CAP 6411 Computer Vision Systems

Fall 2002

CAP 6411 Computer Vision Systems

- Instructor: Dr. Mubarak Shah, shah@cs.ucf.edu, 238 CSB, http://www.cs.ucf.edu/courses/cap6411
- Office Hours:
  - 2PM to 3PM Mon, 4PM-5PM Tu, 5:15PM-6:15 PM Wed
- Grading
  - Mid term 20%, Final 25%, Programs 45% , Homework 10%
- Recommended Book, but not required.
  - Digital Video Processing, A. M. Tekalp, Prentice Hall.

Contents

Contents

Lecture-1

Multimedia

- Text
- Graphics
- Audio
- Images
- Video

Imaging Configurations

- Stationary camera stationary objects
- Stationary camera moving objects
- Moving camera stationary objects
- Moving camera moving objects

Video

- sequence of images
- clip
- mosaic
- key frames

Sequence of Images

Clip

Mosaic

Key Frames

Steps in Video Computing

- Acquire (CCD arrays/synthesize (graphics))
- Process (image processing)
- Analyze (computer vision)
- Transmit (compression/networking)
- Store (compression/databases)
- Retrieve (computer vision/databases)
- Browse (computer vision/databases)
- Visualize (graphics)

Motion

- Motion Detection
- Motion Measurement (optical flow)
- Tracking
- Structure from motion (derive 3-D motion & shape)
- Motion Recognition
- Motion-based Recognition

A Video Clip

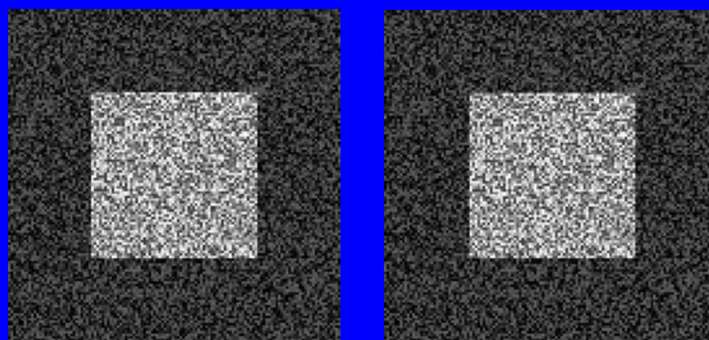
Consecutive Frame Difference

Background Difference

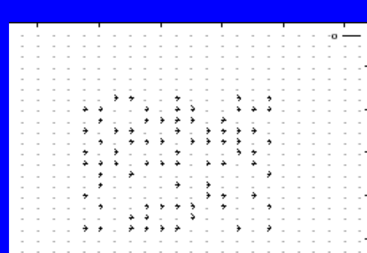
Optical Flow

Measurement of motion at each pixel

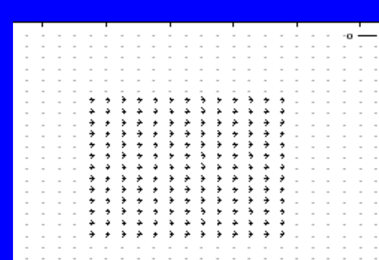
Synthetic Images



Results



One iteration

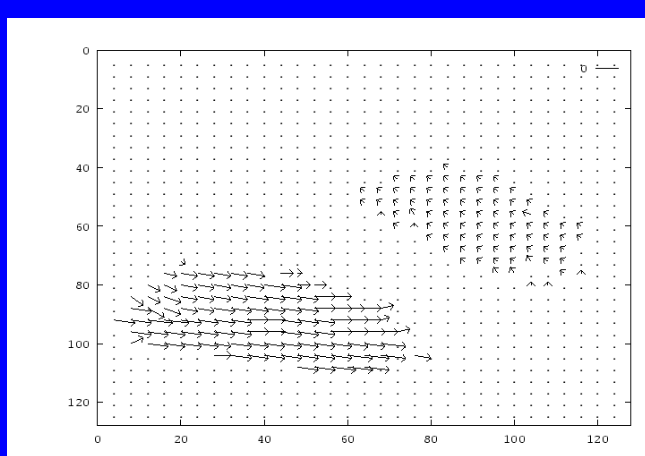


10 iterations

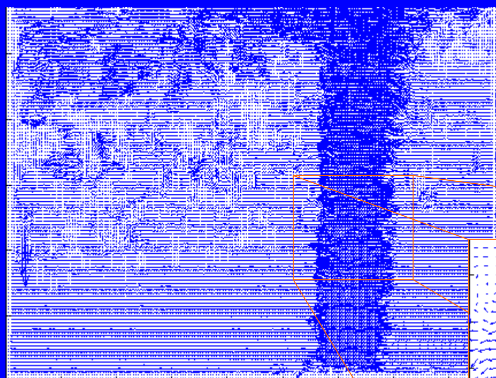
$$\lambda = 4$$

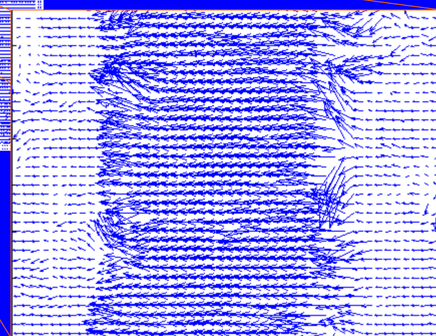
Image from Hamburg Taxi seq

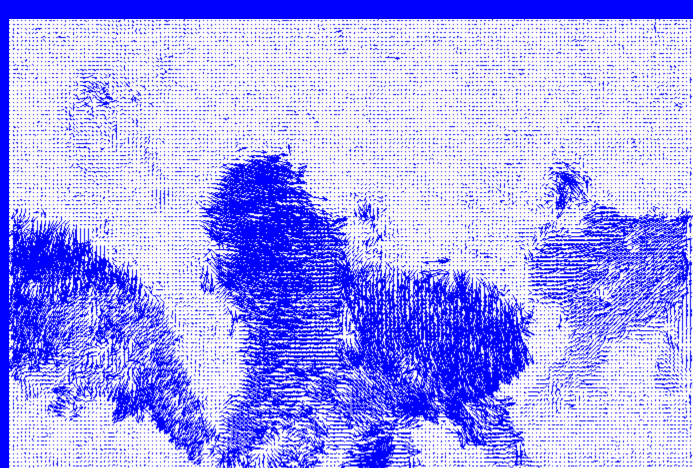
optical flow



Lucas-Kanade with Pyramids Optical Flow

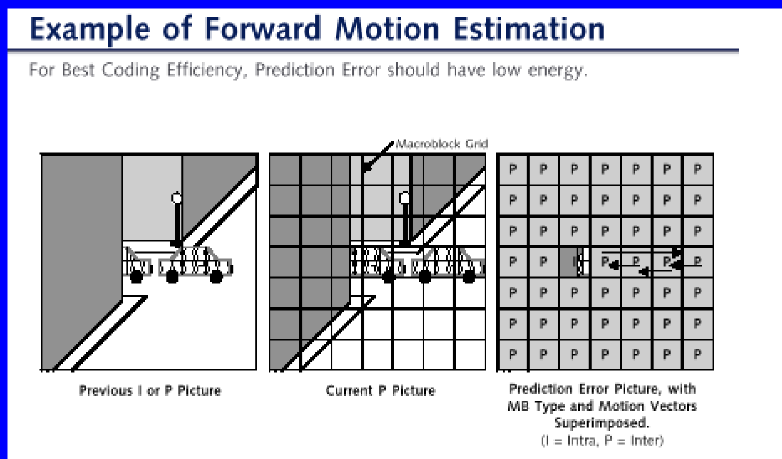




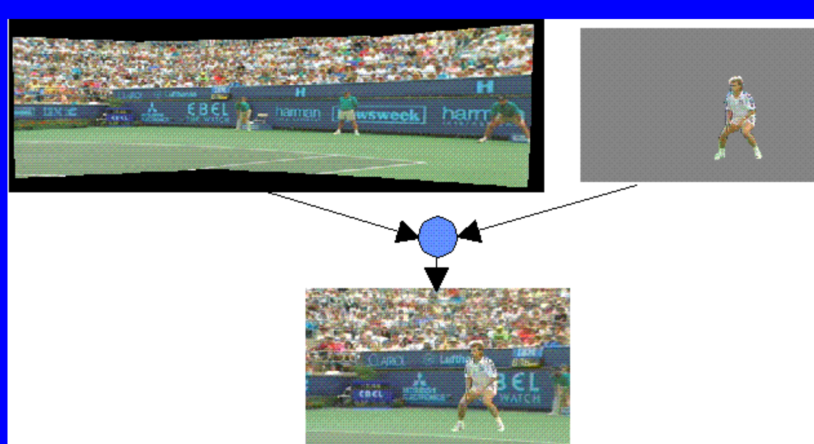


Video Compression

Example



Sprite



Tracking

Tracking & Object Detection In Single Camera

QuickTime™ and a
decompressor
are needed to see this picture.

QuickTime™ and a
decompressor
are needed to see this picture.

PETS-2001

Motion Recognition

Activities

QuickTime™ and a
decompressor
are needed to see this picture.

QuickTime™ and a
decompressor
are needed to see this picture.

QuickTime™ and a
decompressor
are needed to see this picture.

Detecting Violence

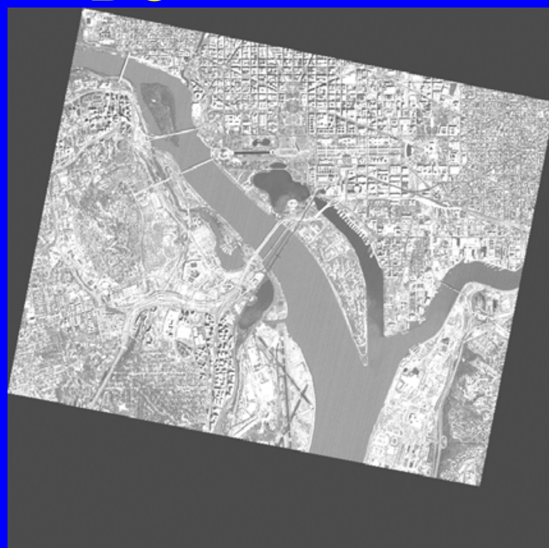
QuickTime™ and a
Microsoft Video 1 decompressor
are needed to see this picture.

QuickTime™ and a
Microsoft Video 1 decompressor
are needed to see this picture.

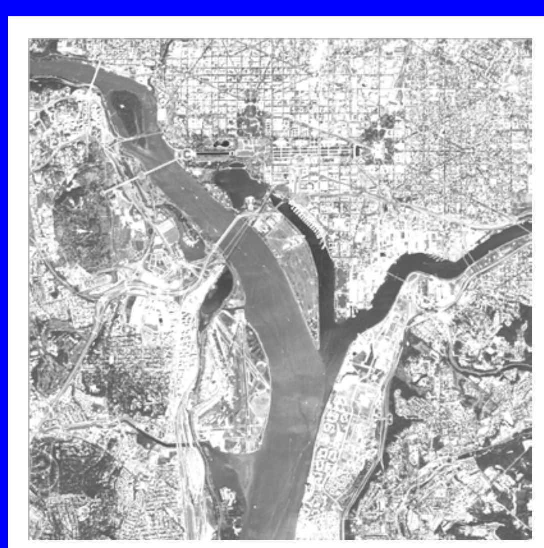
QuickTime™ and a
Microsoft Video 1 decompressor
are needed to see this picture.

Video Registration

IRS-1C - Washington, DC



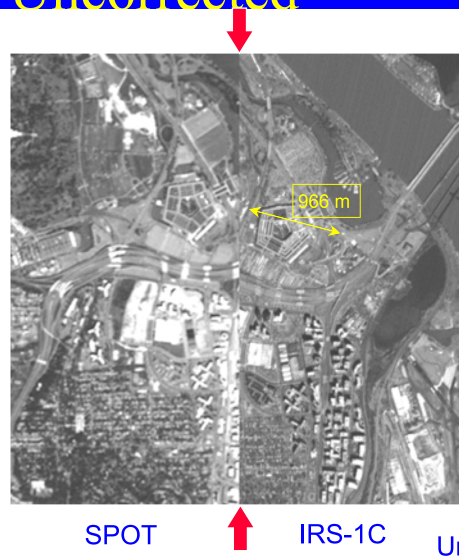
SPOT - Washington, DC



SPOT/IRS-1C
Uncorrected

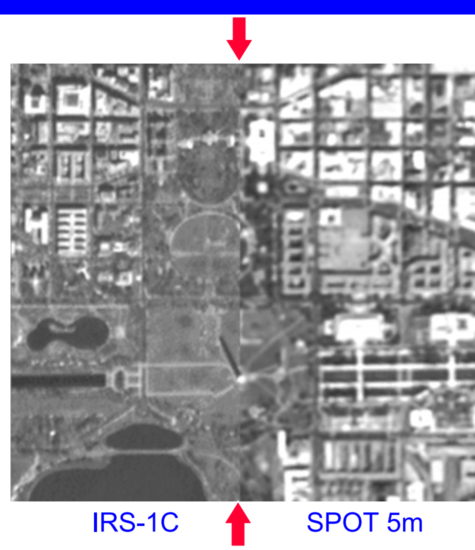
SPOT/IRS-1C
Uncorrected

SPOT/IRS-1C Uncorrected



SPOT IRS-1C Uncorrected

IRS-1C/SPOT Registered



IRS-1C SPOT 5m

Registered IRS-1C to SPOT

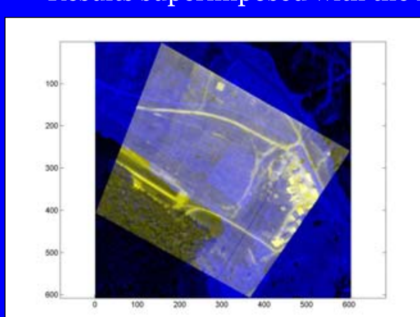
IRS-1C SPOT

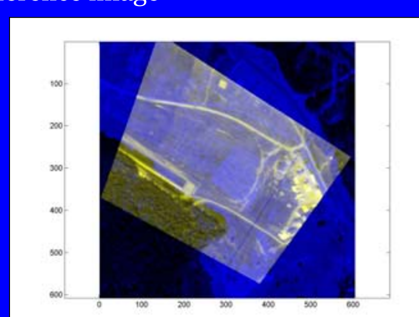
Registered

*geo*Registration

Results and Conclusions

Results superimposed with the reference image





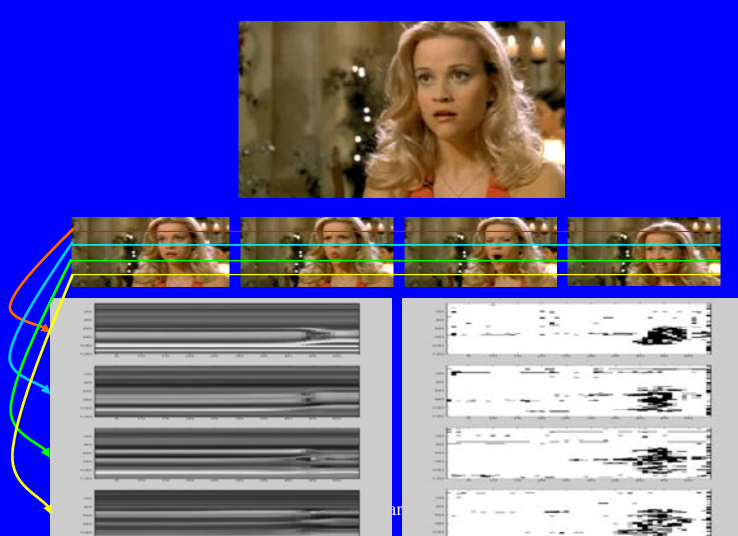
Video Segmentation

Augmented Reality

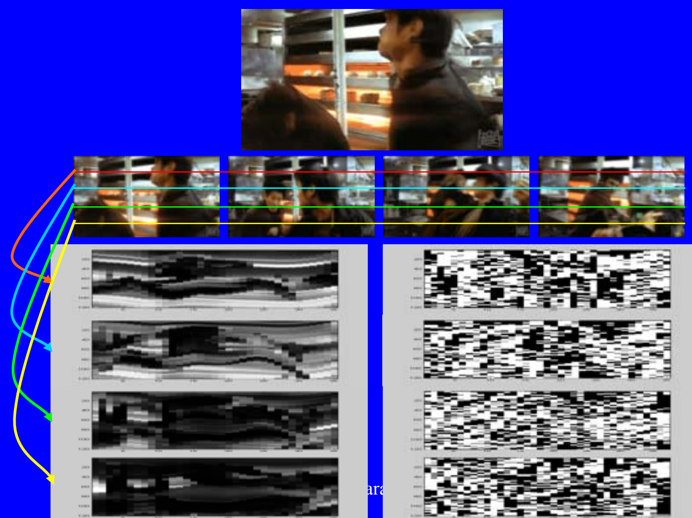
Understanding Hollywood Movies

University of Central Florida VISION

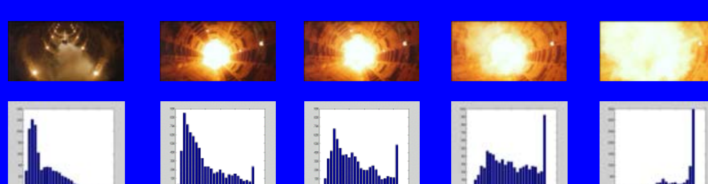
Visual Disturbance



Visual Disturbance



Explosion/fire Detection



Computer Vision

- Measurement of Motion
  - 2-D Motion
    - optical flow
    - point correspondences
  - 3-D Motion (structure from motion (sfm)
    - compute 3D translation, 3D rotation
    - shape from motion (depth)

Computer Vision (contd.)

- Scene Change Detection
  - consecutive frame differencing
  - background differencing
    - median filter
    - pfinder
    - W4
    - Mixture of Gaussians

Computer Vision (contd.)

- Tracking
  - people
  - Vehicles
  - animals

Computer Vision (contd.)

- Video Recognition
  - activity recognition
  - gesture recognition
  - facial expression recognition
  - lipreading
- Video Segmentation
  - shots
  - scenes
  - stories
  - key frames

Image Processing

- Filtering
- Compression
  - MPEG-1
  - MPEG-2
  - MPEG-4
  - MPEG-7 (Multimedia Content Description Interface)

Databases

- Storage
- Retrieval
- Video on demand
- Browsing
  - skim
  - abstract
  - key frames
  - mosaics

Networking

- Transmission
- ATM

Computer Graphics

- Visualization
- Image-based Rendering and Modeling
- Augmented Reality

Video Computing

- Computer Vision
- Image Processing
- Computer Graphics
- Databases
- Networks

Lecture-2

PART I

Measurement of Motion

Contents

- Image Motion Models
- Optical Flow Methods
  - Horn & Schunck
  - Lucas and Kanade
  - Anandan et al
  - Szeliski
  - Mann & Picard
- Video Mosaics

3-D Rigid Motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

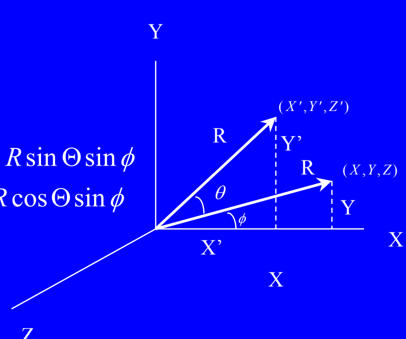
Rotation matrix (9 unknowns)

Translation (3 unknowns)

Rotation

$$X = R\cos\phi$$

$$Y = R\sin\phi$$



$$X' = R\cos(\Theta+\phi) = R\cos\Theta\cos\phi - R\sin\Theta\sin\phi$$

$$Y' = R\sin(\Theta+\phi) = R\sin\Theta\cos\phi + R\cos\Theta\sin\phi$$

$$X' = X\cos\Theta - Y\sin\Theta$$

$$Y' = X\sin\Theta + Y\cos\Theta$$

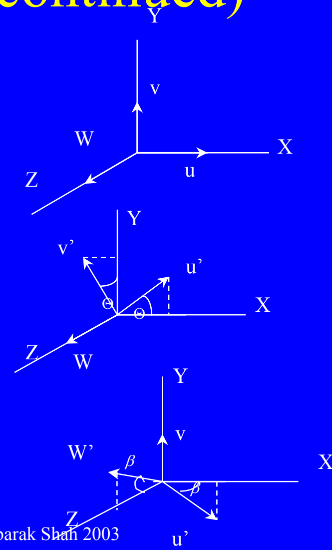
$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Rotation (continued)

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$



Euler Angles

$$R = R_Z^{\alpha} R_Y^{\beta} R_X^{\gamma} = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

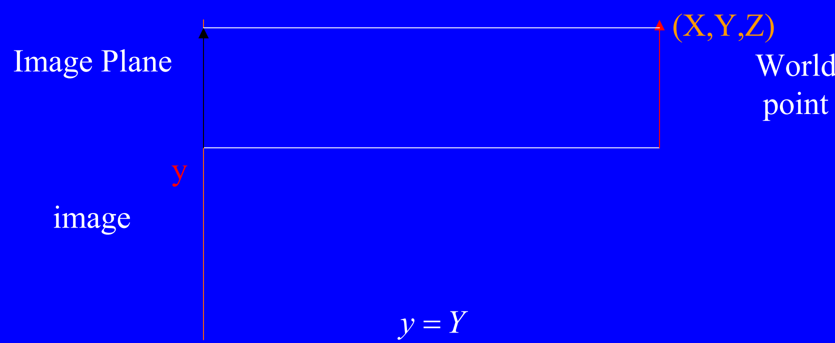
if angles are small($\cos\Theta \approx 1$ $\sin\Theta \approx \Theta$

$$R = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix}$$

Image Motion Models

Displacement Model

Orthographic Projection



$$y = Y$$
$$x = X$$

Orthographic Projection

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$x = X$$
$$y = Y$$

(x,y)=image coordinates,
(X,Y,Z)=world coordinates

$$x' = r_{11}x + r_{12}y + (r_{13}Z + T_X)$$
$$y' = r_{21}x + r_{22}y + (r_{23}Z + T_Y)$$

$$x' = a_1 x + a_2 y + b_1$$
$$y' = a_3 x + a_4 y + b_2$$

Affine Transformation

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b} \qquad \mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

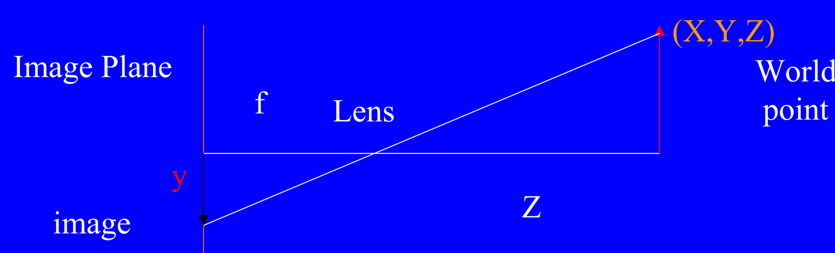
Orthographic Projection (contd.)

$$\begin{bmatrix} X' \\ Y \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$x' = x - \alpha y + \beta Z + T_X$$

$$y' = \alpha x + y - \gamma Z + T_Y$$

Perspective Projection



$$\frac{-y}{Y} = \frac{f}{Z}$$

$$y = -\frac{fY}{Z} \qquad x = -\frac{fX}{Z}$$

Perspective Projection

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$X' = r_{11}X + r_{12}Y + r_{13}Z + T_X$$
$$Y' = r_{21}X + r_{22}Y + r_{23}Z + T_Y$$
$$Z' = r_{31}X + r_{32}Y + r_{33}Z + T_Z$$

$$x' = \frac{X'}{Z'} \qquad y' = \frac{Y'}{Z'}$$

focal length = -1

$$x' = \frac{r_{11}x + r_{12}y + r_{13} + \frac{T_X}{Z}}{r_{31}x + r_{32}y + r_{33} + \frac{T_Z}{Z}}$$

← scale ambiguity

$$y' = \frac{r_{21}x + r_{22}y + r_{23} + \frac{T_Y}{Z}}{r_{31}x + r_{32}y + r_{33} + \frac{T_Z}{Z}}$$

Plane+Perspective(projective)

$$aX + bY + cZ = 1$$

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = 1$$

equation of a plane

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T$$

3d rigid motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad A = R + T\begin{bmatrix} a & b & c \end{bmatrix}$$

Plane+Perspective(projective)

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$x' = \frac{X'}{Z'} \quad y' = \frac{Y'}{Z'}$$

focal length = -1

$$X' = a_1X + a_2Y + a_3Z$$
$$Y' = a_4X + a_5Y + a_6Z$$
$$Z' = a_7X + a_8Y + a_9Z$$

$$x' = \frac{a_1X + a_2Y + a_3Z}{a_7X + a_8Y + a_9Z}$$

$$y' = \frac{a_4X + a_5Y + a_6Z}{a_7X + a_8Y + a_9Z}$$

$$x' = \frac{a_1x + a_2y + a_3}{a_7x + a_8y + a_9}$$

$$y' = \frac{a_4x + a_5y + a_6}{a_7x + a_8y + a_9}$$

$a_9 = 1$

scale ambiguity

Plane+perspective (contd.)

$$x' = \frac{a_1x + a_2y + a_3}{a_7x + a_8y + 1}$$

$$y' = \frac{a_4x + a_5y + a_6}{a_7x + a_8y + 1}$$

$$\mathbf{X}' = \frac{\mathbf{A}\mathbf{X} + b}{C^T\mathbf{X} + 1}$$

$$\mathbf{X}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix},$$

$$b = \begin{bmatrix} a_3 \\ a_6 \end{bmatrix}, C = \begin{bmatrix} a_7 \\ a_8 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Projective

Least Squares

- Eq of a line

$$mx + c = y$$

  - Consider n points

$$mx_1 + c = y_1$$
$$\vdots$$
$$mx_n + c = y_n$$

$$\begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ \\ y_n \end{bmatrix}$$

$$\mathbf{A}p = Y$$

Least Squares Fit

$$\mathbf{A}p = Y$$

$$\mathbf{A}^T\mathbf{A}p = \mathbf{A}^T Y$$

$$p = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T Y$$

$$\min \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

Projective

- If point correspondences (x,y)<-->(x',y') are known
- a's can be determined by least squares fit

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

$$\begin{bmatrix} & & & & \vdots & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i' & -y_i y_i' \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

Affine

- If point correspondences (x,y)<-->(x',y') are known
- a's and b's can be determined by least squares fit

$$x' = a_1 x + a_2 y + b_1$$
$$y' = a_3 x + a_4 y + b_2$$

$$\begin{bmatrix} & & & \vdots & & \\ x_i & y_i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_i & y_i & 1 \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

Summary of Displacement Models

Translation

$$x' = x + b_1$$
$$y' = y + b_2$$

Biquadratic

$$x' = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 y^2 + a_6 xy$$
$$y' = a_7 + a_8 x + a_9 y + a_{10} x^2 + a_{11} y^2 a_{12} xy$$

Rigid

$$x' = x\cos\theta - y\sin\theta + b_1$$
$$y' = x\sin\theta + y\cos\theta + b_2$$

Bilinear

$$x' = a_1 + a_2 x + a_3 y + a_4 xy$$
$$y' = a_5 + a_6 x + a_7 y + a_8 xy$$

Affine

$$x' = a_1 x + a_2 y + b_1$$
$$y' = a_3 x + a_4 y + b_2$$

Pseudo Perspective

$$x' = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy$$
$$y' = a_6 + a_7 x + a_8 y + a_4 xy + a_5 y^2$$

Projective

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$
$$y' = \frac{a_3 x + a_4 y + b_1}{c_1 x + c_2 y + 1}$$

Displacement Models (contd)

- Translation
  - simple
  - used in block matching
  - no zoom, no rotation, no pan and tilt
- Rigid
  - rotation and translation
  - no zoom, no pan and tilt

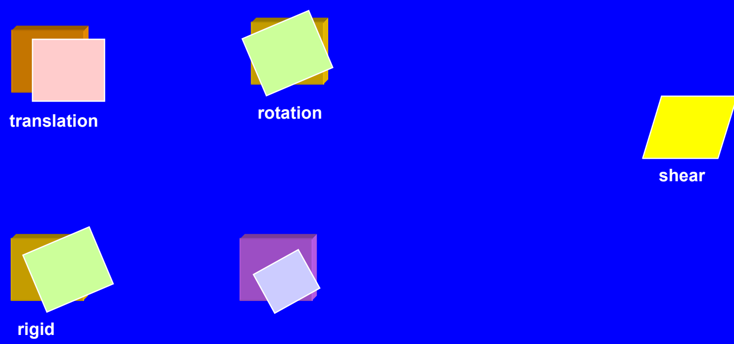
Displacement Models (contd)

- Affine
  - rotation about optical axis only
  - can not capture pan and tilt
  - orthographic projection
- Projective
  - exact eight parameters (3 rotations, 3 translations and 2 scalings)
  - difficult to estimate

Displacement Models (contd)

- Biquadratic
  - obtained by second order Taylor series
  - 12 parameters
- Bilinear
  - obtained from biquadratic model by removing square terms
  - most widely used
  - not related to any physical 3D motion
- Pseudo-perspective
  - obtained by removing two square terms and constraining four remaining to 2 degrees of freedom

Spatial Transformations



affine

Decomposition of Affine

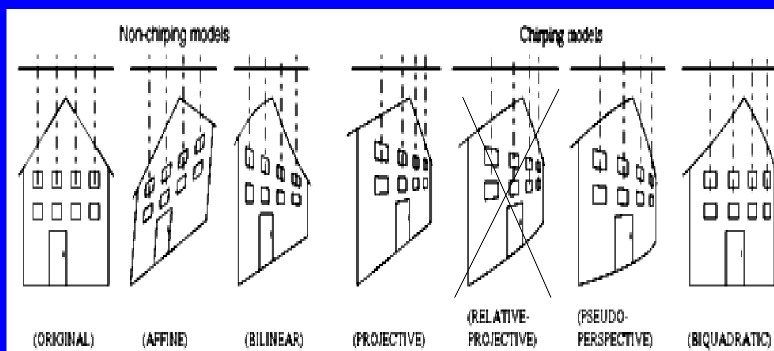
$$A = SVD = S(DD^{-1})VD = (SD)(D^{-1}VD)$$

$$= R(\alpha)C = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} v_1 & v_h \\ v_h & v_2 \end{bmatrix}$$

$$A = \Delta \begin{bmatrix} \frac{1}{\rho} & 0 \\ \alpha & \rho \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix}$$

$$\Delta = scale\_factor = \sqrt{s_x s_y}, \quad \rho = scale\_ratio = \sqrt{\frac{s_x}{s_y}}, \quad \alpha = \text{skew}$$

Displacement Models (contd)



Affine Mosaic

Projective Mosaic

Instantaneous Velocity Model

3-D Rigid Motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} X'-X \\ Y'-Y \\ Z'-Z \end{bmatrix} = \begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \left(\begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

3-D Rigid Motion

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \Omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix}$$

$$\dot{\mathbf{X}} = \Omega \times \mathbf{X} + \mathbf{V}$$

Cross Product

Orthographic Projection

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

$$y = Y$$
$$x = X$$

$$u = \dot{x} = \Omega_2 Z - \Omega_3 y + V_1$$
$$v = \dot{y} = \Omega_3 x - \Omega_1 Z + V_2$$

(u,v) is optical flow

Plane+orthographic(Affine)

$$Z = a + bX + cY$$

$$u = V_1 + \Omega_2 Z - \Omega_3 y$$
$$v = V_2 + \Omega_3 x - \Omega_1 Z$$

$$u = b_1 + a_1 x + a_2 y$$
$$v = b_2 + a_3 x + a_4 y$$

$$\Downarrow$$

$$\mathbf{u} = \mathbf{A}\,\mathbf{x} + \mathbf{b}$$

$$b_1 = V_1 + a\Omega_2$$
$$a_1 = b\Omega_2$$
$$a_2 = c\Omega_2 - \Omega_3$$

$$b_2 = V_2 - a\Omega_1$$
$$a_3 = \Omega_3 - b\Omega_1$$
$$a_4 = -c\Omega_1$$

Perspective Projection (arbitrary flow)

$$x = \frac{fX}{Z} \qquad u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f\frac{\dot{X}}{Z} - x\frac{\dot{Z}}{Z}$$

$$y = \frac{fY}{Z} \qquad v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f\frac{\dot{Y}}{Z} - y\frac{\dot{Z}}{Z}$$

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

$$u = f(\frac{V_1}{Z} + \Omega_2) - \frac{V_3}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = f(\frac{V_2}{Z} - \Omega_1) + \Omega_3 x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

Plane+Perspective (pseudo perspective)

$$u = f(\frac{V_1}{Z} + \Omega_2) - \frac{V_3}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2 \qquad Z = a + bX + cY$$

$$v = f(\frac{V_2}{Z} - \Omega_1) + \Omega_3 x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2 \qquad \frac{1}{Z} = \frac{1}{a} - \frac{b}{a}x - \frac{c}{a}y$$

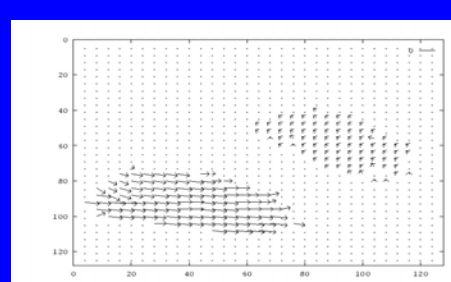
$$\Downarrow$$

$$u = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy$$
$$v = a_6 + a_7 x + a_8 y + a_4 xy + a_5 y^2$$

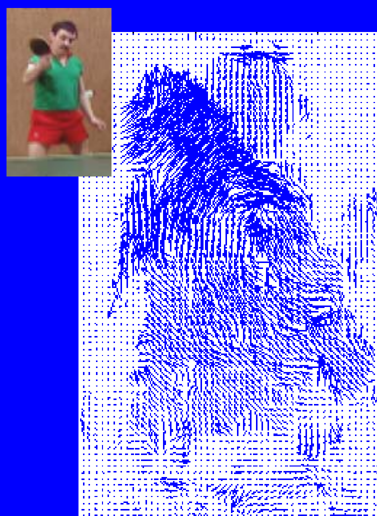
Lecture-3

Computing Optical Flow

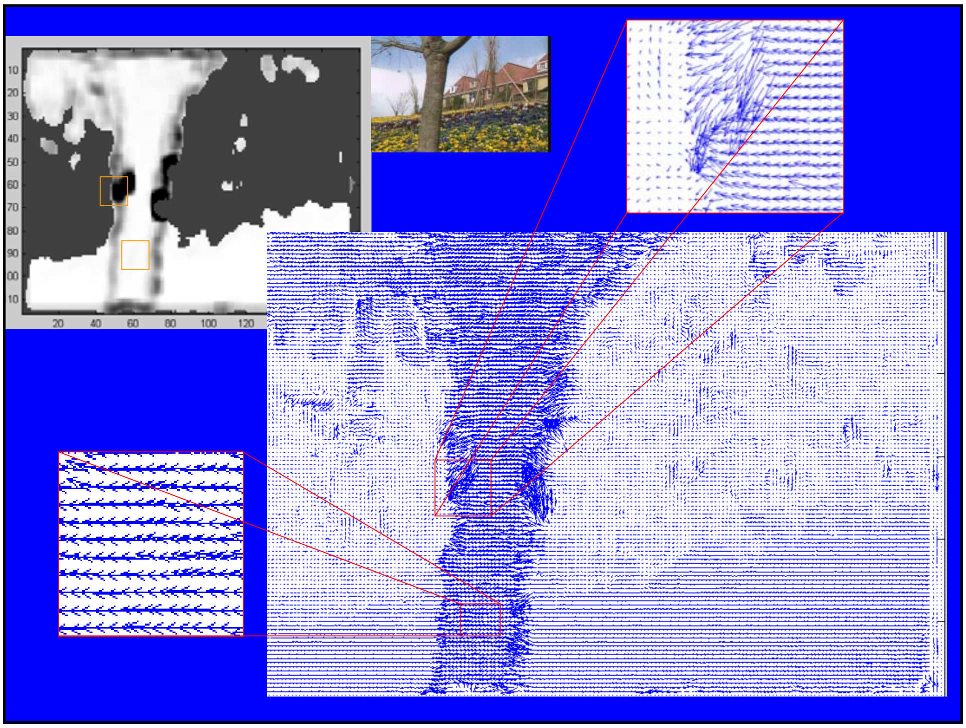
Hamburg Taxi seq



Copyright Mubarak Shah 2003




CLAIROL



Horn&Schunck Optical Flow

$f(x, y, t)$ Image Sequence

$$\frac{df(x,y,t)}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial t} = 0$$

$f_x u + f_y v + f_t = 0$ brightness constancy eq

Horn&Schunck Optical Flow

$$f(x,y,t) = f(x+dx, y+dy, t+dt)$$

⇩ Taylor Series

$$f(x,y,t) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt$$

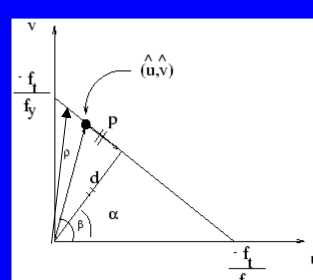
$$f_x dx + f_y dy + f_t dt = 0$$

$$f_x u + f_y v + f_t = 0$$ brightness constancy eq

Interpretation of optical flow eq

$$f_x u + f_y v + f_t = 0$$

$$v = -\frac{f_x}{f_y}u - \frac{f_t}{f_y}$$



d=normal flow

p=parallel flow

$$d = \frac{f_t}{\sqrt{f_x^2 + f_y^2}}$$

Equation of st.line

Horn&Schunck (contd)

$$\iint \{(f_x u + f_y v + f_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_y^2)\}\,dx\,dy$$

min

$$(f_x u + f_y v + f_t) f_x + \lambda(\Delta^2 u) = 0$$

$$(f_x u + f_y v + f_t) f_y + \lambda((\Delta^2 v) = 0$$

discrete version

$$(f_x u + f_y v + f_t) f_x + \lambda(u - u_{av}) = 0$$

$$(f_x u + f_y v + f_t) f_y + \lambda((v - v_{av}) = 0$$

variational calculus

$$u = u_{av} - f_x \frac{P}{D}$$

$$v = v_{av} - f_y \frac{P}{D}$$

$$P = f_x u_{av} + f_y v_{av} + f_t$$

$$D = \lambda + f_x^2 + f_y^2$$

$$\Delta^2 u = u_{xx} + u_{yy}$$

Algorithm-1

- k=0
- Initialize $u^K \quad v^K$
- Repeat until some error measure is satisfied (converges)

$$u^K = u_{av}^{k-1} - f_x \frac{P}{D}$$

$$v^k = v_{av}^{K-1} - f_y \frac{P}{D}$$

$$P = f_x u_{av} + f_y v_{av} + f_t$$

$$D = \lambda + f_x^2 + f_y^2$$

Derivatives

- Derivative: Rate of change of some quantity
  - Speed is a rate of change of a distance
  - Acceleration is a rate of change of speed

Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

$$v = \frac{ds}{dt} \qquad \text{speed}$$

$$a = \frac{dv}{dt} \qquad \text{acceleration}$$

Examples

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

$$y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = \cos x + (-1)e^{-x}$$

Second Derivative

$$\frac{df_x}{dx} = f''(x) = f_{xx}$$

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

$$\frac{d^2y}{dx^2} = 2 + 12x^2$$

Discrete Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

(Finite Difference)

Discrete Derivative

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$ Left difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$ Right difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$ Center difference

Example

	F(x)=10	10	10	10	20	20	20
Left difference	F'(x)=0	0	0	0	10	0	0
	F''(x)=0	0	0	0	10	-10	0

-1 1 left difference
1 -1 right difference
-1 0 1 center difference

Derivatives in Two Dimensions

(partial Derivatives)

$$f(x,y)$$

$$\frac{\partial f}{\partial x} = f_x = \lim_{\Delta x \to 0} \frac{f(x,y) - f(x-\Delta x, y)}{\Delta x}$$

$$\frac{\partial f}{\partial y} = f_y = \lim_{\Delta y \to 0} \frac{f(x,y) - f(x, y-\Delta y)}{\Delta y}$$

(f_x, f_y) Gradient Vector

$$\text{magnitude} = \sqrt{(f_x^2 + f_y^2)}$$

$$\text{direction} = \theta = \tan^{-1} \frac{f_y}{f_x}$$

$$\Delta^2 f = f_{xx} + f_{yy} = \text{Laplacian}$$

Derivatives of an Image

Derivative & average

$$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad \begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

$$f_x \qquad\qquad f_y$$

Prewit

$$I(x,y) = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 30 & 30 & 0 & 0 \\ 0 & 30 & 30 & 0 & 0 \\ 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of an Image

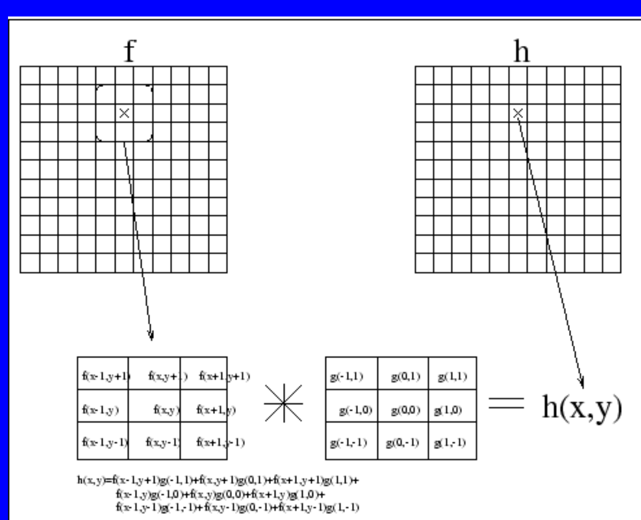
$$I(x,y) = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Laplacian

$$
\begin{matrix}
0 & -\frac{1}{4} & 0 \\
-\frac{1}{4} & 1 & -\frac{1}{4} \\
0 & -\frac{1}{4} & 0
\end{matrix}
$$

$$f_{xx} + f_{yy}$$

Convolution



Convolution (contd)

$$h(x,y)=\sum_{i=-1}^{1}\sum_{j=-1}^{1} f(x+i,y+j)g(i,j)$$

$$h(x,y)=f(x,y)*g(x,y)$$

$$\begin{aligned}h(x,y)=&f(x-1,y-1)g(-1,-1)+f(x,y-1)g(0,-1)+f(x+1,y-1)g(1,-1)\\&+f(x-1,y)g(-1,0)+f(x,y)g(0,0)+f(x+1,y)g(1,0)\\&+f(x-1,y+1)g(-1,1)+f(x,y+1)g(0,1)+f(x+1,y+1)g(1,1)\end{aligned}$$

Derivative Masks

$\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}$ first image

$\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}$ second image

f_x

$\begin{bmatrix}-1 & -1\\ 1 & 1\end{bmatrix}$ first image

$\begin{bmatrix}-1 & -1\\ 1 & 1\end{bmatrix}$ second image

f_y

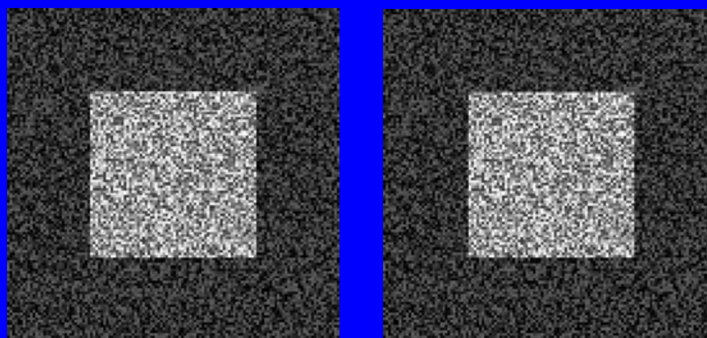
$\begin{bmatrix}-1 & -1\\ -1 & -1\end{bmatrix}$ first image

$\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$ second image

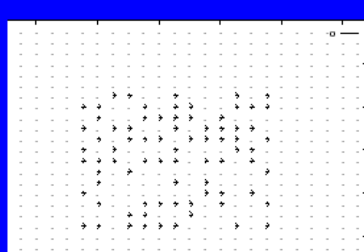
f_t

Apply first mask to 1st image
Second mask to 2nd image
Add the responses to get f_x

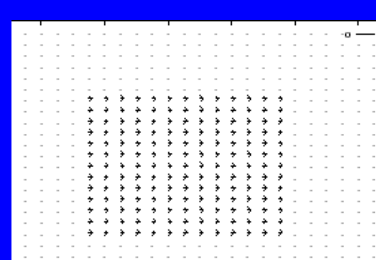
Synthetic Images



Results



One iteration



10 iterations

$$\lambda = 4$$

Homework Due 9/9/02

- Derive Euler Angles matrix from three rotations around x, y and Z. (Lecture-2, page 3, do not hand in).
- Derive bi-quadratic motion model from the projective motion model using Taylor series. (Lecture-2, page 11).
- Verify 3-D rigid motion using instantaneous motion model can be written as a cross product of rotational velocities Ω and object location (X). Lecture-2, page 16.
- Verify that pseudo perspective motion model can be derived assuming planar scene and perspective projection. Lecture-2, page 18.

Pyrramids

Lecture-4

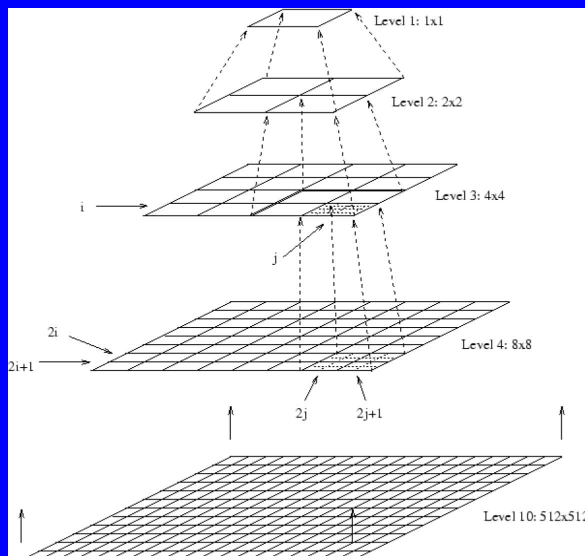
Comments

- Horn-Schunck optical method (Algorithm-1) works only for small motion.
- If object moves faster, the brightness changes rapidly, 2x2 or 3x3 masks fail to estimate spatiotemporal derivatives.
- Pyramids can be used to compute large optical flow vectors.

Pyramids

- Very useful for representing images.
- Pyramid is built by using multiple copies of image.
- Each level in the pyramid is 1/4 of the size of previous level.
- The lowest level is of the highest resolution.
- The highest level is of the lowest resolution.

Pyramid

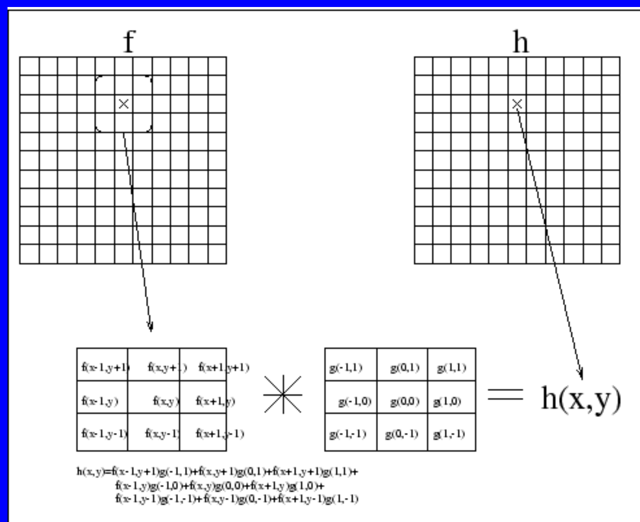


Gaussian Pyramids

$$g_l(i,j)=\sum_{m=-2}^{2}\sum_{n=-2}^{2}w(m,n)g_{l-1}(2i+m,2j+n)$$

$$g_l = REDUCE[g_{l-1}]$$

Convolution



Gaussian Pyramids

$$g_{l,n}(i,j) = \sum_{p=-2}^{2}\sum_{q=-2}^{2} w(p,q)\, g_{l,n-1}\left(\frac{i-p}{2}, \frac{j-q}{2}\right)$$

$$g_{l,n} = EXPAND[g_{l,n-1}]$$

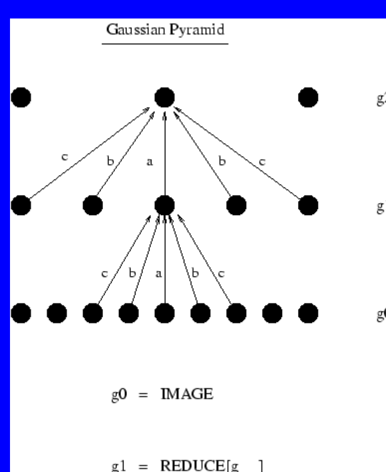
Reduce (1D)

$$g_l(i) = \sum_{m=-2}^{2} \hat{w}(m) g_{l-1}(2i+m)$$

$$g_l(2) = \hat{w}(-2)g_{l-1}(4-2) + \hat{w}(-1)g_{l-1}\hat{w}(4-1) +$$
$$\hat{w}(0)g_{l-1}(4) + \hat{w}(1)g_{l-1}(4+1) + \hat{w}(2)g_{l-1}(4+2)$$

$$g_l(2) = \hat{w}(-2)g_{l-1}(2) + \hat{w}(-1)g_{l-1}\hat{w}(3) +$$
$$\hat{w}(0)g_{l-1}(4) + \hat{w}(1)g_{l-1}(5) + \hat{w}(2)g_{l-1}(6)$$

Reduce



Expand (1D)

$$g_{l,n}(i) = \sum_{p=-2}^{2} \hat{w}(p) g_{l,n-1}(\frac{i-p}{2})$$

$$g_{l,n}(4) = \hat{w}(-2) g_{l,n-1}(\frac{4-2}{2}) + \hat{w}(-1) g_{l,n-1}(\frac{4-1}{2}) +$$

$$\hat{w}(0) g_{l,n-1}(\frac{4}{2}) + \hat{w}(1) g_{l,n-1}(\frac{4+1}{2}) + \hat{w}(2) g_{l,n-1}(\frac{4+2}{2})$$

$$g_{l,n}(4) = \hat{w}(-2) g_{l,n-1}(1) + \hat{w}(0) g_{l,n-1}(2) + \hat{w}(2) g_{l,n-1}(3)$$

Expand (1D)

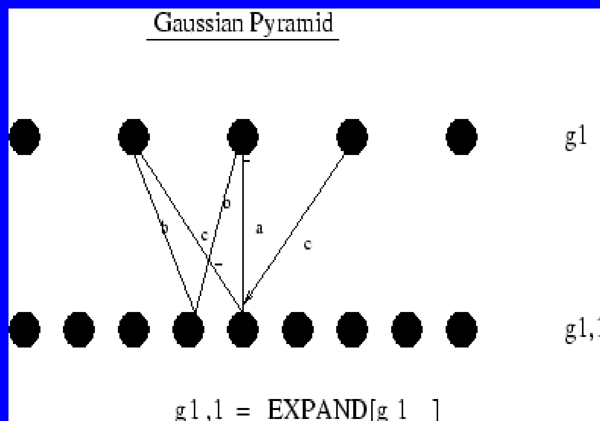
$$g_{l,n}(i) = \sum_{p=-2}^{2} \hat{w}(p) g_{l,n-1}(\frac{i-p}{2})$$

$$g_{l,n}(3) = \hat{w}(-2) g_{l,n-1}(\frac{3-2}{2}) + \hat{w}(-1) g_{l,n-1}(\frac{3-1}{2}) +$$

$$\hat{w}(0) g_{l,n-1}(\frac{3}{2}) + \hat{w}(1) g_{l,n-1}(\frac{3+1}{1}) + \hat{w}(2) g_{l,n-1}(\frac{3+2}{2})$$

$$g_{l,n}(3) = \hat{w}(-1) g_{l,n-1}(1) + \hat{w}(1) g_{l,n-1}(2)$$

Expand



Convolution Mask

$$[w(-2), w(-1), w(0), w(1), w(2)]$$

Convolution Mask

- Separable

$$w(m,n) = \hat{w}(m)\hat{w}(n)$$

•Symmetric

$$\hat{w}(i) = \hat{w}(-i)$$

$$[c, b, a, b, c]$$

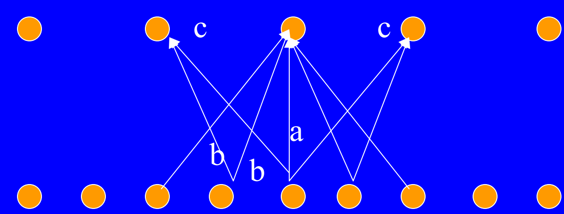
Convolution Mask

- The sum of mask should be 1.

$$a + 2b + 2c = 1$$

•All nodes at a given level must contribute the same total weight to the nodes at the next higher level.

$$a + 2c = 2b$$



Convolution Mask

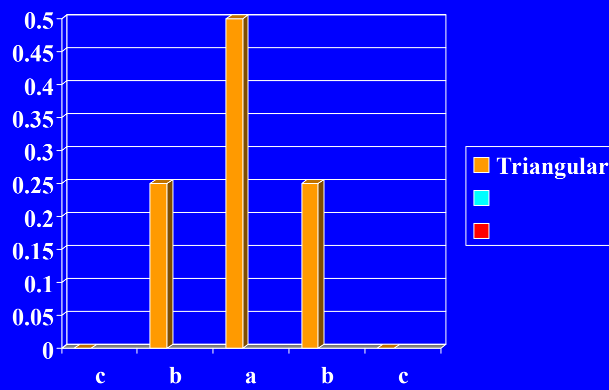
$$\hat{w}(0) = a$$

$$\hat{w}(-1) = \hat{w}(1) = \frac{1}{4}$$

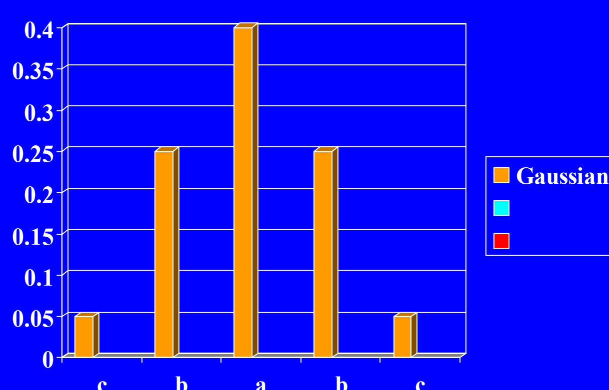
$$\hat{w}(-2) = \hat{w}(2) = \frac{1}{4} - \frac{a}{2}$$

a=.4 GAUSSIAN, a=.5 TRINGULAR

Triangular



Approximate Gaussian



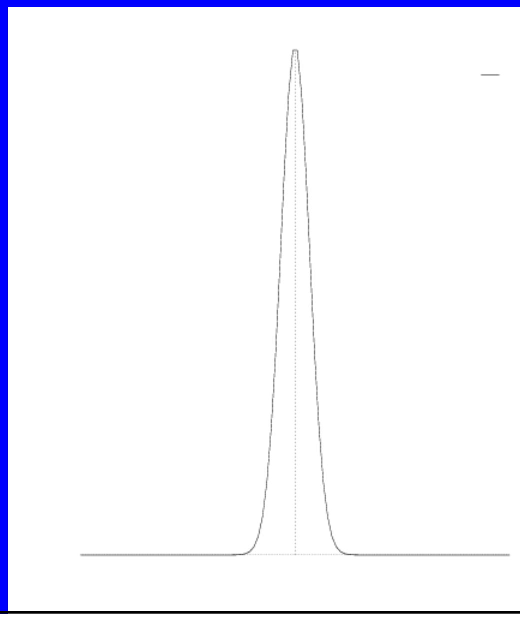
Gaussian

$$g(x) = e^{\frac{-x^2}{2o^2}}$$

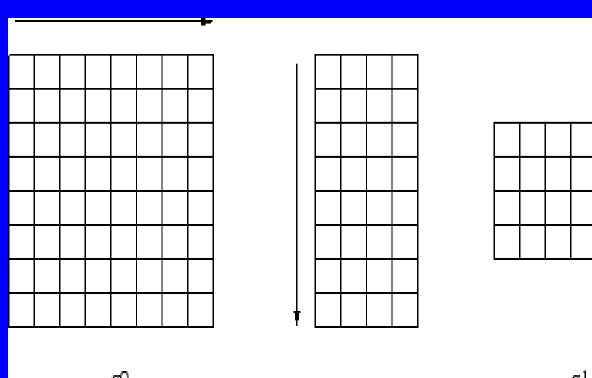
Gaussian

$$g(x) = e^{\frac{-x^2}{2o^2}}$$

x	-3	-2	-1	0	1	2	3
g(x)	.011	.13	.6	1	.6	.13	.011



Separability



Algorithm

- Apply 1-D mask to alternate pixels along each row of image.
- Apply 1-D mask to each pixel along alternate columns of resultant image from previous step.

Gaussian Pyramid

Laplacian Pyramids

- Similar to edge detected images.
- Most pixels are zero.
- Can be used for image compression.

$$L_1 = g_1 - EXPAND[g_2]$$

$$L_2 = g_2 - EXPAND[g_3]$$

$$L_3 = g_3 - EXPAND[g_4]$$

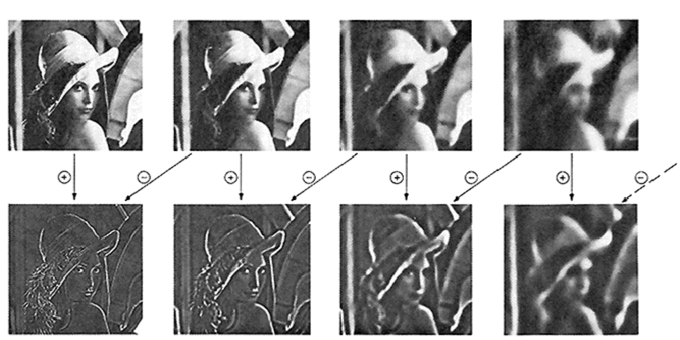


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

Coding using Laplacian Pyramid

- Compute Gaussian pyramid

$$g_1, g_2, g_3, g_4$$

- Compute Laplacian pyramid

$$L_1 = g_1 - EXPAND[g_2]$$
$$L_2 = g_2 - EXPAND[g_3]$$
$$L_3 = g_3 - EXPAND[g_4]$$
$$L_4 = g_4$$

- Code Laplacian pyramid

Decoding using Laplacian pyramid

- Decode Laplacian pyramid.
- Compute Gaussian pyramid from Laplacian pyramid.

$$g_4 = L_4$$
$$g_3 = EXPAND[g_4] + L_3$$
$$g_2 = EXPAND[g_3] + L_2$$
$$g_1 = EXPAND[g_2] + L_1$$

- g_1 is reconstructed image.

Gaussian

- Most natural phenomenon can be modeled by Gaussian.
- Take a bunch of random variables of any distribution, find the mean, the mean will approach to Gaussian distribution.
- Gaussian is very smooth function, it has infinite no of derivatives.

Gaussian

- Fourier Transform of Gaussian is Gaussian.
- If you convolve Gaussian with itself, it is again Gaussian.
- There are cells in human brain which perform Gaussian filtering.
  - Laplacian of Gaussian edge detector

Carl F. Gauss

- Born to a peasant family in a small town in Germany.
- Learned counting before he could talk.
- Contributed to Physics, Mathematics, Astronomy,…
- Discovered most methods in modern mathematics, when he was a teenager.

Carl F. Gauss

- Some contributions
  - Gaussian elimination for solving linear systems
  - Gauss-Seidel method for solving sparse systems
  - Gaussian curvature
  - Gaussian quadrature

Laplacian Pyramid

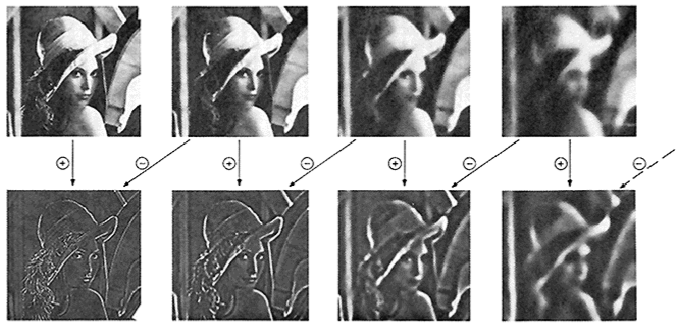
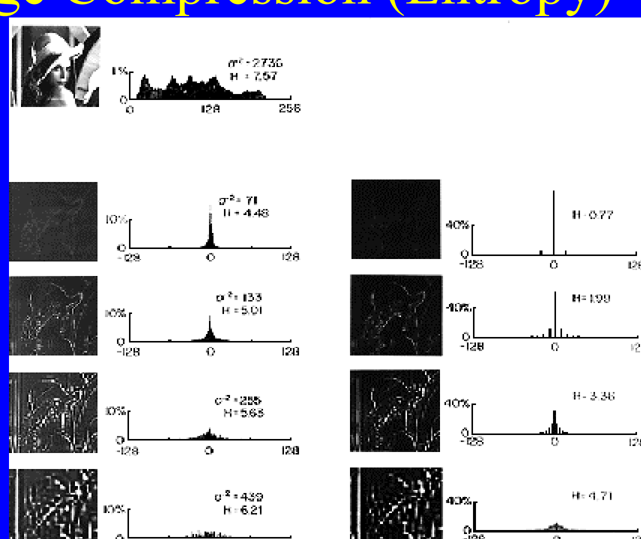


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

Image Compression (Entropy)



7.6

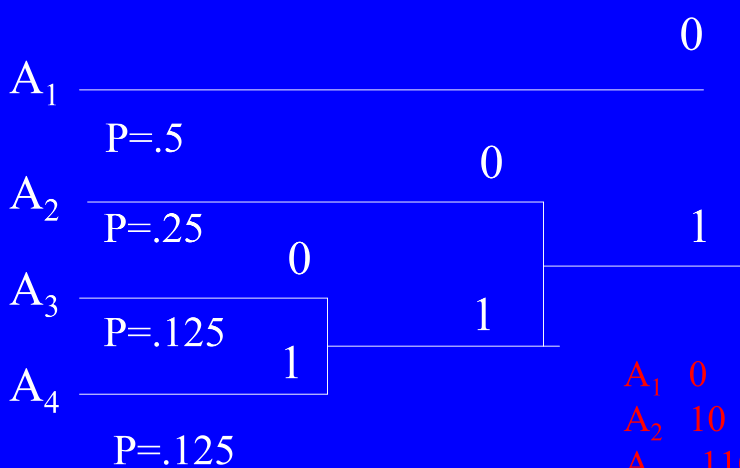
4.4 .77

5.0 1.9

5.6 3.3

6.2 4.2

Huffman Coding (Example-1)



A_1	0
A_2	10
A_3	110
A_4	111

Huffman Coding

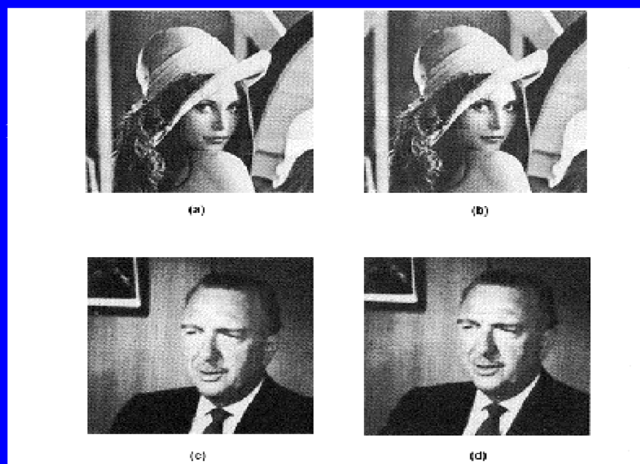
Entropy $$H = -\sum_{i=0}^{255} p(i)\log_2 p(i)$$

$$H = -.5\log.5 - .25\log.25 - .125\log.125 - .125\log.125 = 1.75$$

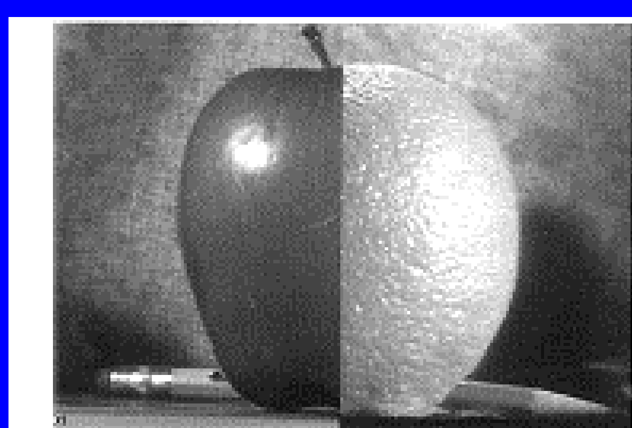
Image Compression

1.58

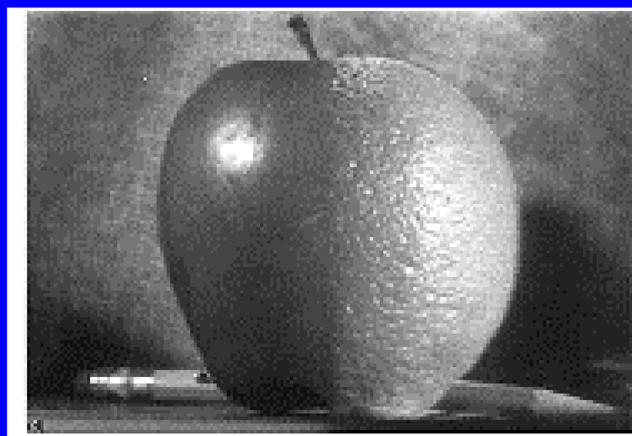
.73



Combining Apple & Orange



Combining Apple & Orange



Algorithm

- Generate Laplacian pyramid Lo of orange image.
- Generate Laplacian pyramid La of apple image.
- Generate Laplacian pyramid Lc by copying left half of nodes at each level from apple and right half of nodes from orange pyramids.
- Reconstruct combined image from Lc.

- http://ww-bcs.mit.edu/people/adelson/papers.html
  – The Laplacian Pyramid as a compact code, Burt and Adelson, IEEE Trans on Communication, 1983.

Algorithm-2 (Optical Flow)

- Create Gaussian pyramid of both frames.
- Repeat
  – apply algorithm-1 at the current level of pyramid.
  – propagate flow by using bilinear interpolation to the next level, where it is used as an initial estimate.
  – Go back to step 2

Lecture-5

Computing Optical Flow: Lucas & Kanade

Global Flow

Lucas & Kanade (Least Squares)

- Optical flow eq

$$f_x u + f_y v = -f_t$$

- Consider 3 by 3 window

$$f_{x1} u + f_{y1} v = -f_{t1}$$

$$\vdots$$

$$f_{x9} u + f_{y9} v = -f_{t9}$$

$$\begin{bmatrix} f_{x1} & f_{y1} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t1} \\ \vdots \\ -f_{t9} \end{bmatrix}$$

$$\mathbf{Au} = \mathbf{f_t}$$

Lucas & Kanade

$$\mathbf{Au} = \mathbf{f_t}$$

$$\mathbf{A^T A u} = \mathbf{A^T f_t}$$

$$\mathbf{u} = (\mathbf{A^T A})^{-1} \mathbf{A^T f_t}$$

$$\min \sum (f_{xi} u + f_{yi} v + f_t)^2$$

Lucas & Kanade

$$\min \sum (f_{xi} u + f_{yi} v + f_t)^2$$

$$\sum (f_{xi} u + f_{yi} v + f_{ti}) f_{xi} = 0$$

$$\sum (f_{xi} u + f_{yi} v + f_{ti}) f_{yi} = 0$$

$$\sum (f_{xi}u + f_{yi}v + f_{ti}) f_{xi} = 0$$

$$\sum (f_{xi}u + f_{yi}v + f_{ti}) f_{yi} = 0$$

$$\sum f_{xi}^2 u + \sum f_{xi} f_{yi} v = -\sum f_{xi} f_{ti}$$

$$\sum f_{xi} f_{yi} u + \sum f_{yi}^2 v = -\sum f_{yi} f_{ti}$$

$$\begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi} f_{yi} \\ \sum f_{xi} f_{yi} & \sum f_{yi}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$

Lucas & Kanade

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi} f_{yi} \\ \sum f_{xi} f_{yi} & \sum f_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi} f_{yi})^2} \begin{bmatrix} \sum f_{yi}^2 & -\sum f_{xi} f_{yi} \\ -\sum f_{xi} f_{yi} & \sum f_{xi}^2 \end{bmatrix} \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$

$$u = \frac{-\sum f_{yi}^2 \sum f_{xi} f_{ti} + \sum f_{xi} f_{yi} \sum f_{yi} f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi} f_{yi})^2}$$

$$v = \frac{\sum f_{xi} f_{ti} \sum f_{xi} f_{yi} - \sum f_{xi}^2 \sum f_{yi} f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi} f_{yi})^2}$$

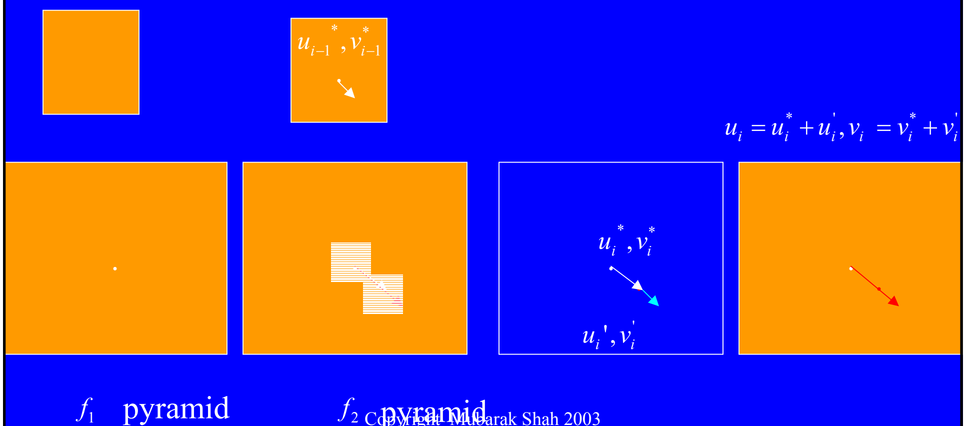
Comments

- Horn-Schunck and Lucas-Kanade optical method works only for small motion.
- If object moves faster, the brightness changes rapidly, 2x2 or 3x3 masks fail to estimate spatiotemporal derivatives.
- Pyramids can be used to compute large optical flow vectors.

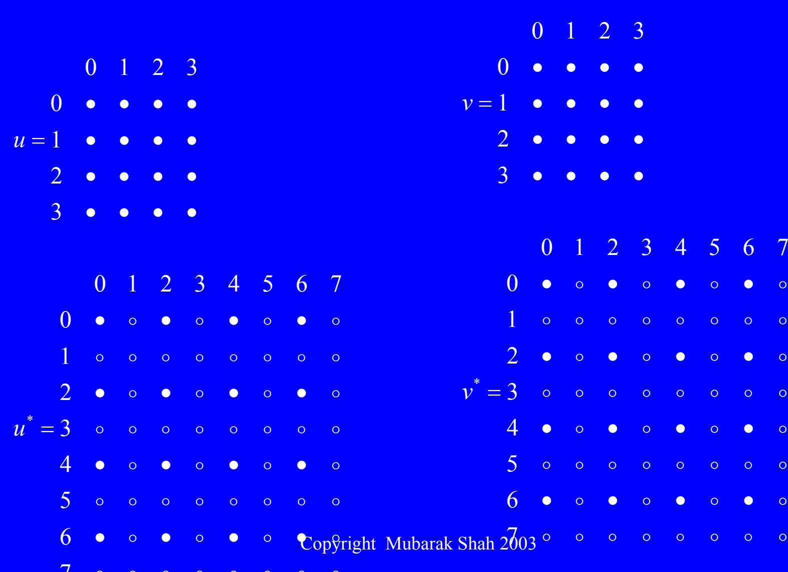
Lucas Kanade with Pyramids

- Compute 'simple' LK at highest level
- At level *i*
  - Take flow u_{i-1}, v_{i-1} from level *i*-1
  - bilinear interpolate it to create u_i^*, v_i^* matrices of twice resolution for level *i*
  - multiply u_i^*, v_i^* by 2
  - compute f_t from a block displaced by $u_i^*(x,y)$, $v_i^*(x,y)$
  - Apply LK to get $u_i'(x, y)$, $v_i'(x, y)$ (the correction in flow)
  - Add corrections u_i' v_i', *i.e.* $u_i = u_i^* + u_i'$, $v_i = v_i^* + v_i'$.

Pyramids

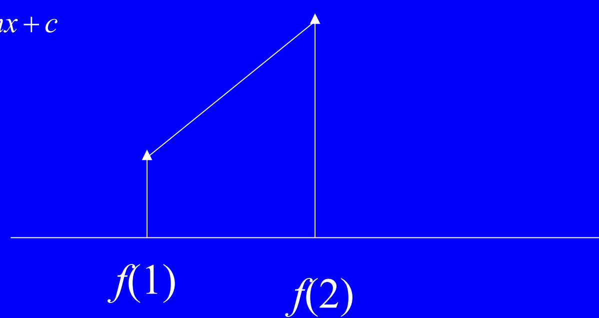


Interpolation



1-D Interpolation

$$y = mx + c$$
$$f(x) = mx + c$$



2-D Interpolation

$$f(x,y)=a_1+a_2x+a_3y+a_4xy$$

Bilinear

X X
O X

Bi-linear Interpolation

Four nearest points of (x,y) are:

$$(\underline{x},\underline{y}),(\overline{x},\underline{y}),(\underline{x},\overline{y}),(\overline{x},\overline{y})$$

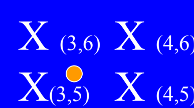
$$(3,5),(4,5),(3,6),(4,6)$$

$\underline{x}=\text{int}(x)$ 3 (3.2,5.6)

$\underline{y}=\text{int}(y)$ 5

$\overline{x}=\underline{x}+1$ 4

$\overline{y}=\underline{y}+1$ 6



$$f'(x,y)=\overline{\varepsilon_x}\,\overline{\varepsilon_y}\,f(\underline{x},\underline{y})+\underline{\varepsilon_x}\,\overline{\varepsilon_y}\,f(\overline{x},\underline{y})+$$
$$\overline{\varepsilon_x}\,\underline{\varepsilon_y}\,f(\underline{x},\overline{y})+\underline{\varepsilon_x}\,\underline{\varepsilon_y}\,f(\overline{x},\overline{y})$$

$\overline{\varepsilon_x}=\overline{x}-x$ $\overline{\varepsilon_x}=\overline{x}-x=4-3.2=.8$

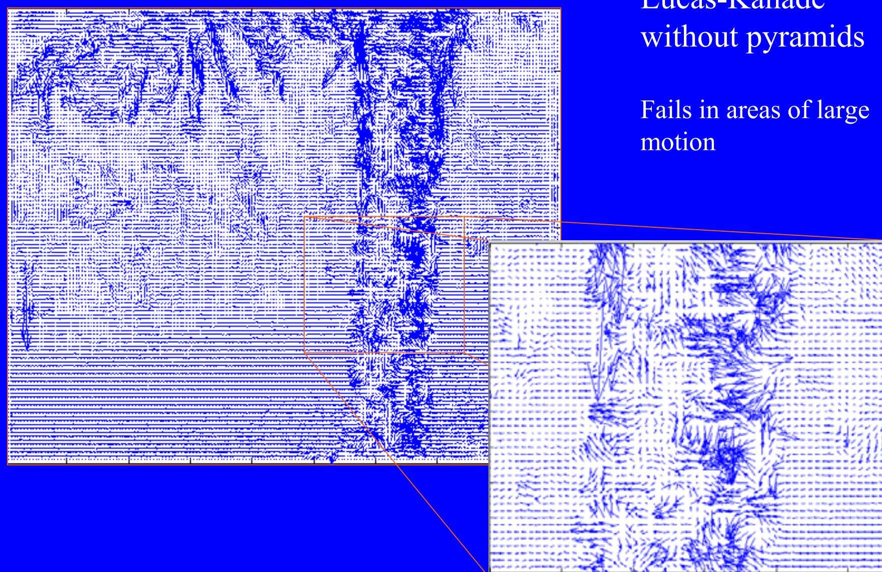
$\overline{\varepsilon_y}=\overline{y}-y$ $\overline{\varepsilon_y}=\overline{y}-y=6-5.6=.4$

$\underline{\varepsilon_x}=x-\underline{x}$ $\underline{\varepsilon_x}=x-\underline{x}=3.2-2=.2$

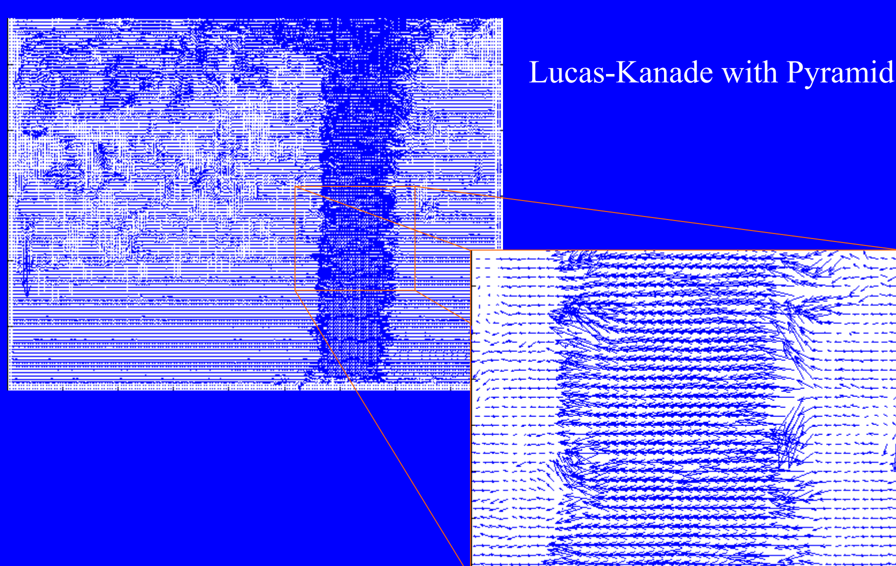
$\underline{\varepsilon_y}=y-\underline{y}$ $\underline{\varepsilon_y}=y-\underline{y}=5.6-5=.6$

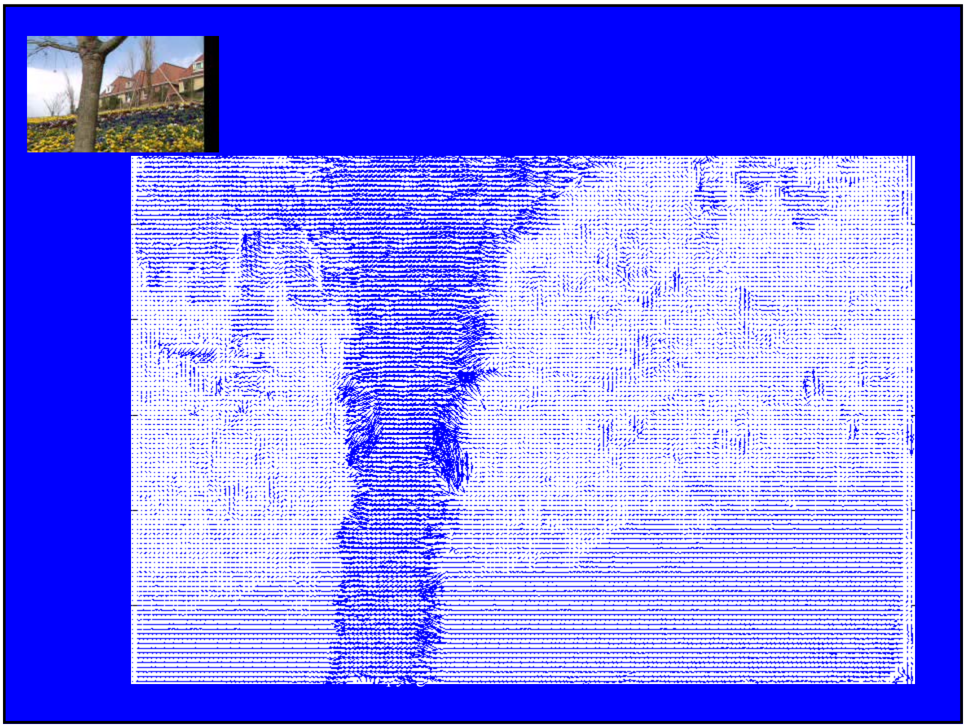
Lucas-Kanade without pyramids

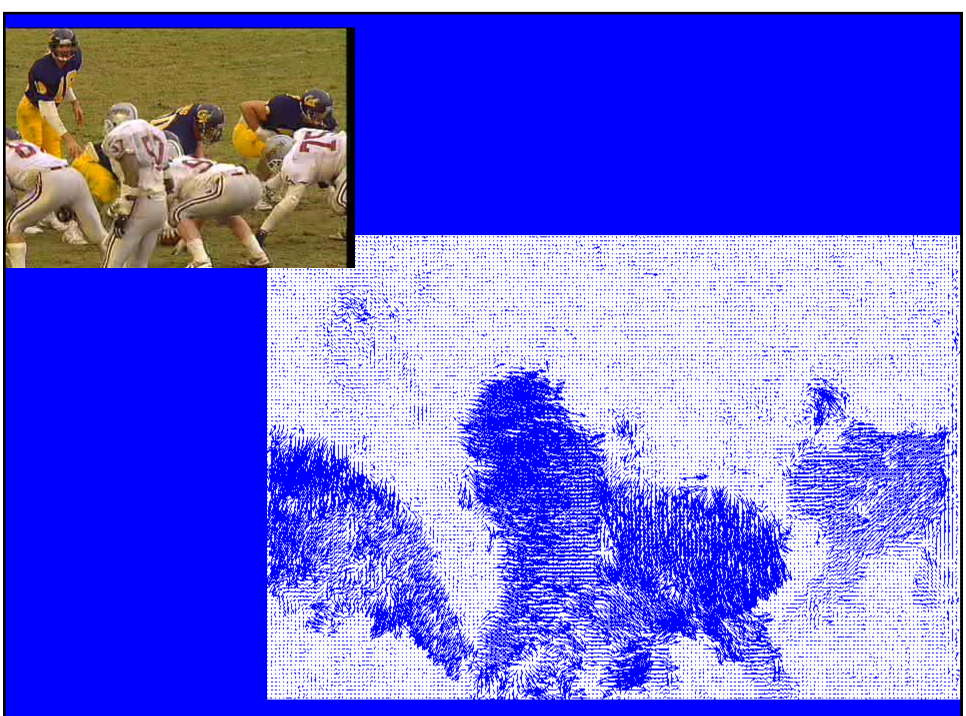
Fails in areas of large motion

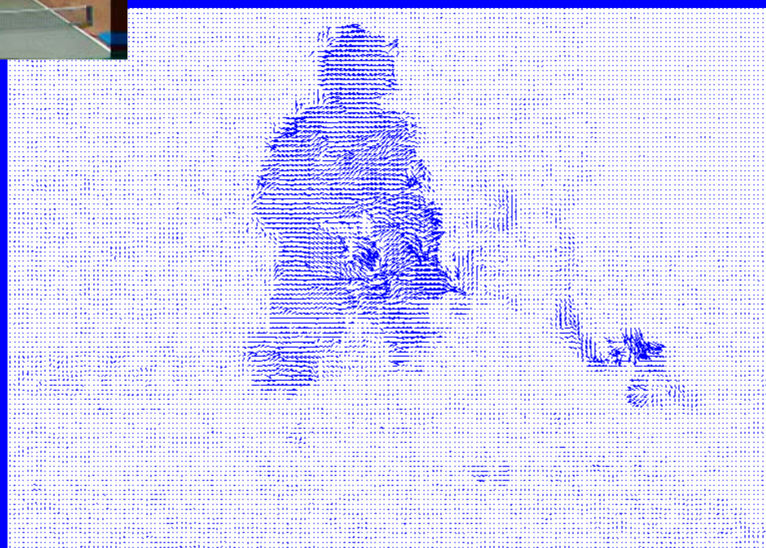


Lucas-Kanade with Pyramids









Global Flow

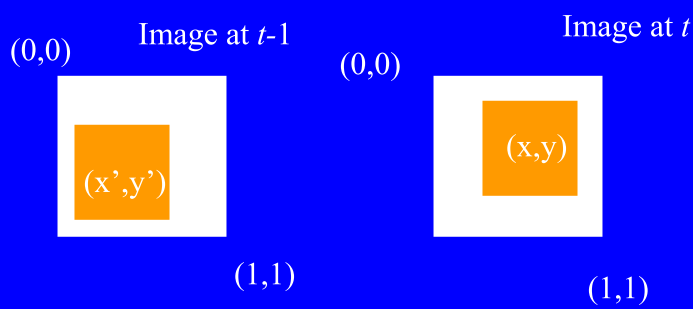
Anandan

Affine

Global Motion

- Estimate motion using all pixels in the image.
- Parametric flow gives an equation, which describes optical flow for each pixel.
  - Affine
  - Projective
- Global motion can be used to
  - generate mosaics
  - Object-based segmentation

Affine



$$u(x,y) = a_1 x + a_2 y + b_1$$

$$v(x,y) = a_3 x + a_4 y + b_2$$

$$X' = X - U$$

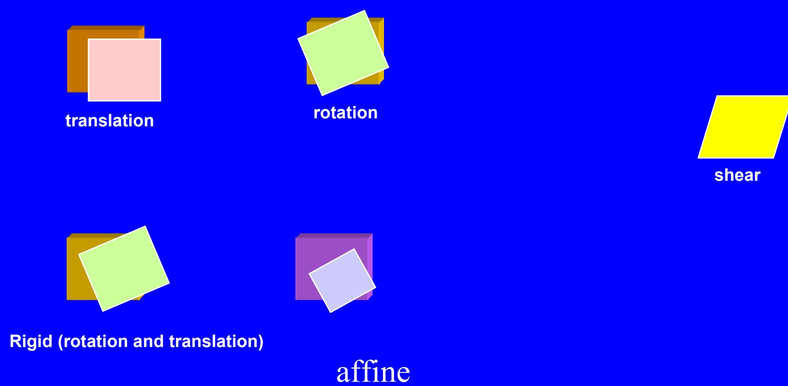
Affine

$$u(x,y) = a_1 x + a_2 y + b_1$$

$$v(x,y) = a_3 x + a_4 y + b_2$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Spatial Transformations



Anandan

$$u(x, y) = a_1 x + a_2 y + b_1$$
$$v(x, y) = a_3 x + a_4 y + b_2$$

•Affine

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2 \end{bmatrix}$$

$$\mathbf{u}(\mathbf{x}) = \mathbf{X}(\mathbf{x})\mathbf{a}$$

Anandan

$$\mathbf{u}(\mathbf{x}) = \mathbf{X}(\mathbf{x})\mathbf{a}$$

Optical flow constraint eq $f_x u + f_y v = -f_t$

$$E(\delta a) = \sum_{\forall x \in f(x,y)} (f_t + f_{\mathbf{x}}^T \delta u)^2 \qquad f_{\mathbf{x}} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

$$E(\delta a) = \sum_{\forall x \in f(x,y)} (f_t + f_{\mathbf{x}}^T \mathbf{X} \delta a\ )^2$$

min

$$\left[\sum X^T (f_X)(f_x)^T X\right]\delta a = -\sum X^T f_X f_t$$

(a) Derive this Due Sept 25

$$Ax = b$$

Linear system

Basic Components

- Pyramid construction
- Motion estimation
- Image warping
- Coarse-to-fine refinement

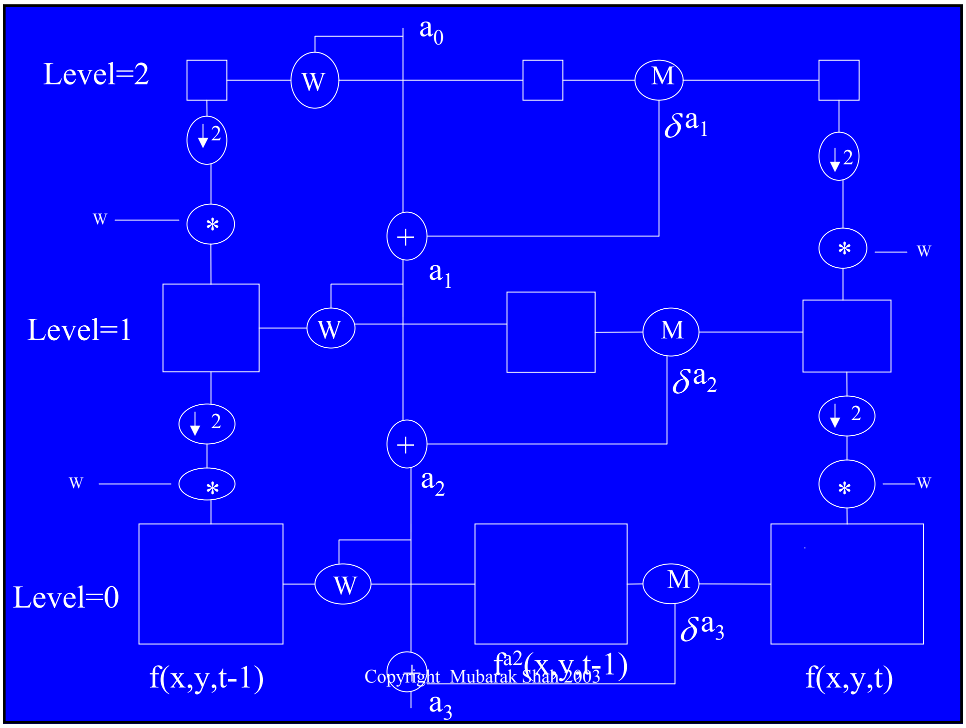


Image Warping

- Warping an image *f* into image *h* using some transformation *g*, involves mapping intensity at each pixel (*x,y*) in image *f* to a pixel (*g(x),* g(*y*)) image *h* such that

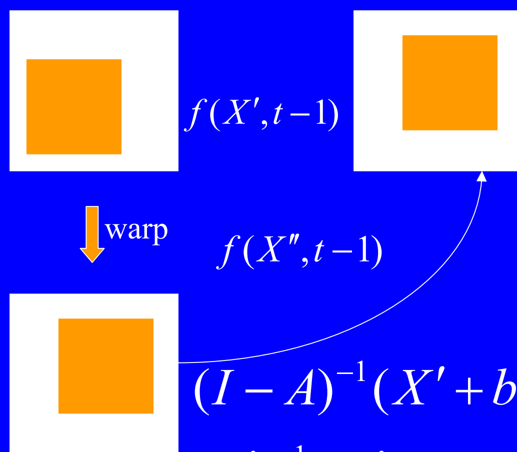
$$(x', y') = (g(x), g(y))$$

- In case of affine transformation, $\mathbf{x} = (x, y)$ is transformed to $\mathbf{x}' = (x', y')$ as:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$$

Image Warping

$$X' = X - U$$
$$= X - (AX + b)$$



$$(I - A)^{-1}(X' + b) = X''$$

$$(A')^{-1}(X' + b) = X''$$

Image Warping

$$X' = X - U = X - (AX + b)$$
$$X' = (I - A)X - b$$
$$X' = A'X - b$$
$$X' + b = A'X$$
$$(A')^{-1}(X' + b) = X$$

Image at time t: X
Image at time t-1: X'

$$(A')^{-1}(X' + b) = X''$$

$$X' \rightarrow X''$$

Image Warping

•How about values in $X'' = (x'', y'')$ are not integer.

•But image is sampled only at integer rows and columns

• Instead of converting X' to X'' and copying $f(X', t-1)$ at $f(X'', t-1)$ we can convert integer values X'' to X' and copy $f(X', t-1)$ at $f(X'', t-1)$

Image Warping

• But how about the values in X' are not integer.

• Perform bilinear interpolation to compute $f(X', t-1)$ at non-integer values.

Image Warping

$$(A')^{-1}(X'+b) = X''$$

$$(X'+b) = (A')X''$$

$$X' = (A')X'' - b \qquad X'' \rightarrow X'$$

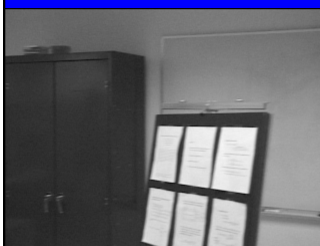
Warping

Copyright Mubarak Shah 2003

Show Demos

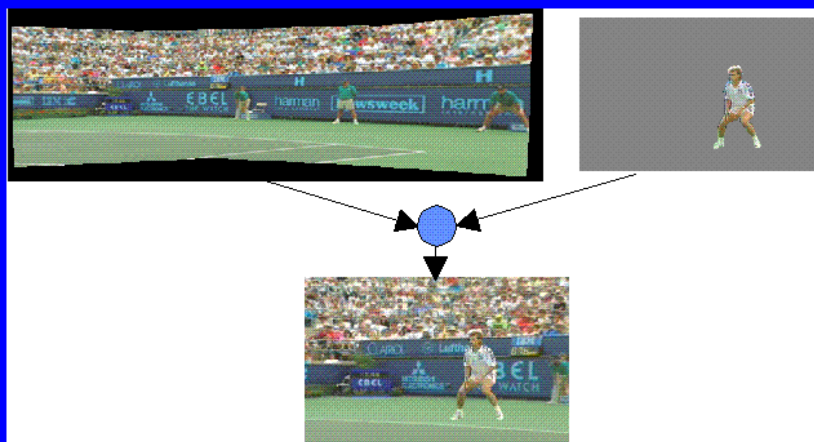
Video Mosaic

Video Mosaic



Video Mosaic

Sprite



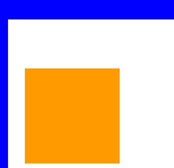
Lecture-6

Szeliski, Mann & Picard

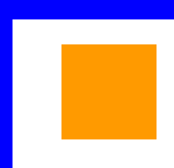
Szeliski

Projective

Projective



$f(X', t-1)$



$f(X, t)$

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$

$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

Szeliski

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$ Projective

$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

$$E = \sum [f(x', y') - f(x, y)]^2 = \sum e^2$$

min

Szeliski

Motion Vector:

$$\mathbf{m} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & b_1 & b_2 & c_1 & c_2 \end{bmatrix}^T$$

Szeliski (Levenberg-Marquadet)

$$\alpha_{kl} = \sum_n \frac{\partial e_n}{\partial m_k}\frac{\partial e_n}{\partial m_l}$$

$$b_k = -\sum e \frac{\partial e_n}{\partial m_k}$$

gradient

$$\Delta m = (A + \lambda I)^{-1} b$$

Approximation of Hessian (*$J^T J$, Jacobian*)

$$\frac{\partial e}{\partial a_1} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_1} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_1}$$

$$E = \sum [f(x', y') - f(x, y)]^2 = \sum e^2$$

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$

$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

Approximation of Hessian

$$J^T = \begin{bmatrix} \frac{\partial e_1}{\partial m_1} & \frac{\partial e_2}{\partial m_1} & \cdots & \frac{\partial e_n}{\partial m_1} \\ \frac{\partial e_1}{\partial m_2} & \frac{\partial e_2}{\partial m_2} & \cdots & \frac{\partial e_n}{\partial m_2} \\ \frac{\partial e_1}{\partial m_3} & \frac{\partial e_2}{\partial m_3} & \cdots & \frac{\partial e_n}{\partial m_3} \\ \frac{\partial e_1}{\partial m_4} & \frac{\partial e_2}{\partial m_4} & \cdots & \frac{\partial e_n}{\partial m_4} \\ \frac{\partial e_1}{\partial m_5} & \frac{\partial e_2}{\partial m_5} & \cdots & \frac{\partial e_n}{\partial m_5} \\ \frac{\partial e_1}{\partial m_6} & \frac{\partial e_2}{\partial m_6} & \cdots & \frac{\partial e_n}{\partial m_6} \\ \frac{\partial e_1}{\partial m_7} & \frac{\partial e_2}{\partial m_7} & \cdots & \frac{\partial e_n}{\partial m_7} \\ \frac{\partial e_1}{\partial m_8} & \frac{\partial e_2}{\partial m_8} & \cdots & \frac{\partial e_n}{\partial m_8} \end{bmatrix} \quad J = \begin{bmatrix} \frac{\partial e_1}{\partial m_1} & \frac{\partial e_1}{\partial m_2} & \frac{\partial e_1}{\partial m_3} & \frac{\partial e_1}{\partial m_4} & \frac{\partial e_1}{\partial m_5} & \frac{\partial e_1}{\partial m_6} & \frac{\partial e_1}{\partial m_7} & \frac{\partial e_1}{\partial m_8} \\ \frac{\partial e_2}{\partial m_1} & \frac{\partial e_2}{\partial m_2} & \frac{\partial e_2}{\partial m_3} & \frac{\partial e_2}{\partial m_4} & \frac{\partial e_2}{\partial m_5} & \frac{\partial e_2}{\partial m_6} & \frac{\partial e_2}{\partial m_7} & \frac{\partial e_2}{\partial m_8} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_n}{\partial m_1} & \frac{\partial e_n}{\partial m_2} & \frac{\partial e_n}{\partial m_3} & \frac{\partial e_n}{\partial m_4} & \frac{\partial e_n}{\partial m_5} & \frac{\partial e_n}{\partial m_6} & \frac{\partial e_n}{\partial m_7} & \frac{\partial e_n}{\partial m_8} \end{bmatrix}$$

$$A = J^T J$$

$$\alpha_{kl} = \sum_n \frac{\partial e_n}{\partial m_k} \frac{\partial e_n}{\partial m_l}$$

A Matrix

Gradient Vector

$$b = \begin{bmatrix} -\sum_n e_n \frac{\partial e_n}{\partial a_1} \\ -\sum_n e_n \frac{\partial e_n}{\partial a_2} \\ -\sum_n e_n \frac{\partial e_n}{\partial a_3} \\ -\sum_n e_n \frac{\partial e_n}{\partial a_4} \\ -\sum_n e \frac{\partial e_n}{\partial b_1} \\ -\sum_n e_n \frac{\partial e_n}{\partial b_2} \\ -\sum_n e_n \frac{\partial e_n}{\partial c_1} \\ -\sum_n e_n \frac{\partial e_n}{\partial c_2} \end{bmatrix}$$

Partial Derivatives wrt motion parameters

$$\frac{\partial e}{\partial a_1} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_1} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_1}$$

$$x' = \frac{a_1x + a_2y + b_1}{c_1x + c_2y + 1}, \quad y' = \frac{a_3x + a_4y + b_2}{c_1x + c_2y + 1}$$

$$\frac{\partial x'}{\partial a_1} = \frac{x}{c_1x + c_2y + 1} \qquad \frac{\partial y'}{\partial a_1} = 0$$

$$\frac{\partial x'}{\partial a_2} = \frac{y}{c_1x + c_2y + 1} \qquad \frac{\partial y'}{\partial a_2} = 0$$

$$\frac{\partial x'}{\partial a_3} = 0 \qquad \frac{\partial y'}{\partial a_3} = \frac{x}{c_1x + c_2y + 1}$$

$$\frac{\partial x'}{\partial a_4} = 0 \qquad \frac{\partial y'}{\partial a_4} = \frac{y}{c_1x + c_2y + 1}$$

$$\frac{\partial x'}{\partial b_1} = \frac{1}{c_1x + c_2y + 1} \qquad \frac{\partial y'}{\partial b_1} = 0$$

$$\frac{\partial x'}{\partial b_2} = 0 \qquad \frac{\partial y'}{\partial b_2} = \frac{1}{c_1x + c_2y + 1}$$

$$\frac{\partial x'}{\partial c_1} = \frac{-x(a_1x + a_2y + b_1)}{(c_1x + c_2y + 1)^2} \qquad \frac{\partial y'}{\partial c_1} = \frac{-x(a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2}$$

$$\frac{\partial x'}{\partial c_2} = \frac{-y(a_1x + a_2y + b_1)}{(c_1x + c_2y + 1)^2} \qquad \frac{\partial y'}{\partial c_2} = \frac{-y(a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2}$$

Partial derivatives wrt image coordinates

$$E = \sum [f(x', y') - f(x, y)]^2 = \sum e^2$$

$$\frac{\partial e}{\partial x'} = f_{x'}$$

$$\frac{\partial e}{\partial y'} = f_{y'}$$

Partial derivatives

$$\frac{\partial e}{\partial a_1} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_1} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_1} = f_{x'}\frac{x}{c_1x + c_2y + 1}$$

$$\frac{\partial e}{\partial a_2} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_2} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_2} = f_{x'}\frac{y}{c_1x + c_2y + 1}$$

$$\frac{\partial e}{\partial a_3} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_3} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_3} = f_{y'}\frac{x}{c_1x + c_2y + 1}$$

$$\frac{\partial e}{\partial a_4} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_4} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_4} = f_{y'}\frac{y}{c_1x + c_2y + 1}$$

$$\frac{\partial e}{\partial b_1} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial b_1} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial b_1} = f_{x'}\frac{1}{c_1x + c_2y + 1}$$

$$\frac{\partial e}{\partial b_2} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial b_2} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial b_2} = f_{y'}\frac{1}{c_1x + c_2y + 1}$$

$$\frac{\partial e}{\partial c_1} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial c_1} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial c_1} = f_{x'}\frac{-x(a_1x + a_2y + b_1)}{(c_1x + c_2y + 1)^2} + f_{y'}\frac{-x(a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2}$$

$$\frac{\partial e}{\partial c_2} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial c_2} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial c_2} = f_{x'}\frac{-y(a_1x + a_2y + b_1)}{(c_1x + c_2y + 1)^2} + f_{y'}\frac{-y(a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2}$$

Gradient Vector

$$\mathbf{b} = \begin{bmatrix} -\sum e f_{x'}\frac{x}{c_1x + c_2y + 1} \\ -\sum e f_{x'}\frac{y}{c_1x + c_2y + 1} \\ -\sum e f_{y'}\frac{x}{c_1x + c_2y + 1} \\ -\sum e f_{y'}\frac{y}{c_1x + c_2y + 1} \\ -\sum e f_{x'}\frac{1}{c_1x + c_2y + 1} \\ -\sum e f_{y'}\frac{1}{c_1x + c_2y + 1} \\ \sum ex\left[\frac{f_{x'}(a_1x + a_2y + b_1) + f_{y'}(a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2}\right] \\ \sum ey\left[\frac{f_{x'}(a_1x + a_2y + b_1) + f_{y'}(a_3x + a_4y + b_2)}{(c_1x + c_2y + 1)^2}\right] \end{bmatrix}$$

Szeliski (Levenberg-Marquadet)

- Start with some initial value of m, and λ=.001
  - For each pixel I at (x_i, y_i)
    - Compute (x', y') using projective transform.
    - Compute $e = f(x', y') - f(x, y)$
    - Compute $\frac{\partial e}{\partial m_k} = \frac{\partial e}{\partial x'}\frac{\partial x'}{\partial m_k} + \frac{\partial e}{\partial y'}\frac{\partial y'}{\partial m_k}$

Szeliski (Levenberg-Marquadet)

-Compute A **and** b

-Solve system

$$(A - \lambda I)\Delta m = b$$

-Update

$$m^{t+1} = m^t + \Delta m$$

Szeliski (Levenberg-Marquadet)

- check if error has decreased, if not increase λ by a factor of 10 and compute a new Δm

- If error has decreased, decrease λ by a factor of 10 and compute a new Δm

- Continue iteration until error is below threshold.

Mann & Picard

Projective

Projective Flow (weighted)

$$u f_x + v f_y + f_t = 0$$

Optical Flow const. equation

$$\mathbf{u}^T \mathbf{f}_\mathbf{x} + f_t = 0$$

$$\mathbf{x}' = \frac{A\mathbf{x} + \mathbf{b}}{\mathbf{C}^\mathbf{T}\mathrm{x} + 1}$$

Projective transform

$$\mathbf{u} = \mathbf{x}' - \mathbf{x} = \frac{A\mathbf{x} + \mathbf{b}}{\mathbf{C}^\mathbf{T}\mathbf{x} + 1} - \mathbf{x}$$

Projective Flow (weighted)

$$\varepsilon_{flow} = \sum (\mathbf{u}^\mathbf{T}\mathbf{f}_\mathbf{X} + f_\mathbf{t})^2$$

$$= \sum ((\frac{A\mathbf{x} + \mathbf{b}}{\mathbf{C}\mathbf{x}^\mathbf{T} + 1} - \mathbf{x})^T \mathbf{f}_\mathbf{x} + f_t)^2$$

$$= \sum ((\mathbf{A}\mathbf{x} + \mathbf{b} - (\mathbf{C}^\mathbf{T}\mathbf{x} + 1)\mathbf{x})^T \mathbf{f}_x + (\mathbf{C}^T\mathbf{x} + 1) f_t)^2$$

⇓ **minimize**

Projective Flow (weighted)

• (b) Homework 2 Derive this equation Due Sept 25

$$(\sum \phi\phi^T)\mathbf{a} = \sum(\mathbf{x}^T\mathbf{f}_x - f_t)\phi$$

$$a = [a_1, a_2, b_1, a_3, a_4, b_2, c_1, c_2]^T$$

$$\phi^t = [f_x x, f_x y, f_x, f_y x, f_y y, f_y, x f_t - x^2 f_x - xy f_y, y f_t - xy f_x - y^2 f_y]$$

Projective Flow (unweighted)

Bilinear

$$\mathbf{x}' = \frac{A\mathbf{x} + \mathbf{b}}{\mathbf{C}^T\mathbf{x} + 1}$$

Taylor Series & remove Square terms

$$u + x = a_1 + a_2 x + a_3 y + a_4 xy$$

$$v + y = a_5 + a_6 x + a_7 y + a_8 xy$$

Pseudo-Perspective

$$\mathbf{x}' = \frac{A\mathbf{x} + \mathbf{b}}{\mathbf{C}^T\mathbf{x} + 1}$$

Taylor Series

$$x + u = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy$$

$$y + v = a_6 + a_7 x + a_8 y + a_4 xy + a_5 y^2$$

Projective Flow (unweighted)

$$\varepsilon_{flow} = \sum (\mathbf{u}^{\mathrm{T}}\mathbf{f}_{\mathbf{x}} + f_{\mathrm{t}})^2$$

Minimize

Bilinear and Pseudo-Perspective

$$(\sum \Phi\Phi^T)\mathbf{q} = -\sum f_t \Phi$$

(c) homewwork Derive these eqs Sept 25

$$\Phi^T = \left[f_x(xy, x, y, 1), \quad f_y\left(xy, x, y, 1\right)\right]$$ **bilinear**

$$\Phi^T = \left[f_x(x, y, 1) \quad f_y(x, y, 1) \quad c_1 \quad c_2 \right]$$

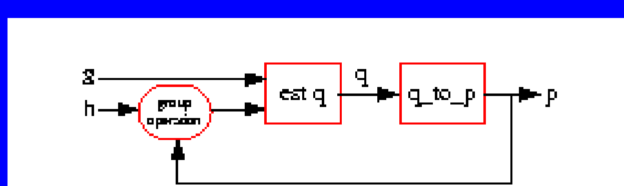
$$c_1 = x^2 f_x + xyf_x$$

Pseudo perspective

$$c_2 = xyf_x + y^2 f_y$$

Algorithm-1

- Estimate "q" (using approximate model, e.g. bilinear model).
- Relate "q" to "p"
  - select four points S1, S2, S3, S4
  - apply approximate model using "q" to compute (x'_k, y'_k)
  - estimate exact "p":



True Projective

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$

$$y' = \frac{a_3 x + a_4 y + b_1}{c_1 x + c_2 y + 1}$$

$$\begin{bmatrix} x'_k \\ y'_k \end{bmatrix} = \begin{bmatrix} x_k & y_k & 1 & 0 & 0 & 0 & -x_k x'_k & -y_k x'_k \\ 0 & 0 & 0 & x_k & y_k & 1 & -x_k y'_k & -y_k y'_k \end{bmatrix} \mathbf{a}$$

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & b_1 & a_3 & a_4 & b_2 & c_1 & c_1 \end{bmatrix}^T$$

$$\begin{bmatrix} x'_1 \\ y'_1 \\ \\ \\ x'_k \\ y'_k \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 x'_1 & -y_1 x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 y'_1 & -y_1 y'_1 \\ \\ \\ x_k & y_k & 1 & 0 & 0 & 0 & -x_k x' & -y_k x'_k \\ 0 & 0 & 0 & x_k & y_k & 1 & -x_k y'_k & -y_k y' \end{bmatrix} \mathbf{a}$$

$$\mathbf{P} = \mathbf{A}\mathbf{a}$$

Perform least squares fit to compute a.

Final Algorithm

- A Gaussian pyramid of three or four levels is constructed for each frame in the sequence.
- The parameters "p" are estimated at the top level of the pyramid, between the two lowest resolution images, "g" and "h", using algorithm-1.

Final Algorithm

- The estimated "p" is applied to the next higher resolution image in the pyramid, to make images at that level nearly congruent.
- The process continues down the pyramid until the highest resolution image in the pyramid is reached.

Video Mosaics

- Mosaic aligns different pieces of a scene into a larger piece, and seamlessly blend them.
  - High resolution image from low resolution images
  - Increased filed of view

Steps in Generating A Mosaic

- Take pictures
- Pick reference image
- Determine transformation between frames
- Warp all images to the same reference view

Applications of Mosaics

- Virtual Environments
- Computer Games
- Movie Special Effects
- Video Compression

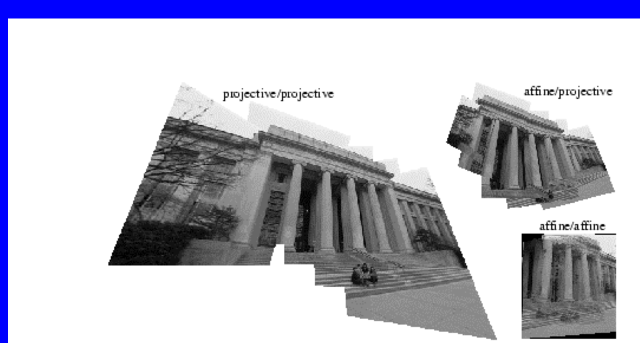
Steve Mann

Sequence of Images

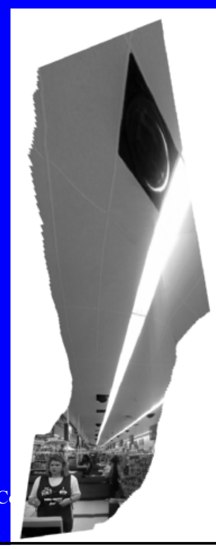
Projective Mosaic

Affine Mosaic

Building



Wal-Mart



Scientific American Frontiers

Scientific American Frontiers

Head-mounted Camera at Restaurant

MIT Media Lab

Webpages

- http://n1nlf1.eecg.toronto.edu/tip.ps.gz
  Video Orbits of the projective group, S. Mann and R. Picard.

- http://wearcam.org/pencigraphy (C code for generating mosaics)

Webpages

- http://ww-bcs.mit.edu/people/adelson/papers.html
  - The Laplacian Pyramid as a compact code, Burt and Adelson, IEEE Trans on Communication, 1983.
- J. Bergen, P. Anandan, K. Hanna, and R. Hingorani, “Hierarchical Model-Based Motion Estimation”, ECCV-92, pp 237-22.

Webpages

- http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/v-source.html (c code for several optical flow algorithms)
- ftp://csd.uwo.ca/pub/vision

  Performance of optical flow techniques (paper)

Barron, Fleet and Beauchermin

Webpages

- http://www.wisdom.weizmann.ac.il/~irani/abstracts/mosaics.html ("Efficient representations of video sequences and their applications", Michal Irani, P. Anandan, Jim Bergen, Rakesh Kumar, and Steve Hsu)
- R. Szeliski. "Video mosaics for virtual environments", IEEE Computer Graphics and Applications, pages,22-30, March 1996.

- M. Irani and P. Anandan, Video Indexing Based on Mosaic Representations. Proceedings of IEEE, May,1998.

- http://www.wisdom.weizmann.ac.il/~irani/abstracts/videoIndexing.html

Homework Due Sept 25

- (a) Derive linear system equation in Anandan's method Lecture 5, page 14, top slide.
- (b) Derive equations for Mann's method (weighted) Lecture 6, page 10.
- (c) Derive equations for Mann's method (un-weighted) Lecture 6, page 13.

Program-1 Due Oct 2

- (a) Implement Anandan's algorithm using affine transformation. To show the results generate a mosaic.
- (b) Implement Szeliski's algorithm using projective transformation. To show the results generate a mosaic.
- (c) Implement Mann's algorithm using projective transformation. To show the results generate a mosaic.
- Implement all four steps:
  - Pyramid construction
  - Motion estimation
  - Image warping
  - Coarse-to-fine refinement
- .

Lecture-7

Feature-based Registration

Steps in Feature-based Registration

- Find features
- Establish correspondence (correlation, point correspondence)
- Fit transformation
- Apply transformation (warp)

Features

- Any pixels
- Corner points
- Interest points
- Features obtained using Gabor/Wavelet filters
- Straight lines
- Line intersections

Transformations

- Affine
- Projective
- Psuedo-perspective
- Rational polynomial

Good Features to Track

- Corner like features
- Moravec's Interest Operator

Corner like features

$$C = \begin{bmatrix} \sum_Q f_x^2 & \sum_Q f_x f_y \\ \sum_Q f_x f_y & \sum_Q f_y^2 \end{bmatrix}$$

↓

$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Corners

- For perfectly uniform region $\lambda_1 = \lambda_2 = 0$
- If Q contains an ideal step edge, then

$$\lambda_2 = 0, \lambda_1 > 0$$

- if Q contains a corner of black square on white background

$$\lambda_1 \geq \lambda_2 \succ 0$$

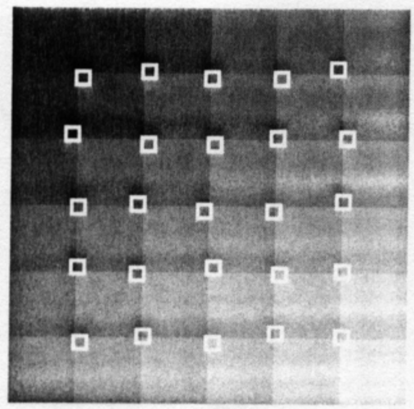
Algorithm Corners

- Compute the image gradient over entire image f.
- For each image point p:
  - form the matrix C over (2N+1)X(2N+1) neighborhood Q of p;
  - compute the smallest eigenvalue of C;
  - if eigenvalue is above some threshold, save the coordinates of p into a list L.

Algorithm Corners

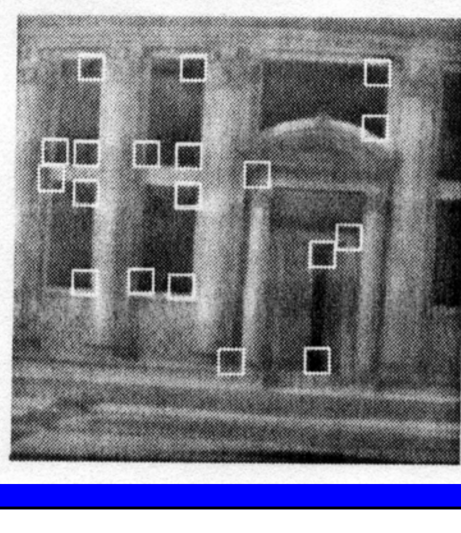
- Sort L in decreasing order of eigenvalues.
- Select the top candidate corner, and perform Non-maxima suppression
  - Scanning the sorted list top to bottom: for each current point, p, delete all other points on the list which belong to the neighborhood of p.

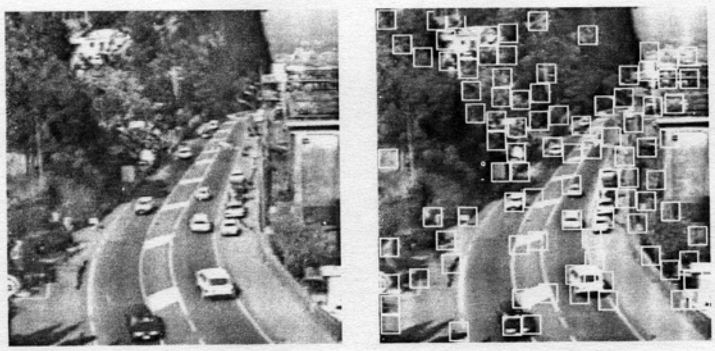
Results



(a)

Results



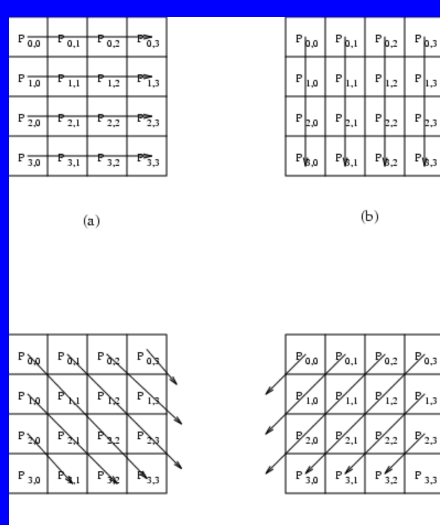


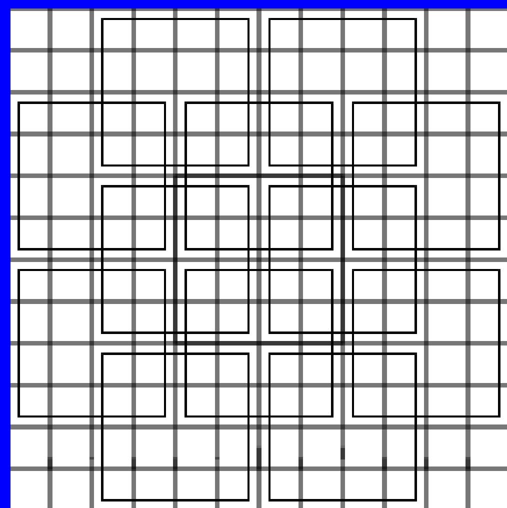
(a) (b)

Moravec's Interest Operator

Algorithm

- Compute four directional variances in horizontal, vertical, diagonal and anti-diagonal directions for each 4 by 4 window.
- If the minimum of four directional variances is a local maximum in a 12 by 12 overlapping neighborhood, then that widow (point) is interesting.





$$V_h = \sum_{j=0}^{3} \sum_{i=0}^{2} (P(x+i, y+j) - P(x+i+1, y+j))^2$$

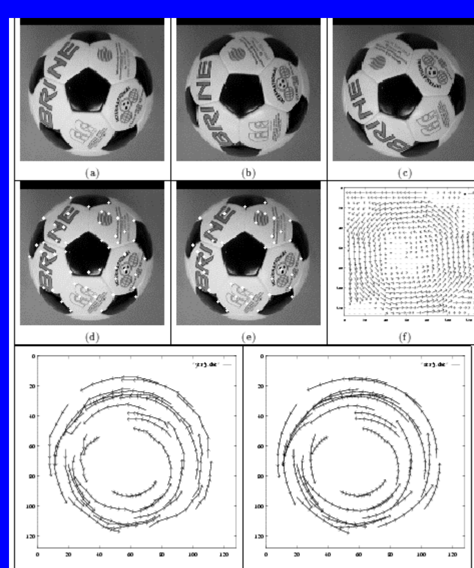
$$V_v = \sum_{j=0}^{2} \sum_{i=0}^{3} (P(x+i, y+j) - P(x+i, y+j+1))^2$$

$$V_d = \sum_{j=0}^{2} \sum_{i=0}^{2} (P(x+i, y+j) - P(x+i+1, y+j+1))^2$$

$$V_a = \sum_{j=0}^{2} \sum_{i=1}^{3} (P(x+i, y+j) - P(x+i-1, y+j+1))^2$$

$$V(x,y) = \min(V_h(x,y), V_v(x,y), V_d(x,y), V_a(x,y))$$

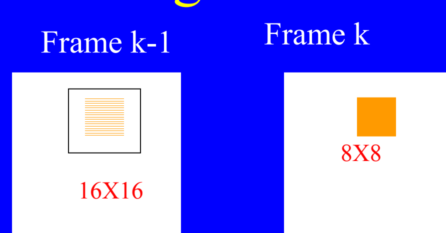
$$I(x,y) = \begin{cases} 1 & if V(x,y) local \max \\ 0 & 0therwise \end{cases}$$



Correlation

- Similarity/Dissimilarity Measures
  - Sum of Squares Difference (SSD)
  - Normalized Correlation
  - Mutual Correlation
  - Mutual information $H(X;Y)=H(X)-H(X|Y)$
- Use
  - Gray levels
  - Laplacian of Gaussian
  - Gradient magnitude

Block Matching



Block Matching

- For each 8X8 block, centered around pixel (x,y) in frame k, B_k
  - Obtain 16X16 block in frame k-1, centered around (x,y), B_{k-1}
  - Compute Sum of Squares Differences (SSD) between 8X8 block, $B_{k,}$ and all possible 8X8 blocks in B_{k-1}
  - The 8X8 block in B_{k-1} centered around (x',y'), which gives the least SSD is the match
  - The displacement vector (optical flow) is given by u=x-x'; v=y-y'

Sum of Squares Differences (SSD)

$$(u(x,y),v(x,y)) = \arg\min_{u,v=-3\ldots3} \sum_{i=0}^{-7}\sum_{j=0}^{-7}\left(f_k(x+i,y+j)-f_{k-1}(x+i+u,y+j+v)\right)^2$$

Minimum Absolute Difference (MAD)

$$(u(x,y),v(x,y)) = \arg\min_{u,v=-3\ldots3} \sum_{i=0}^{-7}\sum_{j=0}^{-7} |\left(f_k(x+i,y+j) - f_{k-1}(x+i+u,y+j+v)\right)|$$

Maximum Matching Pixel Count (MPC)

$$T(x,y;u,v) = \begin{cases} 1 & \text{if } |f_k(x,y) - f_{k-1}(x+u,y+v)| \le t \\ 0 & \text{Otherwise} \end{cases}$$

$$(u(x,y),v(x,y)) = \arg\max_{u,v=-3\ldots3} \sum_{i=0}^{-7}\sum_{j=0}^{-7} T(x+i,y+j;u,v)$$

Cross Correlation

$$(u,v) = \arg\max_{u,v=-3\ldots3} \sum_{i=0}^{-7}\sum_{j=0}^{-7}\left(f_k(x+i,y+j)).(f_{k-1}(x+i+u,y+j+v)\right)$$

Normalized Correlation

$$(u,v) = \arg\max_{u,v=-3\ldots3} \frac{\sum_{i=0}^{j=-7}\sum_{j=0}^{-7}\left(f_k(x+i,y+j)).(f_{k-1}(x+i+u,y+j+v)\right)}{\sqrt{\sum_{i=0}^{-7}\sum_{j=0}^{-7} f_{k-1}(x+i+u,y+j+v).f_{k-1}(x+i+u,y+j+v)}}$$

Copyright Mubarak Shah 2003

Mutual Correlation

$$(u,v) = \arg\max_{u,v=-3\ldots3} \frac{1}{64\sigma_1\sigma_2}\sum_{i=0}^{-7}\sum_{j=0}^{-7}\left(f_k(x+i,y+j)-\mu_1\right).\left(f_{k-1}(x+i+u,y+j+v)-\mu_2\right)$$

Sigma and mu are standard deviation and mean of patch-1 and patch-2 respectively

Issues with Correlation

- Patch Size
- Search Area
- How many peaks

Spatiotemporal Models

- First order Taylor series

$$f(x,y,t) = f(x+dx, y+dy, t+dt)$$

$$f(x,y,t) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt$$

$$\left[\sum X^T \mathbf{f_x}\mathbf{f_x^T} X\right]\delta a = -\sum X^T \mathbf{f_x} f_t$$

$$(u(x,y), v(x,y)) = \arg\min_{u,v=-3..3} \sum_{i=-7}^{7}\sum_{j=-7}^{7} \left| f_k(x+i, y+j) - f_{k-1}(x+i+u, y+j+v) \right|$$

Correlation

Bilinear and Pseudo-Perspective

$$(\sum \Phi\Phi^T)\mathbf{q} = -\sum f_t \Phi$$

$$\Phi^T = \left[f_x(xy, x, y, 1), \quad f_y(xy, x, y, 1)\right]$$ **bilinear**

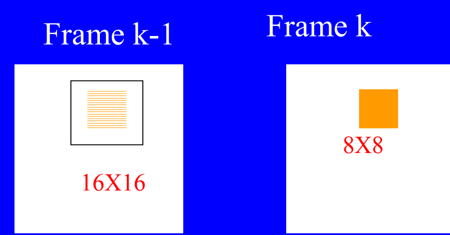
$$\Phi^T = \left[f_x(x,y,1) \quad f_y(x,y,1) \quad c_1 \quad c_2\right]$$

$$c_1 = x^2 f_x + xyf_x$$

Pseudo perspective

$$c_2 = xyf_x + y^2 f_y$$

Correlation Vs Spatiotemporal



Correlation Complexity

- m*m multiplications and additions
- 2*m*m additions and 2 divisions for two means
- 2*m*m multiplications and additions for variances

$$(u,v) = \arg\max_{u,v=-3\ldots3} \frac{1}{64\sigma_1\sigma_2} \sum_{i=0}^{-7}\sum_{j=0}^{-7}(f_k(x+i,y+j)-\mu_1).(f_{k-1}(x+i+u,y+j+v)-\mu_2)$$

Spatiotemporal Complexity

- 3* m*m subtractions for sptiotemporal derivatives
- (36+6)*m*m additions for generating linear system
- 6*6*6 multiplications and additions for solving 6 by 6 linear system

$$\left[\sum X^T \mathbf{f_X} \mathbf{f_X^T} X\right]\delta a = -\sum X^T \mathbf{f_X} f_t$$

Feature-based Matching

Feature-based Matching

- The input is formed by f1 and f2, two frames of an image sequence.
- Let Q1, Q2 and Q' be three NXN image regions.
- Let "d" be the unknown displacement vector between f1 and f2 of a feature point "p", on which Q1 is centered.

Algorithm

- Set d=0, center Q1 on p1.
- Estimate the displacement "d0" of "p", center of "Q1", using Lucas and Kanade method. Let d=d+d0.
- Let Q' bet the patch obtained by warping Q1 according to "d0". Compute Sum of Square (SSD) difference between new patch Q' and corresponding patch Q2 in frame f2.
- If SSD more than threshold, set Q1=Q' and go to step 1, otherwise exit.

Lucas & Kanade (Least Squares)

- Optical flow eq

$$f_x u + f_y v = -f_t$$

- Consider 3 by 3 window

$$f_{x1} u + f_{y1} v = -f_{t1}$$

$$\vdots$$

$$f_{x9} u + f_{y9} v = -f_{t9}$$

$$\begin{bmatrix} f_{x1} & f_{y1} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t1} \\ \vdots \\ -f_{t9} \end{bmatrix}$$

$$\mathbf{Au} = \mathbf{f_t}$$

Shi-Tomasi-Kanade(STK) Tracker

$$I(\delta(\mathrm{x}), t+\tau) = I(\mathrm{x}, t)$$

$$\delta(\mathrm{x}) = A\mathrm{x} + d$$

$$\varepsilon = \sum_W \left[I(A\mathrm{x} + d, t+\tau) - I(\mathrm{x}, t) \right]^2$$

After the first order Taylor expansion at A=I and d=0, we can get a linear 6×6 system:

$$Tz = f$$

$$z = \begin{bmatrix} A_{11} & A_{12} & A_{21} & A_{22} & d_1 & d_2 \end{bmatrix}$$

$$f = I_t \sum_W \begin{bmatrix} xI_x & yI_x & xI_y & yI_y & I_x & I_y \end{bmatrix}$$

Shi-Tomasi-Kanade(STK) Tracker

- Advantages:
  - Easy to implement .
  - Works very well with a small transformation.
- Drawbacks:
  - Fail to track in presence of a large rotation.
- Reason:
  - The rotation component implied in affine model is non-linear.

Useful Links

http://twtelecom.dl.sourceforge.net/sourceforge/opencvlibrary/OpenCVReferenceManual.pdf

http://vision.stanford.edu/~birch/klt/.

Lecture-8

Structure from Motion

Problem

- Given optical flow or point correspondences, compute 3-D motion (translation and rotation) and shape (depth).

3-D Rigid Motion (displacement)

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$X' = X - \alpha Y + \beta Z + T_X$$
$$Y' = \alpha X + Y - \gamma Z + T_Y$$
$$Z' = -\beta X + \gamma Y + Z + T_Z$$

Orthographic Projection (displacement model)

$$X' = X - \alpha Y + \beta Z + T_X$$
$$Y' = \alpha X + Y - \gamma Z + T_Y$$
$$Z' = -\beta X + \gamma Y + Z + T_Z$$

$$x' = x - \alpha y + \beta Z + T_X$$
$$y' = \alpha x + y - \gamma Z + T_Y$$

Perspective Projection (displacement)

$$X' = X - \alpha Y + \beta Z + T_X$$

$$Y' = \alpha X + Y - \gamma Z + T_Y$$

$$Z' = -\beta X + \gamma Y + Z + T_Z$$

$$x' = \frac{X - \alpha Y + \beta Z + T_X}{-\beta X + \gamma Y + Z + T_Z} \qquad x' = \frac{x - \alpha y + \beta + \frac{T_X}{Z}}{-\beta x + \gamma y + 1 + \frac{T_Z}{Z}}$$

$$y' = \frac{\alpha X + Y - \gamma Z + T_Y}{-\beta X + \gamma Y + Z + T_Z} \qquad y' = \frac{\alpha x + y - \gamma + \frac{T_Y}{Z}}{-\beta x + \gamma y + 1 + \frac{T_Z}{Z}}$$

Instantaneous Velocity Model

Optical Flow

3-D Rigid Motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} X'-X \\ Y'-Y \\ Z'-Z \end{bmatrix} = \begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \left(\begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

3-D Rigid Motion

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \Omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix}$$

$$\dot{\mathbf{X}} = \Omega \times \mathbf{X} + \mathbf{V}$$

Cross Product

Orthographic Projection

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

$$y = Y$$
$$x = X$$

$$u = \dot{x} = \Omega_2 Z - \Omega_3 y + V_1$$
$$v = \dot{y} = \Omega_3 x - \Omega_1 Z + V_2$$

(u,v) is optical flow

Perspective Projection (arbitrary flow)

$$x = \frac{fX}{Z}$$
$$y = \frac{fY}{Z}$$

$$u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f\frac{\dot{X}}{Z} - x\frac{\dot{Z}}{Z}$$
$$v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f\frac{\dot{Y}}{Z} - y\frac{\dot{Z}}{Z}$$

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

$$u = \dot{x} = f\frac{\dot{X}}{Z} - x\frac{\dot{Z}}{Z} = f\frac{\Omega_2 Z - \Omega_3 Y + V_1}{Z} - x\frac{\Omega_1 Y - \Omega_2 X + V_3}{Z}$$
$$v = \dot{y} = f\frac{\dot{Y}}{Z} - y\frac{\dot{Z}}{Z} = f\frac{\Omega_3 X - \Omega_1 Z + V_2}{Z} - y\frac{\Omega_1 Y - \Omega_2 X + V_3}{Z}$$

$$u = f(\frac{V_1}{Z} + \Omega_2) - \frac{V_3}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$
$$v = f(\frac{V_2}{Z} - \Omega_1) + \Omega_3 x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

Perspective Projection (optical flow)

$$u = f\left(\frac{V_1}{Z} + \Omega_2\right) - \frac{V_3}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = f\left(\frac{V_2}{Z} - \Omega_1\right) + \Omega_3 x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

$$u = \frac{fV_1 - V_3 x}{Z} + f\Omega_2 - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = \frac{fV_2 - V_3 y}{Z} - f\Omega_1 + \Omega_3 x + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

Pure Translation (FOE)

$$u^{(T)} = \frac{fV_1 - V_3 x}{Z}$$

$$v^{(T)} = \frac{fV_2 - V_3 y}{Z}$$

$$x_0 = f\frac{V_1}{V_3}, y_0 = f\frac{V_2}{V_3}$$

$$u^{(T)} = (x_0 - x)\frac{V_3}{Z}$$

$$p_0 = (x_0, y_0)$$

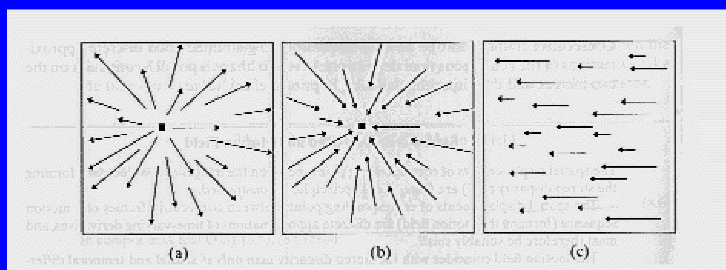
$$v^{(T)} = (y_0 - y)\frac{V_3}{Z}$$

Pure Translation (FOE)

- p_0 is the vanishing point of the direction of translation.
- p_0 is the intersection of the ray parallel to the translation vector with the image plane.

Pure Translation (FOE)

- If V3 is not zero, the flow field is radial, and all vectors point towards (or away from) a single point.
- The length of flow vectors is inversely proportional to the depth, if V_3 is not zero, then it is also proportional to the distance between p and p_0.



Pure Translation (FOE)

$$u^{(T)} = \frac{fV_1 - V_3 x}{Z}$$

$$v^{(T)} = \frac{fV_2 - V_3 y}{Z}$$

$$u^{(T)} = \frac{fV_1}{Z}$$

$$v^{(T)} = \frac{fV_2}{Z}$$

- If V_3=0, the flow field is parallel.

Structure From Motion

ORTHOGRAPHIC PROJECTION

Orthographic Projection (displacement)

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & \gamma \\ -\beta & \gamma & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$x' = x - \alpha y + \beta Z + T_X$$

$$y' = \alpha x + y - \gamma Z + T_Y$$

Simple Method

- **Two Steps Method**

-Assume depth is known, compute motion

$$x' = x - \alpha y + \beta Z + T_X$$
$$y' = \alpha x + y - \gamma Z + T_Y$$

$$\begin{bmatrix} x' - x \\ y' - y \end{bmatrix} = \begin{bmatrix} -y & Z & 0 & 1 & 0 \\ x & 0 & -Z & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ T_x \\ T_y \end{bmatrix}$$

Simple Method

-Assume motion is known, refine depth

$$x' = x - \alpha y + \beta Z + T_X$$
$$y' = \alpha x + y - \gamma Z + T_Y$$

$$\begin{bmatrix} \beta \\ -\gamma \end{bmatrix} [Z] = \begin{bmatrix} x' - x - \alpha y - T_x \\ y' - y - \alpha x - T_y \end{bmatrix}$$

Tomasi and Kanade

Orthographic Projection

Assumptions

- The camera model is orthographic.
- The positions of "p" points in "f" frames (f>=3), which are not all coplanar, have been tracked.
- The entire sequence has been acquired before starting (batch mode).
- Camera calibration not needed, if we accept 3D points up to a scale factor.

Tomasi & Kanade

Image point $\left\{(u_{fp}, v_{fp}) \mid f = 1,\ldots,F,\ p = 1,\ldots,P\right\}$

$$W = \begin{bmatrix} u_{11} \ldots u_{1P} \\ \vdots \\ u_{F1} \ldots u_{FP} \\ v_{11} \ldots v_{1P} \\ \vdots \\ v_{F1} \ldots v_{FP} \end{bmatrix} \qquad W = \begin{bmatrix} U \\ - \\ V \end{bmatrix}$$

Tomasi & Kanade

$$a_f = \frac{1}{P}\sum_{p=1}^{P} u_p \qquad b_f = \frac{1}{P}\sum_{p=1}^{P} v_p$$

$$\tilde{u}_{fP} = u_{fP} - a_{fP}$$

$$\tilde{v}_{fP} = v_{fP} - b_{fP}$$

$$s_p = (X_p, Y_P, Z_P)$$

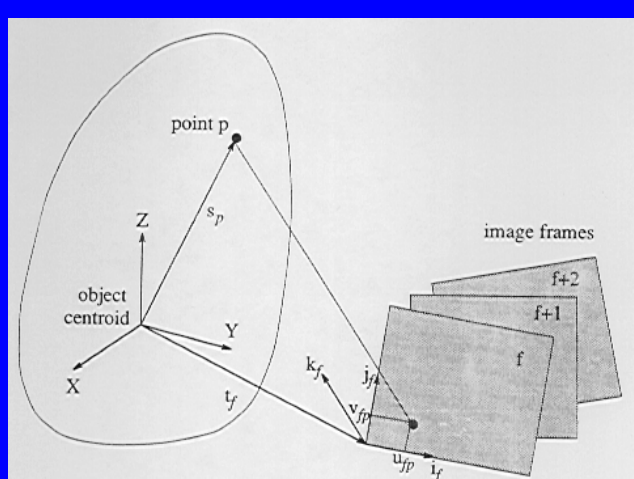
3D world point

$$u_{fP} = i_f^T (s_P - t_f)$$

$$v_{fP} = j_f^T (s_P - t_f)$$

Orthographic projection

$$k_f = i_f \times j_f$$



$$\tilde{u}_{fp} = u_{fP} - a_f$$

$$= i_f^T (s_p - t_f) - \frac{1}{P} \sum_{q=1}^{P} i_f^T (s_q - t_f)$$

$$= i_f^T \left[s_P - \frac{1}{P} \sum_{q=1}^{P} s_q \right]$$

$$= i_f^T s_P$$

Origin of world is at the centroid of object points

$$\tilde{u}_{fP} = i_f^T s_P$$

$$\tilde{v}_{fP} = j_f^T s_P$$

$$\tilde{W} = \begin{bmatrix} \tilde{U} \\ - \\ \tilde{V} \end{bmatrix}$$

$$\tilde{u}_{fP} = i_f^T s_P$$

$$\tilde{v}_{fP} = j_f^T s_P$$

$$\tilde{W} = \begin{bmatrix} \tilde{U} \\ - \\ \tilde{V} \end{bmatrix}$$

$$\tilde{W} = \begin{bmatrix} i_1^T \\ \vdots \\ i_f^T \\ j_1^T \\ \vdots \\ j_f^T \end{bmatrix} \begin{bmatrix} s_1 & \dots & s_P \end{bmatrix} = RS$$

2FX3 **3XP**

Rank of ***S*** is 3, because points in 3D space are not Co-planar

Rank Theorem

Without noise, the registered measurement matrix $\tilde{W}$ is at most of rank three.

$$\tilde{W} = \begin{bmatrix} i_1^T \\ \vdots \\ i_f^T \\ j_1^T \\ \vdots \\ j_f^T \end{bmatrix} \begin{bmatrix} s_1 & \dots & s_P \end{bmatrix} = RS$$

2FX3 **3XP**

Translation

$$\tilde{u}_{fp} = u_{fP} - a_f$$

$$u_{fp} = \tilde{u}_{fP} + a_f \qquad \tilde{u}_{fp} = i_f^T s_P$$

$$u_{fp} = i_f s_p + a_f \qquad u_{fp} = i_f^T (s_p - t_f)$$

a_f is projection of camera translation along x-axis

Translation

$$u_{fp} = i_f s_p + a_f \quad v_{fp} = j_f s_p + b_f$$

$$\mathbf{W} = \mathbf{RS} + \mathbf{te_p^T}$$

2FX3 3XP 2FX1 1XP

$$\mathbf{t} = (a_1, \ldots, a_f, b_1, \ldots, b_f)^T$$

$$\mathbf{e_p^T} = (1, \ldots 1)$$

Translation

Projected camera translation can be computed:

$$-i_f^T t_f = a_f = \frac{1}{P}\sum_{p=1}^{P} u_p$$

$$-j_f^T t_f = b_f = \frac{1}{P}\sum_{p=1}^{P} v_p$$

Noisy Measurements

- Without noise, the matrix $\tilde{W}$ must be at most of rank 3. When noise corrupts the images, however, $\tilde{W}$ will not be rank 3. Rank theorem can be extended to the case of noisy measurements.

Approximate Rank

SVD

$$\widetilde{W} = O_1 \Sigma O_2$$

2FXP PXP PXP

Singular Value Decomposition (SVD)

- For some linear systems Ax=b, Gaussian Elimination or LU decomposition does not work, because matrix A is singular, or very close to singular. SVD will not only diagnose for you, but it will solve it.

Singular Value Decomposition (SVD)

Theorem: Any m by n matrix A, for which $m \geq n$, can be written as

$$A = O_1 \Sigma O_2$$

mxn mxn nxn nxn

Σ is diagonal

O_1, O_2 are orthogonal

$O_1^T O_1 = O_2^T O_2 = I$

Singular Value Decomposition (SVD)

If A is square, then O_1, Σ, O_2 are all square.

$$O_1^{-1} = O_1^T$$

$$O_2^{-1} = O_2^T$$

$$\Sigma^{-1} = diag(\frac{1}{w_j})$$

$$A = O_1 \Sigma O_2$$

$$A^{-1} = O_2 diag(\frac{1}{w_j}) O_1$$

Singular Value Decomposition (SVD)

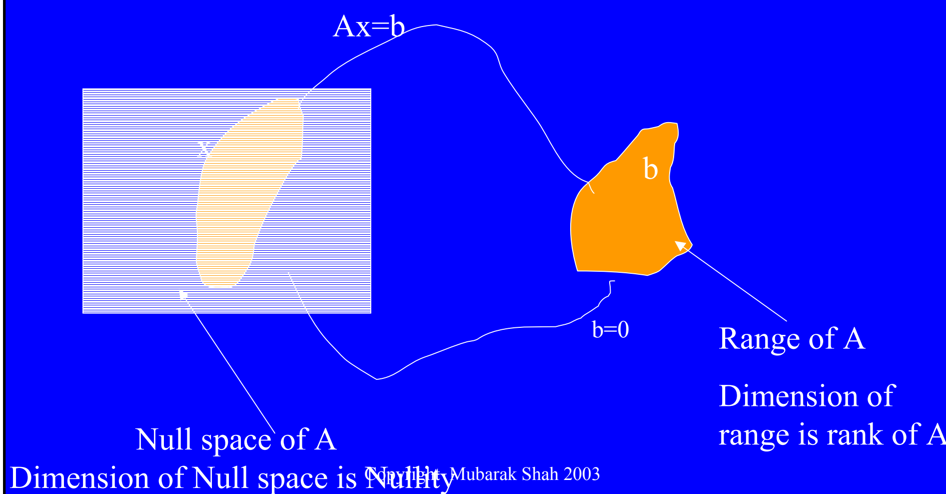
The condition number of a matrix is the ratio of the largest of the w_j to the smallest of w_j. A matrix is singular if the condition number is infinite, it is ill-conditioned if the condition number is too large.

Singular Value Decomposition (SVD)

$$Ax = b$$

- If A is singular, some subspace of “x” maps to zero; the dimension of the null space is called “nullity”.
- Subspace of “b” which can be reached by “A” is called range of “A”, the dimension of range is called “rank” of A.

Range and Null Space



Singular Value Decomposition (SVD)

- If A is non-singular its rank is “n”.
- If A is singular its rank <n.
- Rank+nullity=n

Singular Value Decomposition (SVD)

$$A = O_1 \Sigma O_2$$

- SVD constructs orthonormal basses of null space and range.
- Columns of O_1 with non-zero w_j spans range.
- Columns of O_2 with zero w_j spans null space.

Solution of Linear System

- How to solve Ax=b, when A is singular?
- If "b" is in the range of "A" then system has many solutions.
- Replace $\frac{1}{w_j}$ by zero if $w_j = 0$

$$x = O_2 [diag(\frac{1}{w_j})] O_1^T b$$

Solution of Linear System

If b is not in the range of A, above eq still gives the solution, which is the best possible solution, it minimizes:

$$r \equiv | Ax - b |$$

Approximate Rank

$$\tilde{W} = O_1 \Sigma O_2$$

$$O_1 \Sigma O_2 = O_1' \Sigma' O_2' + O_1'' \Sigma'' O_2''$$

3 P-3

$$O_1 = \begin{bmatrix} O_1' & O_1'' \end{bmatrix}$$ 2F

3 P-3

$$\Sigma = \begin{bmatrix} \Sigma' & 0 \\ 0 & \Sigma'' \end{bmatrix}$$ 3, P-3

$$O_2 = \begin{bmatrix} O_2' \\ O_2'' \end{bmatrix}$$ 3, P-3

P

Approximate Rank

$$\hat{W} = O_1' \Sigma' O_2'$$

The best rank 3 approximation to the ideal registered measurement matrix.

Rank Theorem for noisy measurement

The best possible shape and rotation estimate is obtained by considering only 3 greatest singular values of $\tilde{W}$ together with the corresponding left, right eigenvectors.

Approximate Rank

$\hat{R} = O_1'[\Sigma']^{1/2}$ Approximate Rotation matrix

$\hat{S} = [\Sigma']^{1/2} O_2'$ Approximate Shape matrix

$\hat{W} = \hat{R}\hat{S}$ This decomposition is not unique

$\hat{W} = (\hat{R}Q)(Q^{-1}\hat{S})$ Q is *any* 3X3 invertable matrix

Approximate Rank

$R = \hat{R}Q$

$S = Q^{-1}\hat{S}$

R and ***S*** are linear transformation of approximate Rotation and shape matrices

How to determine ***Q?***

$$\hat{i}_f^T Q Q^T \hat{i}_f = 1$$

$$\hat{j}_f^T Q Q^T \hat{j}_f = 1$$

$$\hat{i}_f^T Q Q^T \hat{j}_f = 0$$

Rows of rotation matrix are unit vectors and orthogonal

How to determine *Q*: Newton's Method

$$f_1(\mathbf{q}) = \hat{i}_i^T QQ^T \hat{i}_i^T - 1 = 0$$
$$f_2(\mathbf{q}) = \hat{j}_1^T QQ^T \hat{j}_1^T - 1 = 0$$
$$f_3(\mathbf{q}) = \hat{i}_1^T QQ^T \hat{j}_1^T = 0$$
$$\vdots$$
$$f_{3f-2}(\mathbf{q}) = \hat{i}_f^T QQ^T \hat{i}_f^T - 1 = 0$$
$$f_{3f-1}(\mathbf{q}) = \hat{j}_f^T QQ^T \hat{j}_f^T - 1 = 0$$
$$f_{3f}(\mathbf{q}) = \hat{i}_f^T QQ^T \hat{j}_f^T = 0$$

$$\mathbf{M}\Delta\mathbf{q} = \varepsilon$$

$$\Delta\mathbf{q} = [\Delta q_1, \ldots, \Delta q_9]$$

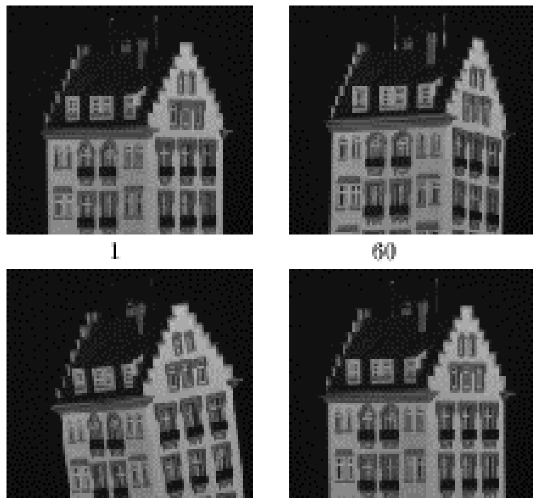
$$\mathbf{M}_{ij} = \frac{\partial f_i}{\partial q_j}$$

ε is error

Algorithm

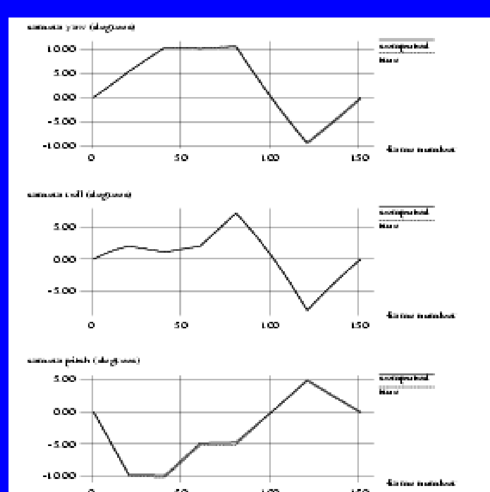
- Compute SVD of $\tilde{W} = O_1 \Sigma O_2$
- define $\hat{R} = O_1'[\Sigma']^{1/2}$ $\hat{S} = [\Sigma']^{1/2} O_2'$
- Compute Q
    - Compute $R = \hat{R}Q$ $S = Q^{-1}\hat{S}$

Hotel Sequence

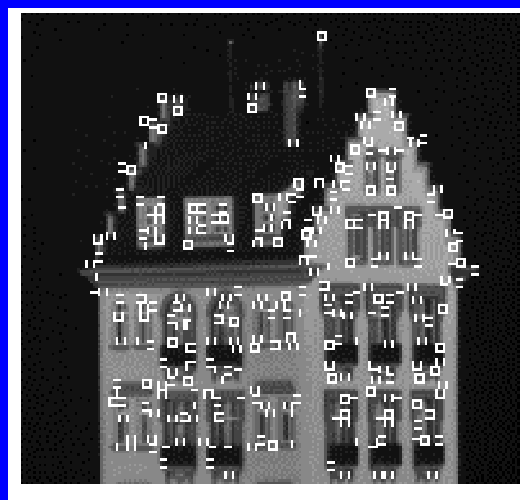


1 60 120 150

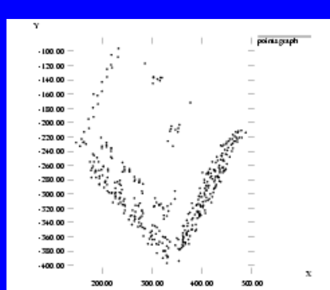
Results (rotations)

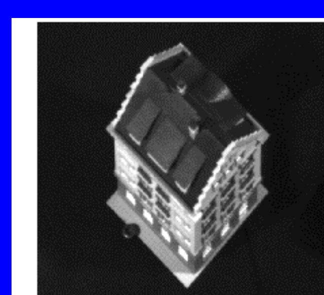


Selected Features

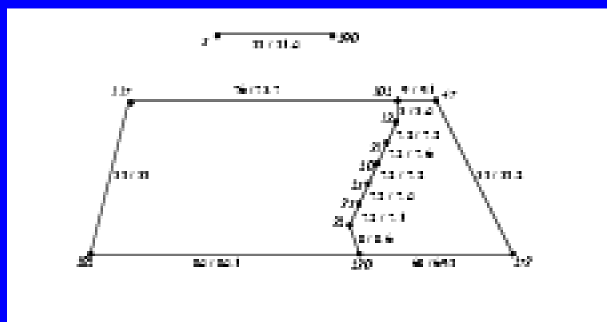


Reconstructed Shape

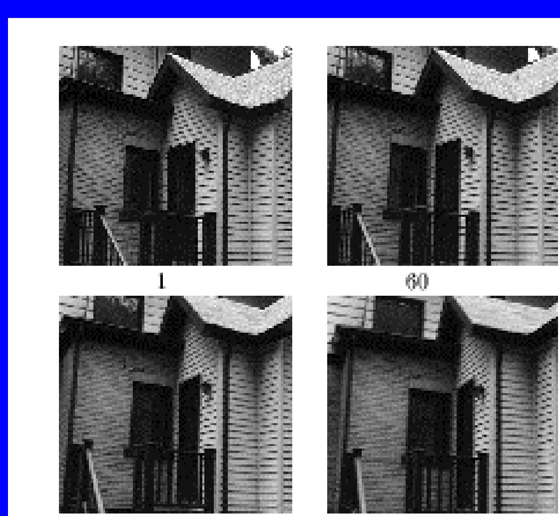




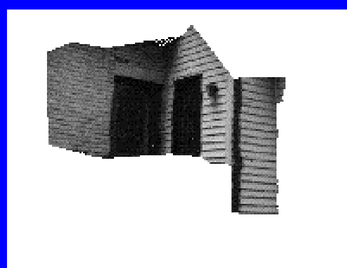
Comparison

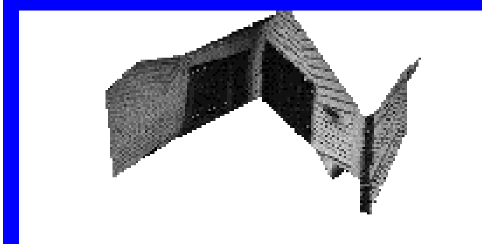


House Sequence



Reconstructed Walls





../tomasiTr92Figures.pdf

Web Page

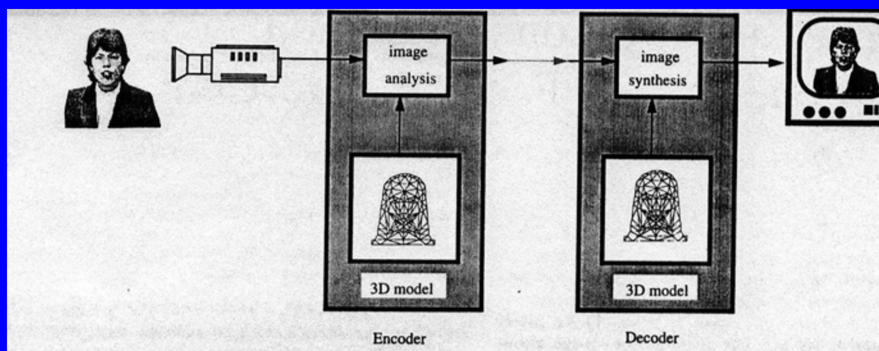
- http://vision.stanford.edu/cgi-bin/svl/publication/publication1992.cgi

Lecture-9

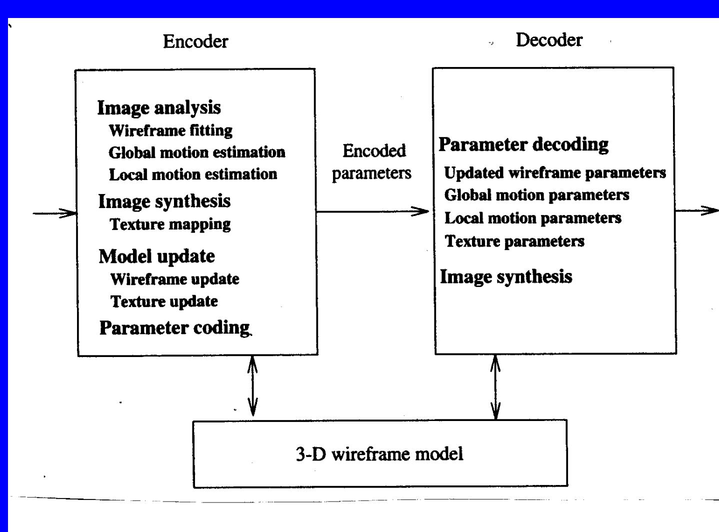
Model-base Video Compression

Li, Teklap

Model-Based Image Coding



Model-Based Image Coding



Model-Based Image Coding

- The transmitter and receiver both posses the same 3D face model and texture images.
- During the session, at the transmitter the facial motion parameters: global and local, are extracted.
- At the receiver the image is synthesized using estimated motion parameters.
- The difference between synthesized and actual image can be transmitted as residuals.

Candide Model

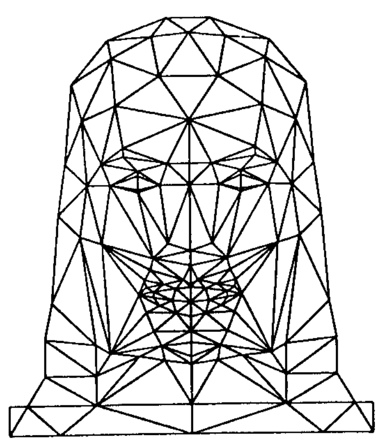


Fig. 2. Wire-frame model of the face.

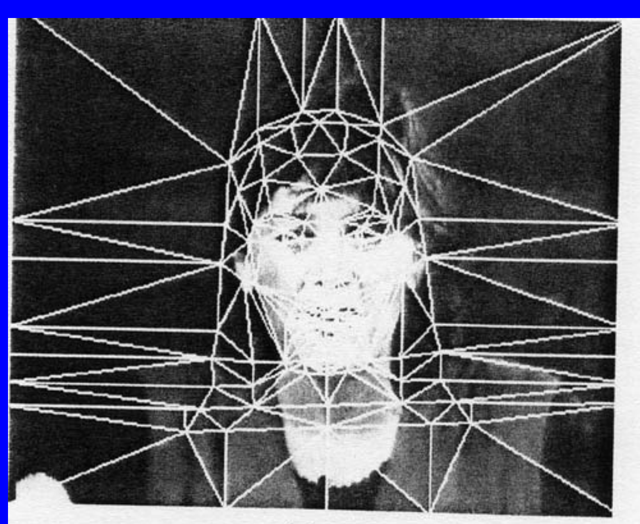
../CANDIDE.HTM

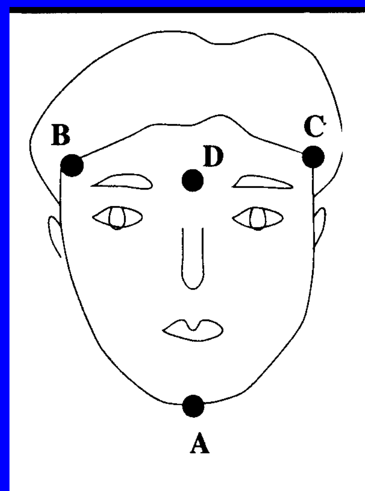
Face Model

- Candide model has 108 nodes, 184 polygons.
- Candide is a generic head and shoulder model. It needs to be conformed to a particular person's face.
- Cyberware scan gives head model consisting of 460,000 polygons.

Wireframe Model Fitting

- Fit orthographic projection of wireframe to the frontal view of speaker using Affine transformation.
  - Locate three to four features in the image and the projection of a model.
  - Find parameters of Affine transformation using least squares fit.
  - Apply Affine to all vertices, and scale $\sqrt{(a_1^2 + a_4^2)^2 / 2}$ depth.





QuickTime™ and a YUV420 codec decompressor are needed to see this picture.

QuickTime™ and a YUV420 codec decompressor are needed to see this picture.

Synthesis

- Collapse initial wire frame onto the image to obtain a collection of triangles.
- Map observed texture in the first frame into respective triangles.
- Rotate and translate the initial wire frame according to global and local motion, and collapse onto the next frame.
- Map texture within each triangle from first frame to the next frame by interpolation.

Texture Mapping

Video Phones

Motion Estimation

Perspective Projection (optical flow)

$$u = f\left(\frac{V_1}{Z} + \Omega_2\right) - \frac{V_3}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = f\left(\frac{V_2}{Z} - \Omega_1\right) + \Omega_3 x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

Optical Flow Constraint Eq

$$f_x u + f_y v + f_t = 0$$

$$f_x(f(\frac{V_1}{Z}+\Omega_2)-\frac{V_3}{Z}x-\Omega_3 y-\frac{\Omega_1}{f}xy+\frac{\Omega_2}{f}x^2)+f_y$$

$$(f(\frac{V_2}{Z}-\Omega_1)+\Omega_3 x-\frac{V_3}{Z}y+\frac{\Omega_2}{f}xy-\frac{\Omega_1}{f}y^2)+f_t=0$$

$$(f_x\frac{f}{Z})V_1+(f_y\frac{f}{Z})V_2+(\frac{f}{Z}(f_x x-f_y y)V_3+$$

$$(-f_x\frac{xy}{f}+f_y\frac{y^2}{f}-f_y f)\Omega_1+(f_x f+f_x\frac{x^2}{f}+f_y\frac{xy}{f})\Omega_2+$$

$$(f_x y+f_y x)\Omega_3=-f_t$$

$$(f_x \frac{f}{Z})V_1 + (f_y \frac{f}{Z})V_2 + (\frac{f}{Z}(f_x x - f_y y)V_3 +$$
$$(-f_x \frac{xy}{f} + f_y \frac{y^2}{f} - f_y f)\Omega_1 + (f_x f + f_x \frac{x^2}{f} + f_y \frac{xy}{f})\Omega_2 +$$
$$(f_x y + f_y x)\Omega_3 = -f_t$$

$$\mathbf{Ax} = \mathbf{b}$$

Solve by Least Squares

$$\mathbf{x} = (V_1, V_2, V_3, \Omega_1, \Omega_2, \Omega_3)$$

$$\mathbf{Ax} = \mathbf{b}$$

$$\begin{bmatrix} & & \vdots & & & \\ (f_x \frac{f}{Z}) & (f_y \frac{f}{Z}) & (\frac{f}{Z}(f_x x - f_y y) & (-f_x \frac{xy}{f} + f_y \frac{y^2}{f} - f_y f) & (f_x f + f_x \frac{x^2}{f} + f_y \frac{xy}{f}) & (f_x y + f_y x) \\ & & \vdots & & & \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} = \begin{bmatrix} \vdots \\ f_t \\ \vdots \end{bmatrix}$$

Comments

- This is a simpler (linear) problem than sfm because depth is assumed to be known.
- Since no optical flow is computed, this is called “direct method”.
- Only spatiotemporal derivatives are computed from the images.

Problem

- We have used 3D rigid motion, but face is not purely rigid!
- Facial expressions produce non-rigid motion.
- Use global rigid motion and non-rigid deformations.

3-D Rigid Motion

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \left(\begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

3-D Rigid Motion

$$\begin{bmatrix} X' - X \\ Y' - Y \\ Z' - Z \end{bmatrix} = \begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\dot{\mathbf{X}} = \Omega \times \mathbf{X} + \mathbf{V}$$

3-D Rigid+Non-rigid Motion

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T} + \mathbf{E}(\mathbf{X})\Phi$$

Facial expressions

Action Units:
-opening of a mouth
-closing of eyes
-raising of eyebrows

$$\Phi = (\phi_1, \phi_2, \ldots, \phi_m)^T$$

3-D Rigid+Non-rigid Motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^{m} E_{1i}(\mathbf{X})\phi_i \\ T_Y + \sum_{i=1}^{m} E_{2i}(\mathbf{X})\phi_i \\ T_Z + \sum_{i=1}^{m} E_{3i}(\mathbf{X})\phi_i \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \left(\begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^{m} E_{1i}(\mathbf{X})\phi_i \\ T_Y + \sum_{i=1}^{m} E_{2i}(\mathbf{X})\phi_i \\ T_Z + \sum_{i=1}^{m} E_{3i}(\mathbf{X})\phi_i \end{bmatrix}$$

3-D Rigid+Non-rigid Motion

$$\begin{bmatrix} X'-X \\ Y'-Y \\ Z'-Z \end{bmatrix} = \begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^{m} E_{1i}(\mathbf{X})\phi_i \\ T_Y + \sum_{i=1}^{m} E_{2i}(\mathbf{X})\phi_i \\ T_Z + \sum_{i=1}^{m} E_{3i}(\mathbf{X})\phi_i \end{bmatrix}$$

3-D Rigid+Non-rigid Motion

$$\dot{X} = -\Omega_3 Y + \Omega_2 Z + V_1 + \sum_{i=1}^{m} E_{1i}\phi_i$$

$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2 + \sum_{i=1}^{m} E_{2i}\phi_i$$

$$\dot{Z} = -\Omega_2 X + \Omega_1 Z + V_3 + \sum_{i=1}^{m} E_{3i}\phi_i$$

Perspective Projection (arbitrary flow)

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

$$u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f\frac{\dot{X}}{Z} - x\frac{\dot{Z}}{Z}$$

$$v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f\frac{\dot{Y}}{Z} - y\frac{\dot{Z}}{Z}$$

Perspective Projection (arbitrary flow)

$$\dot{X} = -\Omega_3 Y + \Omega_2 Z + V_1 + \sum_{i=1}^{m} E_{1i}\phi_i$$

$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2 + \sum_{i=1}^{m} E_{2i}\phi_i$$

$$\dot{Z} = -\Omega_2 X + \Omega_1 Z + V_3 + \sum_{i=1}^{m} E_{3i}\phi_i$$

$$u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f\frac{\dot{X}}{Z} - x\frac{\dot{Z}}{Z}$$

$$v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f\frac{\dot{Y}}{Z} - y\frac{\dot{Z}}{Z}$$

$$u = f\left(\frac{V_1 + \sum_{i=1}^{m} E_{1i}\phi_i}{Z} + \Omega_2\right) - \frac{V_3 + \sum_{i=1}^{m} E_{3i}\phi_i}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}x + \frac{\Omega_2}{f}x^2$$

$$v = f\left(\frac{V_2 + \sum_{i=1}^{m} E_{2i}\phi_i}{Z} - \Omega_1\right) + \Omega_3 x - \frac{V_3 + \sum_{i=1}^{m} E_{3i}\phi_i}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y$$

Optical Flow Constraint Eq

$$f_x u + f_y v + f_t = 0$$

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{x} = (V_1, V_2, V_3, \Omega_1, \Omega_2, \Omega_3, \phi_1, \phi_2, \ldots, \phi_m)$$

Lecture-10

Estimation Using Flexible Wireframe Model

Main Points

- Model photometric effects
- Simultaneously compute 3D motion and adapt the wireframe model.

Generalized Optical Flow Constraint

Lambertian Model

$$f(x,y,t) = \rho N(t).L$$

Albedo

Surface Normal

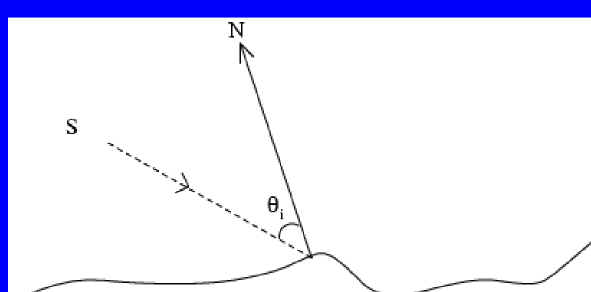
(-p,-q,1)

$$\frac{df(x,y,t)}{dt} = \rho L.\frac{dN}{dt}$$

$$f_x u + f_y v + f_t = \rho L.\frac{dN}{dt}$$

Lambertian Model

S=L, light source



$$f(x,y) = n.L = (n_x, n_y, n_z).(l_x, l_y, l_z)$$

$$f(x,y) = n.L = (\frac{1}{\sqrt{p^2+q^2+1}}(-p,-q,1)).(l_x, l_y, l_z)$$

Sphere

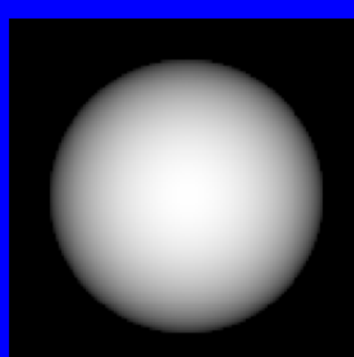
$$z = \sqrt{\left(R^2 - x^2 - y^2\right)}$$

$$p = \frac{\partial z}{\partial x} = -\frac{x}{z}$$

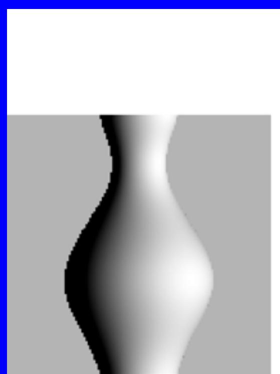
$$q = \frac{\partial z}{\partial y} = -\frac{y}{z}$$

$$(n_x, n_y, n_z) = \frac{1}{R}(x, y, z)$$

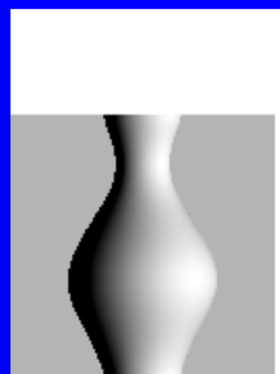
Sphere



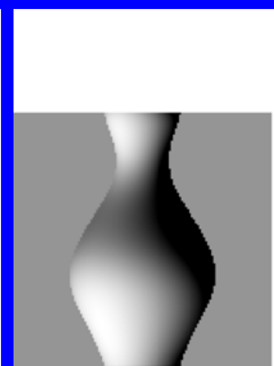
Vase



(1, 0, 1)



(-1, 1, 1)



(-1,-1, 1)

Orthographic Projection

$$u = \dot{x} = \Omega_2 Z - \Omega_3 y + V_1$$
$$v = \dot{y} = \Omega_3 x - \Omega_1 Z + V_2$$

(u,v) is optical flow

$$\dot{\mathbf{X}} = \Omega \times \mathbf{X} + \mathbf{V}$$
$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$
$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$
$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

Optical flow equation

$$f_x(\Omega_2 Z - \Omega_3 y + V_1) + f_y(\Omega_3 x - \Omega_1 Z + V_2) + f_t = \rho L.\frac{dN}{dt}$$

$$f_x(\Omega_2 Z - \Omega_3 y + V_1) + f_y(\Omega_3 x - \Omega_1 Z + V_2) + f_t =$$
$$\rho L.\left[\frac{(-p',-q',1)^T}{\sqrt{p'^2 + q'^2 + 1}} - \frac{(-p,-q,1)^T}{\sqrt{p^2 + q^2 + 1}}\right]$$

Homework 3.1
Show this.

Equation 24.8, page 473.

Error Function

$$E = \sum_i \sum_{(x,y)\varepsilon ithpatch} e_i^2$$

$$p_i x_1^{(ij)} + q_i x_2^{(ij)} + c_i = p_j x_1^{(ij)} + q_j x_2^{(ij)} + c_j$$

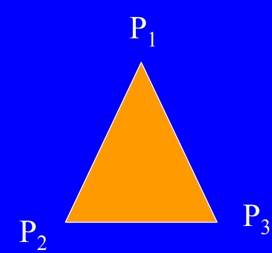
constraint

$$e_i(x,y) = f_x(\Omega_3 y - \Omega_2(p_i x + q_1 y + c_i) + V_1)$$
$$+ f_y(-\Omega_3 x + \Omega_1(p_i x + q_1 y + c_i) + V_2) + f_t$$
$$-\rho(L_1,L_2,L_3).(\frac{(-\frac{-\Omega_2 + p_i}{1+\Omega_2 p_i}, \frac{-\Omega_1 + q_i}{1-\Omega_1 q_i})}{((\frac{-\Omega_2 + p_i}{1+\Omega_2 p_i})^2 + (\frac{-\Omega_1 + q_i}{1-\Omega_1 q_i})^2 + 1)^{1/2}} -$$
$$\frac{(-p_i,-q_i,1)}{(p_i^2 + q_i^2 + 1)^{1/2}}$$

Homework 3.3
Show this.

Equation of a Planar Patch

$$P_1^{(i)} = (X_1^{(i)}, Y_1^{(i)}, Z_1^{(i)})$$
$$P_2^{(i)} = (X_2^{(i)}, Y_2^{(i)}, Z_2^{(i)})$$
$$P_3^{(i)} = (X_3^{(i)}, Y_3^{(i)}, Z_3^{(i)})$$



$$P^{(i)} = (X^{(i)}, Y^{(i)}, Z^{(i)})$$

$$\overline{P^{(i)}P_1^{(i)}}.(\overline{P_2^{(i)}P_1^{(i)}} \times \overline{P_3^{(i)}P_1^{(i)}}) = 0$$

Equation of a Planar Patch

$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

Homework 3.2
Show this.

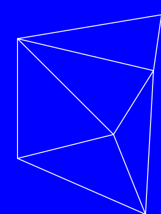
$$p_i = -\frac{(Y_2^{(i)} - Y_1^{(i)})(Z_3^{(i)} - Z_1^{(i)}) - (Z_2^{(i)} - Z_1^{(i)})(Y_3^{(i)} - Y_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

$$q_i = -\frac{(Z_2^{(i)} - Z_1^{(i)})(X_3^{(i)} - X_1^{(i)}) - (X_2^{(i)} - X_1^{(i)})(Z_3^{(i)} - Z_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

$$c_i = Z_1^{(i)} + X_1^{(i)} \frac{(Y_2^{(i)} - Y_1^{(i)})(Z_3^{(i)} - Z_1^{(i)}) - (Z_2^{(i)} - Z_1^{(i)})(Y_3^{(i)} - Y_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})} +$$
$$Y_1^{(i)} \frac{(Z_2^{(i)} - Z_1^{(i)})(X_3^{(i)} - X_1^{(i)}) - (X_2^{(i)} - X_1^{(i)})(Z_3^{(i)} - Z_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

Structure of Wireframe Model

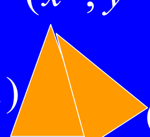
- Each triangular patch is either surrounded by two (if it is on the boundary of the wireframe) or three other triangles.



Neighboring patches must intersect at a straight line.

$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

$$p_i x^{(ij)} + q_i y^{(ij)} + c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j$$



Main Points of Algorithm

- Stochastic relaxation.
- In each iteration visit all patches in a sequential order.
  - If, at present iteration none of neighboring patches of i have been visited yet, then p_i, q_i, c_i are all independently perturbed.
  - If, only one of the neighbor, j, has been visited, then two parameters, say p_i, q_i, are independent and perturbed. The dependent variable c_i is calculated from the equation:

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

Main Points of Algorithm

- If two of the neighboring patches, say j and k, have already been visited, i.e., the variables p_k, q_k, c_{ik} and p_j, q_j, c_j have been updated, then only one variable p_i is independent, and is perturbed. q_i, c_i can be evaluated as

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

$$q_i = \frac{p_k x^{(ik)} + q_k y^{(ik)} + c_k - p_i x^{(ik)} - c_i}{y^{(ik)}}$$

Main Points of Algorithm

- The perturbation of structure parameters (surface normal) for each patch, results in the change of coordinates (X,Y,Z) of the nodes of wireframe.

Updating of (X,Y,Z):

Patches i, j, k intersect at node n.

$$p_i X^{(n)} + q_i Y^{(n)} + c_i = p_j X^{(n)} + q_j Y^{(n)} + c_j$$

$$p_i X^{(n)} + q_i Y^{(n)} + c_i = p_k X^{(n)} + q_k Y^{(n)} + c_k$$

$$\begin{bmatrix} X^{(n)} \\ Y^{(n)} \end{bmatrix} = \begin{bmatrix} p_i - p_j & q_i - q_j \\ p_i - p_k & q_i - q_k \end{bmatrix}^{-1} \begin{bmatrix} c_j - c_i \\ -c_i + c_k \end{bmatrix}$$

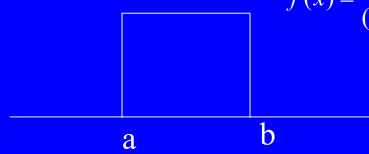
$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

Algorithm

- Estimate light source direction
- Initialize coordinates of all nodes using approximately scaled wireframe model (program-2). Determine initial values of surface normals for each triangle.
- Determine initial motion parameters based on selected feature correspondences and their depth values from wireframe model. (Assume motion parameters.)
- (A) Compute the value of error function E.

- If error E is less than some threshold, then stop
- Else
  - Perturb motion parameters (3 rotations and 2 translations) by random amount (zero mean Gaussian, s.d. equal to error E) (you can use uniform distribution)
  - Perturb structure parameters (p,q,c):
    - Perturb p, q, and c of first patch by adding random amount (zero mean Gaussian, s.d. equal to error E)
    - Increment count for all neighbors of patch-1 by 1

Uniform Distribution



$$f(x) = \frac{1}{(b-a)}$$

$$\overline{X} = mean = \frac{(a+b)}{2}$$

$$\sigma^2 = \text{variance} = \frac{(a-b)^2}{12}$$

Use rand() in C
to generate random
number between a range.

- For patch 2 to n
  - If the count==1
    - » Perturb p and q
    - » Compute c using equation for c_i
    - » Increment the count

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

– If count==2

» Perturb p_i

» Compute c_i and q_i using equations

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

$$q_i = \frac{p_k x^{(ik)} + q_k y^{(ik)} + c_k - p_i x^{(ik)} - c_i}{y^{(ik)}}$$

» Increment the count

– If p, q and c for at least three patches intersecting at node are updated, then update the coordinates of the node using equation.

$$\begin{bmatrix} X^{(n)} \\ Y^{(n)} \end{bmatrix} = \begin{bmatrix} p_i - p_j & p_j - p_k \\ q_i - q_j & q_j - q_k \end{bmatrix}^{-1} \begin{bmatrix} c_j - c_i \\ -c_j + c_k \end{bmatrix}$$

$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

- Go to step (A)

Lecture-11

Synthesizing Realistic Facial Expressions from Photographs:

Pighin et al

SIGGRAPH'98

Synthesizing Realistic Facial Expressions from Photographs

Pighin et al

SIGGRAPH'98

The Artist's Complete Guide to Facial Expression: Gary Faigin

- There is no landscape that we know as well as the human face. The twenty-five-odd square inches containing the features is the most intimately scrutinized piece of territory in existence, examined constantly, and carefully, with far more than an intellectual interest. Every detail of the nose, eyes, and mouth, every regularity in proportion, every variation from one individual to the next, are matters about which we are all authorities.

Main Points

- One view is not enough.
- Fitting of wire frame model to the image is a complex problem (pose estimation)
- Texture mapping is an important problem

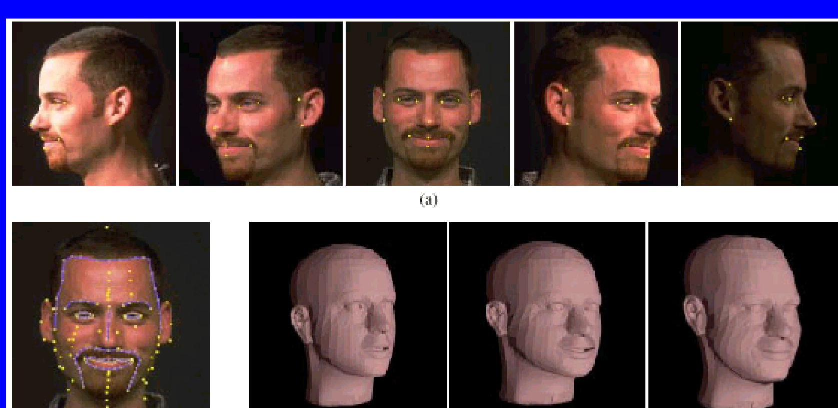
Synthesizing Realistic Facial Expressions

- Select 13 feature points manually in face image corresponding to points in face model created with Alias.
- Estimate camera poses and deformed 3d model points.
- Use these deformed values to deform the remaining points on the mesh using interpolation.

Synthesizing Realistic Facial Expressions

- Introduce more feature points (99) manually, and compute deformations as before by keeping the camera poses fixed.
- Use these deformed values to deform the remaining points on the mesh using interpolation as before.
- Extract texture.
- Create new expressions using morphing.

Synthesizing Realistic Facial Expressions


(a)

3D Rigid Transformation

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

Camera coordinates

Wireframe coordinates

$$x_i'^k = f^k \frac{X_i'^k}{Z_i'^k}, y_i'^k = f^k \frac{Y_i'^k}{Z_i'^k}$$ perspective

K=camera no.

3D Rigid Transformation

$$x_i'^k = f^k \frac{X_i'^k}{Z_i'^k}, y_i'^k = f^k \frac{Y_i'^k}{Z_i'^k}$$

$$x_i'^k = f_k \frac{r_{11}^k X_i + r_{12}^k Y_i + r_{13}^k Z_i + T_X^k}{r_{31}^k X_i + r_{32}^k Y_i + r_{33}^k Z_i + T_Z^k}$$

$$y_i'^k = f_k \frac{r_{21}^k X_i + r_{22}^k Y_i + r_{23}^k Z_i + T_Y^k}{r_{31}^k X_i + r_{32}^k Y_i + r_{33}^k Z_i + T_Z^k}$$

Model Fitting

$$x_i'^k = f_k \frac{\mathbf{r}_x^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_X^k}{\mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Z^k}$$

$$y_i'^k = f_k \frac{\mathbf{r}_y^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Y^k}{\mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Z^k}$$

Model Fitting

$$x_i'^k = f_k \frac{\mathbf{r}_x^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_X^k}{\mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Z^k}$$

$$y_i'^k = f_k \frac{\mathbf{r}_y^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Y^k}{\mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Z^k} \qquad \eta^k = \frac{1}{T_Z^k}, s^k = f^k \eta^k$$

$$x_i'^k = s_k \frac{\mathbf{r}_x^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_X^k}{1 + \eta^k \mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}}}$$

$$y_i'^k = s_k \frac{\mathbf{r}_y^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Y^k}{1 + \eta^k \mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}}}$$

Model Fitting

$$x_i'^k = s_k \frac{\mathbf{r}_x^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_X^k}{1 + \eta^k \mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}}}$$

$$y_i'^k = s_k \frac{\mathbf{r}_y^{\mathbf{k}} \mathbf{p}_{\mathbf{i}} + T_Y^k}{1 + \eta^k \mathbf{r}_z^{\mathbf{k}} \mathbf{p}_{\mathbf{i}}} \qquad w_i^k = (1 + \eta^k (\mathbf{r}_z^k . \mathbf{p}_i))^{-1}$$

$$w_i^k (x_i'^k + x_i'^k \eta^k (\mathbf{r}_{\mathbf{z}}^{\mathbf{k}} . \mathbf{p}_{\mathbf{i}}) - s^k (\mathbf{r}_X^{\mathbf{k}} . \mathbf{p}_{\mathbf{i}} + T_X^k)) = 0$$

$$w_i^k (y_i'^k + y_i'^k \eta^k (\mathbf{r}_{\mathbf{z}}^{\mathbf{k}} . \mathbf{p}_{\mathbf{i}}) - s^k (\mathbf{r}_Y^{\mathbf{k}} . \mathbf{p}_{\mathbf{i}} + T_Y^k)) = 0$$

Model Fitting

- Solve for unknowns in five steps:

  $s^k ; \mathbf{p}_i ; \mathbf{R}^k ; T_X^k , T_Y^k ; \eta^k$

- Use linear least squares fit.
- When solving for an unknown, assume other parameters are known.

Least Squares Fit

$$a_j.x - b_j = 0$$

$$\sum_j (a_j.x - b_j)^2$$

$$\sum_j (a_j a_j^T)x = \sum_j b_j a_j$$

$$w_i^k(x_i'^k + x_i'^k \eta^k (\mathbf{r_z^k}.\mathbf{p_i}) - s^k(\mathbf{r_X^k}.\mathbf{p_i} + T_X^k)) = 0$$

$$w_i^k(y_i'^k + y_i'^k \eta^k (\mathbf{r_z^k}.\mathbf{p_i}) - s^k(\mathbf{r_Y^k}.\mathbf{p_i} + T_Y^k)) = 0$$

Update for p

$$a_{2k+0} = w_i^k(x_i^k \eta^k r_z^k - s^k r_x^k) \qquad \mathrm{b}_{2k+0} = w_i^k(s^k T_x^k - x_i^k)$$

$$a_{2k+1} = w_i^k(y_i^k \eta^k r_z^k - s^k r_y^k) \qquad \mathrm{b}_{2k+1} = w_i^k(s^k T_y^k - y_i^k)$$

$$a_j.x - b_j = 0$$

$$\sum_j (a_j.x - b_j)^2$$

$$\sum_j (a_j a_j^T)x = \sum_j b_j a_j$$

$$w_i^k(x_i'^k + x_i'^k \eta^k (\mathbf{r_z^k}.\mathbf{p_i}) - s^k(\mathbf{r_X^k}.\mathbf{p_i} + T_X^k)) = 0$$

$$w_i^k(y_i'^k + y_i'^k \eta^k (\mathbf{r_z^k}.\mathbf{p_i}) - s^k(\mathbf{r_Y^k}.\mathbf{p_i} + T_Y^k)) = 0$$

Update for s^k

$$a_{2k+0} = w_i^k(r_x^k.p_i + t_x^k) \qquad \mathrm{b}_{2k+0} = w_i^k(x_i^k + x_i^k \eta^k (r_z^k.p_i))$$

$$a_{2k+1} = w_i^k(r_y^k.p_i + t_y^k) \qquad \mathrm{b}_{2k+1} = w_i^k(y_i^k + y_i^k \eta^k (r_z^k.p_i))$$

$$x_i' + x_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_x.\mathbf{p}_i + T_x) = 0$$

$$y_i' + y_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_y.\mathbf{p}_i + T_y) = 0$$

Solving for T_x and T_y

$$sT_x = x_i' + x_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_x.\mathbf{p}_i)$$

$$a_0 = s, b_0 = x_i' + x_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_x.\mathbf{p}_i)$$

$$sT_y = y_i' + y_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_y.\mathbf{p}_i)$$

$$a_0 = s, b_0 = y_i' + y_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_y.\mathbf{p}_i)$$

$$x_i' + x_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_x.\mathbf{p}_i + T_x) = 0$$

$$y_i' + y_i'\eta(\mathbf{r}_z.\mathbf{p}_i) - s(\mathbf{r}_y.\mathbf{p}_i + T_y) = 0$$

Solving for η

$$a_0 = x_i'(\mathbf{r}_z.\mathbf{p}_i), b_0 = s(\mathbf{r}_x.\mathbf{p}_i + T_x) - x_i'$$

$$a_1 = y_i'(\mathbf{r}_z.\mathbf{p}_i), b_1 = s(\mathbf{r}_y.\mathbf{p}_i + T_y) - y'$$

Rotation Around an Arbitrary Axis (Rodriguez's Formula)

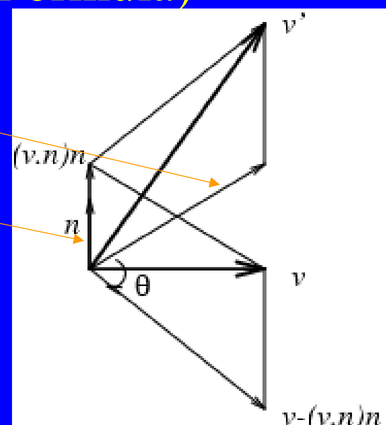
$$V = (V.n)n + (V - (V.n)n)$$

HW 4.1

$$V'_{\perp} = \cos\theta(V - (V.n)n) + \sin\theta(n \times (V - (V.n)n)$$

$$V'_{\amalg} = (V.n)n$$

$$V' = V'_{\perp} + V'_{\amalg}$$



$$V' = \cos\theta\, V + \sin\theta\, n \times V + (1 - \cos\theta)(V.n)n$$

$$V' = V + \sin\theta\, n \times V + (1 - \cos\theta) n \times (n \times V)$$

$$n \times (n \times V) = (V.n)n - V$$

$$n \times (n \times V) + V = (V.n)n$$

Rotation Around an Arbitrary Axis (Rodriguez's Formula)

$$V' = V + \sin\theta\, n \times V + (1 - \cos\theta)\, n \times (n \times V)$$

$$V' = R(n,\theta)V$$

$$R(n,\theta) = I + \sin\theta\, X(n) + (1 - \cos\theta) X^2(n)$$

HW 4.2

$$X(n) = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

Rotation Around an Arbitrary Axis (Rodriguez's Formula)

$$r = \|r\| \frac{r}{\|r|} = \theta n \qquad n = \frac{r}{\|r|}$$

$$R(n,\theta) = I + \sin\theta X(n) + (1-\cos\theta) X^2(n)$$

$$X(n) = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

$$R(r,\theta) = I + \sin\theta \frac{X(r)}{\|r\|} + (1-\cos\theta)\frac{X^2(r)}{\|r\|^2}$$

$$X(r) = \begin{bmatrix} 0 & -r_z & r_x \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

$$R^{it+1} \leftarrow \widetilde{R} R^{it}$$

$$R(m) = I + \sin\theta \frac{X(m)}{\theta} + \frac{X^2(m)}{\theta^2}(1-\cos\theta)$$

$$\widetilde{R} \approx I + X(m)$$

$$m = \theta n = (m_x, m_y, m_z)$$

$$\widetilde{r}_x^{\,k} = (1, -m_z, m_y)$$

$$\widetilde{r}_y^{\,k} = (m_z, 1, -m_x)$$

$$\widetilde{r}_z^{\,k} = (-m_y, m_x, 1)$$

$$w_i^k(x_i'^k + x_i'^k \eta^k(\mathbf{r}_\mathbf{z}^\mathbf{k}.\mathbf{p}_\mathbf{i}) - s^k(\mathbf{r}_X^\mathbf{k}.\mathbf{p}_\mathbf{i} + T_X^k)) = 0$$
$$w_i^k(y_i'^k + y_i'^k \eta^k(\mathbf{r}_\mathbf{z}^\mathbf{k}.\mathbf{p}_\mathbf{i}) - s^k(\mathbf{r}_Y^\mathbf{k}.\mathbf{p}_\mathbf{i} + T_Y^k)) = 0$$

$$q_i = R^{it} p_i$$
$$\tilde{r}_x^k = (1, -m_z, m_y)$$
$$\tilde{r}_y^k = (m_z, 1, -m_x)$$

$$R^{it+1} \leftarrow \tilde{R} R^{it}$$

$$\tilde{r}_z^k = (-m_y, m_x, 1)$$

$$w_i^k(x_i^k + x_i^k \eta(\tilde{r}_z^k.q_i) - s^k(\tilde{r}_x^k.q_i + t_x^k)) = 0$$
$$w_i^k(y_i^k + y_i^k (\tilde{r}_z^k.q_i) - s^k(\tilde{r}_y^k.q_i + t_y^k)) = 0$$

Solving for rotation:

$$R^{it+1} \leftarrow \tilde{R} R^{it}$$
$$q_i = R^{it} p_i$$
$$\tilde{r}_z^k = (-m_y, m_x, 1)$$
$$\tilde{r}_x^k = (1, -m_z, m_y)$$
$$\tilde{r}_y^k = (m_z, 1, -m_x)$$

$$w_i^k(x_i'^k + x_i'^k \eta^k(\mathbf{r}_\mathbf{z}^\mathbf{k}.\mathbf{p}_\mathbf{i}) - s^k(\mathbf{r}_X^\mathbf{k}.\mathbf{p}_\mathbf{i} + T_X^k)) = 0$$
$$w_i^k(y_i'^k + y_i'^k \eta^k(\mathbf{r}_\mathbf{z}^\mathbf{k}.\mathbf{p}_\mathbf{i}) - s^k(\mathbf{r}_Y^\mathbf{k}.\mathbf{p}_\mathbf{i} + T_Y^k)) = 0$$

$$w_i^k(x_i'^k + x_i'^k \eta(\tilde{r}_z^k.q_i) - s^k(\tilde{r}_x^k.q_i + T_x^k)) = 0$$
$$w_i^k(y_i'^k + y_i'^k \eta(\tilde{r}_z^k.q_i) - s^k(\tilde{r}_y^k.q_i + T_y^k)) = 0$$

$$x_i' + x_i'\eta(-m_y q_x + m_x q_y + q_z) - s(q_x - m_z q_y + m_y q_z + T_x) = 0$$

$$y_i' + y_i'\eta(-m_y q_x + m_x q_y + q_z) - s(m_z q_x + q_y - m_x q_z + T_y) = 0$$

$$x_i' + x_i'\eta(-m_y q_x + m_x q_y + q_z) - s(q_x - m_z q_y + m_y q_z + T_x) = 0$$
$$y_i' + y_i'\eta(-m_y q_x + m_x q_y + q_z) - s(m_z q_x + q_y - m_x q_z + T_y) = 0$$

$$\begin{bmatrix} x'\eta q_y & -x'\eta q_x - sq_z & sq_y \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} sT_x - x' - x'\eta q_z + sq_x \end{bmatrix}$$

$$\begin{bmatrix} y'\eta q_y + sq_z & -y'\eta q_x & -sq_x \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} sT_y - y' - y'\eta q_z + sq_y \end{bmatrix}$$

$$a_0 = \begin{bmatrix} x'\eta q_y & -x'\eta q_x + sq_z & sq_y \end{bmatrix}, b_0 = \begin{bmatrix} sT_x - x' - x'\eta q_z + sq_x \end{bmatrix}$$

$$a_1 = \begin{bmatrix} y'\eta q_y - sq_z & -y'\eta q_x & sq_x \end{bmatrix}, b_1 = \begin{bmatrix} sT_y - y' - y'\eta q_z + sq_y \end{bmatrix}$$

$$\sum_j (a_j a_j^T)x = \sum_j b_j a_j$$

QuickTime™ and a DV/DVCPRO - NTSC decompressor are needed to see this picture.

Video-realistic Speech Animation

QuickTime™ and a YUV420 codec decompressor are needed to see this picture.

Ezzat & Poggio, MIT

Recognizing Facial Expressions

Lecture-12

- Facial expressions reflect the emotional stage of a person.
- Recognizing facial expression from video sequences is a challenging problem.
- Applications
  - Perceptual user interface
  - Video compression (MPEG-4)
  - Synthesis of facial expressions

Facial Expressions

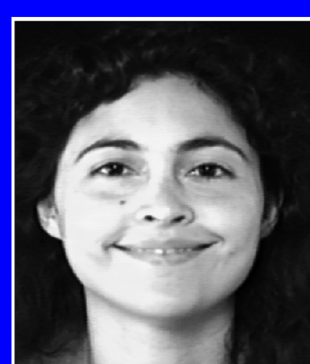
- Joy
  - The eyebrows are relaxed. The mouth is open, and mouth corners pulled back toward ears.
- Sadness
  - The inner eyebrows are bent upward. The eyes are slightly closed. The mouth is relaxed.
- Anger
  - The inner eyebrows are pulled downward and together. The eyes are wide open. The lips are pressed against each other or opened to expose teeth.

Facial Expressions

- Fear
  - The eyebrows are raised and pulled together. The inner eyebrows are bent upward. The eyes are tense and alert.
- Disgust
  - The eyebrows and eyelids are relaxed. The upper lip is raised and curled, often asymmetrically.
- Surprise
  - The eyebrows are raised. The upper eyelids are wide open, the lower relaxed. The jaw is open.

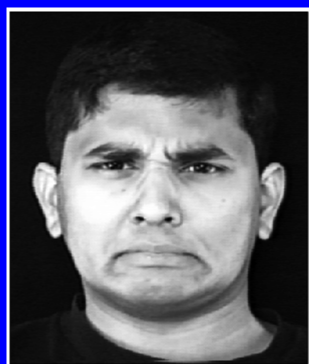
FACIAL EXPRESSIONS

RAISE EYE BROWS



SMILE

FACIAL EXPRESSIONS

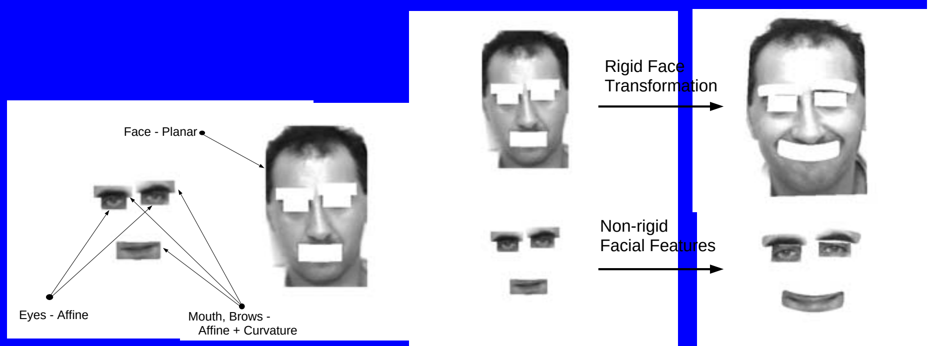


DISGUST

ANGER

Black and Yacoob Algorithm

- Given the location of the face, eyes, brows, and mouth estimate the rigid motion of the face using pseudo perspective motion model.
- Use the face motion to register images through warping.
- Estimate relative motion of face features (eyes, mouth, brows).
- The estimated feature motions are used to predict locations of features in the next frame, and the process is repeated.
- The estimated motion is used to classify the facial expressions.



Affine

$$u(x,y) = a_1 x + a_2 y + b_1$$
$$v(x,y) = a_3 x + a_4 y + b_2$$

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2 \end{bmatrix}$$

Affine

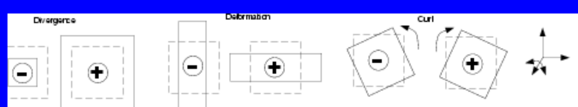
$$u(x,y) = a_1 x + a_2 y + b_1$$

$$v(x,y) = a_3 x + a_4 y + b_2$$

Expansion or contraction: $divergence = u_x + v_y = a_1 + a_4$

Rotation around Z: $curl = -(u_y - v_x) = -(a_2 - a_3)$

Squashing or stretching: $deformation = (u_x - v_y) = (a_1 - a_4)$



Pseudo Perspective

$$u(x,y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy$$

$$v(x,y) = a_6 + a_7 x + a_8 y + a_4 xy \ + a_5 y^2$$

a_4=yaw: rotation around y-axis

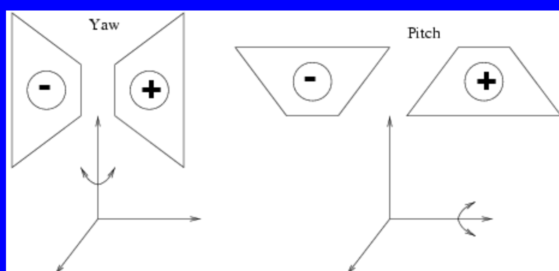
a_5=pitch: rotation around x-axis

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} 1 & x & y & x^2 & xy & 0 & 0 & 0 \\ 0 & 0 & 0 & xy & y^2 & 1 & x & y \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix}$$

Pseudo Perspective

$$u(x,y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy$$

$$v(x,y) = a_6 + a_7 x + a_8 y + a_4 xy \ \ + a_5 y^2$$



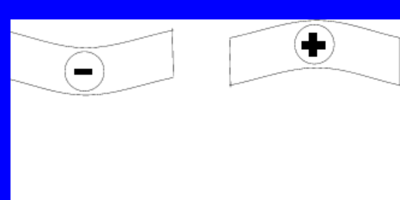
a_4=yaw
a_5=pitch

Affine with Curvature

$$u(x,y) = a_1 x + a_2 y + b_1$$

$$v(x,y) = a_3 x + a_4 y + b_2 + cx^2$$

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 & x^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2 \\ c \end{bmatrix}$$



Rules for Classifying Expressions

- Anger
  - B: inward lowering of brows and mouth contraction
  - E: outward raising of brows and mouth expansion
- Disgust
  - B: mouth horizontal expansion and lowering of brows
  - E: mouth contraction and raising of brows
- Happiness
  - B: upward curving of mouth and expansion or horizontal deformation
  - E: downward curving of mouth and contraction or horizontal deformation

Rules for Classifying Expressions

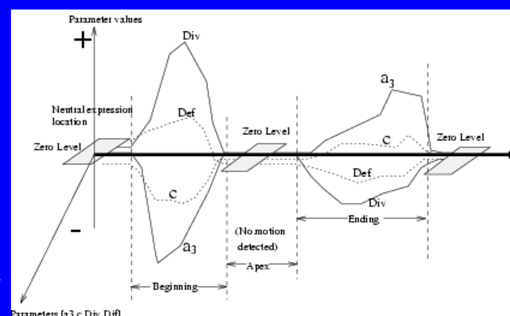
- Surprise
  - B: raising brows and vertical expansion of mouth
  - E: lowering brows and vertical contraction of mouth
- Sadness
  - B: downward curving of mouth and upward-inward motion in the inner parts of brows
  - E: upward curving of mouth and downward-outward motion in inner parts of brows
- Fear
  - B: expansion of mouth and raising-inwards inner parts of brows
  - E: contraction of mouth and lowering inner parts of brows

Smile Expression

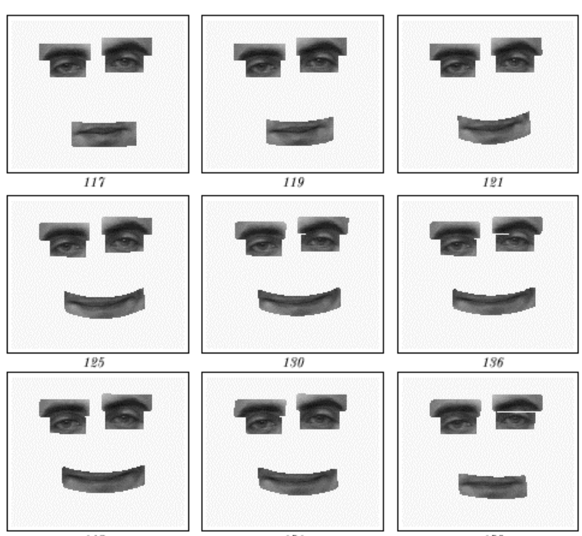
Upward-outward motion of mouth corners results in –ve curvature

Horizontal and overall vertical stretching result in +ve div & def.

Some upward trans is caused by raising of lower and upper lips due to stretching of the mouth (a3 is –ve).



Smile



117 119 121 125 130 136 143 151 159

Figure 8: Smile experiment: facial expression tracking.

Smile Mouth Parameters

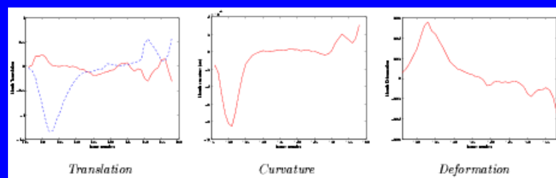


Figure 9: Smile mouth parameters. For translation, solid and dashed lines indicate horizontal and vertical motion respectively.

16

Anger

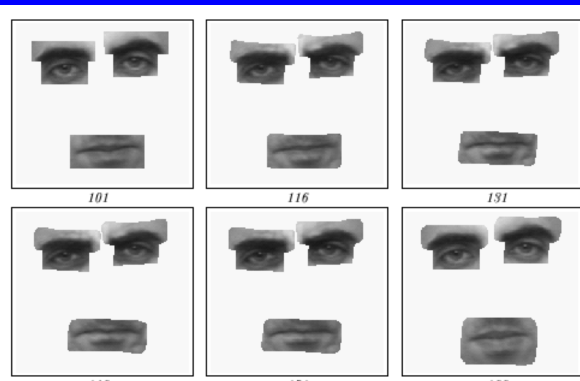


Figure 10: Anger experiment: facial expression tracking. Features every 15 frames.

Anger Motion Parameters

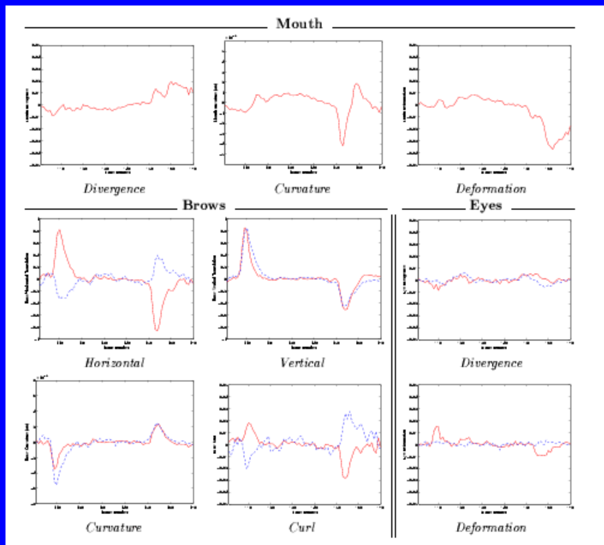
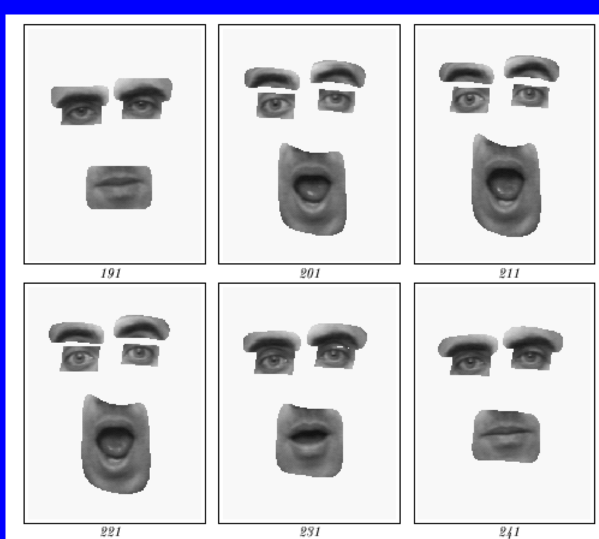
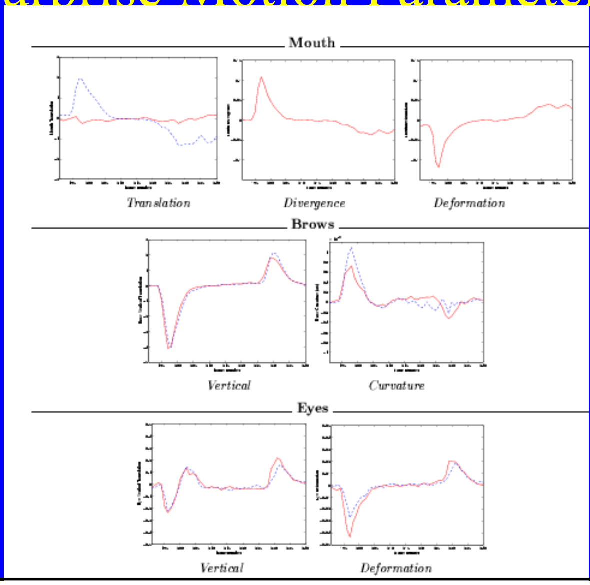


Figure 11: Anger motion parameters; the solid line indicates the right eye or brow while the dashed line indicates the left eye or brow.

Surprise



Surprise Motion Parameters



Blinking

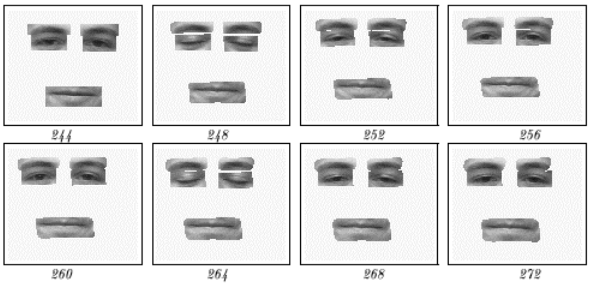
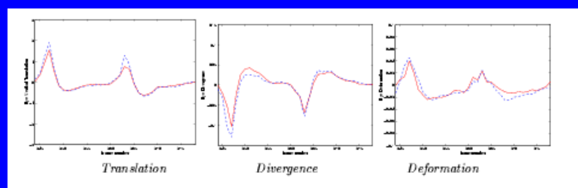


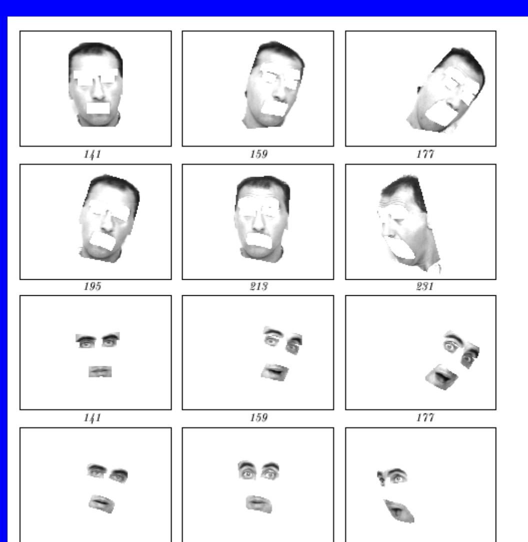
Figure 14: Blinking experiment: facial feature tracking. Features every four frames.

Blinking Motion Parameters for Eyes



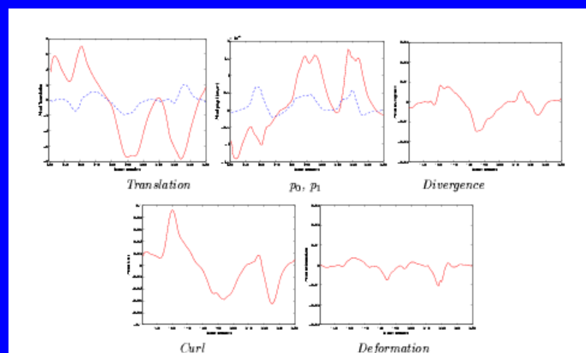
Copyright Mubarak Shah 2003

Rotation

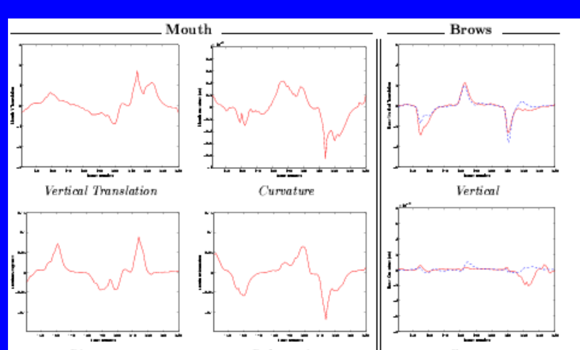


Rotate Face motion parameters

P_0 rot y
P_1 rot X



Rotation Motion Parameters



Mid-level predicates for Mouth

Table 3: The mid-level predicates derived from deformation and motion parameter estimates.

Parameter	Threshold	Derived Predicates
a_0	> 0.25	Mouth rightward
	< -0.25	Mouth leftward
a_3	< -0.1	Mouth upward
	> 0.1	Mouth downward
Div	> 0.02	Mouth expansion
	< -0.02	Mouth contraction
Def	> 0.005	Mouth horizontal deformation
	< -0.005	Mouth vertical deformation
Curl	> 0.005	Mouth clockwise rotation
	< -0.005	Mouth counterclockwise rotation
c	< -0.0001	Mouth curving upward ('U' like)
	> 0.0001	Mouth curving downward

Mid-level predicates for Head

Table 4: The mid-level predicates derived from deformation and motion parameter estimates as applied to head motion.

Parameter	Threshold	Derived Predicates
a_0	> 0.5	Head rightward
	< -0.5	Head leftward
a_3	< -0.5	Head upward
	> 0.5	Head downward
Div	> 0.01	Head expansion
	< -0.01	Head contraction
Def	> 0.01	Head horizontal deformation
	< -0.01	Head vertical deformation
Curl	> 0.005	Head clockwise rotation
	< -0.005	Head counterclockwise rotation
p_0	< -0.00005	Head rotating rightward around the neck
	> 0.00005	Head rotating leftward around the neck
p_1	< -0.00005	Head rotating forward
	> 0.00005	Head rotating backward

Parameter values used for classifying expressions

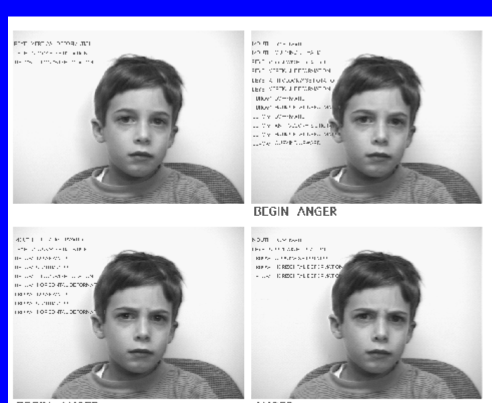
Expr.	B/E	Feature	a_0	a_3	Div	Curl	Def	c
Anger	B	Mouth		−		0	+	−
		R. Brow	+	+		+	+	−
		L. Brow	−	+		−	+	−
		R. Eye	+		−		+	
		L. Eye	−		−		+	
Anger	E	Mouth		+		0	−	+
		R. Brow	−	−		−	−	+
		L. Brow	+	−		+	−	+
		R. Eye	−		+		−	
		L. Eye	+		+		−	
Happiness	B	Mouth		−			+	−
Happiness	E	Mouth		+			−	+
Surprise	B	Mouth		+	+	0	−	
		R. Brow	−	−		−		+
		L. Brow	+	−		+		+
		R. Eye	−	−	+		−	
		L. Eye	+	−	+		−	
Surprise	E	Mouth		−	−	0	+	
		R. Brow	+	+		+		−
		L. Brow	−	+		−		−
		R. Eye	+	+	−		+	
		L. Eye	−	+	−		+	

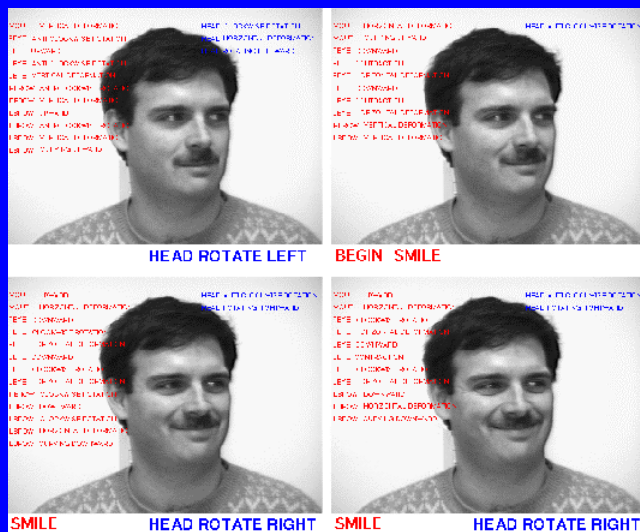
Forty Test Subjects

Results

Expression	Rate
Surprise	91%
Happiness	95%
Anger	90%
Disgust	93%
Fear	83%
Sadness	100%

Beginning of Anger Expression





Frames from 10 Video Clips

Results

Expression	Rate
Surprise	86%
Happiness	95%
Anger	80%
Disgust	50%
Fear	100%
Sadness	60%

http://www.cfar.umd.edu/ftp/TRs/CVL-Reports-1995/TR3401-Black.ps.gz

Lecture-13

Face Recognition & Visual Lipreading

Face Recognition

Simple Approach

- Recognize faces (mug shots) using gray levels (appearance)
- Each image is mapped to a long vector of gray levels
- Several views of each person are collected in the model-base during training
- During recognition a vector corresponding to an unknown face is compared with all vectors in the model-base
- The face from model-base, which is closest to the unknown face is declared as a recognized face.

Problems and Solution

- Problems :
  - Dimensionality of each face vector will be very large (250,000 for a 512X512 image!)
  - Raw gray levels are sensitive to noise, and lighting conditions.
- Solution:
  - Reduce dimensionality of face space by finding principal components (eigen vectors) to span the face space
  - Only a few most significant eigen vectors can be used to represent a face, thus reducing the dimensionality

Eigen Vectors and Eigen Values

The eigen vector, x, of a matrix *A* is a special vector, with the following property

$$Ax = \lambda x$$ Where λ is called eigen value

To find eigen values of a matrix A first find the roots of:

$$\det(A - \lambda I) = 0$$

Then solve the following linear system for each eigen value to find corresponding eigen vector

$$(A - \lambda I)x = 0$$

Example

$$A=\begin{bmatrix}-1 & 2 & 0\\ 0 & 3 & 4\\ 0 & 0 & 7\end{bmatrix}$$

Eigen Values

$$\lambda_1=7,\ \lambda_2=3, \lambda_3=-1$$

$$\mathbf{x_1}=\begin{bmatrix}1\\4\\4\end{bmatrix},\ \mathbf{x_2}=\begin{bmatrix}1\\2\\0\end{bmatrix},\ \mathbf{x_3}=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

Eigen Vectors

Eigen Values

$$\det(A-\lambda I)=0$$

$$\det(\begin{bmatrix}-1 & 2 & 0\\ 0 & 3 & 4\\ 0 & 0 & 7\end{bmatrix}-\lambda\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix})=0$$

$$\det(\begin{bmatrix}-1-\lambda & 2 & 0\\ 0 & 3-\lambda & 4\\ 0 & 0 & 7-\lambda\end{bmatrix})=0$$

$$(-1-\lambda)((3-\lambda)(7-\lambda)-0)=0$$

$$(-1-\lambda)(3-\lambda)(7-\lambda)=0$$

$$\lambda=-1, \quad \lambda=3, \quad \lambda=7$$

Eigen Vectors

$\lambda = -1$ $\quad (A - \lambda I)x = 0$

$$\left(\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 8 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$0 + 2x_2 + 0 = 0$$

$$0 + 4x_2 + 4x_3 = 0$$

$$0 + 0 + 8x_3 = 0$$

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 0$$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Face Recognition

Collect all gray levels in a long vector *u:*

$$u = (I(1,1),\ldots,I(1,N),I(2,1),\ldots,I(2,N),\ldots,I(M,1),\ldots,I(M,N))^T$$

Collect *n* samples (views) of each of *p* persons in matrix A (MN X pn):

$$A = \left[u_1^1,\ldots u_n^1, u_1^2 \ldots, u_n^2, \ldots, u_1^p \ldots, u_n^p\right]$$

Form a correlation matrix L (MN X MN):

$$L = AA^T$$

Compute eigen vectors, $\phi_1, \phi_2, \phi_3, \ldots \phi n_1$, of L, which form a bases for whole face space

Face Recognition

Each face, u, can now be represented as a linear combination of eigen vectors

$$u = \sum_{i=1}^{n_1} a_i \phi_i$$

Eigen vectors for a symmetric matrix are orthonormal:

$$\phi_i^T . \phi_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Face Recognition

$$u_x^T . \phi_i = (\sum_{i=1}^{n} a_i \phi_i)^T . \phi_i$$

$$= (\mathrm{a}_1 \phi_1^T + \mathrm{a}_2 \phi_2^T + \ldots + \mathrm{a}_i \phi_i^T + \ldots + \mathrm{a}_n \phi_n^T) \phi_i$$

$$u_x^T . \phi_i = (\mathrm{a}_1 \phi_1^T . \phi_i + \mathrm{a}_2 \phi_2^T . \phi_i + \ldots + \mathrm{a}_i \phi_i^T . \phi_i + \ldots + \mathrm{a}_n \phi_n^T . \phi_i)$$

$$u_x^T . \phi_i = a_i$$

Therefore: $a_i = u_x^T . \phi_i$

Face Recognition

L is a large matrix, computing eigen vectors of a large matrix is time consuming. Therefore compute eigen vectors of a smaller matrix, C:

$$C = A^T A$$

Let α_i be eigen vectors of C, then $A\alpha_i$ are the eigen vectors of L:

$$C\alpha_i = \lambda_i \alpha_i$$

$$A^T A\alpha_i = \lambda_i \alpha_i$$

$$AA^T(A\alpha_i) = \lambda_i (A\alpha_i)$$

$$L(A\alpha_i) = \lambda_i (A\alpha_i)$$

Training

- Create A matrix from training images
- Compute C matrix from A.
- Compute eigenvectors of C.
- Compute eigenvectors of L from eigenvectors of C.
- Select few most significant eigenvectors of L for face recognition.
- Compute coefficient vectors corresponding to each training image.
- For each person, coefficients will form a cluster, compute the mean of cluster.

Recognition

- Create a vector *u* for the image to be recognized.
- Compute coefficient vector for this *u*.
- Decide which person this image belongs to, based on the distance from the cluster mean for each person.


```
load faces.mat
C=A'*A;
[vectorC,valueC]=eig(C);
ss=diag(valueC);
[ss,iii]=sort(-ss);
vectorC=vectorC(:,iii);
vectorL=A*vectorC(:,1:5);
Coeff=A'*vectorL;
for I=1:30
        model(i, :)=mean(coeff((5*(i-1)+1):5*I,:));
end
while (1)
        imagename=input('Enter the filename of the image to
        Recognize(0 stop):');
        if (imagename <1)
        break;
        end;
        imageco=A(:,imagename)'*vectorL;
        disp ('');
        disp ('The coefficients for this image are:');
```

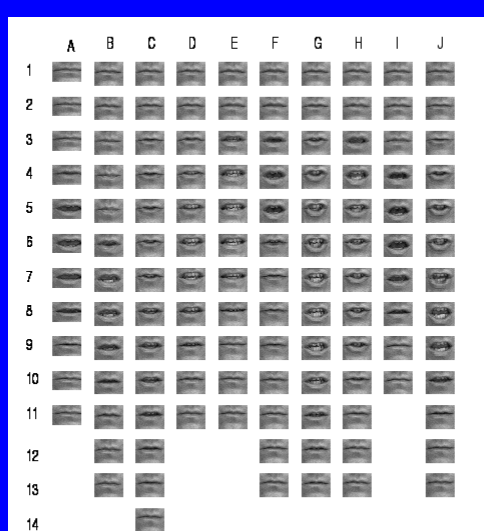
```
mess1=sprintf(‘%.2f %.2f %.2f %.2f %.2f’,
imageco(1),imageco(2),imageco(3),imageco(4),
imageco(5));
disp(mess1);
top=1;
for I=2:30
        if (norm(model(i,:)-imageco,1)<norm(model
        (top, : )-imageco,1))
        top=i
        end
end
mess1=sprintf(‘The image input was a image of person
number %d’,top);
disp(mess1);
end
b=A(:,81);
b=reshape(b,34,51);
imshow(b,gray(255)):
```


Webpage

http://vismod.www.media.mit.edu/vismod/demos/

Visual Lipreading

Image Sequences of "A" to "J"



Particulars

- Problem: Pattern differ spatially
- Solution: Spatial registration using SSD
- Problem : Articulations vary in length, and thus, in number of frames.
- Solution: Dynamic programming for temporal warping of sequences.
- Problem: Features should have compact representation.
- Solution: Principle Component Analysis.

Feature Subspace Generation

- Generate a lower dimension subspace onto which image sequences are projected to produce a vector of coefficients.
- Components
  - Sample Matrix
  - Most Expressive Features

Generating the Sample Matrix

- Consider ε letters, each of which has a training set of K sequences. Each sequence is compose of images:

$$I_1, I_2, \ldots, I_P$$

- Collect all gray-level pixels from all images in a sequence into a vector:

$$u = (I_1(1,1), \ldots, I_1(M,N), I_2(1,1), \ldots, I_2(M,N), \ldots I_P(1,1), \ldots, I_P(M,N))$$

. Generating the Sample Matrix

- For letter ω , collect vectors into matrix T

$$T_\omega = \left[u^1, u^2, \ldots u^K\right]$$

- Create sample matrix A:

$$A = \left[T_1, T_2, \ldots T_\varepsilon\right]$$

- The eigenvectors of a matrix $L = AA^T$ are defined as:

$$L\phi_i = \lambda_i \phi_i$$

The Most Expressive Features

- ϕ is an orthonormal basis of the sample matrix.

- Any image sequence, u, can be represented as:

$$u = \sum_{n=1}^{Q} a_n \phi_n$$

- Use Q most significant eigenvectors.
  - The linear coefficients can be computed as:

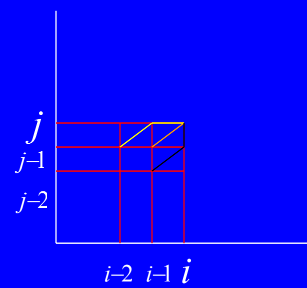
$$a_n = u^T \phi_n$$

Training Process

- Model Generation
  - Warp all the training sequences to a fixed length.
  - Perform spatial registration (SSD).
  - Perform PCA.
  - Select Q most significant eigensequences, and compute coefficient vectors "a".
  - Compute mean coefficient vector for each letter.

Warping

$$A = [a_1, a_2, \ldots, a_i, a_I]$$
$$B = [b_1, b_2, \ldots, b_j, b_J]$$



$$d_{ij} = |a_i - b_j|$$

$$g_{11} = 2d_{11}$$

$$g(i,j) = \min \begin{bmatrix} g(i-1, j-2) + 2d(i, j-1) + d(i,j) \\ g(i-1, j-1) + 2d(i,j) \\ g(i-2, j-1) + 2d(i-1, j) + d(i,j) \end{bmatrix}$$

Recognition

- Warp the unknown sequence.
- Perform spatial registration.
- Compute:

$$a_i^x = u_x^T . \phi_i$$

$$d^w = \| a^w - a^x \|$$

- Determine best match by $\min_\omega (d^\omega)$

Extracting letters from Connected Sequences

- Average absolute intensity difference function

$$f(n) = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} \| I_n(x,y) - I_{n-1}(x,y) \|$$

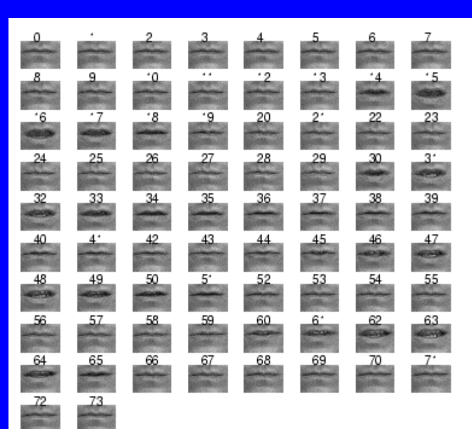
- f is smoothed to obtain g.
- Articulation intervals correspond to peaks and non-articulation intervals correspond to valleys in “g”.

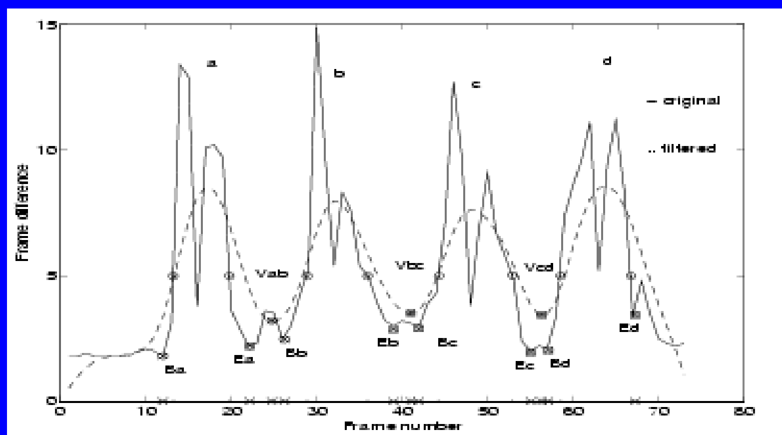
A 12-22

B 26-39

C 42-55

D 57-67





Extracting letters from Connected Sequences

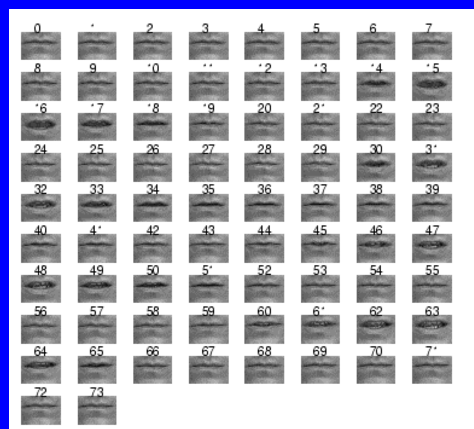
- Detect valleys in g.
- From valley locations in g, find locations where f crosses high threshold.
- Locate beginning and ending frames.

A 12-22

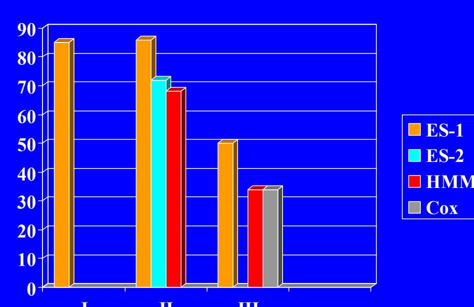
B 26-39

C 42-55

D 57-67

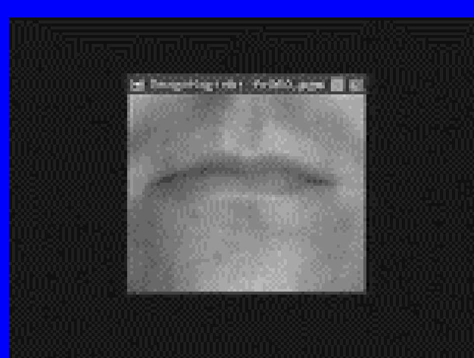


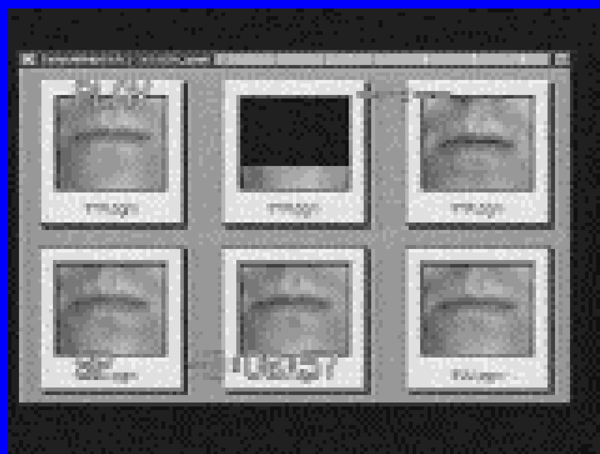
Results



I: "A" to "J" one speaker, 10 training seqs
II. "A" to "M", one speaker, 10 training seqs
III. "A" to "Z", ten speakers, two training seqs/letter/person

Show Video Clip






PLAY
SP -0:08:13

paper

http://www.cs.ucf.edu/~vision/papers/shah/97/NDS97.pdf

Change Detection, Skin Detection, Color Tracking

Lecture-14

Mixture of Gaussians

Grimson

Algorithm

- Learn background model by watching 30 second video
- Detect moving object by measuring deviations from background model, and applying connected component to foreground pixels.
- Predict position of a region in the next frame using Kalman filter
- Update background and blob statistics

Summary

- Each pixel is an independent statistical process, which may be combination of several processes.
  - Swaying branches of tree result in a bimodal behavior of pixel intensity.
- The intensity is fit with a mixture of K Gaussians.

$$\Pr(X_t) = \sum_{j=1}^{K} \frac{\omega_j}{(2\pi)^{\frac{m}{2}} |\Sigma_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_j)^T \Sigma_j^{-1} (X_t-\mu_j)}$$

Summary

- Where $X=[R,G,B]^T$, μ is a 3x 1 mean vector,Σ is a 3x3 covariance matrix.
- The method assumes that RGB color channels are independent and have the same variance σ . In this case $\Sigma= \sigma I$. Where I is a 3x3 unit matrix.

Learning Background Model

- Every new pixel is checked against all existing distributions. The match is the first distribution such that the pixel value lies within 2 standard deviations of mean.
- Another way of measuring distance from a Gaussian distribution is the Mahalanobis distance i.e the match is the distribution with M-distance less than a threshold

$$d = (X_t - \mu_j)^T \Sigma_j^{-1} (X_t - \mu_j)$$

Updating

- The mean and s.d. of unmatched distributions remain unchanged. For the matched distributions they are updated as:

$$\mu_{j,t} = (1-\rho)\mu_{j,t-1} + \rho X_t$$

$$\sigma_{j,t} = (1-\rho)\sigma_{j,t-1}^2 + \rho (X_t - \mu_{j,t})^T (X_t - \mu_{j,t})$$

- The weights are adjusted:

$$\omega_{j,t} = (1-\alpha)\omega_{j,t-1} + \alpha(M_{j,t}) \qquad M_{j,t} = \begin{cases} 1 & \text{if distribution matches} \\ 0 & \text{otherwise} \end{cases}$$

Segmenting Background

- The K distributions are stored in descending order of the term $\frac{\omega_j}{\sigma_j}$

- Out of "k" distributions, the first B are selected

$$B = \arg\min_b \left[\frac{\sum_{j=1}^{b} \omega_j}{\sum_{j=1}^{K} \omega_j} > T \right]$$

Segmenting Background

- Foreground= Pixels matched with distributions not in the fist B distributions + unmatched pixels
- The lowest weight distribution at the unmatched pixel location is replaced with a new distribution with mean = unmatched pixel color and with some initial covariance.

Segmenting Background

- Results

Kanade

Summary

- Very similar to k-Gaussian with following differences:
  - uses only single Gaussian
  - uses gray level images, the mean and variance are scalar values

Algorithm

- Learn background model by watching 30 second video
- Detect moving object by measuring deviations from background model, and applying connected component to foreground pixels.
- Update background and region statistics

Detection

- During detection if intensity value is more than two sigma away from the background it is considered foreground:
  - keep original mean and variance
  - track the object with new mean and variance
  - if new mean and variance persists for sometime, then substitute the new mean and variance as the background model
  - If object is no longer visible, it is incorporated as part of background

W4 (Who, When, Where, What)

Davis

W4

- Compute "minimum"(M(x)), "maximum" (N(x)), and "largest absolute difference" (L(x)).

$$D_i(x,y) = \begin{cases} 1 \quad if \quad |M(x,y) - f_i(x,y)| > L(x,y) or \\ \quad |N(x,y) - f_i(x,y)| > L(x,y) \\ 0 \quad \ldots \quad otherwise \end{cases}$$

- Theoretically, the performance of this tracker should be worse than others.
- Even if one value is far away from the mean, then that value will result in an abnormally high value of L.
- Having short training time is better for this tracker.

Limitations

- Multiple people
- Occlusion
- Shadows
- Slow moving people
- Multiple processes (swaying of trees..)
- Quick Illumination Changes

Webpage

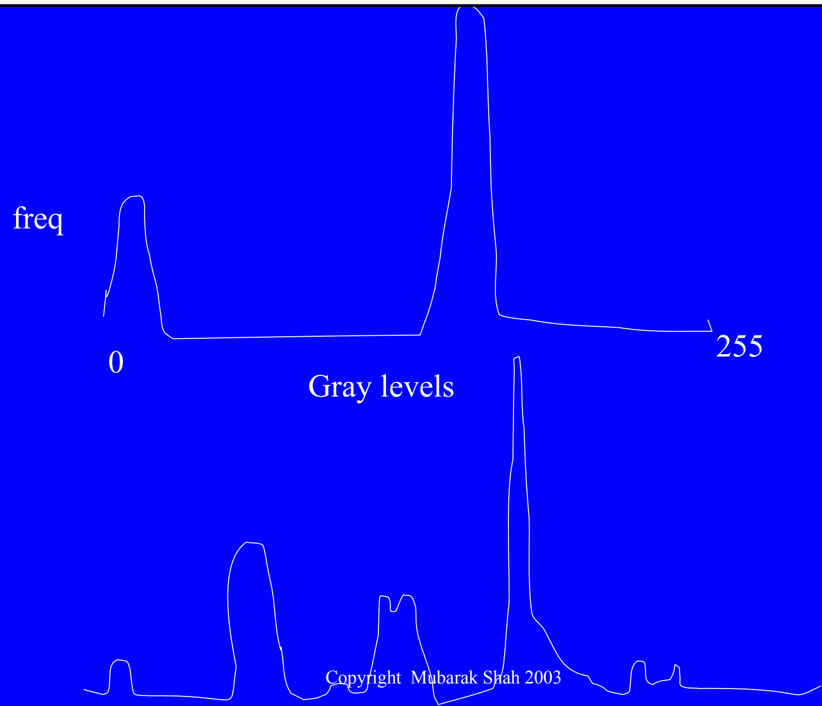
- Http://www.cs.cmu.edu/~vsam

Skin Detection

Kjeldsen and Kender

Training

- Crop skin regions in the training images.
- Build histogram of training images.
- Ideally this histogram should be bi-modal, one peak corresponding to the skin pixels, other to the non-skin pixels.
- Practically there may be several peaks corresponding to skin, and non-skin pixels.



Training

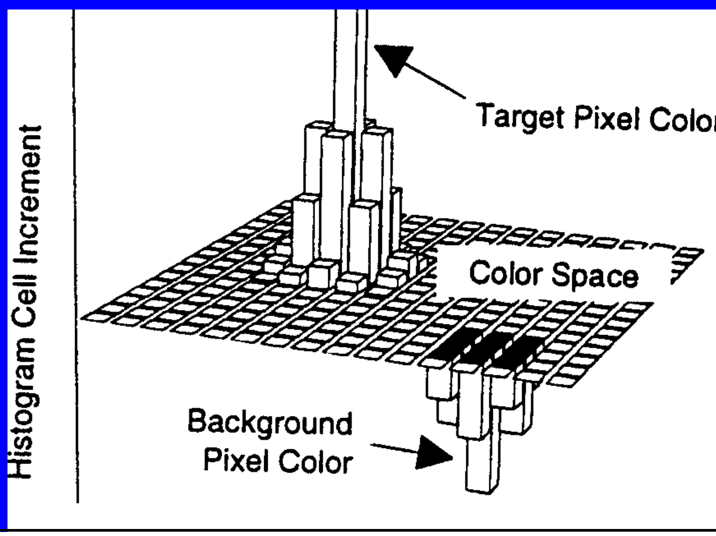
- Apply threshold to skin peaks to remove small peaks.
- Label all gray levels (colors) under skin peaks as “skin”, and the remaining gray levels as “non-skin”.
- Generate a look-up table for all possible colors in the image, and assign “skin” or “non-skin” label.

Detection

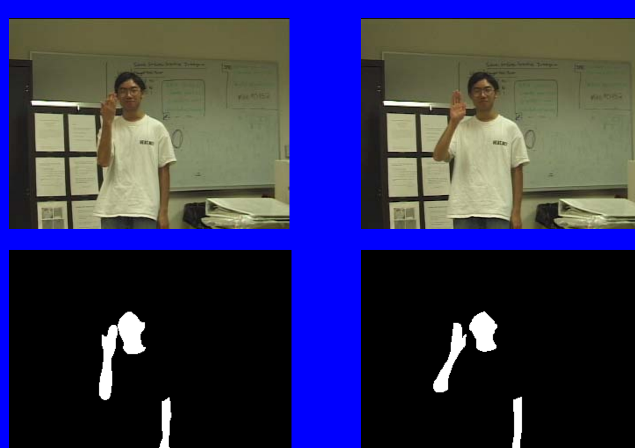
- For each pixel in the image, determine its label from the "look-up table" generated during training.

Building Histogram

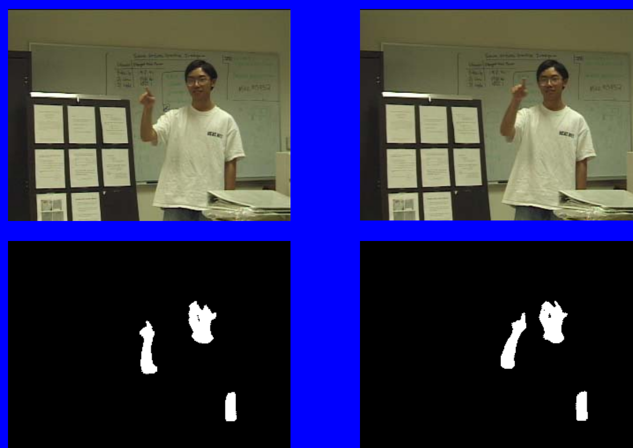
- Instead of incrementing the pixel counts in a particular histogram bin:
  - for skin pixel increment the bins centered around the given value by a Gaussian function.
  - For non-skin pixels decrement the bins centered around the given value by a smaller Gaussian function.



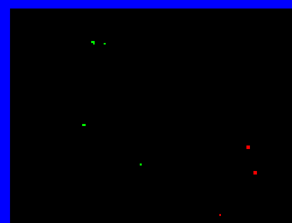
Example training images



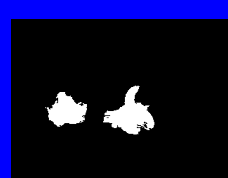
Results of skin detection



Results of skin detection

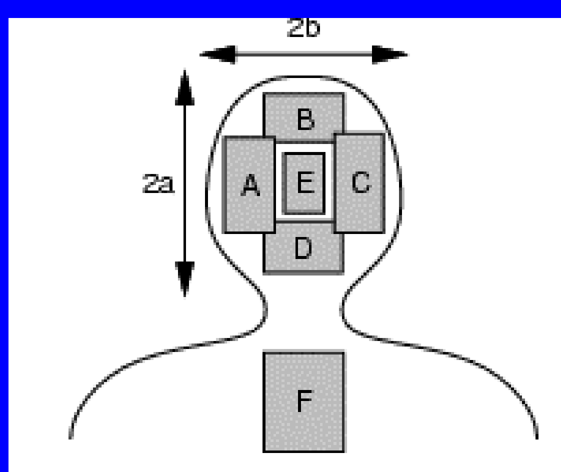


Detecting Fire



Tracking People Using Color

Fieguth and Terzopoulos



Fieguth and Terzopoulos

- Compute mean color vector for each sub region R_i.

$$(r_i, g_i, b_i) = \frac{1}{|R_i|} \sum_{(x,y)\in R_i} (r(x,y), g(x,y), b(x,y))$$

Fieguth and Terzopoulos

- Compute goodness of fit.

$$\Psi_i = \frac{\max\left\{\frac{r_i}{\bar{r}_i}, \frac{g_i}{\bar{g}_i}, \frac{b_i}{\bar{b}_i}\right\}}{\min\left\{\frac{r_i}{\bar{r}_i}, \frac{g_i}{\bar{g}_i}, \frac{b_i}{\bar{b}_i}\right\}}$$

$(\bar{r}_i, \bar{g}_i, \bar{b}_i)$

Target

(r_i, g_i, b_i)

Measurement

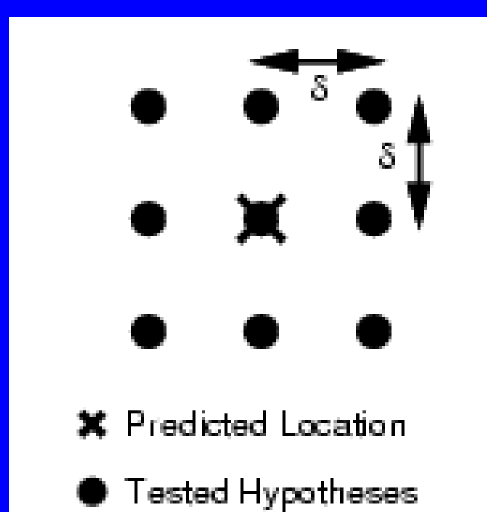
Fieguth and Terzopoulos

- Tracking

$$\Psi(x_H, y_H) = \sum_{i=1}^{N} \frac{\Psi_i(x_H + x_i, y_H + y_i)}{N}$$

$$(\hat{x}, \hat{y}) = \arg_{(x_H, y_H)} \min\{\Psi(x_H, y_H)\}$$

Fieguth and Terzopoulos



Fieguth and Terzopoulos

- Non-linear velocity estimator

$$v(f) = v(f-1)$$

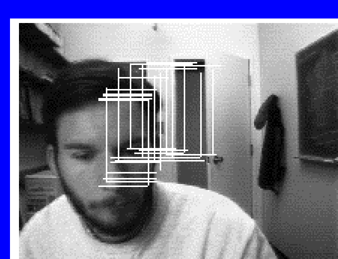
$$if \quad (\rho(f).\rho(f-1) > 0) \quad v(f) \quad += \quad \delta \frac{\mathrm{sgn}(\rho(f))}{\Delta t}$$

$$if \quad (\rho(f).v(f-1) < 0) \quad v(f) \quad += \quad \delta \frac{\mathrm{sgn}(\rho(f))}{\Delta t}$$

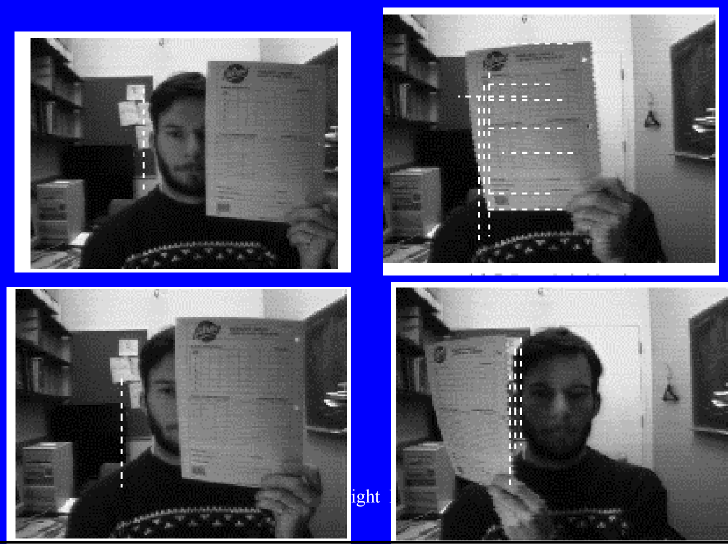
$$if \quad (\rho(f) = 0) \quad v(f) \quad -= \quad \delta \frac{\mathrm{sgn}(v(f))}{2\Delta t}$$

ρ is an error in prediction.

Fieguth and Terzopoulos



Fieguth and Terzopoulos



Bibliography

- .J. K. Aggarwal and Q. Cai, “Human Motion Analysis: A Review”, *Computer Vision and Image Understanding*, Vol. 73, No. 3, March, pp. 428-440, 1999
- .Azarbayejani, C. Wren and A. Pentland, “Real-Time 3D Tracking of the Human Body”, MIT Media Laboratory, Perceptual Computing Section, TR No. 374, May 1996
- .W.E.L. Grimson *et. al.,* “Using Adaptive Tracking to Classify and Monitor Activities in a Site”, *Proceedings of Computer Vision and Pattern Recognition,* Santa Barbara, June 23-25, 1998, pp. 22-29

Bibliography

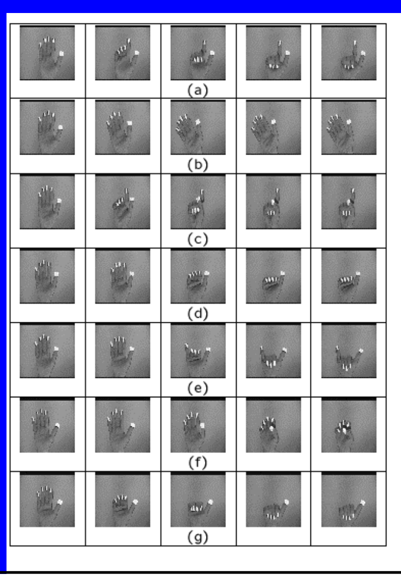
- .Takeo Kanade *et. al.* "Advances in Cooperative Multi-Sensor Video Surveillance", *Proceedings of Image Understanding workshop*, Monterey California, Nov 20-23, 1998, pp. 3-24
- .Haritaoglu I., Harwood D, Davis L, "W^4 - Who, Where, When, What: A Real Time System for Detecting and Tracking People", *International Face and Gesture Recognition Conference,* 1998
- .Paul Fieguth, Demetri Terzopoulos, "Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates", *CVPR 1997,* pp. 21-27

Hand Gesture Recognition, Aerobic exercises, Events

Lecture-15

Hand Gesture Recognition

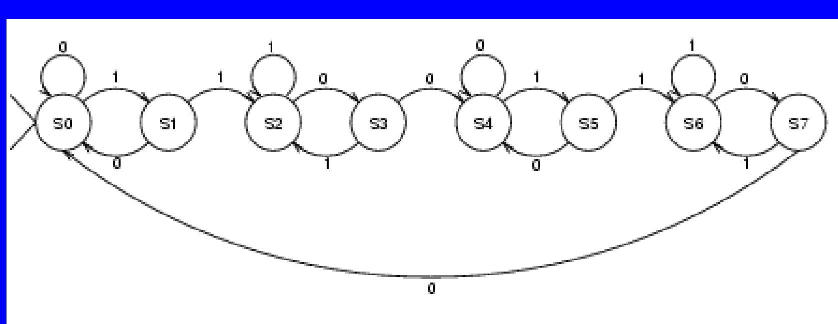
Seven Gestures



Gesture Phases

- Hand fixed in the start position.
- Fingers or hand move smoothly to gesture position.
- Hand fixed in gesture position.
- Fingers or hand return smoothly to start position.

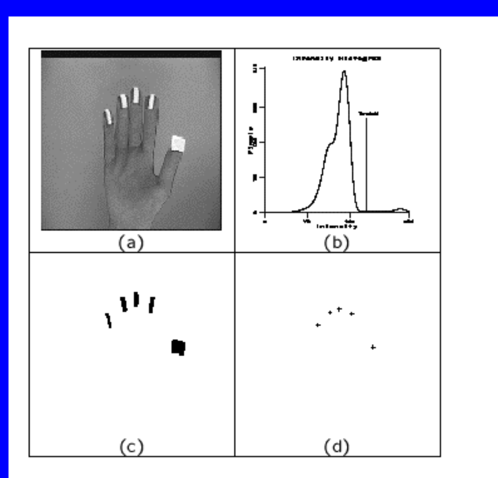
Finite State Machine



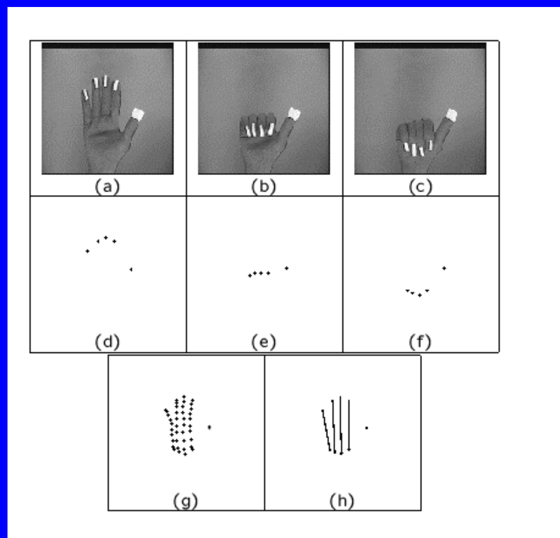
Main Steps

- Detect fingertips.
- Create fingertip trajectories using motion correspondence of fingertip points.
- Fit vectors and assign motion code to unknown gesture.
- Match

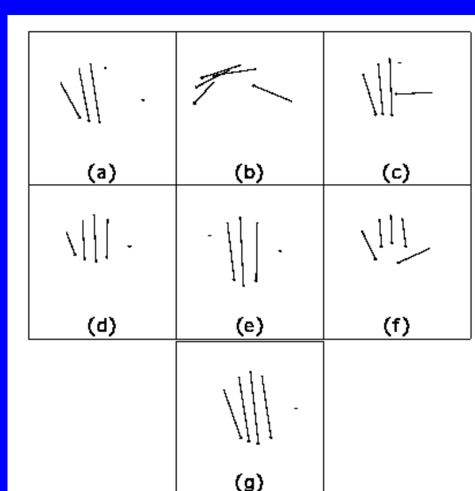
Detecting Fingertips



Vector Extraction



Vector Representation of Gestures



Results

Results

Run	Frames	L	R	U	D	T	G	S
1	200	√	√	√	√	√	√	√
2	250	√	√	√	√	√	√	√
3	250	√	√	√	X	√	√	√
4	250	√	√	√	√	√	√	√
5	300	√	√	√	√	√	√	√
6	300	√	√	√	√	√	√	√
7	300	√	√	√	√	√	√	√
8	300	√	√	√	√	√	√	√
9	300	√	√	√	√	*	*	*
10	300	√	√	√	√	√	√	√

L = Left, R = Right, U = Up, D = Down, T = Rotate, G = Grab, S = Stop, √ - Recognized, X - Not Recognized, * - Error in Sequence.

Action Recognition Using Temporal Templates

Jim Davis and Aaron Bobick

Main Points

- Compute a sequence of difference pictures from a sequence of images.
- Compute Motion Energy Images (MEI) and Motion History Images (MHI) from difference pictures.
- Compute Hu moments of MEI and MHI.
- Perform recognition using Hu moments.

MEI and MHI

Motion-Energy Images (MEI)

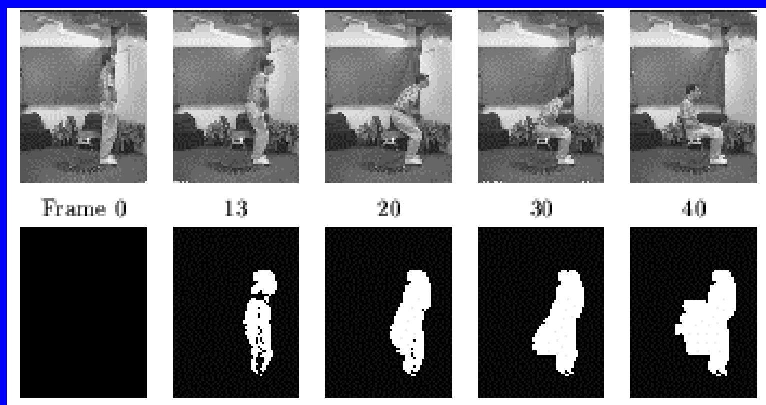
$$E_\tau(x,y,t) = \bigcup_{i=0}^{\tau-1} D(x,y,t-i)$$

Change Detected Images

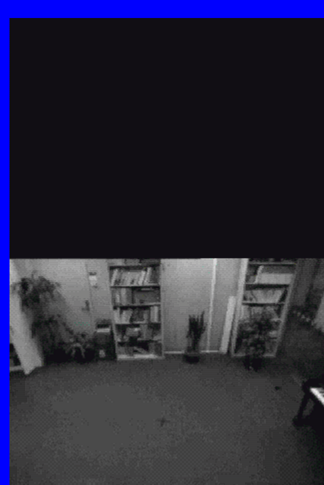
Motion History Images (MHI)

$$H_\tau(x,y,t) = \begin{cases} \tau & \text{if } D(x,y,t)=1 \\ \max(0, H_\tau(x,y,t-1)-1) & \text{otherwise} \end{cases}$$

MEIs



Color MHI Demo



Summary

- Use seven Hu moments of MHI and MEI to recognize different exercises.
- Use seven views (-90 degrees to +90 degrees in increments of 30 degrees).
- For each exercise several samples are recorded using all seven views, and the mean and covariance matrices for the seven moments are computed as a model.
- During recognition, for an unknown exercise all seven moments are computed, and compared with all 18 exercises using Mahalanobis distance.
- The exercise with minimum distance is computed as the match.
- They present recognition results with one and two view sequences, as compared to seven view sequences used for model generation.

Moments

Binary image

General Moments

$$m_{pq} = \int \int x^p y^q \rho(x, y) dxdy$$

Central Moments (Translation Invariant)

$$\mu_{pq} = \int\int (x-\bar{x})^p (y-\bar{y})^q \rho(x,y) \; d(x-\bar{x}) d(y-\bar{y})$$

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}$$

centroid

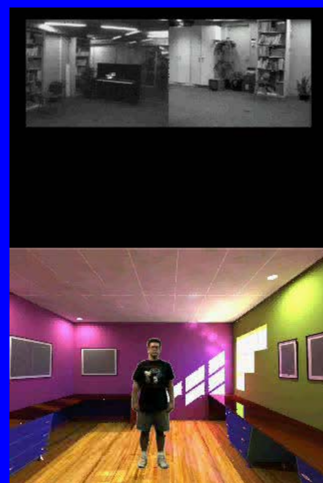
Central Moments

$$\mu_{00} = m_{00} \equiv \mu$$
$$\mu_{01} = 0$$
$$\mu_{10} = 0$$
$$\mu_{20} = m_{20} - \mu\bar{x}^2$$
$$\mu_{11} = m_{11} - \mu\bar{x}\bar{y}$$
$$\mu_{02} = m_{02} - \mu\bar{y}^2$$
$$\mu_{30} = m_{30} - 3m_{20}\bar{x} + 2\mu\bar{x}^3$$
$$\mu_{21} = m_{21} - m_{20}\bar{y} - 2m_{11}\bar{x} + 2\mu\bar{x}^2 y$$
$$\mu_{12} = m_{12} - m_{02}\bar{x} - 2m_{11}\bar{y} + 2\mu\bar{x}y^2$$
$$\mu_{03} = m_{03} - 3m_{02}\bar{y} + 2\mu\bar{y}^3$$

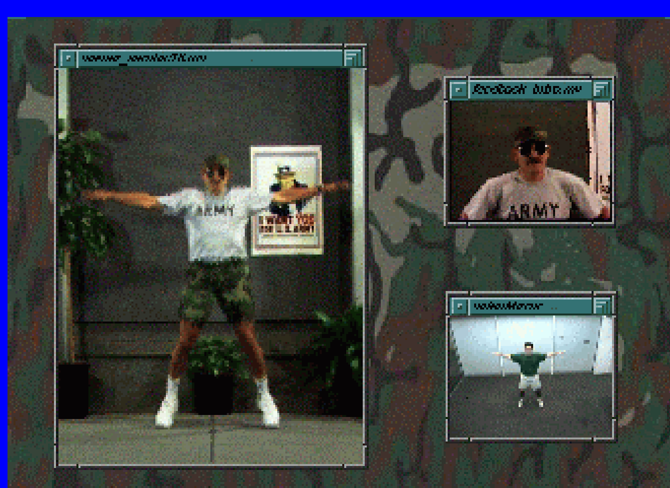
Moments

Hu Moments: translation, scaling and rotation invariant

$$\upsilon_1 = \mu_{20} + \mu_{02}$$
$$\upsilon 2 = (\mu_{20} - \mu_{02})^2 + \mu_{11}{}^2$$
$$\upsilon_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{12} - \mu_{03})^2$$
$$\upsilon_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$
$$\vdots$$



PAT (Personal Aerobic Trainer)



PAT (Personal Aerobic Trainer)

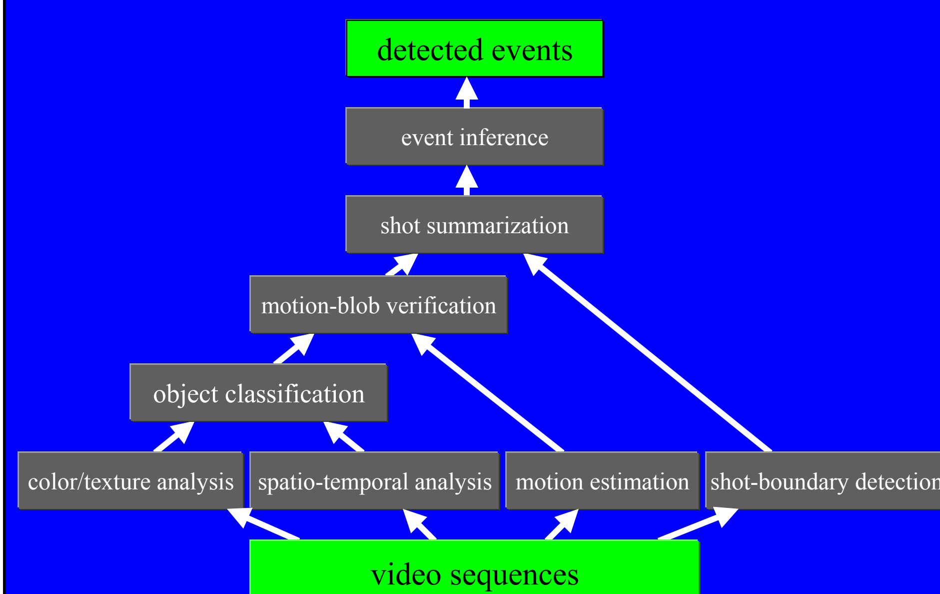
http://vismod.www.media.mit.edu/vismod/demos/actions/mhi_generation.mov

PAT (Personal Aerobic Trainer)

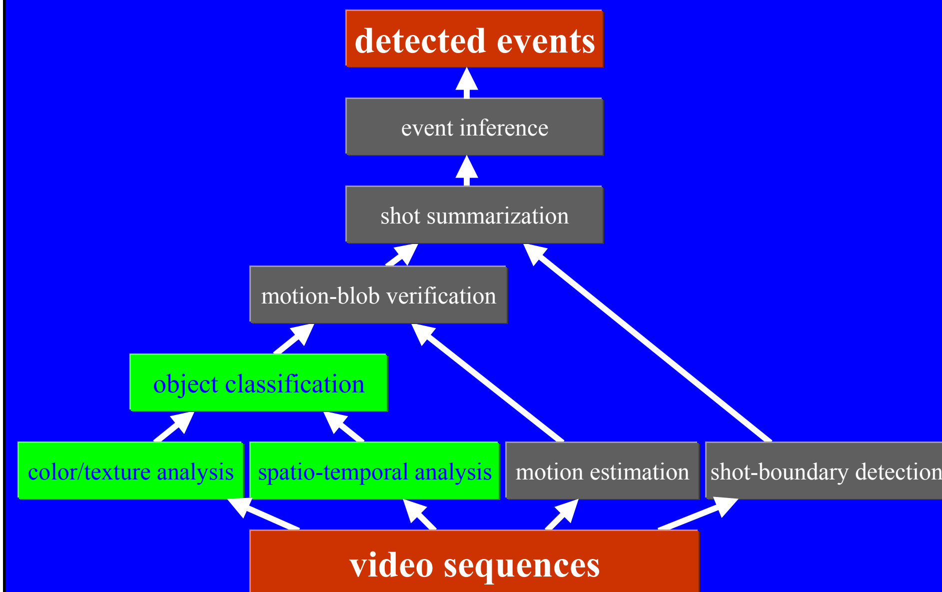
A Framework for the Design of Visual Event Detectors

Niels Haering

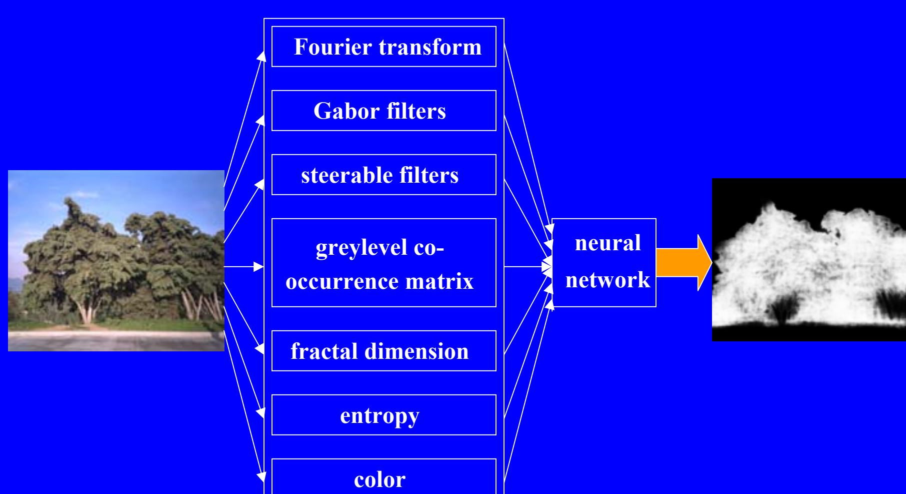
Our Framework



Object Classification



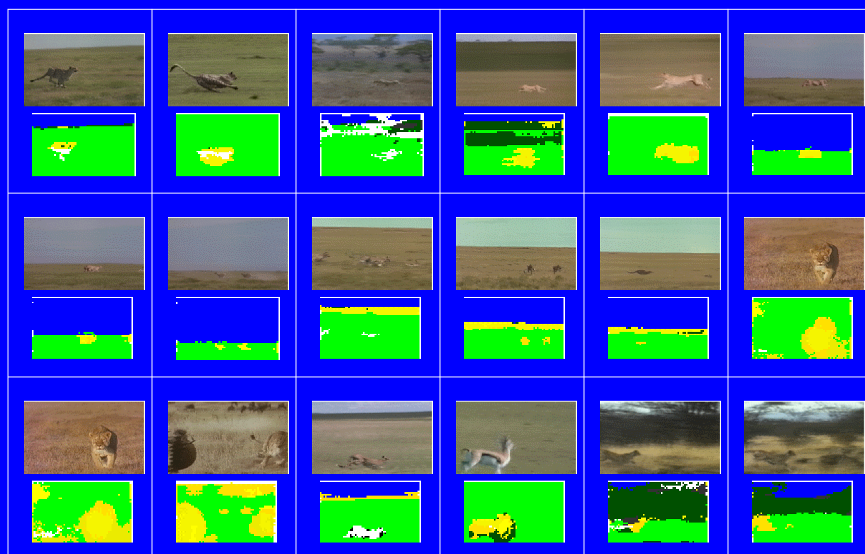
Object Classification



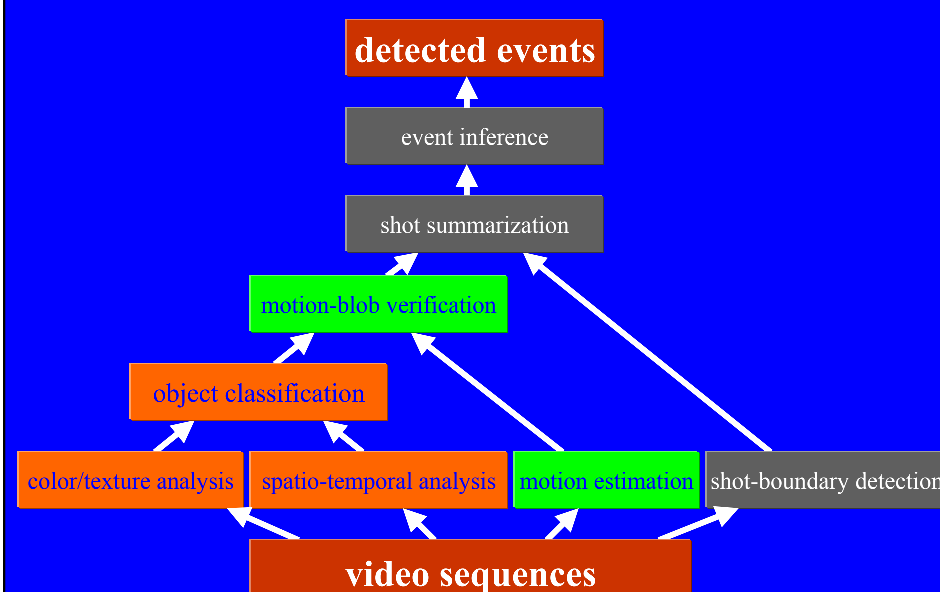
Deciduous Trees

Sky

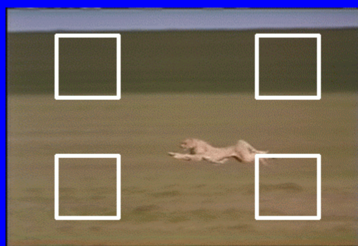
Animals, Sky, Grass, Trees, Rock

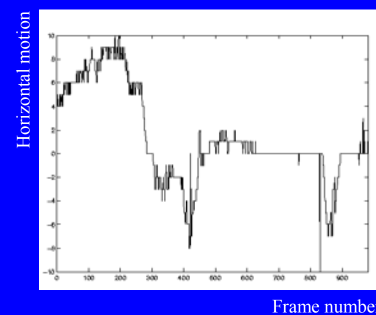


Motion-blob Verification



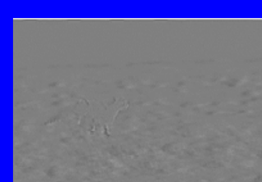
Motion Estimation





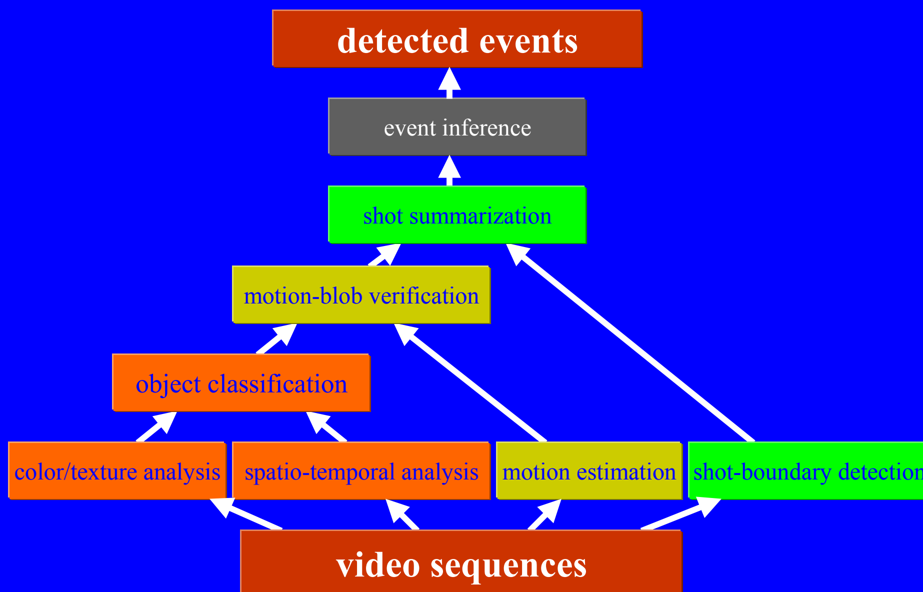
- three parameter system: x-, y-translation, and zoom,
- 4 motion estimates based on pyramid,
- 4 motion estimates based on previous best match,
- “texture” measure prevents ambiguous matches

Motion-blob detection



Motion estimate
Δx = -7
Δy = 0
zoom = 1.0

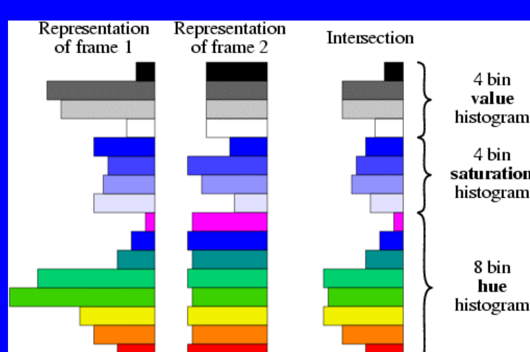
Shot Summarization



Shot Detection

Characteristics of shot boundaries:

- Change of camera/viewpoint
- Change of color characteristics



4 Bins for Value
4 Bins for Saturation
8 bins for hue

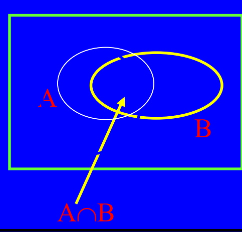
= 0.79

$$\cap = \frac{\sum_{n=0}^{15} \min(f_1(n), f_2(n))}{\min(\sum_{n=0}^{15}(f_1(n), \sum_{n=0}^{15} f_2(n))}$$

Shot Summaries

General Information	
Forced/Real Shot Summary	0
First Frame of Shot	64
Last Frame of Shot	263
Global motion estimate (x,y)	(-4.48, 0.01)
Within Frame animal motion estimate (x,y)	(-0.17, 0.23)
Initial Position (x,y)	(175, 157)
Final Position (x,y)	(147, 176)
Initial Size (x,y)	(92, 67)
Final Size (x,y)	(100, 67)
Motion smoothness throughout Shot (x,y)	(0.83, 0.75)
Precision throughout Shot	0.84
Recall throughout Shot	0.16

Hunt Information	
Tracking	1
Fast	1
Animal	1
Beginning of Hunt	1
Number of Hunt Shots	1
End of Hunt	0
Valid Hunt	0



A = Ground Truth

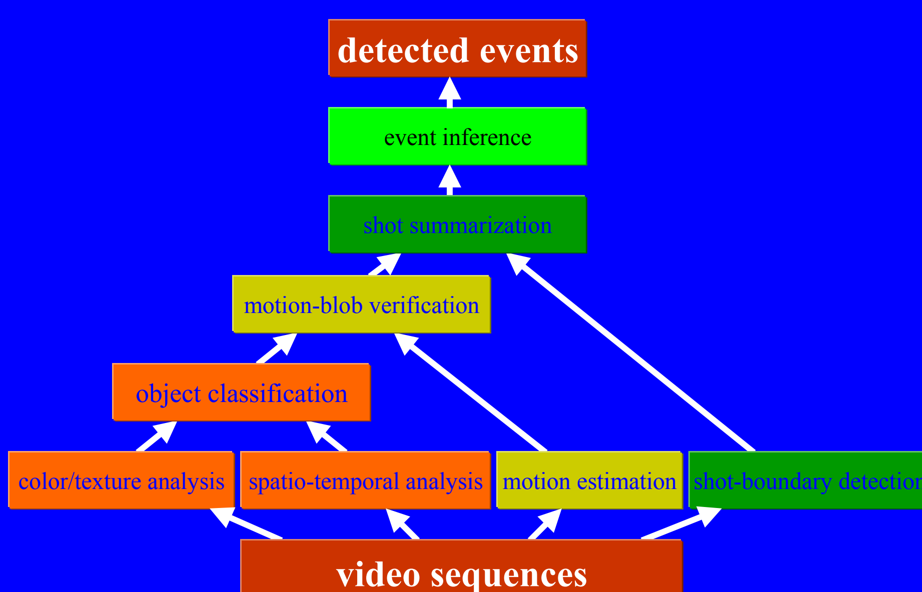
B = Result of Algorithm

A∩B = Correct detection

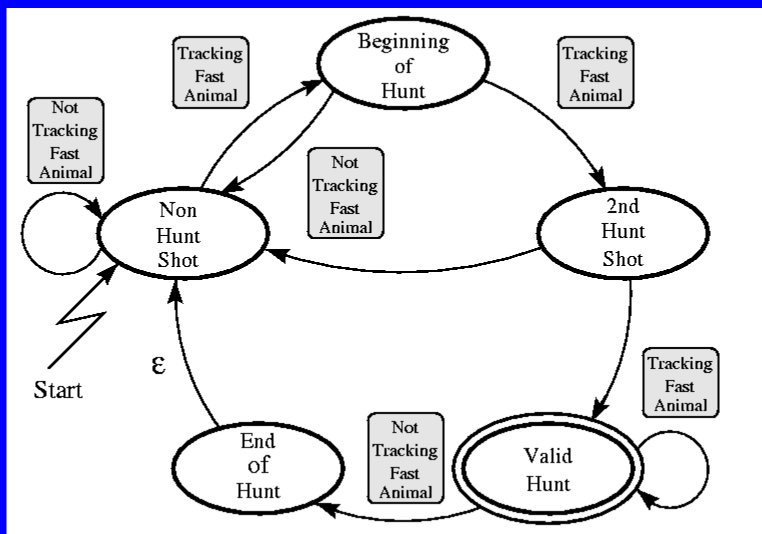
$$recall = \frac{A \cap B}{A}$$

$$precision = \frac{A \cap B}{B}$$

Event Inference



Hunt events



Hunts

Hunt

Non-hunt

Hunts

Non-hunt

Hunt

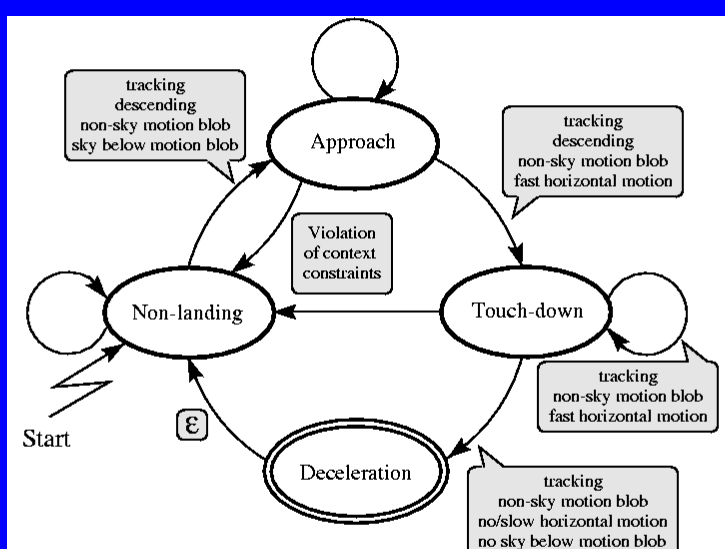
Non-hunt

Event Detection

Sequence Name	Actual Hunt Frames	Detected Hunt Frames	Precision	Recall
Hunt1	305 – 1375	305 – 1375	100%	100%
Hunt2	2472 – 2696	2472 – 2695	100%	99.6%
Hunt3	3178 – 3893	3178 – 3856	100%	94.8%
Hunt4	6363 – 7106	6363 – 7082	100%	96.8%
Hunt5	9694 – 10303	9694 – 10302	100%	99.8%
Hunt6	12763 – 14178	12463 – 13389	67.7%	44.2%
Hunt7	16581 – 17293	16816 – 17298	99.0%	67.0%
average			95.3%	86.0%

Landing Events

Landing Events




Landing Events
Non-landing
Approach
Touch-down
Deceleration
Non-landing


Landing Events
Non-landing
Approach
Touch-down
Deceleration
Non-landing

Landing Events

Non-landing

Approach

Touch-down

Deceleration

Non-landing

Conclusions

- Many natural objects are easily recognized by their color and texture signatures (shape is often not needed)
- Many events are easily detected and recognized by the classes of the comprising objects and their approximate motions
- The proposed visual event detection is robust to changes in scale, color, shape, occlusion, lighting conditions, view points and distances, and image compression

Monitoring Human Behavior

Lecture-16

Monitoring Human Behavior

http://www.cs.ucf.edu/~vision/projects/Office/Office.html

Goals of the System

- Recognize human actions in a room for which **prior knowledge** is available.
- Handle multiple people
- Provide a textual description of each action
- Extract "key frames" for each action

Possible Actions

- **Enter**
- **Leave**
- **Sitting** or **Standing**
- **Picking Up Object**
- **Put Down Object**
- **…..**

Prior Knowledge

- Spatial layout of the scene:
  - Location of **entrances** and **exits**
  - Location of **objects** and some information about how they are use
- Context can then be used to improve recognition and save computation

Layout of Scene 1

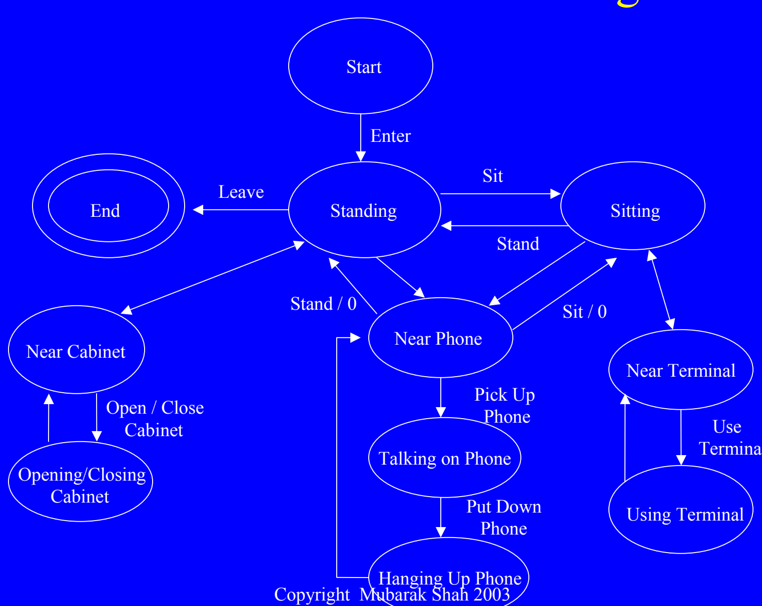
Layout of Scene 2

Layout of Scene 4

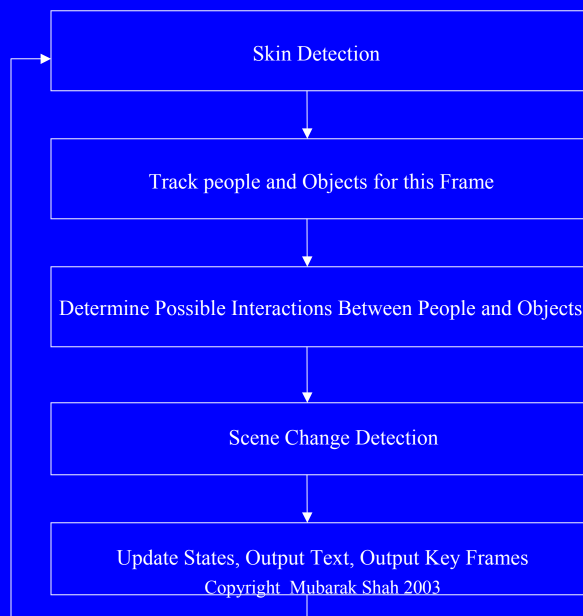
Major Components

- Skin Detection
- Tracking
- Scene Change Detection
- Action Recognition

State Model For Action Recognition



Flow of the System



Key Frames

- Why get key frames?
  - Key frames take less space to store
  - Key frames take less time to transmit
  - Key frames can be viewed more quickly
- We use heuristics to determine when key frames are taken
  - Some are taken before the action occurs
  - Some are taken after the action occurs

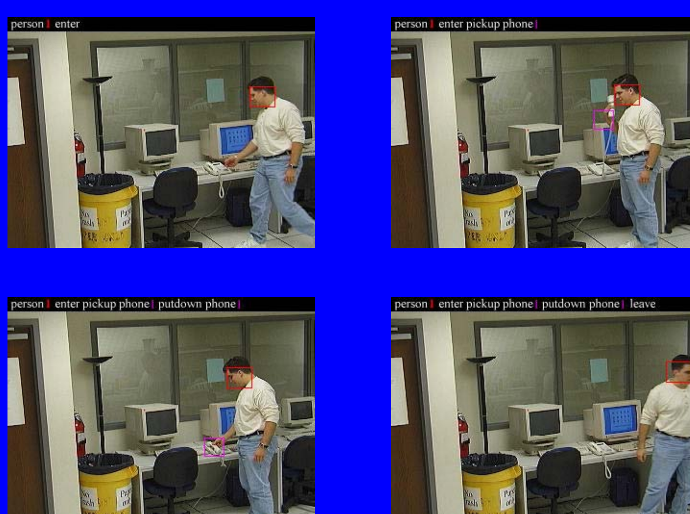
Key Frames

- <u>"Enter" key frames</u>: as the person leaves the entrance/exit area
- <u>"Leave" key frames</u>: as the person enters the entrance/exit area
- <u>"Standing/Sitting" key frames</u>: after the tracking box has stopped moving up or down respectively
- <u>"Open/Close" key frames</u>: when the % of changed pixels stabilizes

Key Frames Sequence 1 (350 frames), Part 1

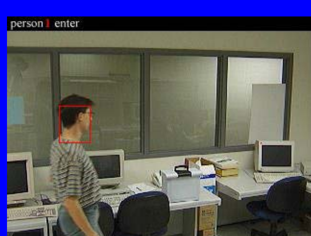
Key Frames Sequence 1 (350 frames), Part 2

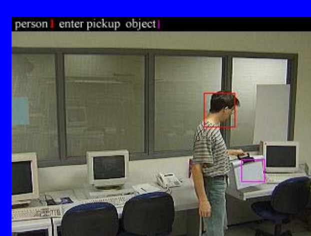
Key Frames Sequence 2 (200 frames)



Copyright Mubarak Shah 2003

Key Frames Sequence 3 (200 frames)





Key Frames Sequence 4 (399 frames), Part 1

Key Frames Sequence 4 (399 frames), Part 2

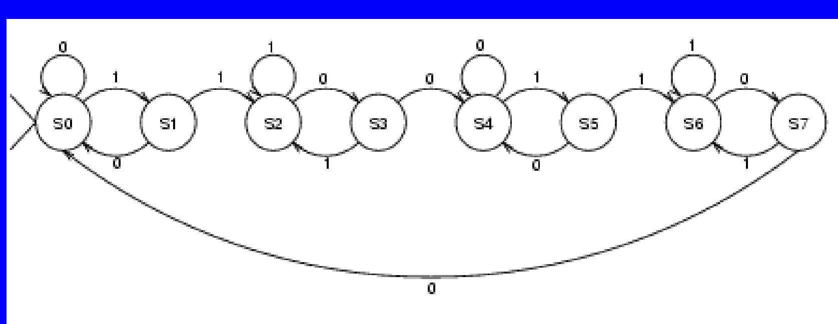
Action Recognition

Approaches

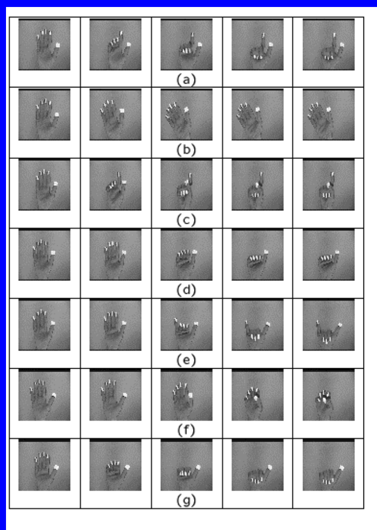
- FSA
- HMMs/NNs
- Rule-based
- ---

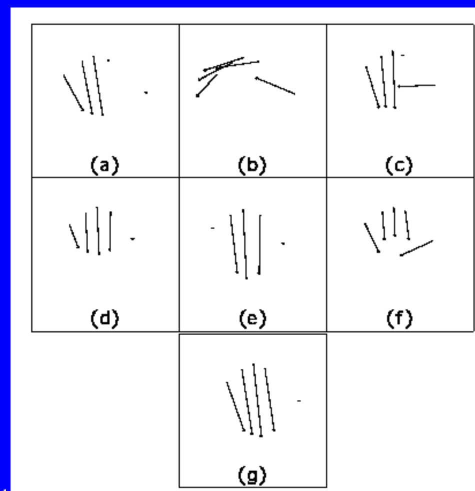
- Representation is important

FSA: Hand Gesture Recognition

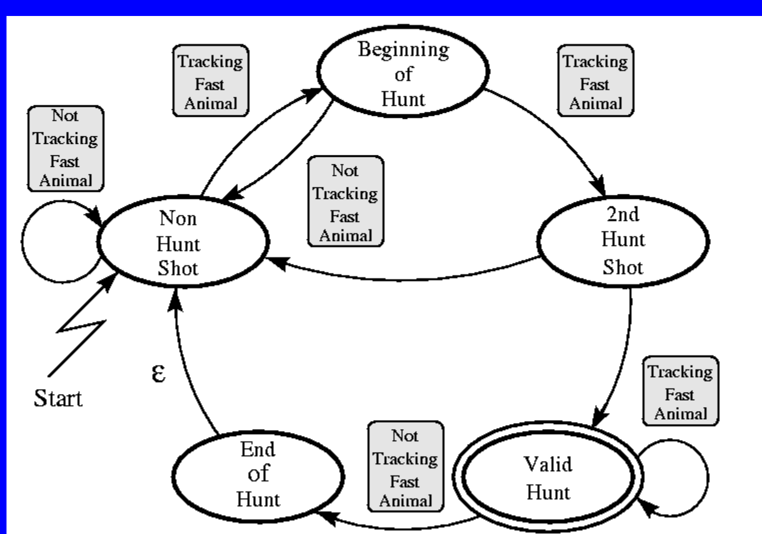


FSA: Hand Gesture Recognition

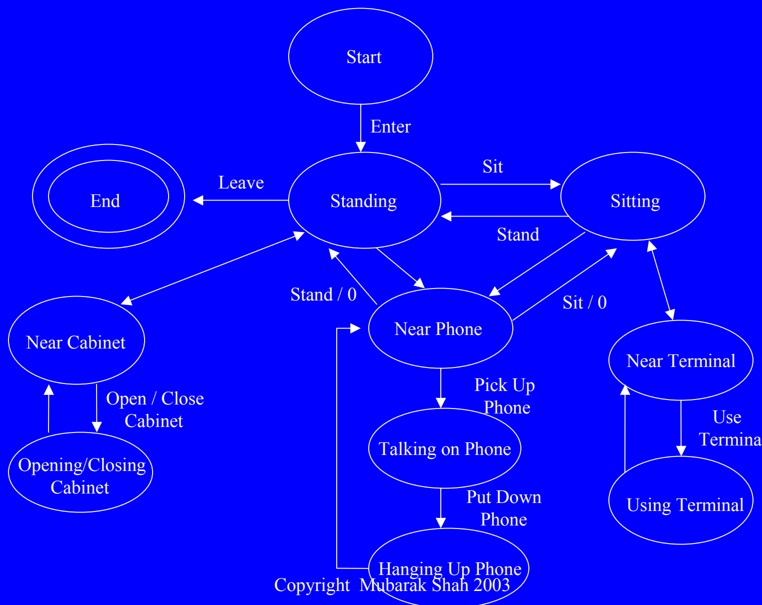




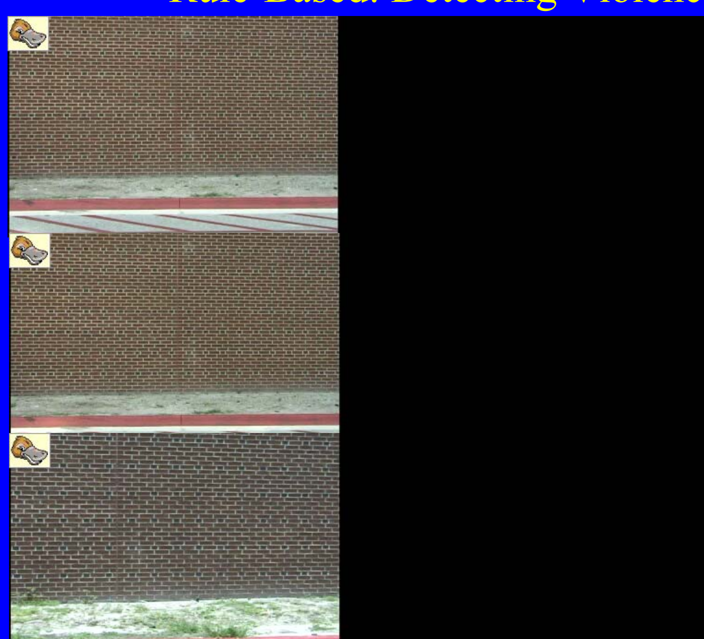
Hunt events



FSA: Recognizing Human Behavior in Office Environment



Rule-Based: Detecting Violence



Rule-Based: Detecting Violence

Rule-Based: Recognizing Outdoor Activities

Limitations

- A priori knowledge
- Extensive training
- No explanation
- No learning
- Representation
- View invariance

View-invariant Representation and Recognition of Human Action

Hand Actions Recognition

- hand generates a 3-D trajectory with respect to time.
- analyze 2-D projection of this 3-D trajectory.
- View invariance issues.

Generation of Hand Trajectory

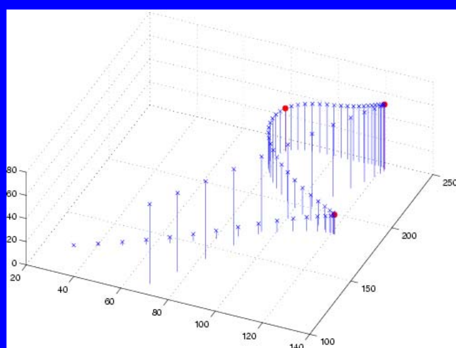
- For each frame:
  - Skin detection + Mean-shift tracker.

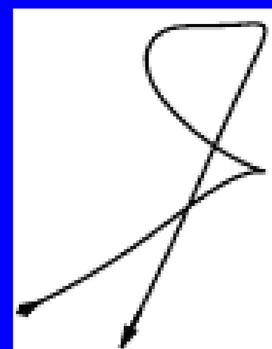
QuickTime™ and a YUV420 codec decompressor are needed to see this picture.

Example of tracking

Spatio-temporal Curve

$$r_{st} = [x(t) \quad y(t) \quad t]$$





Spatio-temporal Curvature

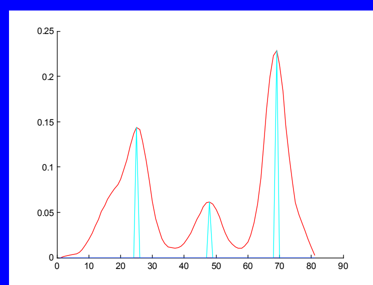
$$k = \frac{\sqrt{A^2 + B^2 + C^2}}{\left((x')^2 + (y')^2 + (t')^2\right)^{3/2}}$$

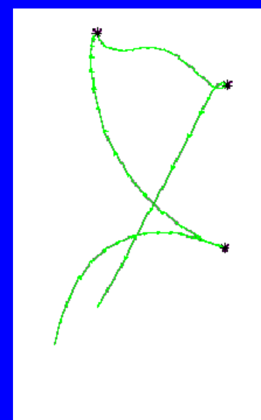
$$A = \begin{vmatrix} y' & t' \\ y'' & t'' \end{vmatrix}, B = \begin{vmatrix} t' & x' \\ t'' & x'' \end{vmatrix}, C = \begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix}$$

Spatiotemporal curvature captures both the **speed** and **direction** changes in **one quantity**.

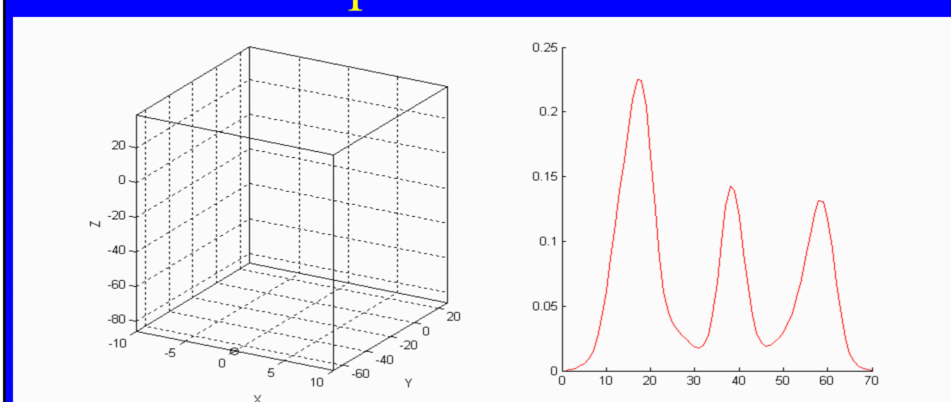
Representation of Actions

- Dynamic Instants:
  - Maximum in spatiotemporal curvature represents an important change of motion characteristic.
- Intervals





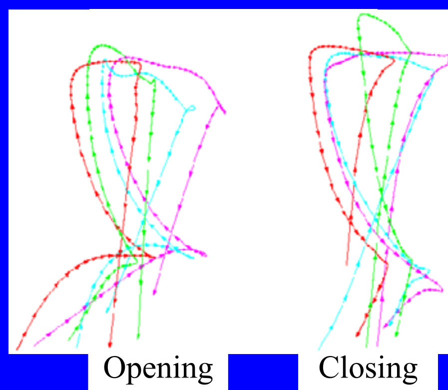
Viewing direction and spatio-temporal curvature



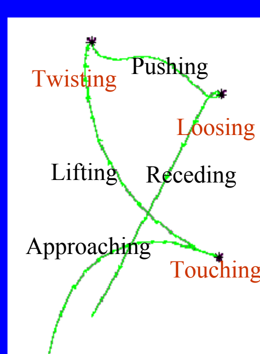
Although the viewing directions are quite different, the peak locations are consistent

View-invariant Representation of Actions (4)

- Opening and closing an overhead cabinet



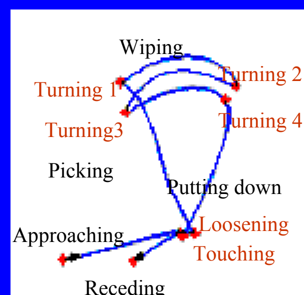
Explanation



Opening

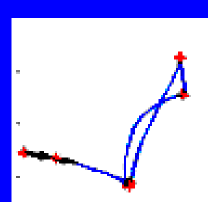
View-invariant Representation of Actions (2)

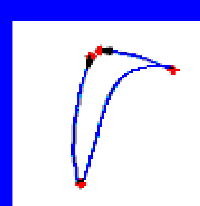
- Erasing white board.



Erasing

View-invariant Representation of Actions (3)





Generalized Spatio-temporal Curvature

- Input: x, y, *orientation* θ and calculate generalized spatio-temporal curvature

$$k = \frac{\sqrt{A^2 + B^2 + C^2 + D^2 + E^2 + F^2}}{\left((x')^2 + (y')^2 + (\theta')^2 + (t')^2\right)^{\frac{3}{2}}}$$

$$A = \begin{vmatrix} y' & t' \\ y'' & t'' \end{vmatrix}, B = \begin{vmatrix} t' & x' \\ t'' & x'' \end{vmatrix}, C = \begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix}, D = \begin{vmatrix} \theta & t' \\ \theta' & t'' \end{vmatrix}, E = \begin{vmatrix} \theta & x' \\ \theta' & x'' \end{vmatrix}, F = \begin{vmatrix} \theta & y' \\ \theta' & y'' \end{vmatrix}$$

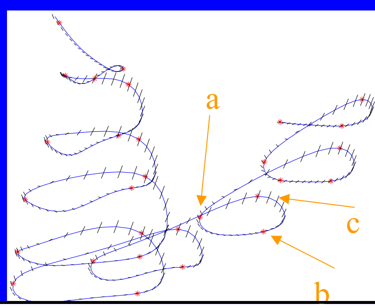
- Detect local maxima
- More characteristics – size, color, …

Foot Trajectory

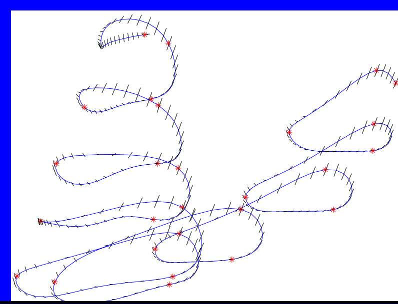
Walking trajectories

- The dynamic events are: a) The foot touches the ground, b) the foot leaves the ground, c) the foot moves forward.

Right foot trajectory

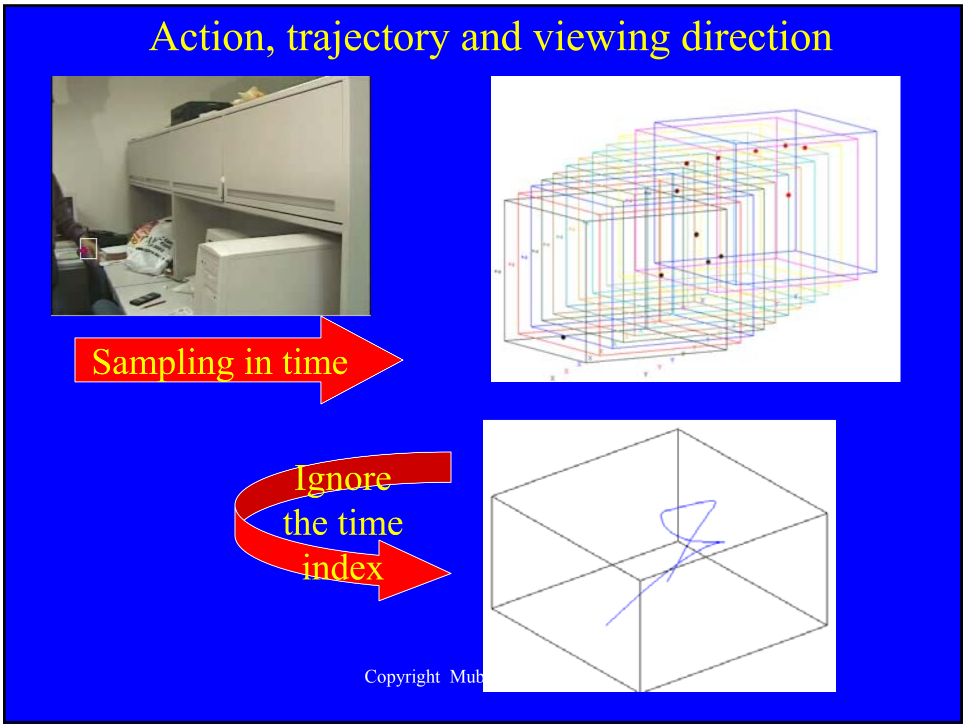


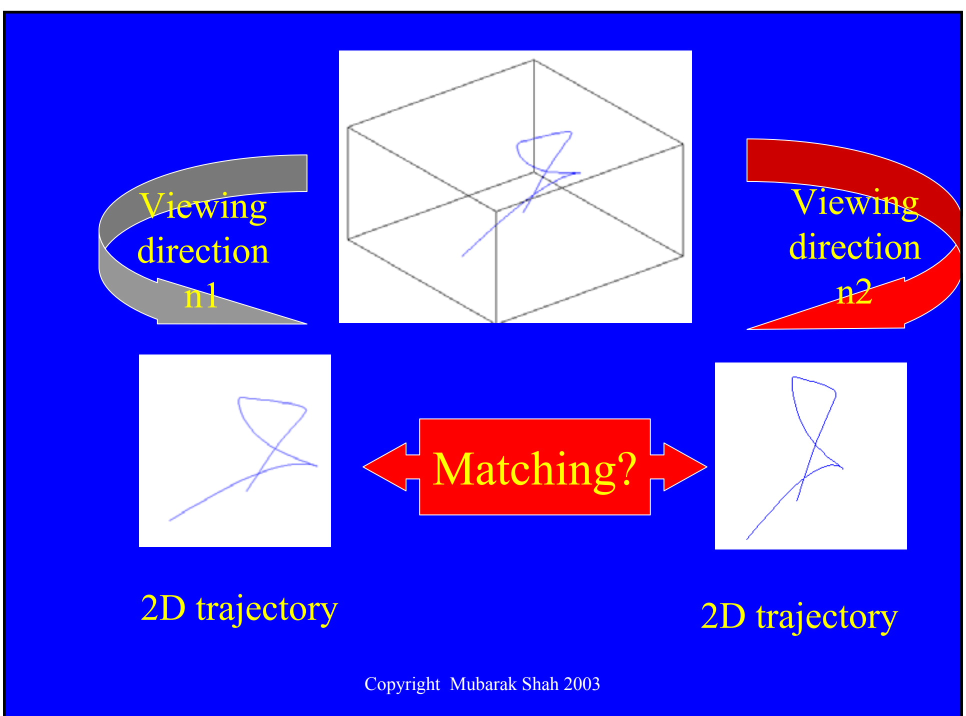
Left foot trajectory



View-invariant Matching

- Consider 3D trajectories as 3D objects.
- View-invariant object recognition.
- Instant and interval information.


Action, trajectory and viewing direction
Sampling in time
Ignore
the time
index
Copyright Mub


Viewing
direction
n1
Viewing
direction
n2
Matching?
2D trajectory
2D trajectory
Copyright Mubarak Shah 2003

Rank Theorem

Without noise, the registered measurement matrix is at most of rank three.

$$M = P \bullet S = \begin{bmatrix} \Pi_{v_1} \\ \vdots \\ \Pi_{v_k} \end{bmatrix} \bullet \begin{bmatrix} X_1\ X_2 & & X_n \\ Y_1\ Y_2 & \cdots & Y_n \\ Z_1\ Z_2 & & Z_n \end{bmatrix}$$

$$M = \begin{bmatrix} I_{v_1} \\ I_{v_2} \\ \vdots \\ I_{v_k} \end{bmatrix}, and \quad I_v = \begin{bmatrix} \mu_1^v & \mu_2^v & \dots & \mu_n^v \\ \nu_1^v & \nu_2^v & \dots & \nu_n^v \end{bmatrix}$$

M is a $2k$ by n matrix

$$\Pi_v = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix}$$

S is 3 by n matrix, projection matrix is $2k$ by 3 matrix, then the rank of M is at most 3.

(Tomasi & Kanade, Shapiro & Zisserman)

Generalized Rank Theorem

- Under affine camera model a set of image tracks belong to a single 3-D object if and only if measurement matrix M is of rank at most 3.

Shapiro-Zisserman, Seitz-Dyer

Action Matching

- A set of action trajectories match if and only if M (which is in term of instants) is of rank at most 3.

$$M = \begin{bmatrix} \mu_1^i & \mu_2^i & \cdots & \mu_n^i \\ \nu_1^i & \nu_2^i & \cdots & \nu_n^i \\ \mu_1^j & \mu_2^j & \cdots & \mu_n^j \\ \nu_1^j & \nu_2^j & \cdots & \nu_n^j \end{bmatrix}$$

Distance measurement

M is the set of image coordinates, and its singular value decomposition is U Σ V. Σ' is the minimum perturbation of *M* that makes the rank of *M* to be 3. Therefore,

$$M = U\Sigma V$$
$$M = U\hat{\Sigma}V + U\Sigma' V$$

$$\hat{\Sigma} = \left[\begin{array}{ccc|cccc} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \sigma_3 & & & & \\ \hline & & & 0 & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{array}\right]$$

$$\Sigma' = \left[\begin{array}{ccc|cccc} 0 & & & & & & \\ & 0 & & & & & \\ & & 0 & & & & \\ \hline & & & \sigma_4 & & & \\ & & & & \sigma_5 & & \\ & & & & & \ddots & \\ & & & & & & \sigma_n \end{array}\right]$$

$$dist = \sqrt{\frac{1}{2kn}\sum_{i=4}^{n}\sigma_i^2}$$

View invariant matching (2)

- Distance between two actions i and j *is*:

$$M = \begin{bmatrix} \mu_1^i & \mu_2^i & \dots & \mu_n^i \\ \nu_1^i & \nu_2^i & \dots & \nu_n^i \\ \mu_1^j & \mu_2^j & \dots & \mu_n^j \\ \nu_1^j & \nu_2^j & \dots & \nu_n^j \end{bmatrix}$$

$$dist_{i,j} = \frac{|\sigma_4|}{2\sqrt{n}}$$

- where σ_4 is the fourth singular values of $\boldsymbol{M}$.
- This distance gives the average amount necessary to additively perturb the coordinates of each instant in order to produce projections of a single action.

Action Learning

- For every action trajectory:
  - Determine its category based on the number of instants and the permutation of signs.
  - Compare this action with all other actions and find 3 best matches whose matching error are under threshold.
- Use transitive property to get the transitive closure for each action.

Experiments

- 1st open the cabinet.
- 2nd pick up an object (umbrella) from the cabinet.
- 3rd put down the object in cabinet, then close the door.
- 4th open the cabinet, with touching the door an extra time.
- 5th pick up an object (disks) with twisting hand around.
- 6th put back the object (disks) and then close the door.
- 7th open the cabinet door, wait, then close the door.
- 8th open the cabinet door, wait, then close the door.
- 9th pick up an object from top the of the cabinet.
- 10th put the object back to the top of cabinet.
- 11th pick up an object from the desk.
- 12th put the object back to the desk.
- 13th pick up an object, then make random motions.
- 14th open the cabinet.
- 15th pick up an object, put it in the cabinet, then close the door.
- 16th open the cabinet.

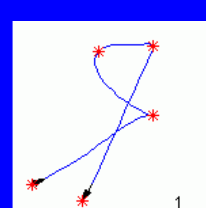
Experiments (2)

- 17th pick up an object (umbralla) from the cabinet.
- 18th put the object (umbralla) back to the cabinet.
- 19th pick up a bag from the desk.
- 20th make random motions.
- 21st open the cabinet.
- 22nd pick up an object (a bag of disks).
- 23rd put donw an object (a bag of disks) back to the cabinet, then close the door.
- 24th pick up an object from the top of the cabinet.
- 25th put the object back to the cabinet top.
- 26th make random motions with two hands.
- 27th continue the action 26.
- 28th close the door, with some random motion.
- 29th open the cabinet.
- 30th pick up an object (remote controller) from the cabinet, put it down on the desk, pick up another object (pencil) from the desk, put it in the cabinet, then close the door.

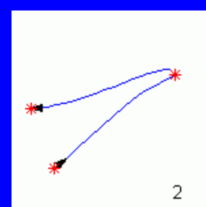
Experiments (3)

- 31st open the cabinet door, with the door half pushed, pick up an object (pencil) from the cabinet.
- 32nd pick up an object (remote controller) from the desk, put it in the cabinet, then close the door.
- 33rd open the cabinet door, wait, then close the door.
- 34th open the cabinet door, make random motions, then close the door.
- 35th pick up some objects.
- 36th open the door, pick up an object, with the door half opened.
- 37th close the half opened door.
- 38th open the cabinet door.
- 39th pick up an object, move it within the cabinet, pick up another object, move it, then close the door.
- 40th open the cabinet door, wait, then close the door.
- 41st pick up an object from the top of the cabinet.
- 42nd close the cabinet.
- 43rd open the cabinet.

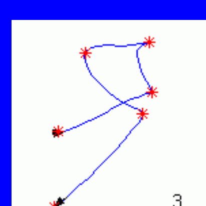
open the cabinet



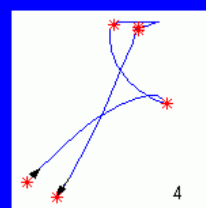
pick up an object (umbrella) from the cabinet



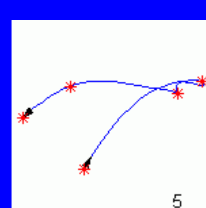
put down the object in cabinet, then close the door



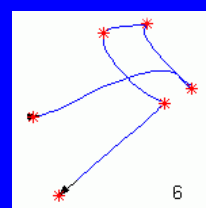
open the cabinet, with touching the door an extra time.



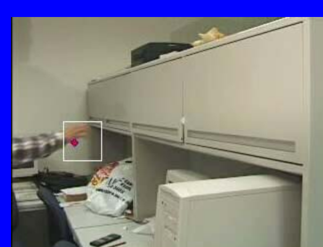
pick up an object (disks) with twisting hand around

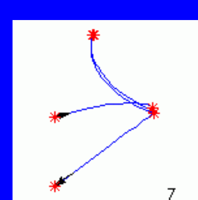


put back the object (disks) and then close the door.

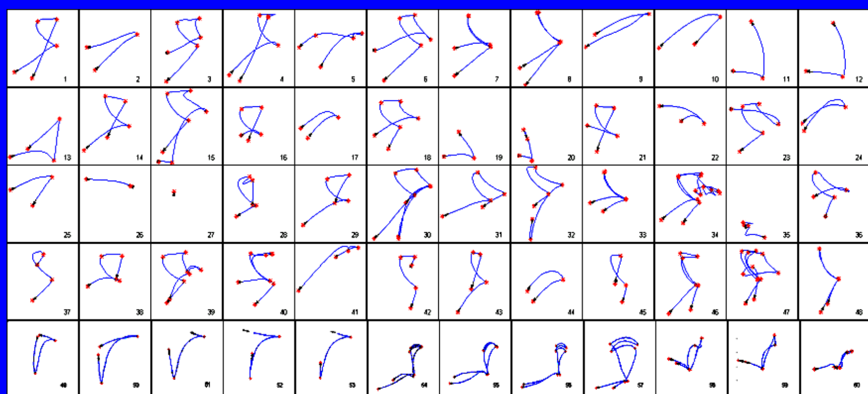


open the cabinet door, wait, then close the door.



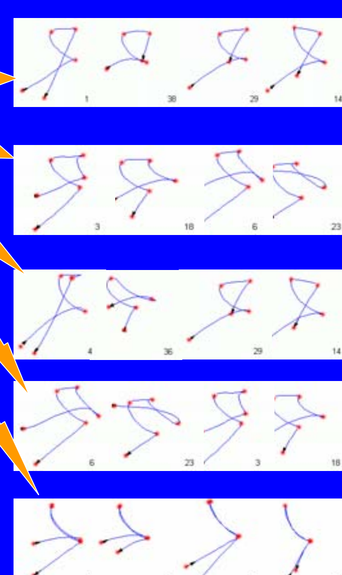


Action Trajectories



Experiment results (1)

Actions	3 Best matches	Evaluation & comments
1	38 29 14	Correct
2	Pick up	Correct
3	18 6 23	Correct
4	**36** 29 14	One wrong
5		Unique action
6	23 3 18	Correct
7	33 8 48	correct
8	33 7 **60**	One wrong
9	Pick up	Correct
10	Put down	Correct
11	Pick up	Correct
12	Put down	Correct
13		Unique action
14	16 1 29	Correct
15		Unique action
16	38 14 29	Correct
17	**Pick up**	Incorrect, object hidden
18	3 23 6	Correct
19	Pick up	Correct
20		Unique random motion



Lecture-17

Kalman Filter

Main Points

- Very useful tool.
- It produces an optimal estimate of the state vector based on the noisy measurements (observations).
- For the state vector it also provides confidence (certainty) measure in terms of a covariance matrix .
- It integrates estimate of state over time.
- It is a sequential state estimator.

State-Space Model

State-transition equation

$$\mathbf{z}(k) = \Phi(k, k-1)\mathbf{z}(k-1) + \mathbf{w}(k)$$

State model error With covariance Q(k)

State Vector

Measurement (observation) equation

$$\mathbf{y}(k) = \mathbf{H}(k)\mathbf{z}(k) + \mathbf{v}(k)$$

Observation Noise with covariance R(k)

Measurement Vector

Kalman Filter Equations

State Prediction

$$\hat{\mathbf{z}}_b(k) = \Phi(k, k-1)\hat{\mathbf{z}}_a(k-1)$$

Covariance Prediction

$$\mathbf{P}_b(k) = \Phi(k, k-1)\mathbf{P}_a(k-1)\Phi^T(k, k-1) + \mathbf{Q}(k)$$

Kalman Gain

$$\mathbf{K}(k) = \mathbf{P}_b(k)\mathbf{H}^T(k)(\mathbf{H}(k)\mathbf{P}_b(k)\mathbf{H}^T(k) + \mathbf{R}(k))^{-1}$$

State-update

$$\hat{\mathbf{z}}_a(k) = \hat{\mathbf{z}}_b(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{H}(k)\hat{\mathbf{z}}_b(k)]$$

Covariance-update

$$\mathbf{P}_a(k) = \mathbf{P}_b(k) - \mathbf{K}(k)\mathbf{H}(k)\mathbf{P}_b(k)$$

Two Special Cases

- Steady State

$$\Phi(k,k-1)=\Phi$$

$$\mathbf{Q}(k)=\mathbf{Q}$$

$$\mathbf{H}(k)=\mathbf{H}$$

$$\mathbf{R}(k)=\mathbf{R}$$

- Recursive least squares

$$\Phi(k,k-1)=\mathbf{I}$$

$$\mathbf{Q}(k)=0$$

Comments

- In some cases, state transition equation and the observation equation both may be non-linear.
- We need to linearize these equation using Taylor series.

Extended Kalman Filter

$$\mathbf{z}(k) = \mathbf{f}(\mathbf{z}(k-1)) + \mathbf{w}(k)$$

$$\mathbf{y}(k) = \mathbf{h}(\mathbf{z}(k)) + \mathbf{v}(k)$$

$$\mathbf{f}(\mathbf{z}(k-1)) \approx \mathbf{f}(\hat{\mathbf{z}}_a(k-1)) + \frac{\partial \mathbf{f}(\mathbf{z}(k-1))}{\partial \mathbf{z}(k-1)}(\mathbf{z}(\mathrm{k}-1) - \hat{\mathbf{z}}_a(k-1))$$

Taylor series

$$\mathbf{h}(\mathbf{z}(k)) \approx \mathbf{h}(\hat{\mathbf{z}}_b(k)) + \frac{\partial \mathbf{h}(\mathbf{z}(k))}{\partial \mathbf{z}(k)}(\mathbf{z}(k) - \hat{\mathbf{z}}_b(k-1))$$

Extended Kalman Filter

$$\mathbf{z}(k) = \mathbf{f}(\mathbf{z}(k-1)) + \mathbf{w}(k)$$

$$\mathbf{z}(k) = \mathbf{f}(\hat{\mathbf{z}}_a(k-1)) + \frac{\partial \mathbf{f}(\mathbf{z}(k-1))}{\partial \mathbf{z}(k-1)}(\mathbf{z}(\mathrm{k}-1) - \hat{\mathbf{z}}_a(k-1)) + \mathbf{w}(k)$$

$$\mathbf{z}(k) \approx \Phi(k, k-1)\mathbf{z}(k-1) + \mathbf{u}(k) + \mathbf{w}(k)$$

$$\mathbf{u}(k) = \mathbf{f}(\hat{\mathbf{z}}_a(k-1)) - \Phi(k, k-1)\hat{\mathbf{z}}_a(k-1)$$

$$\Phi(k, k-1) = \frac{\partial \mathbf{f}(\mathbf{z}(k-1))}{\partial \mathbf{z}(k-1)}$$

Extended Kalman Filter

$$\mathbf{y}(k) = \mathbf{h}(\mathbf{z}(k)) + \mathbf{v}(k)$$

$$\mathbf{y}(k) = \mathbf{h}(\hat{\mathbf{z}}_b(k)) + \frac{\partial \mathbf{h}(\mathbf{z}(k))}{\partial \mathbf{z}(k)}(\mathbf{z}(k) - \hat{\mathbf{z}}_b(k-1)) + \mathbf{v}(k)$$

$$\tilde{\mathbf{y}}(k) \approx \mathbf{H}(k)\mathbf{z}(k) + \mathbf{v}(k)$$

$$\tilde{\mathbf{y}}(k) = \mathbf{y}(k) - \mathbf{h}(\hat{\mathbf{z}}_b(k)) + \mathbf{H}(k)\hat{\mathbf{z}}_b(k)$$

$$\mathbf{H}(k) = \frac{\partial \mathbf{h}(\mathbf{z}(k))}{\partial \mathbf{z}(k)}$$

Multi-Frame Feature Tracking

Application of Kalman Filter

- Assume feature points have been detected in each frame.
- We want to track features in multiple frames.
- Kalman filter can estimate the position and uncertainty of feature in the next frame.
  - Where to look for a feature
  - how large a region should be searched

$$\mathbf{p}_k = [x_k, y_k]^T$$ Location

$$\mathbf{v}_k = [u_k, v_k]^T$$ Velocity

$$\mathbf{Z} = [x_k, y_k, u_k, v_k]^T$$ State Vector

System Model

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \mathbf{v}_{k-1} + \xi_{k-1}$$

$$\mathbf{v}_k = \mathbf{v}_{k-1} + \eta_{k-1}$$

noise

$$\mathbf{Z}_k = \Phi_{k-1}\mathbf{Z}_{k-1} + \mathbf{w}_{k-1}$$

$$\Phi_{k-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{w}_{k-1} = \begin{bmatrix} \xi_{k-1} \\ \eta_{k-1} \end{bmatrix}$$

Measurement Model

$$\mathbf{y}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{bmatrix} + \mu_k$$

$$\mathbf{y}_k = \mathbf{H} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{bmatrix} + \mu_k$$

Measurement matrix

Kalman Filter Equations

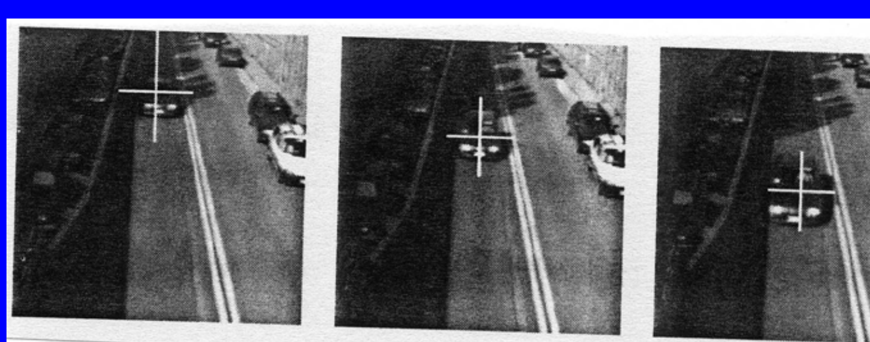
State Prediction $$\hat{\mathbf{z}}_b(k) = \Phi(k, k-1)\hat{\mathbf{z}}_a(k-1)$$

Covariance Prediction $$\mathbf{P}_b(k) = \Phi(k, k-1)\mathbf{P}_a(k-1)\Phi^T(k, k-1) + \mathbf{Q}(k)$$

Kalman Gain $$\mathbf{K}(k) = \mathbf{P}_b(k)\mathbf{H}^T(k)(\mathbf{H}(k)\mathbf{P}_b(k)\mathbf{H}^T(k) + \mathbf{R}(k))^{-1}$$

State-update $$\hat{\mathbf{z}}_a(k) = \hat{\mathbf{z}}_b(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{H}(k)\hat{\mathbf{z}}_b(k)]$$

Covariance-update $$\mathbf{P}_a(k) = \mathbf{P}_b(k) - \mathbf{K}(k)\mathbf{H}(k)\mathbf{P}_b(k)$$



Kalman Filter: Relation to Least Squares

$$f_i(\mathbf{Z}, \mathbf{y}_i) = 0$$

↓ Taylor series

$$f_i(\mathbf{Z}, \mathbf{y}_i) = 0 \approx f_i(\hat{\mathbf{Z}}_{i-1}, \hat{\mathbf{y}}_i) + \frac{\partial f_i}{\partial \mathbf{y}}(\mathbf{y} - \hat{\mathbf{y}}_i) + \frac{\partial f_i}{\partial \mathbf{z}}(\mathbf{z} - \hat{\mathbf{z}}_\mathbf{i})$$

$$\mathbf{Y}_i = H_i \mathbf{Z} + w_i$$

$$\mathbf{Y}_i = -f_i(\hat{\mathbf{Z}}_{i-1}, \hat{\mathbf{y}}_i) + \frac{\partial f_i}{\partial \mathbf{z}} \hat{\mathbf{z}}_{i-1}, H_i = \frac{\partial f_i}{\partial \mathbf{z}}$$

$$w_i = \frac{\partial f_i}{\partial \mathbf{y}}(\mathbf{y} - \hat{\mathbf{y}}_i)$$

Kalman Filter: Relation to Least Squares

$$C = (\hat{\mathbf{Z}}_0 - \mathbf{Z})^T P_0^{-1} (\hat{\mathbf{Z}}_0 - \mathbf{Z}) + \sum_{i=1}^{k} (\mathbf{Y}_i - H_i \mathbf{Z})^T W^{-1}{}_i (\mathbf{Y}_i - H_i \mathbf{Z})$$

↓ minimize

$$\hat{\mathbf{Z}} = [P_0^{-1} + \sum_{i=1}^{k} H_i^T W_i^{-1} H_i]^{-1} [P_0^{-1} \hat{\mathbf{Z}}_0 + \sum_{i=1}^{k} H_i^T W_i^{-1} \mathbf{Y}_i]$$

Batch Mode

Kalman Filter: Relation to Least Squares

$$\hat{\mathbf{Z}}_k = [P_0^{-1} + \sum_{i=1}^{k} H_i^{\ T} W_i^{-1} H_i]^{-1}[P_0^{-1}\hat{\mathbf{Z}}_0 + \sum_{i=1}^{k} H_i^{\ T} W_i^{-1} \mathbf{Y}_i]$$

$$\hat{\mathbf{Z}}_{k-1} = [P_0^{-1} + \sum_{i=1}^{k-1} H_i^{\ T} W_i^{-1} H_i]^{-1}[P_0^{-1}\hat{\mathbf{Z}}_0 + \sum_{i=1}^{k-1} H_i^{\ T} W_i^{-1} \mathbf{Y}_i]$$

Recursive Mode

Kalman Filter: Relation to Least Squares

$$\mathbf{Z}_k = \mathbf{Z}_{k-1} + K_k (Y_k - H_k \mathbf{Z}_{k-1})$$

$$K_k = P_{k-1} H^T{}_k \ (W_k + H_k P_{k-1} H_k^{\ T})^{-1}$$

$$P_k = (I - K_k H_k) P_{k-1}$$

$$\Phi(k, k-1) = \mathbf{I}$$

$$\mathbf{Q}(k) = 0$$

$$Y_k = -f^T(\mathbf{Z}_{k-1}, \mathbf{y}_{k-1}) + \frac{\partial f}{\partial \mathbf{Z}} \mathbf{Z}_{k-1}$$

$$H_k = \frac{\partial f}{\partial \mathbf{Z}}$$

$$W^k = \frac{\partial f}{\partial \mathbf{y}} \mathrm{A}_k \frac{\partial f}{\partial \mathbf{y}}^T$$

Covariance matrix for measurement Vector y

Kalman Filter (Least Squares)

State Prediction

$$\hat{\mathbf{z}}_b(k) = \Phi(k, k-1)\hat{\mathbf{z}}_a(k-1)$$

$$\hat{\mathbf{z}}_b(k) = \hat{\mathbf{z}}_a(k-1)$$

Covariance Prediction

$$\mathbf{P}_b(k) = \Phi(k, k-1)\mathbf{P}_a(k-1)\Phi^T(k, k-1) + \mathbf{Q}(k)$$

$$\mathbf{P}_b(k) = \mathbf{P}_a(k-1)$$

Kalman Gain

$$\mathbf{K}(k) = \mathbf{P}_b(k)\mathbf{H}^T(k)(\mathbf{H}(k)\mathbf{P}_b(k)\mathbf{H}^T(k) + \mathbf{R}(k))^{-1}$$

$$\mathbf{K}(k) = \mathbf{P}_b(k)\mathbf{H}^T(k)(\mathbf{H}(k)\mathbf{P}_b(k)\mathbf{H}^T(k) + \mathbf{W}(k))^{-1}$$

Kalman Filter (Least Squares)

State-update

$$\hat{\mathbf{z}}_a(k) = \hat{\mathbf{z}}_b(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{H}(k)\hat{\mathbf{z}}_b(k)]$$

$$\hat{\mathbf{z}}\ (k) = \hat{\mathbf{z}}\ (k-1) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{H}(k)\hat{\mathbf{z}}\ (k-1)]$$

Covariance-update

$$\mathbf{P}_a(k) = \mathbf{P}_b(k) - \mathbf{K}(k)\mathbf{H}(k)\mathbf{P}_b(k)$$

$$\mathbf{P}\ (k) = \mathbf{P}\ (k-1) - \mathbf{K}(k)\mathbf{H}(k)\mathbf{P}\ (k-1)$$

Computing Motion Trajectories

Algorithm For Computing Motion Trajectories

- Compute tokens using Moravec's interest operator (intensity constraint).
- Remove tokens which are not interesting with respect to motion (optical flow constraint).
  - Optical flow of a token should differ from the mean optical flow around a small neighborhood.

Algorithm For Computing Motion Trajectories

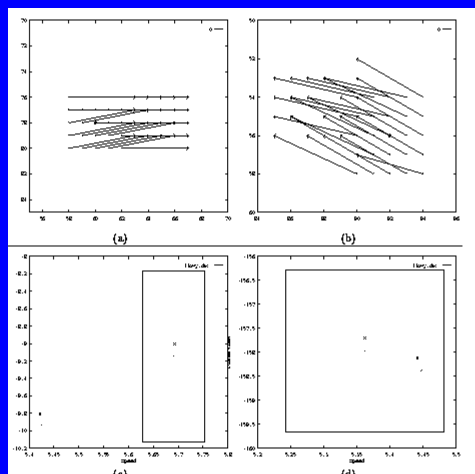
- Link optical flows of a token in different frames to obtain motion trajectories.
  - Use optical flow at a token to predict its location in the next frame.
  - Search in a small neighborhood around the predicted location in the next frame for a token.
- Smooth motion trajectories using Kalman filter.

Kalman Filter (Ballistic Model)

$$x(t) = .5a_x t^2 + v_x t + x_0 \qquad \mathbf{Z} = (a_x, a_y, v_x, v_y)$$

$$y(t) = .5a_y t^2 + v_y t + y_0 \qquad \mathbf{y} = (x(t), y(t))$$

$$f(\mathbf{Z}, \mathbf{y}) = (x(t) - .5a_x t^2 - v_x t - x_0, y(t) - .5a_y t^2 - v_y t - y_0)$$



Kalman Filter (Ballistic Model)

$$\mathbf{Z}(k) = \mathbf{Z}(k-1) + K(k)(Y(k) - H(k)\mathbf{Z}(k-1))$$

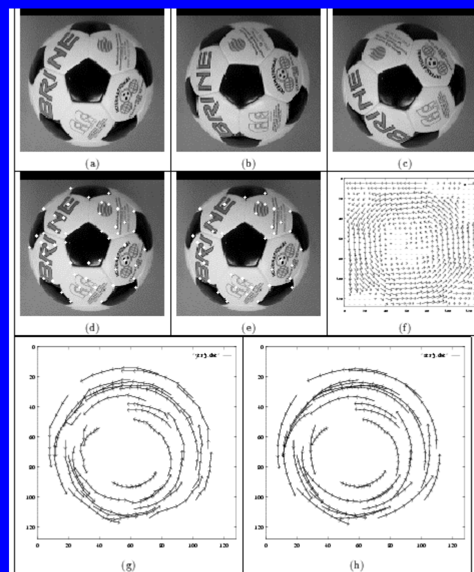
$$K(k) = P(k-1)H^T(k)\ \ (W(k) + H^T P(k-1)H^T(k))^{-1}$$

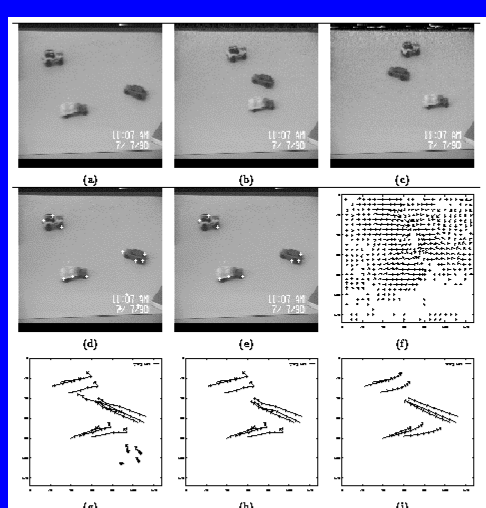
$$P(k) = (I - K(k)H(k))P(k-1)$$

$$Y(k) = -f^T(\mathbf{Z(k-1)},\mathbf{y}) + \frac{\partial f}{\partial \mathbf{Z}}\mathbf{Z}(k-1)$$

$$H(k) = \frac{\partial f}{\partial \mathbf{Z}}$$

$$W(k) = \frac{\partial f}{\partial \mathbf{y}}\mathrm{A}(k)\frac{\partial f}{\partial \mathbf{y}}^T$$


(a) (b) (c)
(d) (e) (f)
(g) (h)


(a) (b) (c)
(d) (e) (f)
(g) (h) (i)