flow (Summary)

Local Methods (Lucas-Kanade) – assume *(u,v)* is locally constant:

$$E_{LK} := K_\sigma * (I_x u + I_y v + I_t)^2$$

- Pros: robust under noise.
- Cons: if image is locally constant, need interpolation steps.

Global Methods (Horn-Schunck) – use global regularization term:

$$E_{HS} := \int_{Spatial} \left((I_x u + I_y v + I_t)^2 + \lambda(|\nabla u|^2 + |\nabla v|^2) \right) dxdy$$

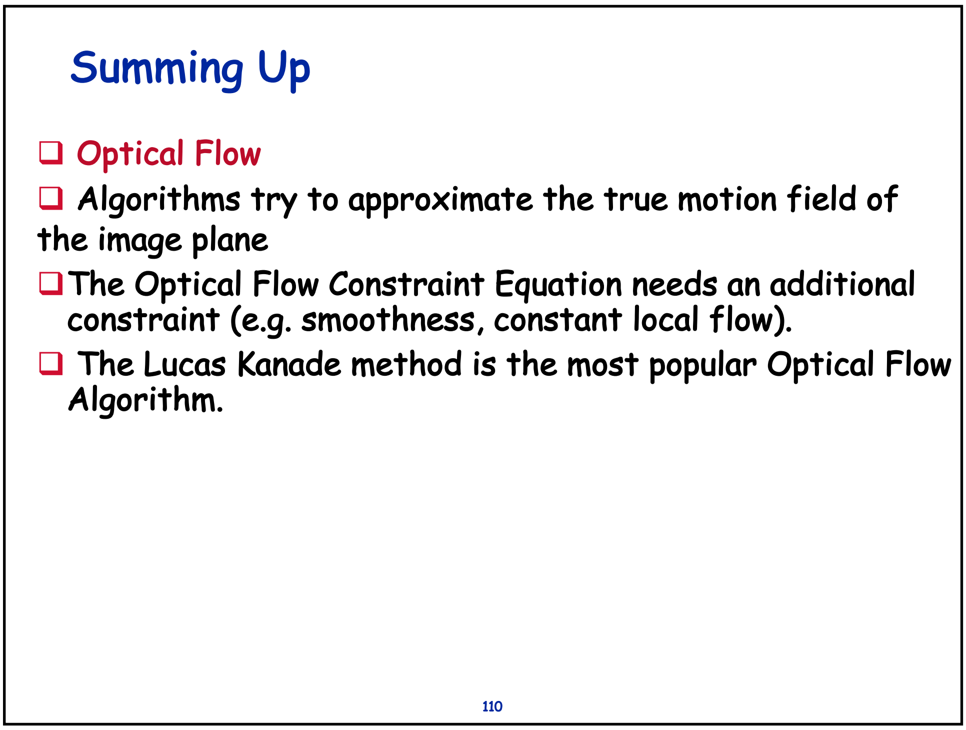
- Pros: automatic filling-in in places where image is constant.
- Cons: less robust under noise.

Combined Local-Global Method (Weickert et al.)

$$E_{CLG} := \int_{Spatial} \left(K_\sigma * (I_x u + I_y v + I_t)^2 + \lambda(|\nabla u|^2 + |\nabla v|^2) \right) dxdy$$

K_σ – smoothing kernel (spatial or spatio-temporal)

Gordon and Milman, Learning Optical Flow slides



Summing Up

- □ Optical Flow
- □ Algorithms try to approximate the true motion field of the image plane
- □ The Optical Flow Constraint Equation needs an additional constraint (e.g. smoothness, constant local flow).
- □ The Lucas Kanade method is the most popular Optical Flow Algorithm.