BMB5113
COMPUTER VISION

MOVING IMAGES

& OPTICAL FLOW

Why Estimate Visual Motion?

- Visual motion can be annoying
  - Camera instabilities, jitter
  - Measure it; remove it (stabilize)
- Visual motion indicates dynamics in the scene
  - Moving objects, behavior
  - Track objects and analyze trajectories
- Visual motion reveals spatial layout
  - Motion parallax

Camera versus Object Movements

- Static camera- Static image
- Static camera- Dynamic image
- Dynamic camera- Static image
- Dynamic camera- Dynamic image

Motion Estimation Techniques

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

Motion Analysis

- Try to find answers to the questions:
  - How many objects move?
  - What are their directions?
  - What are their velocity?
  - What kind of movements are they?
    - For humans: running, walking etc…

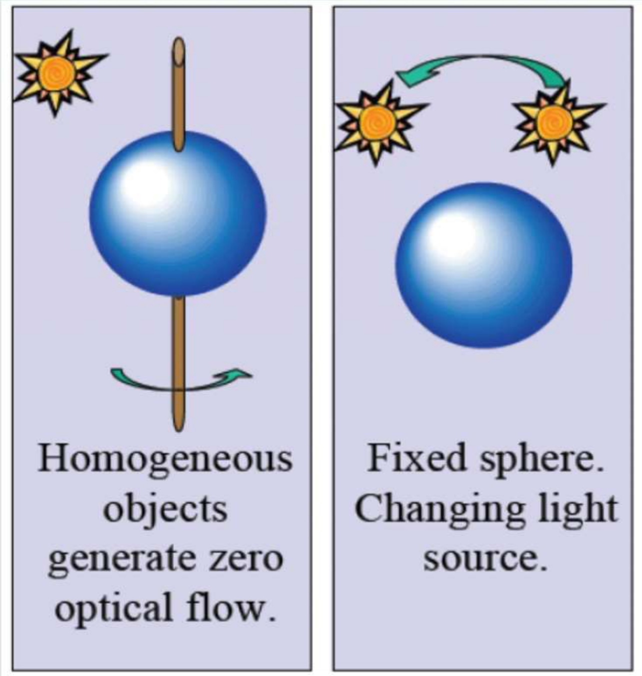
Optical Flow

- Apparent motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion
  - Think of a uniform rotating sphere under fixed lighting vs. a stationary sphere under moving illumination

Optical Flow versus Motion Field

- Motion field = Real world 3-D motion
- Optical flow field = Image intensity movement

Optical flow field is not necessarily equal to motion field!



Typical Problems of Moving Images

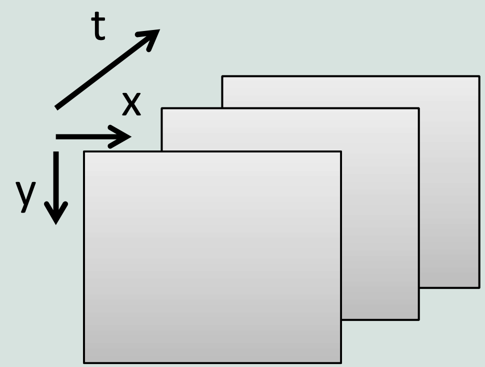
- Background subtraction
  - Frame differencing
  - Mean filtering
  - Running average
  - Gaussian mixture model
- Key frame detection
  - Pixel comparison
  - Block-based comparison
  - Histogram comparison
  - Motion-based comparison
  - Object-based comparison
- Visual motion estimation
  - Patch-based motion estimation
  - Dense motion estimation
- Jitter removal (Deghosting)
  - Local alignment
  - Region-based methods
  - Cut out-based methods
- Spatio-temporal feature extraction
- Action recognition
- Object tracking

Background Subtraction

- Frame is composed of foreground $F(x, y)$ and background $B(x, y)$
  - Foreground: moving objects
  - Background: static objects, scene
- Knowing background, foreground can be determined as

$$F(x, y) = |I_t(x, y) - B(x, y)| > Threshold$$

- Threshold should be an optimized value



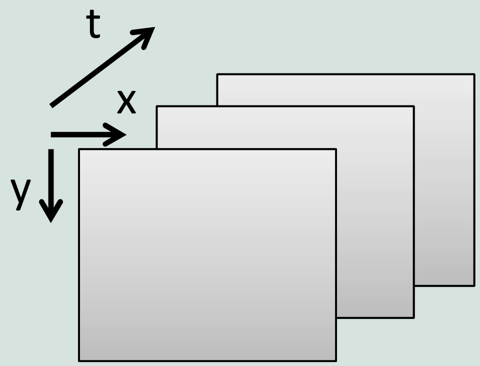
Video

Background Subtraction: Frame Differencing

- Determines foreground simply subtracting two consecutive frames

$$F_t(x,y) = |I_t(x,y) - I_{t-1}(x,y)| > Threshold$$

- Threshold should be an optimized value



Video

Background Subtraction: Mean Filtering

- Average N frames to determine background
- Accepts small movements such as those of tree leaves as background
- Algorithm:
  1. Capture N images with no objects and calculate the average background $B_{ave}(x,y)$

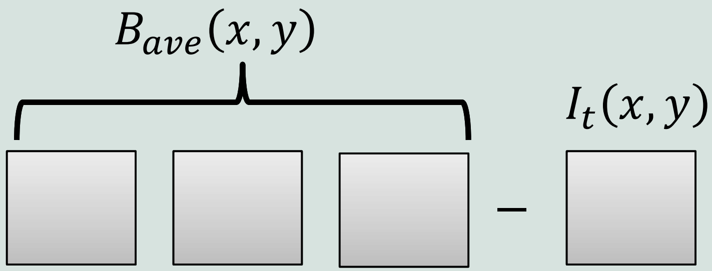
$$B_{ave}(x,y) = \frac{1}{N}\sum_{i=t-N}^{t-1} I_i(x,y)$$

  2. Capture camera image at time t. Then,

$$F_t(x,y) = |I_t(x,y) - B_{ave}(x,y)| > Threshold$$

- Disadvantages
  - Needs space to hold frames of B_{ave} in memory unnecessarily
  - In videos with high time resolution, fastly moving objects may seem as background.

for N=3



Background Subtraction: Running Average

- Background image is updated using a proportion of the current image and the previous background
- Algorithm:
  1. Compute background $B_{t+1}(x, y)$ at time t

$$B_{t+1}(x, y) = \alpha . I_t \, (x, y) + (1 - \alpha) B_t(x, y)$$

  2. Foreground at time $\mathrm{t} + 1$ is then,

$$F_{t+1}(x, y) = |I_{t+1}(x, y) - B_{t+1}(x, y)| > Threshold$$

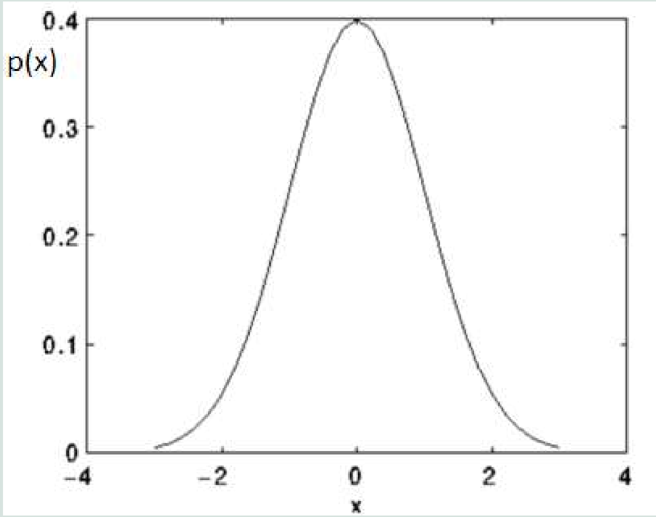
- α is updating rate
  - Needs to be a small value like 0.005
  - Then background changes slowly
- Advantage
  - There is no need to hold N frames in memory

Background Subtraction: Gaussian Mixture Model

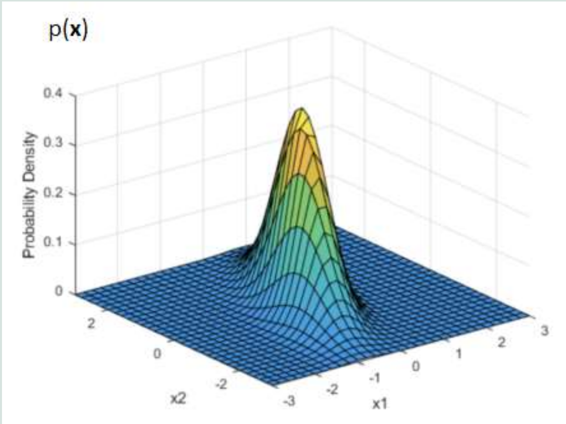
- 1-D probability density function

$$p(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \qquad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$$



Background Subtraction: Gaussian Mixture Model



- 2-D probability density function

$$p(\boldsymbol{x}|\boldsymbol{\mu},\Sigma) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}{2}}$$

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \qquad \mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i\,, \qquad \boldsymbol{\mu} = \begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}$$

$$\sigma_{xy} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)}\,, \qquad \Sigma = \begin{bmatrix}{\sigma^2}_{xx} & {\sigma^2}_{xy}\\ {\sigma^2}_{yx} & {\sigma^2}_{yy}\end{bmatrix} \approx \begin{bmatrix}\sigma^2 & 0\\ 0 & \sigma^2\end{bmatrix}$$

Background Subtraction: Gaussian Mixture Model

- M-D probability density function

$$p(\boldsymbol{x}|\boldsymbol{\mu},\Sigma) = \frac{1}{2\pi^{M/2}\sqrt{|\Sigma|}}e^{-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}}$$

  - Σ is mxm covariance matrix
  - $\boldsymbol{\mu}$ is mx1 mean vector

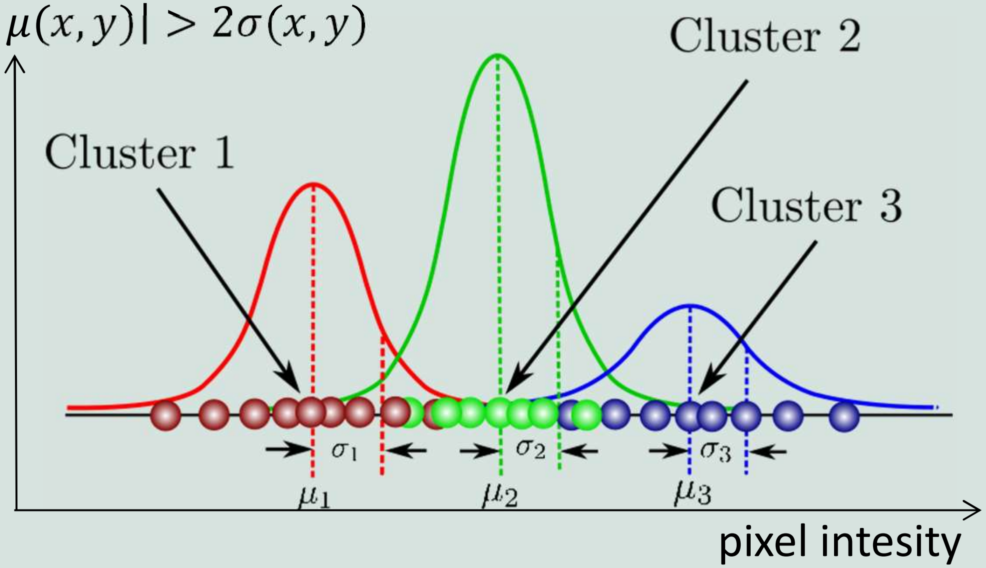
Background Subtraction: Gaussian Mixture Model

- Mixture of Gaussians can be used for background subtraction
- Algorithm:
  1. Compute background model $\mu(x, y)$

$$\mu(x,y) = \frac{1}{T}\sum_t I_t(x,y)\,, \qquad \sigma^2(x,y) = \frac{1}{T}\sum_t (I_t(x,y) - \mu(x,y)\,)^2$$

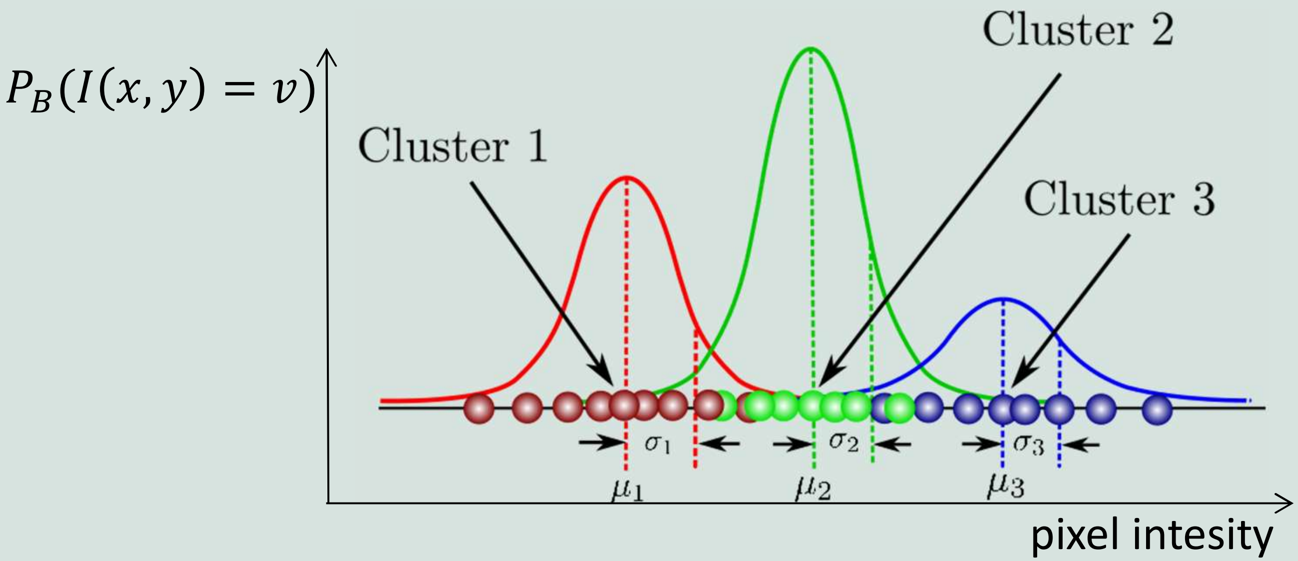
  2. Foreground segmentation,

$$F(x,y) = |I(x,y) - \mu(x,y)| > 2\sigma(x,y)$$



Background Subtraction: Gaussian Mixture Model

- Training:
  - Using frames with no foreground choose a cluster number such as k=3, 5
  - Determine k-Gaussians
  - For each pixel a similar graph can be obtained.
- Testing:
  - Evaluate every pixel with respect to T frames
  - Determine which of the k-Gaussians the pixel fits to
  - If pixel value does not change much, it will be dominant in a Gaussian function so it is background.



Background Subtraction: Gaussian Mixture Model

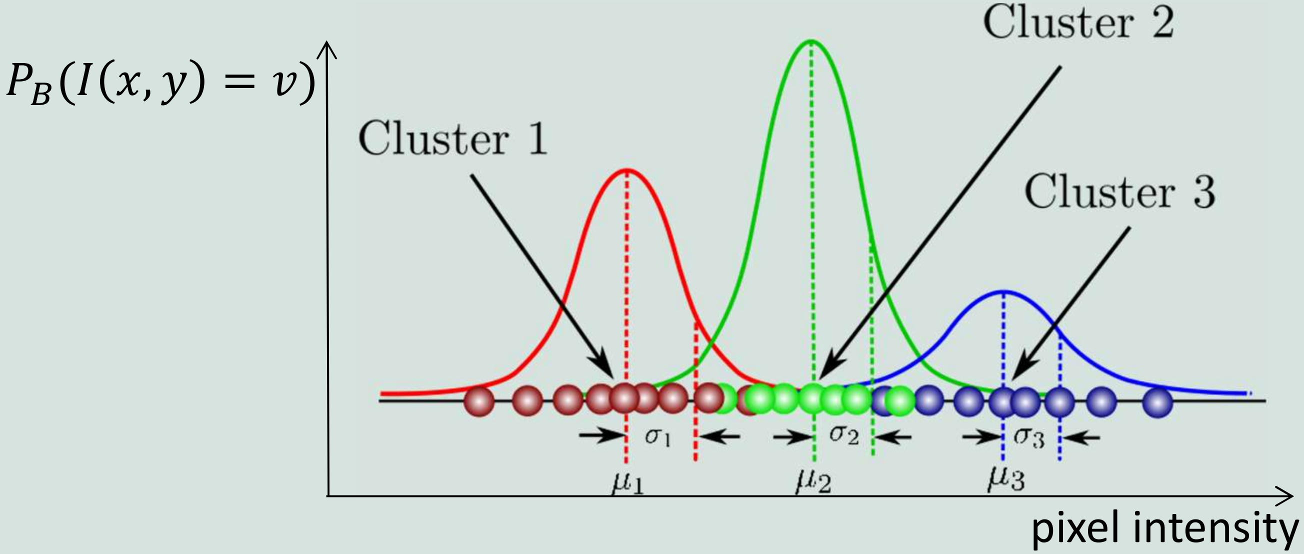
- Training update:

$$\rho = \alpha P_B(V_{t+1}), \qquad P_B(I(x,y) = v) = \sum_i w_i \, e^{-\frac{(v-\mu_i)^2}{2\sigma_i^2}}$$

- Then updating of μ and σ

$$\mu_{t+1}(x,y) = \rho . V_{t+1}(x,y) + (1-\rho)\mu_t(x,y)$$

$$\sigma^2{}_{t+1}(x,y) = \rho(V_{t+1}(x,y) - \mu_{t+1}(x,y))^2 + (1-\rho)\sigma^2{}_t(x,y)$$

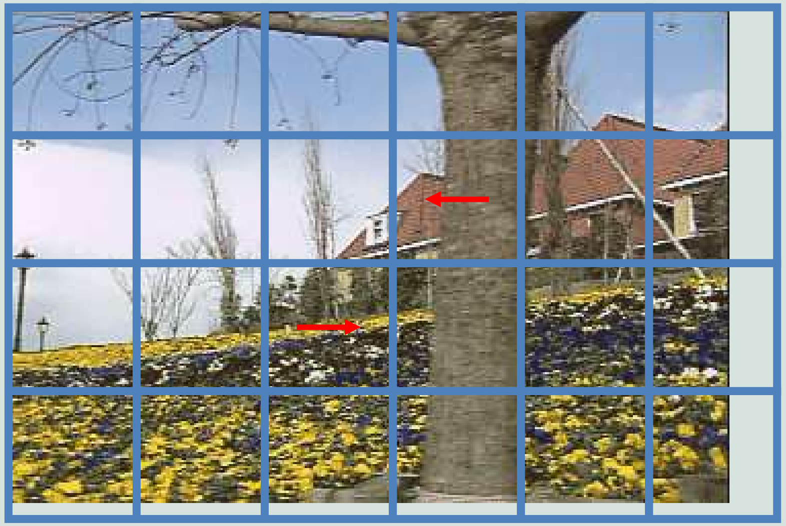


Motion Representations

- How can we describe this scene?

Block-based Motion Estimation

- Break image up into square blocks
- Estimate translation for each block
- Use this to predict next frame, code difference (MPEG-2)



Block-based Motion Estimation

- Assumes that a block of pixels is moved collectively in the frames
- By dividing an image into blocks of equal size,
  - the next position of the block is searched in the next frame
- Using one of following metrics similarity or distance is measured
  - Sum of squared difference (SSD)
  - Sum of squared error (SSE)
  - Normalized cross correlation (NCC)

Block-based Motion Estimation: Exhaustive Block Matching

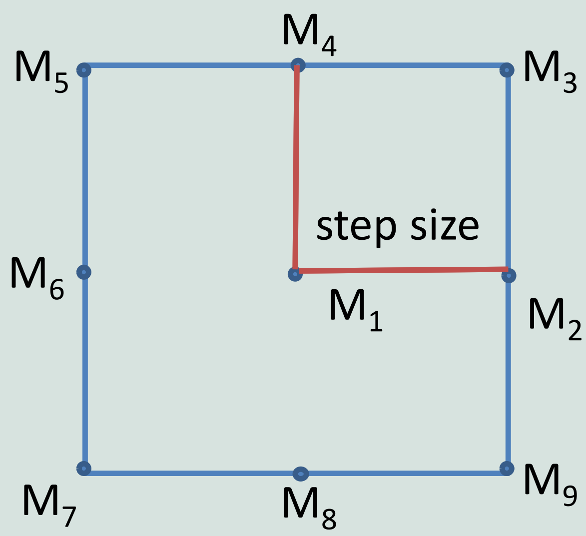
- Searches the block exhaustively in the next frame without having an assumption as to where the block might have moved.
- Guaranteed optimality within search range but has high computational complexity.

$$SSE = \sum_{Block} [I_t(x,y) - I_{t+1}(x+u, y+v)]^2 \,, \qquad \forall\ (u,v)$$

Block-based Motion Estimation: 3-Step Search Algorithm

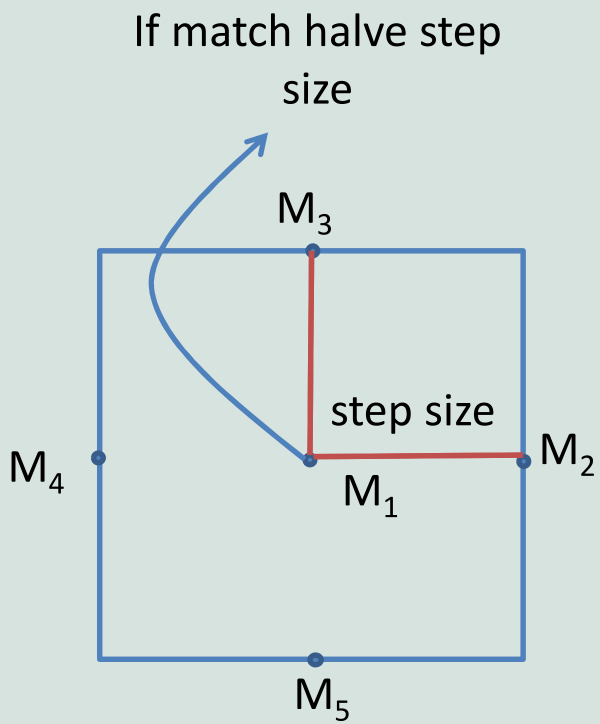
- Searches vicinity of the block first, then farther locations.
- Therefore faster searches are possible.
  1. Initial step size is set
  2. Eight blocks at a distance of step size from the center are picked for comparison
  3. The block center is moved to the location where minimum distance is obtained.
  4. The step size is halved.
- If step size is 3 algorithm stops in 3 run.

8-neighbourhood for block center movement



Block-based Motion Estimation: 2-D Logarithmic Search

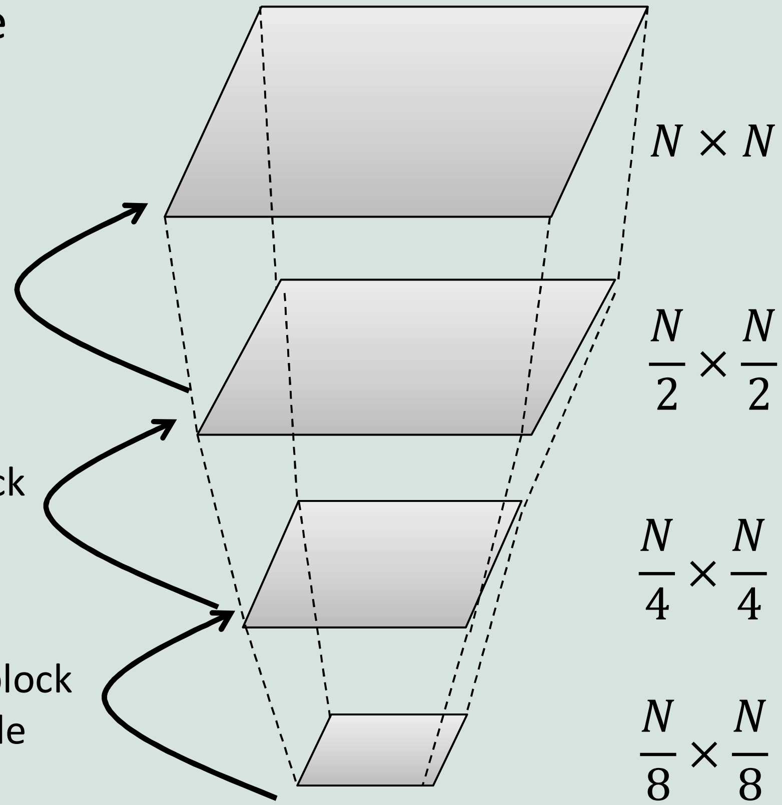
- Step size is changed only when a match with center occurs.
  1. Initial step size is set to block size/2
  2. 4-neighbourhood of the block is compared with the given step size
  3. If position of the best match is at the center, halve the step size.
  4. Otherwise move the center to the best matched position and repeat step 2 without changing the step size.
  5. When step size is 1, 8-neighbourhood is searched to find the best matching block.



Block-based Motion Estimation: Hierarchical Block Matching

- Similar to the approaches of image pyramids
- Starts searching with coarse resolution to narrow down the search range.
- Then searches higher resolutions to refine motion estimation.

8-neighbourhood checked for block center movement



Using best matched block move to upper scale

Search with e.g., 2-D logarithmic search at upper scales

Using best matched block move to upper scale

Search at this coarsest scale exhaustively

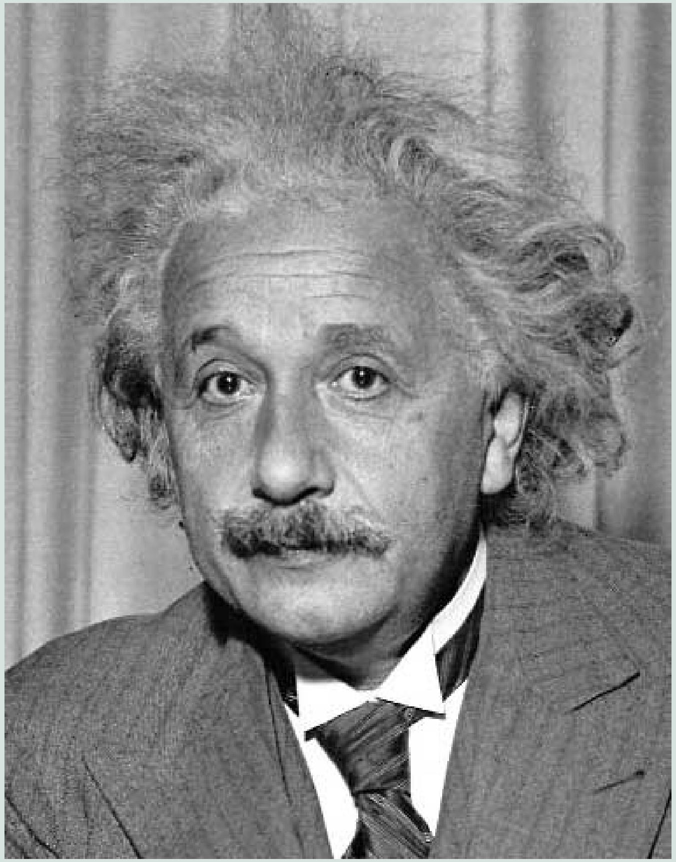
Using best matched block move to upper scale

Correlation and SSD

- For larger displacements template matching can be preferred
  - Define a small area around a pixel as the template
  - Match the template against each pixel within a search area in next image.
  - Use a match measure such as correlation, normalized correlation, or sum-of-squared difference
  - Choose the maximum (or minimum) as the match
  - Sub-pixel estimate (Lucas-Kanade)

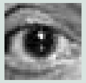
Distance Metric

- Goal: find  in image,
  assume translation only: no scale change or rotation, using search (scanning the image)
- What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum of Squared Difference
  - Normalized Cross Correlation



```
res = cv2.matchTemplate(img,template,method)
```

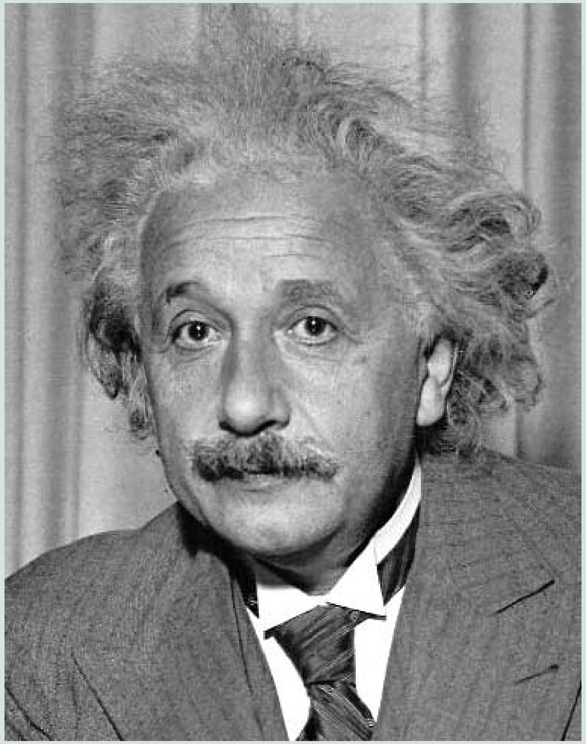
Correlation

Goal: find  in image

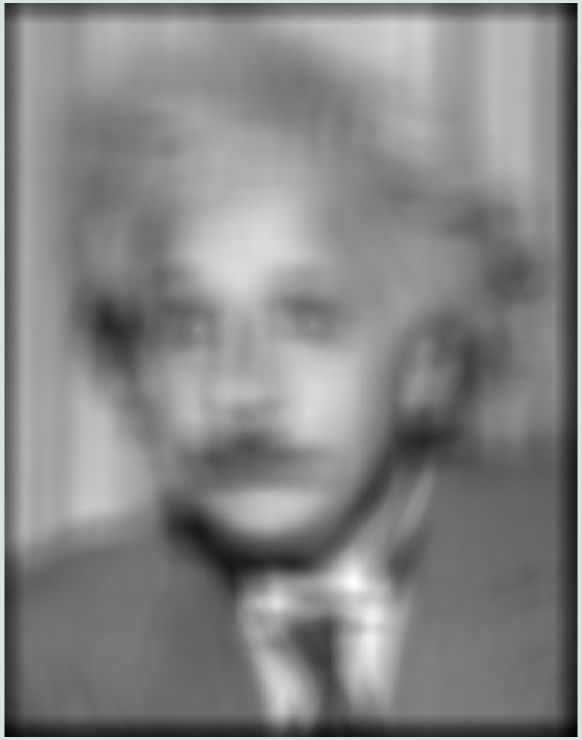
- filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

f = image
g = filter



Input



Filtered Image

What went wrong?

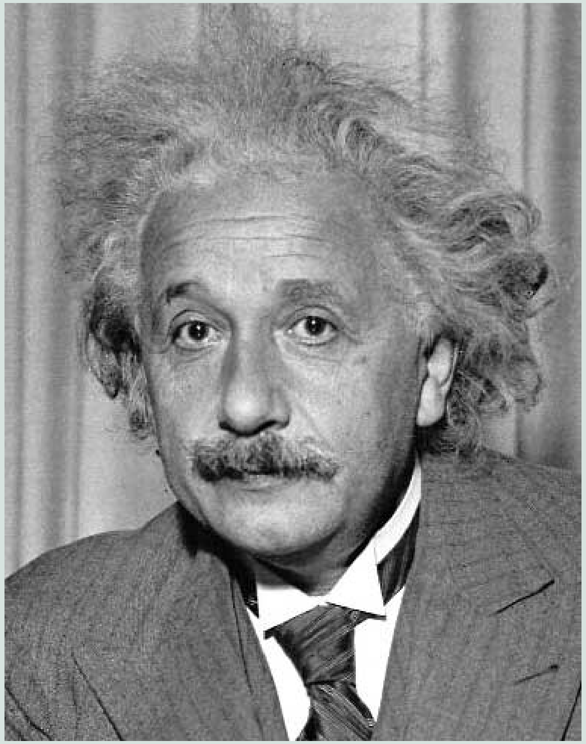
response is stronger for higher intensity

Zero-Mean Correlation

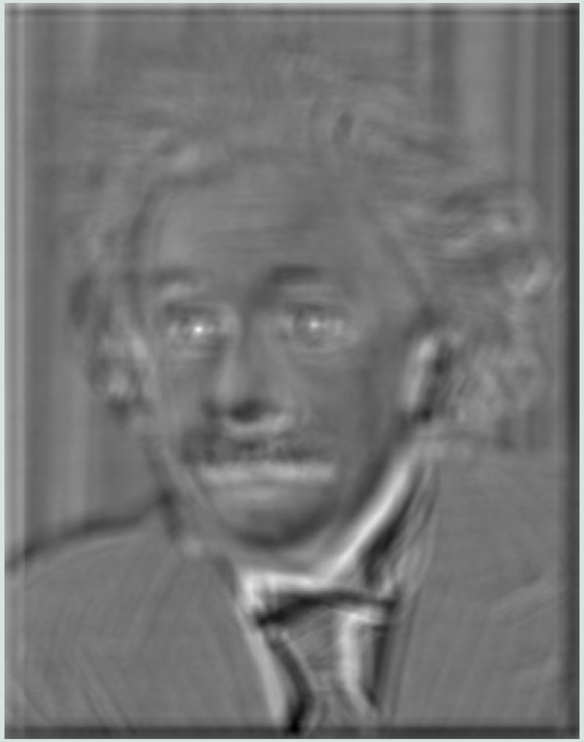
- Goal: find  in image
- filter the image f with zero-mean eye g

mean of g

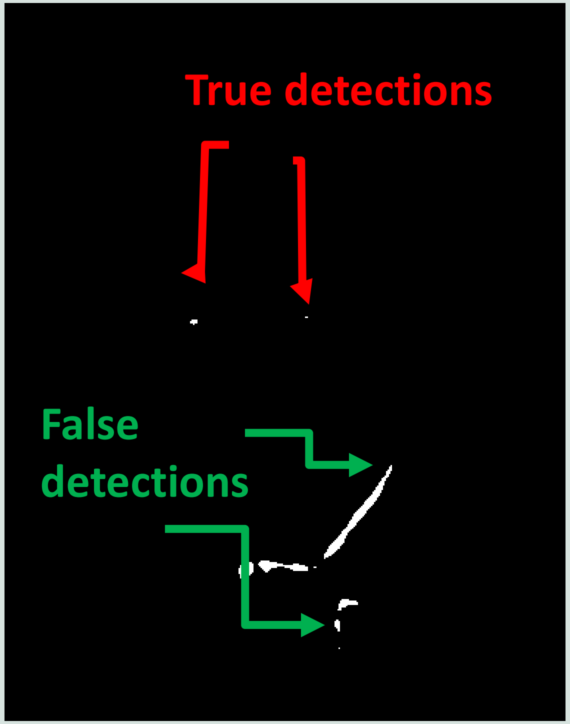
$$h[m,n]=\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l])$$



Input



Filtered Image (scaled)

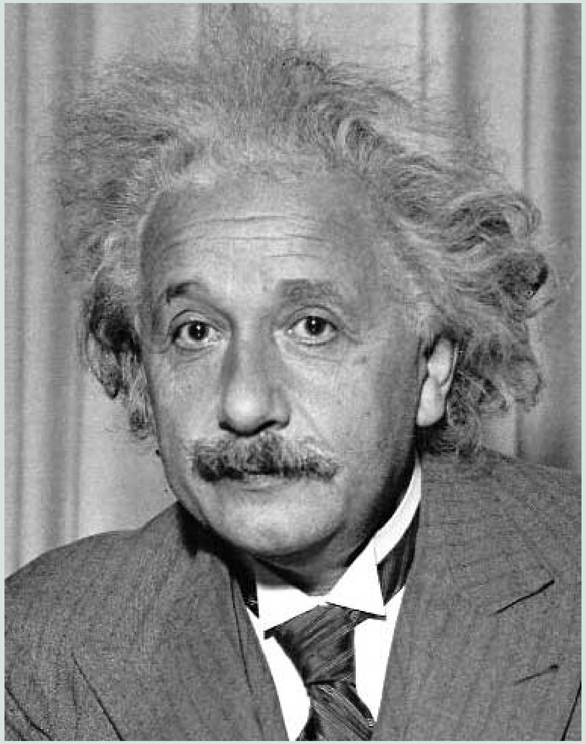


Thresholded Image

SSD (L2 Distance)

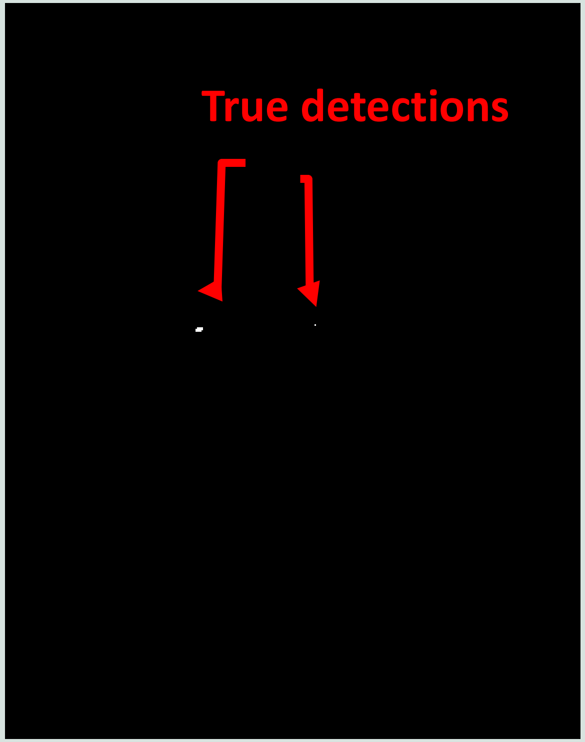
- Goal: find  in image
- SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$



Input

1- sqrt(SSD)

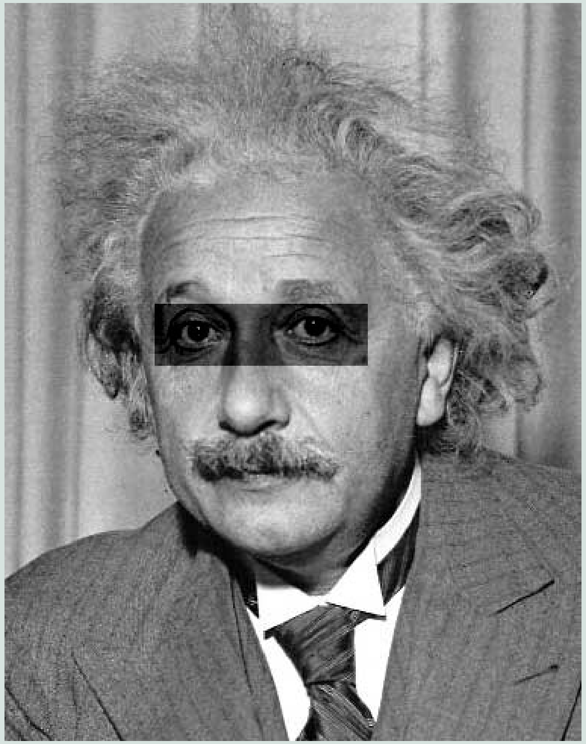


Thresholded Image

SSD (L2 Distance)

- Goal: find  in image
- Method 2: SSD

$$h[m,n]=\sum_{k,l}(g[k,l]-f[m+k,n+l])^2$$



Input

1- sqrt(SSD)

One potential downside of SSD:

Brightness Constancy Assumption

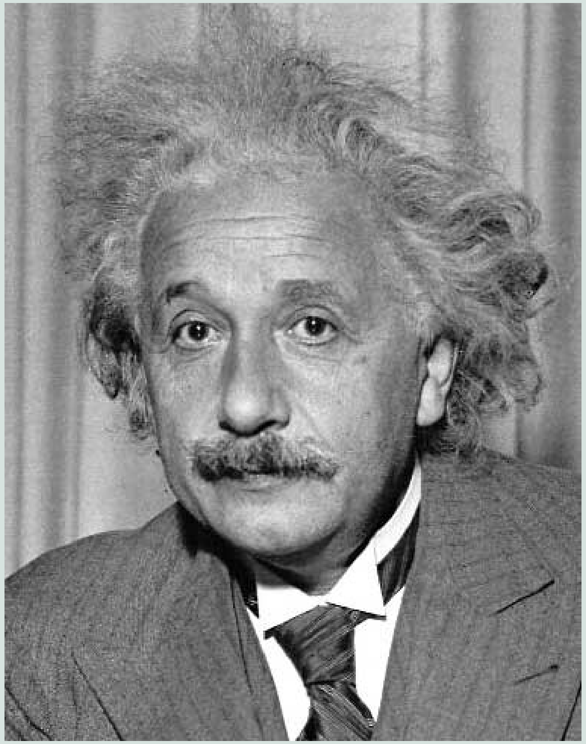
Normalized Cross-Correlation

- Goal: find  in image
- Normalized cross-correlation
  (= angle between zero-mean vectors)

$$h[m,n] = \frac{\sum_{k,l} (g[k,l] - \overset{\text{mean template}}{\bar{g}})(f[m-k,n-l] - \overset{\text{mean image patch}}{\bar{f}_{m,n}})}{\left(\sum_{k,l} (g[k,l] - \bar{g})^2 \sum_{k,l} (f[m-k,n-l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

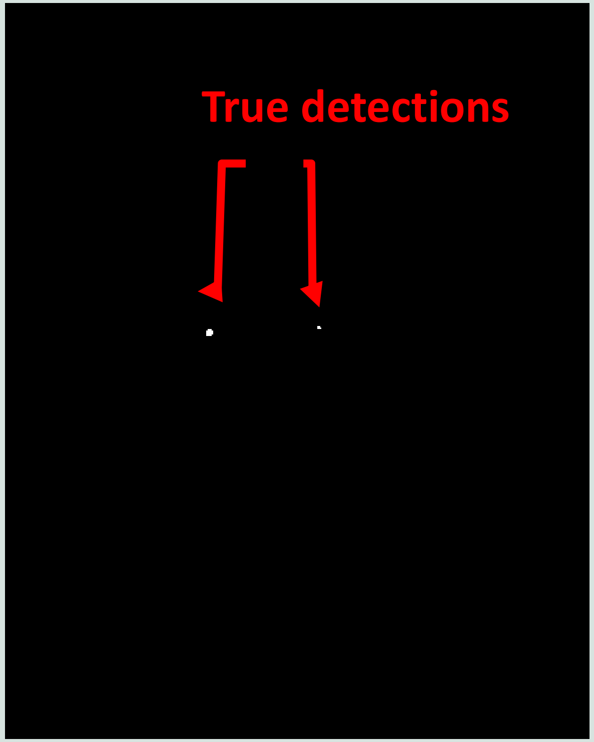
Normalized Cross-Correlation

- Goal: find  in image
- Normalized cross-correlation



Input

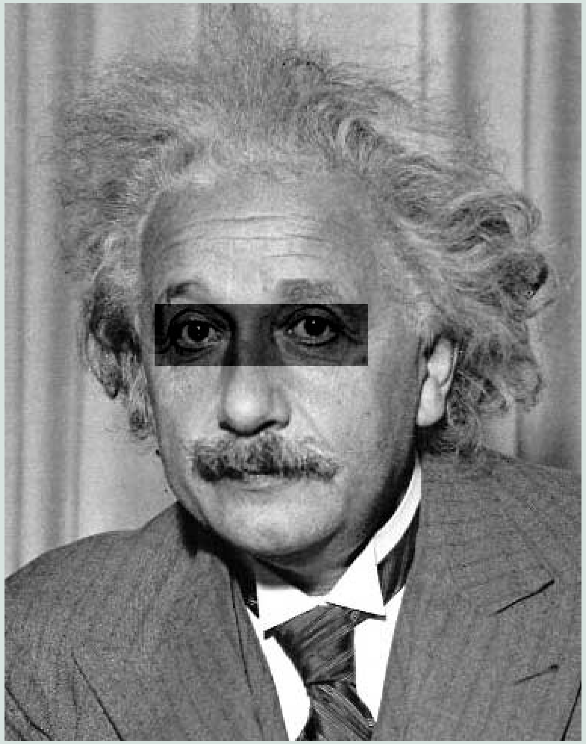
Normalized X-Correlation



Thresholded Image

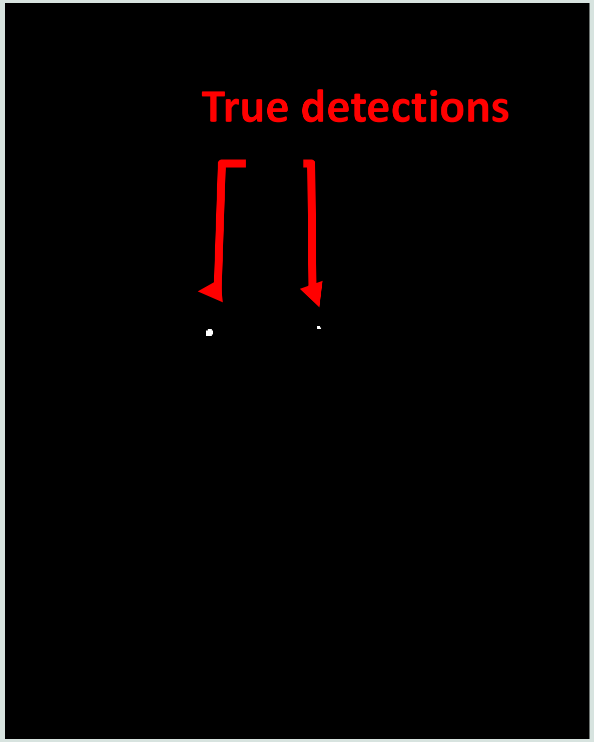
Normalized Cross-Correlation

- Goal: find  in image
- Normalized cross-correlation



Input

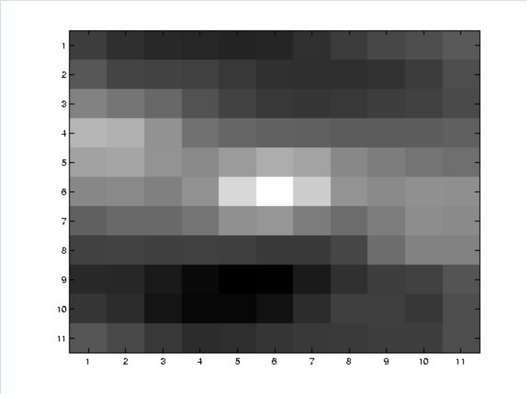
Normalized X-Correlation

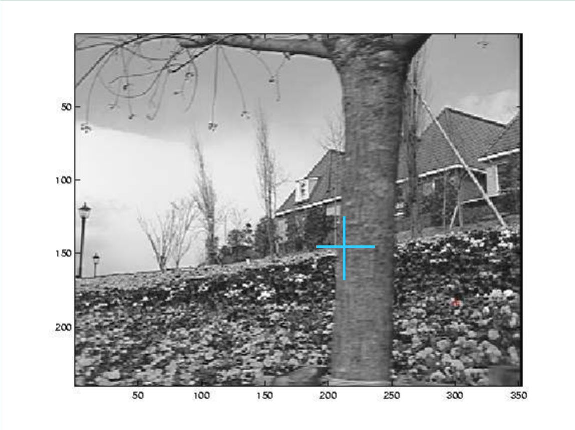


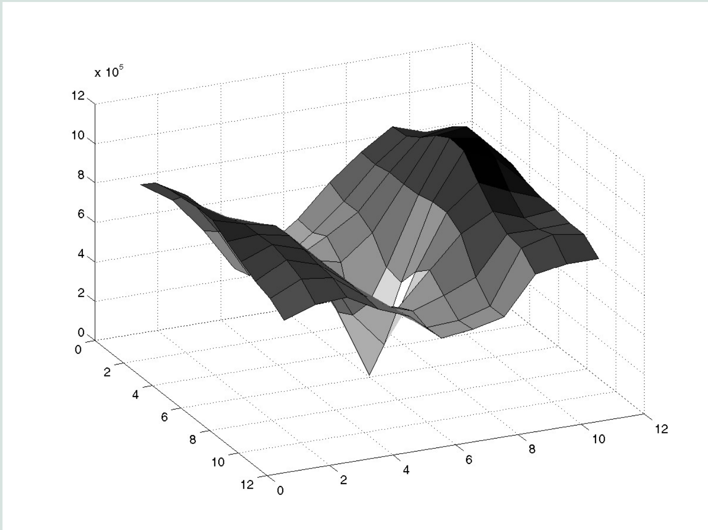
Thresholded Image

SSD Surface – Textured Area

- SSD of textured templates results in a local minimum in search results

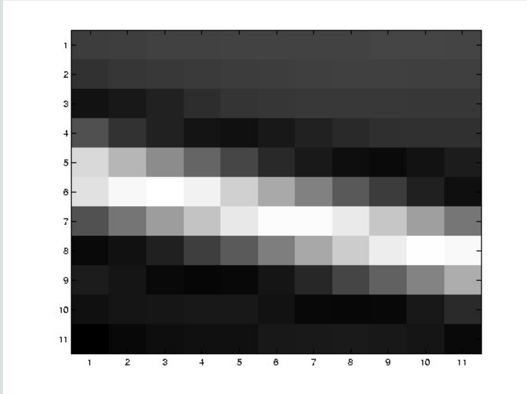


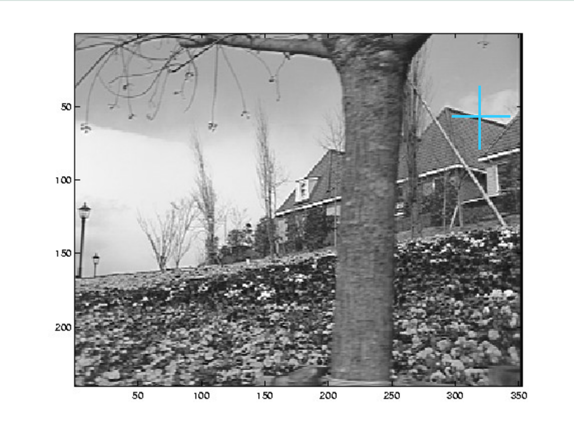


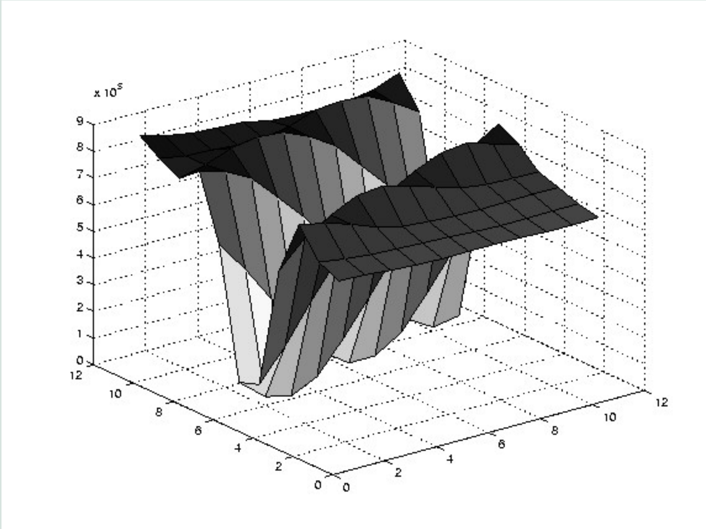


SSD Surface -- Edge

- SSD of edge-like templates results in local minima in search results along the edge

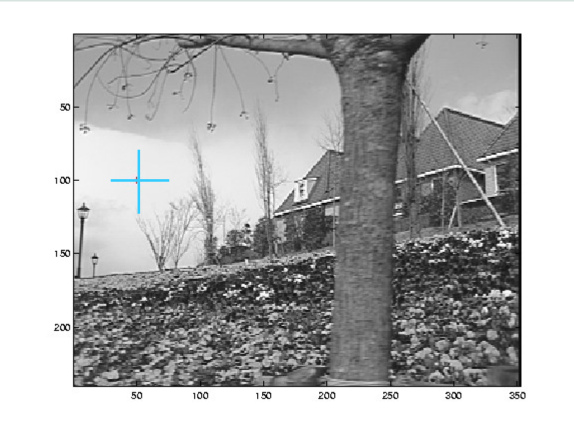


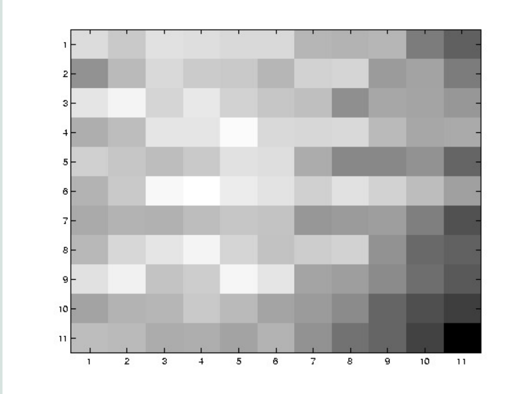


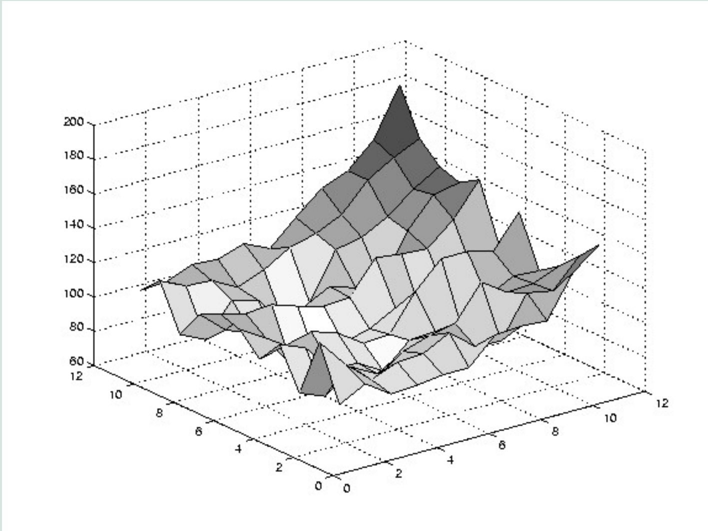


SSD – Homogeneous Area

- SSD of homogeneous templates results in an insignificant output in search results

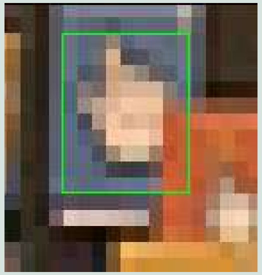




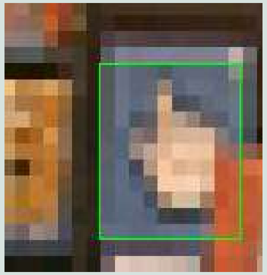


Visual Motion Estimation: Patch-based Motion Estimation

How do we determine correspondences?



I_t



I_{t+1}

Assume all change between frames is due to **motion:**

$$\boldsymbol{I}_{t+1}(\boldsymbol{x}, \boldsymbol{y}) \approx \boldsymbol{I}(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{y} + \boldsymbol{v}(\boldsymbol{x}, \boldsymbol{y}))$$

The Brightness Constraint

- Brightness Constancy Equation:

$$I_{t+1}(x,y) \approx \boxed{I(x+u(x,y),\, y+v(x,y))}$$

- Or, equivalently, minimize :

$$E(u,v) = (I_{t+1}(x,y) - I(x+u,\, y+v))^2$$

- Linearizing (assuming small (u,v)) using Taylor series expansion:

$$I_{t+1}(x,y) \approx \boxed{I(x,y) + I_x(x,y)\cdot u(x,y) + I_y(x,y)\cdot v(x,y)}$$

The Optical Flow Constraint

derivative with respect to x, y, and time

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

Minimizing:

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

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

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

In general $I_x, I_y \neq 0$

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

Lucas-Kanade: Patch Translation

- Assume a single velocity for all pixels within an image patch

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

- Remember:
$$\boldsymbol{I}_x(\boldsymbol{I}_x\boldsymbol{u} + \boldsymbol{I}_y\boldsymbol{v} + \boldsymbol{I}_t) = 0$$
$$\boldsymbol{I}_y(\boldsymbol{I}_x\boldsymbol{u} + \boldsymbol{I}_y\boldsymbol{v} + \boldsymbol{I}_t) = 0$$

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

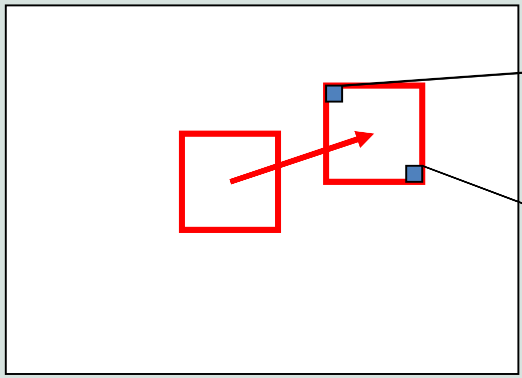
$$\left(\sum \nabla \boldsymbol{I} \ \nabla \boldsymbol{I}^T\right) \vec{\boldsymbol{U}} = -\sum \nabla \boldsymbol{I} \boldsymbol{I}_t$$

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

Lucas-Kanade: Patch Translation

(u,v) is assumed to be constant in a local patch.

$$I_x u + I_y v = -I_t \quad \Rightarrow \quad \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$



$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix}$$

$$\boxed{A\vec{u} = b}$$

Lucas-Kanade: Patch Translation

- In a local patch of 5x5 pixels in total 25 equations are obtained.
- Therefore to compute (u,v):

$$\underbrace{\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{x25} & I_{y25} \end{bmatrix}}_{A_{25\times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d_{2\times 1}} = - \underbrace{\begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \\ I_{t25} \end{bmatrix}}_{b_{25\times 1}}$$

Lucas-Kanade: Patch Translation

For colored image 75 equations are obtained :

$$\underbrace{\begin{bmatrix} I_{x1}[0] & I_{y1}[0] \\ I_{x1}[1] & I_{y1}[1] \\ I_{x1}[2] & I_{y1}[2] \\ \vdots & \vdots \\ I_{x25}[0] & I_{y25}[0] \\ I_{x25}[1] & I_{y25}[1] \\ I_{x25}[2] & I_{y25}[2] \end{bmatrix}}_{A_{75\times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d_{2\times 1}} = - \underbrace{\begin{bmatrix} I_{t1}[0] \\ I_{t1}[1] \\ I_{t1}[2] \\ \vdots \\ I_{t25}[0] \\ I_{t25}[1] \\ I_{t25}[2] \end{bmatrix}}_{b_{75\times 1}}$$

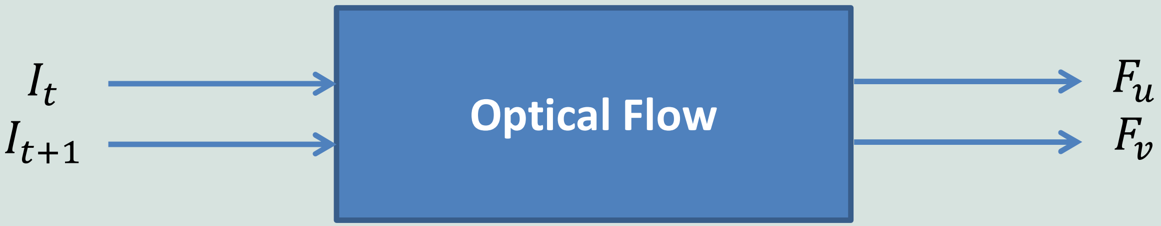
Lucas-Kanade: Solution to Equation

- Solution to equation: least squares estimation

$$(A^T A)\vec{\boldsymbol{u}} = A^T b$$

$$\vec{\boldsymbol{u}} = (A^T A)^{-1} A^T b$$

$$\underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_{(A^T A)_{2\times 2}} \underbrace{\begin{pmatrix} u \\ v \end{pmatrix}}_{\vec{\boldsymbol{u}}_{2\times 1}} = - \underbrace{\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}}_{(A^T b)_{2\times 1}}$$



$$F_{mag} = \sqrt{F_u^2 + F_v^2}$$

$$F_{ori} = \tan^{-1}\left(\frac{F_v}{F_u}\right)$$

Lucas – Kanade Optical Flow: Algorithm

1. Determine the window size
2. Compute image gradients both spatially and temporally for two consecutive images
    - The masks above applied to the first image I_t
    - The masks below applied to the second image I_{t+1}
    - Sum the obtained results

$$\underbrace{\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \text{first image} \quad \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \text{second image}}_{f_x}$$

$$\underbrace{\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \text{first image} \quad \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \text{second image}}_{f_y}$$

$$\underbrace{\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} \text{first image} \quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{second image}}_{f_t}$$

Lukas – Kanade Optical Flow: Algorithm

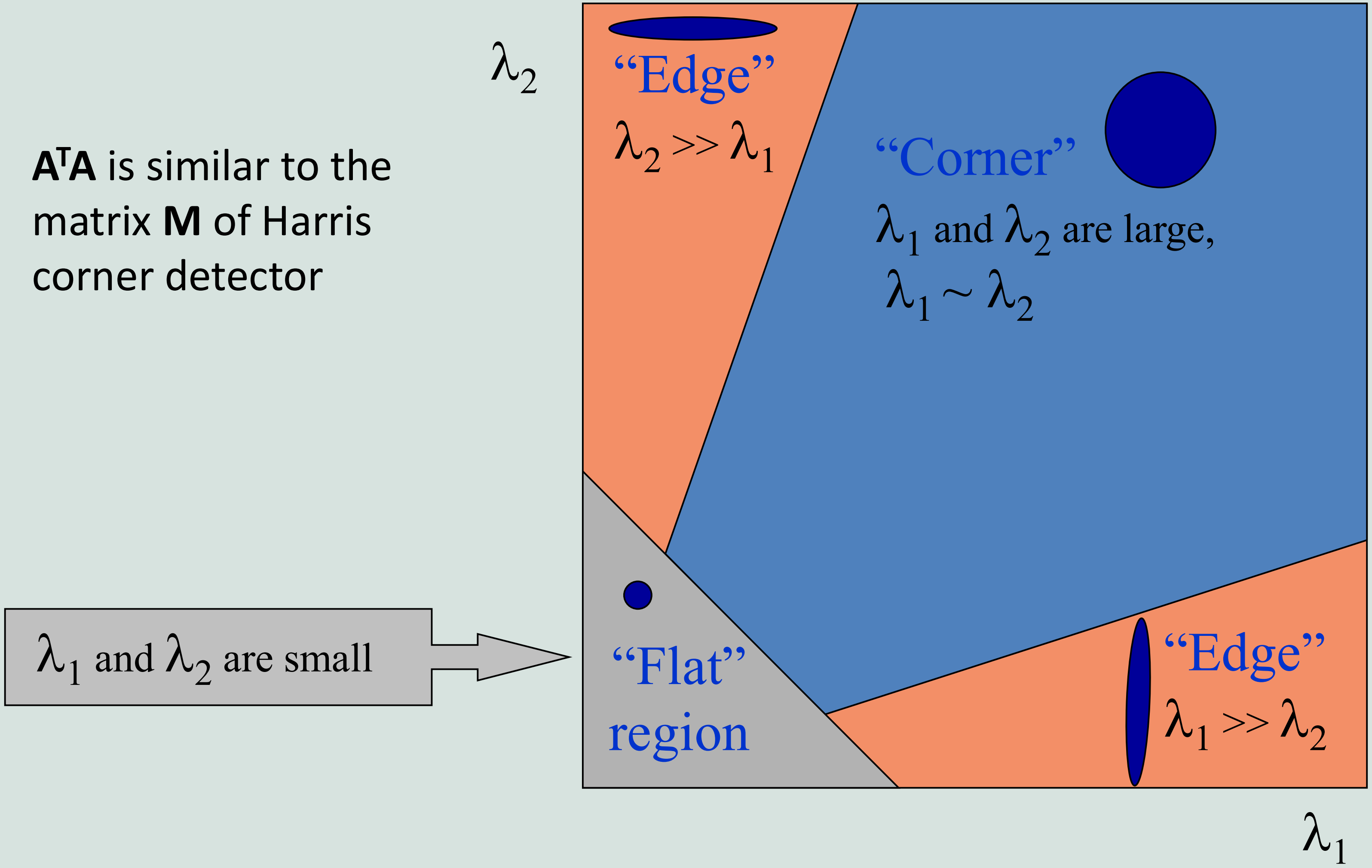
3. Compute vector **u**
   - Assume **u**=(u,v) is initially zero
   - Optical flow is computed when the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$ is greater than a threshold value
   - Stop when iterations of n and n+1 are close to each other when varying window size

$$\boxed{\vec{u} = \left(A^T A\right)^{-1} A^T b}$$

$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

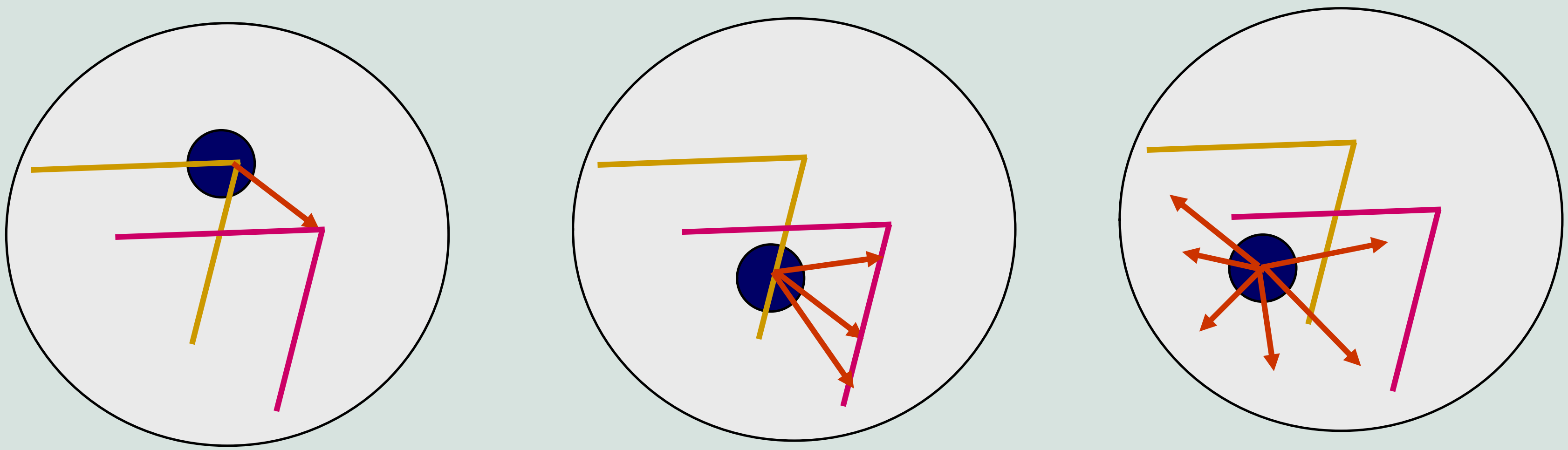
Lukas – Kanade Optical Flow: Algorithm

A^T**A** is similar to the matrix **M** of Harris corner detector



Local Patch Analysis

- How *certain* are the motion estimates?

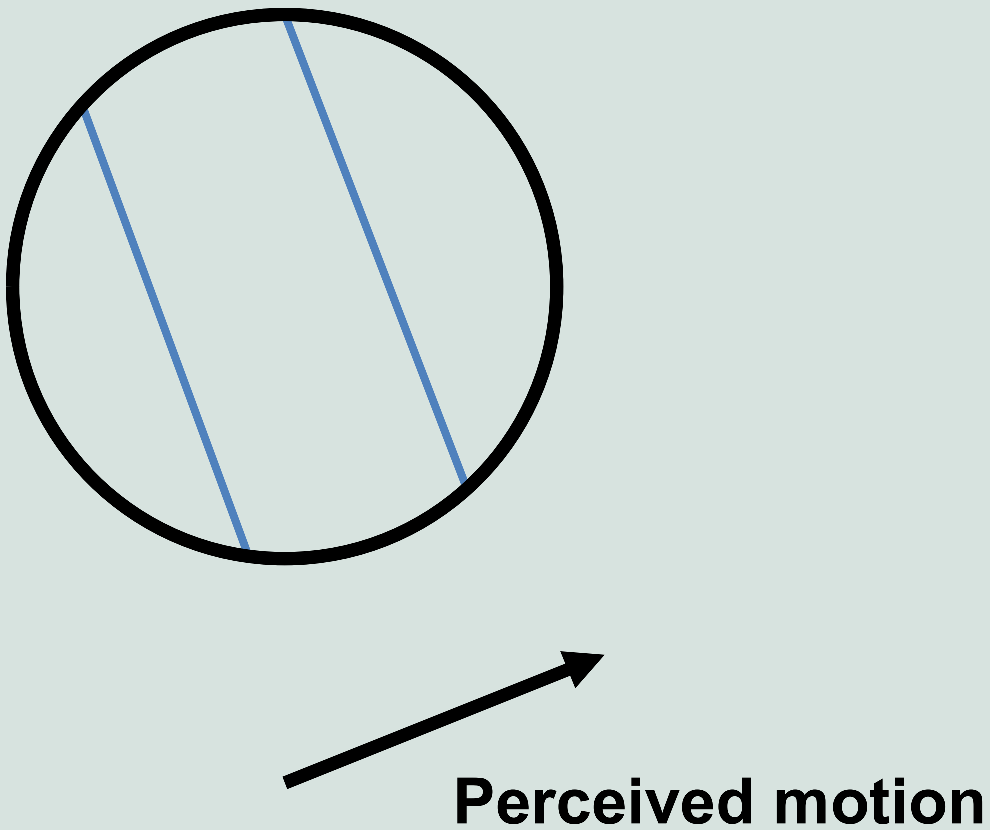


The Aperture Problem

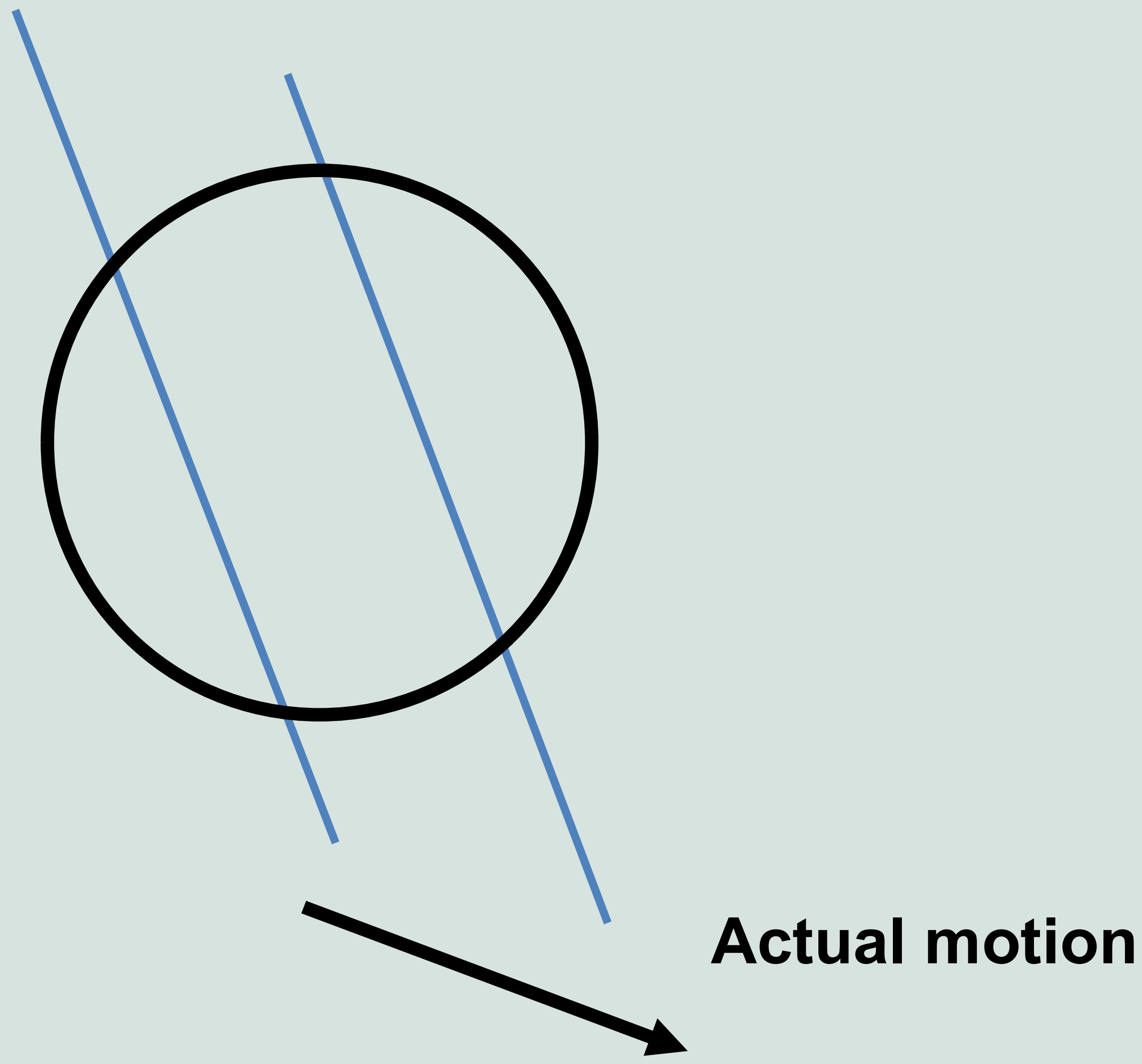
Let $M = \sum (\nabla I)(\nabla I)^T$ and $b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

- Algorithm: At each pixel compute U by solving $MU = b$
- M is singular if all gradient vectors point in the same direction
  - e.g., along an edge
- trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only normal flow is available (aperture problem)
- Corners and textured areas are OK

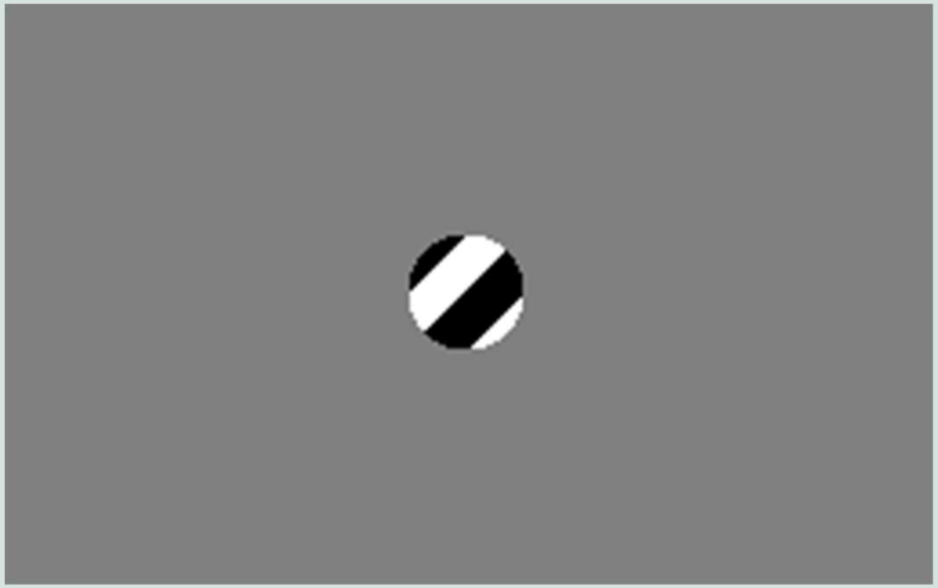
The Aperture Problem



The Aperture Problem



The Barber Pole Illusion



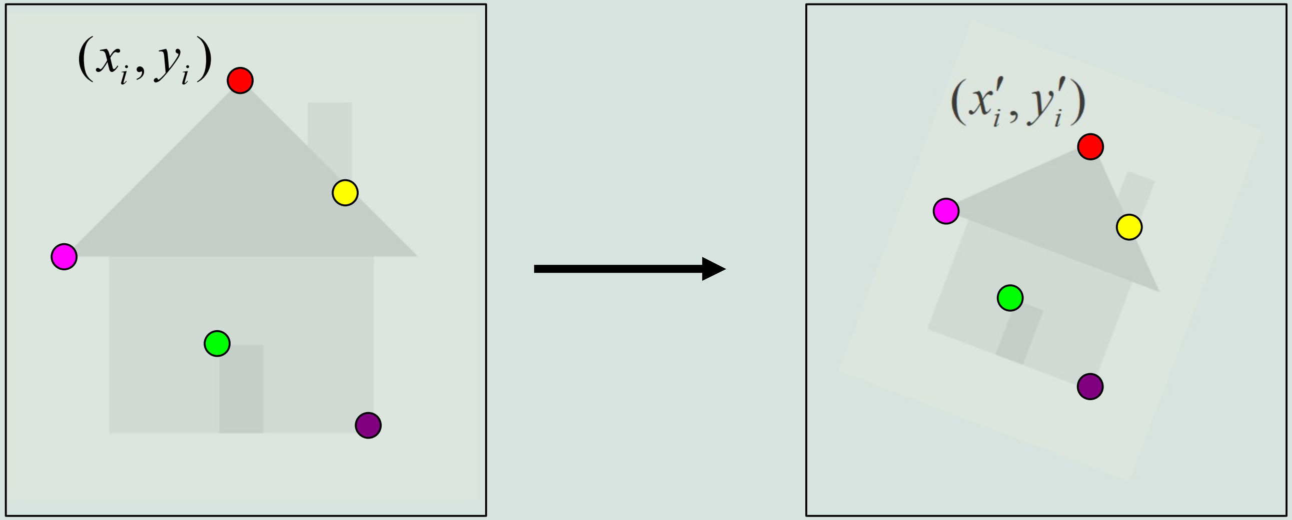
http://en.wikipedia.org/wiki/Barberpole_illusion

The Barber Pole Illusion

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

Alignment / Motion Warping

- "Alignment": Assuming we know the correspondences, how do we get the transformation?



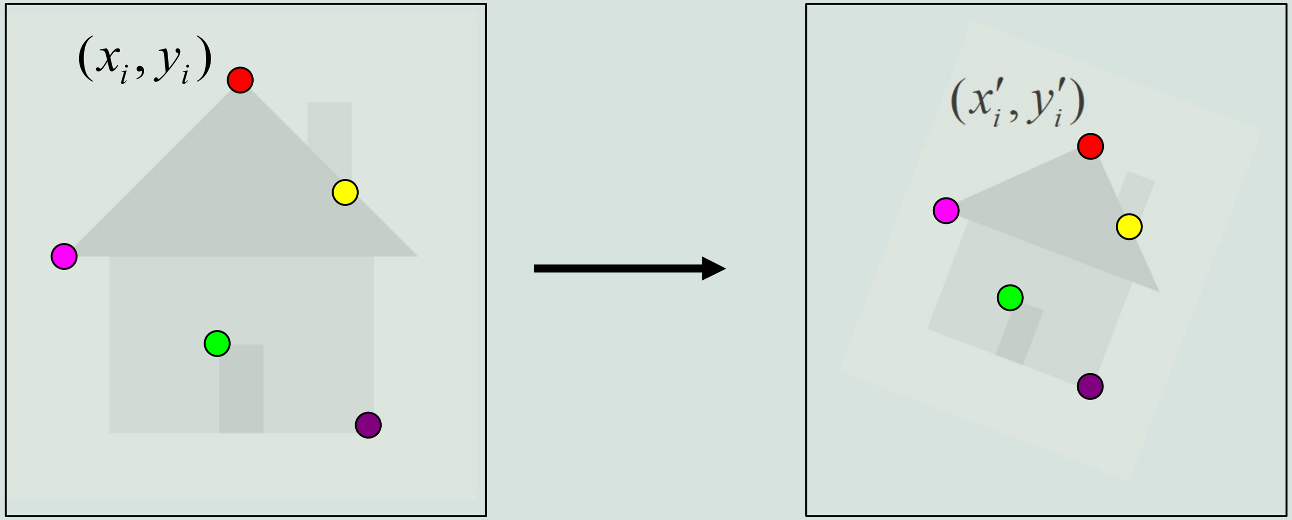
e.g., affine model in abs. coords…

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

- Expressed in terms of absolute coordinates of corresponding points…
- Generally presumed features separately detected in each frame

Flow, Parametric Motion

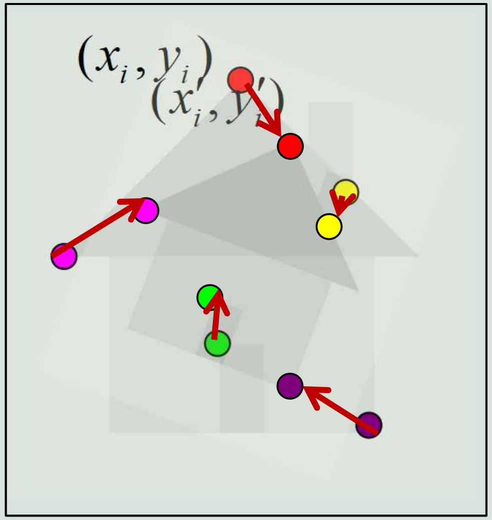
- Two views presumed in temporal sequence…**track** or analyze **spatio-temporal gradient**



- Sparse or dense in first frame
- Search in second frame
- Motion models expressed in terms of position change

Flow, Parametric Motion

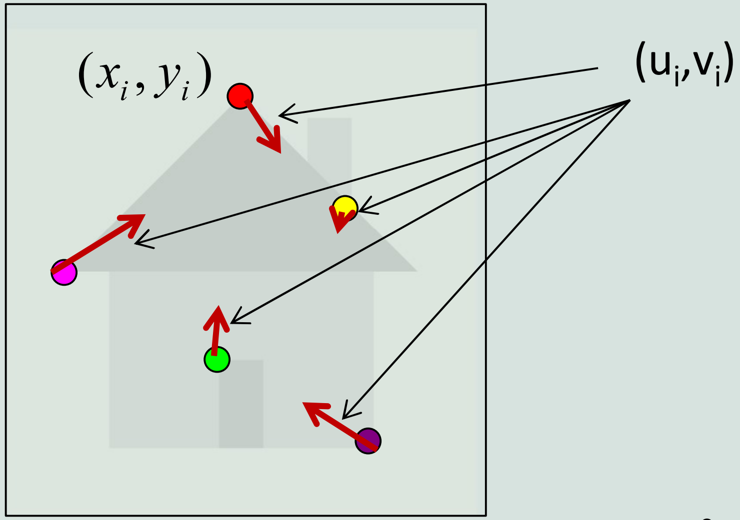
- Two views presumed in temporal sequence…**track** or analyze **spatio-temporal gradient**



- Sparse or dense in first frame
- Search in second frame
- Motion models expressed in terms of position change

Flow, parametric motion

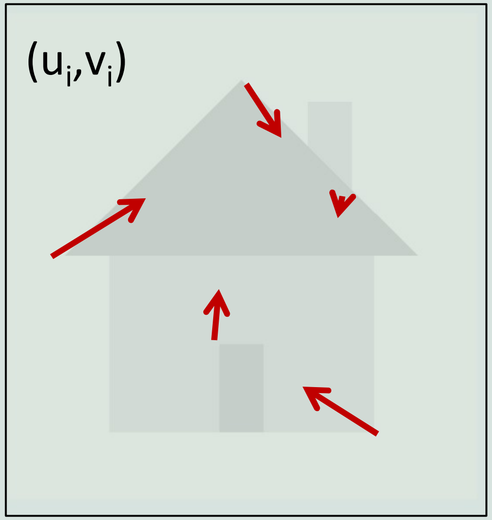
- Two views presumed in temporal sequence…**track** or analyze **spatio-temporal gradient**



- Sparse or dense in first frame
- Search in second frame
- Motion models expressed in terms of position change

Flow, parametric motion

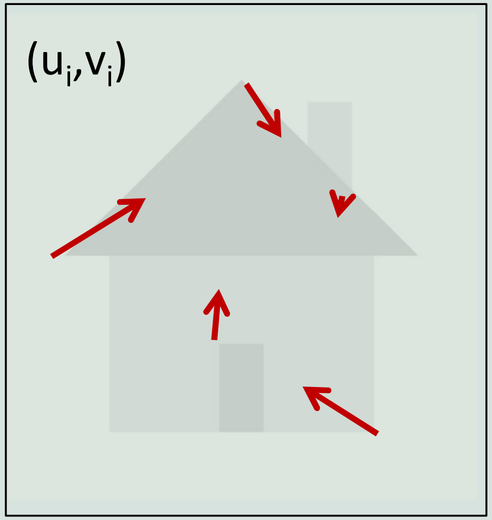
- Two views presumed in temporal sequence…**track** or analyze **spatio-temporal gradient**



- Sparse or dense in first frame
- Search in second frame
- Motion models expressed in terms of position change

Flow, parametric motion

- Two views presumed in temporal sequence…**track** or analyze **spatio-temporal gradient**



Previous Alignment model:

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Now, Displacement model:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} a_2 & a_3 \\ a_5 & a_6 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} a_1 \\ a_4 \end{bmatrix}$$

- Sparse or dense in first frame
- Search in second frame
- Motion models expressed in terms of position change

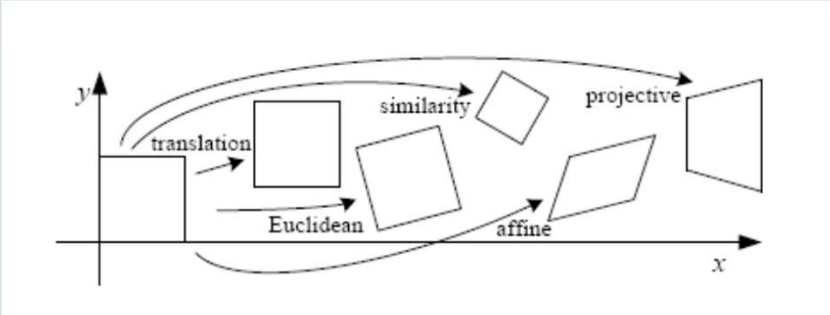
$$u(x,y) = a_1 + a_2 x + a_3 y$$
$$v(x,y) = a_4 + a_5 x + a_6 y$$

Global (Parametric) Motion Models

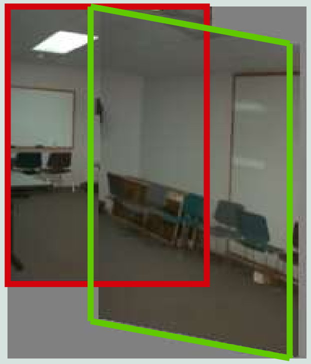
- ***2D Models:***
  - Affine
  - Quadratic
  - Planar projective transform (Homography)

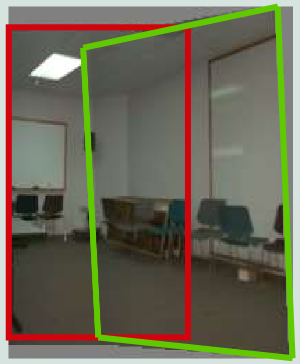
- ***3D Models:***
  - Instantaneous camera motion models
  - Homography + epipole
  - Plane + Parallax

Motion models



Translation | **Affine** | **Perspective** | **3D rotation**



2 unknowns | **6 unknowns** | **8 unknowns** | **3 unknowns**

Example: Affine Motion

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

- Substituting into the equation:

$$I_x u + I_y v + I_t \approx 0$$

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

Each pixel provides 1 linear constraint in 6 *global* unknowns

- Least Square Minimization (over all pixels):

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

Other 2D Motion Models

Quadratic – instantaneous approximation to planar motion

$$u = q_1 + q_2 x + q_3 y + q_7 x^2 + q_8 xy$$
$$v = q_4 + q_5 x + q_6 y + q_7 xy + q_8 y^2$$

Projective – exact planar motion

$$x' = \frac{h_1 + h_2 x + h_3 y}{h_7 + h_8 x + h_9 y}$$
$$y' = \frac{h_4 + h_5 x + h_6 y}{h_7 + h_8 x + h_9 y}$$

and

$$u = x' - x, \quad v = y' - y$$

3D Motion Models

Instantaneous camera motion:

Global parameters: $\boxed{\Omega_X, \Omega_Y, \Omega_Z, T_X, T_Y, T_Z}$

Local Parameter: $\boxed{Z(x, y)}$

$$\boxed{\begin{aligned} u &= -xy\Omega_X + (1+x^2)\Omega_Y - y\Omega_Z + (T_X - T_Z x)/Z \\ v &= -(1+y^2)\Omega_X + xy\Omega_Y - x\Omega_Z + (T_Y - T_Z x)/Z \end{aligned}}$$

Homography+Epipole

Global parameters: $\boxed{h_1, \ldots, h_9, t_1, t_2, t_3}$

Local Parameter: $\boxed{\gamma(x, y)}$

$$x' = \frac{h_1 x + h_2 y + h_3 + \gamma t_1}{h_7 x + h_8 y + h_9 + \gamma t_3}$$

$$y' = \frac{h_4 x + h_5 y + h_6 + \gamma t_1}{h_7 x + h_8 y + h_9 + \gamma t_3}$$

$$\text{and}: \quad u = x' - x, \quad v = y' - y$$

Residual Planar Parallax Motion

Global parameters: $\boxed{t_1, t_2, t_3}$

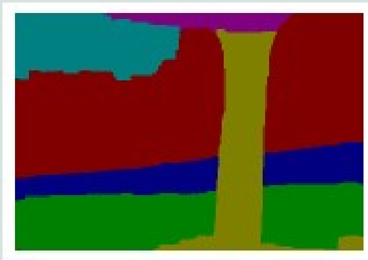
Local Parameter: $\boxed{\gamma(x, y)}$

$$u = x^w - x = \frac{\gamma}{1 + \gamma t_3}(t_3 x - t_1)$$

$$v = y^w - x = \frac{\gamma}{1 + \gamma t_3}(t_3 y - t_2)$$

Layered Motion

- Break image sequence up into “layers”:



=

- Describe each layer’s motion

Layered Motion

- Advantages:
  - can represent occlusions / disocclusions
  - each layer's motion can be smooth
  - video segmentation for semantic processing
- Difficulties:
  - how do we determine the correct number?
  - how do we assign pixels?
  - how do we model the motion?

Layers for Video Summarization

Frame 0

Frame 50

Frame 80

Background scene (players removed)

Complete synopsis of the video

Background Modeling (MPEG-4)

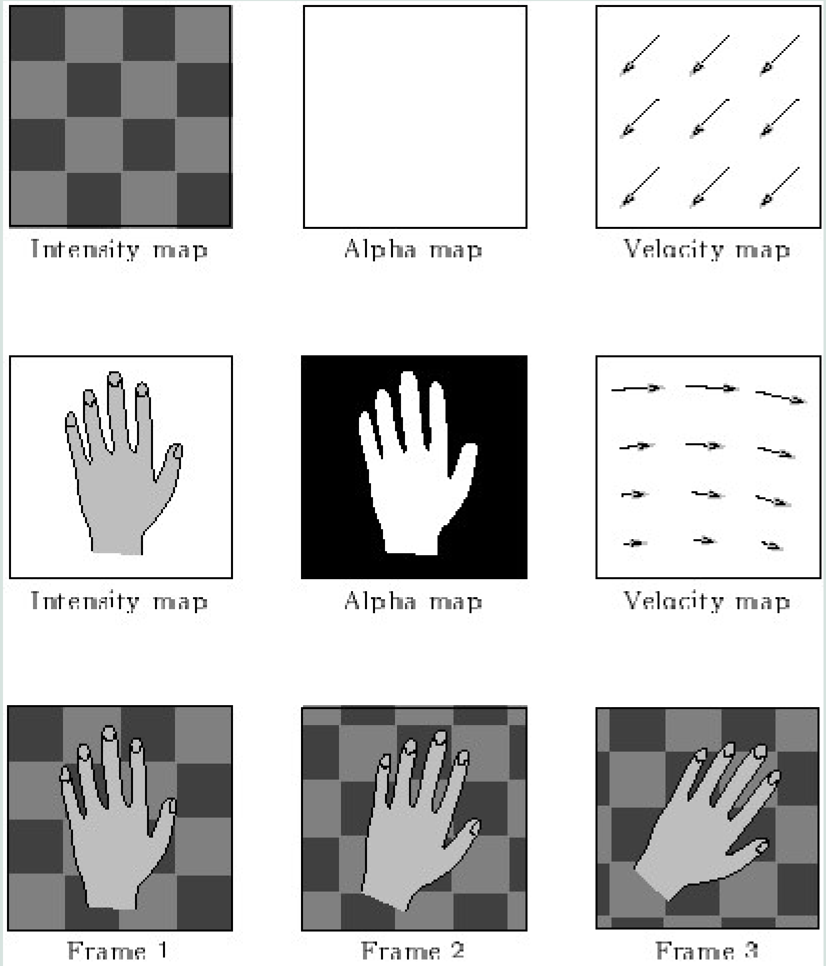
- Convert masked images into a background sprite for layered video coding




=

What Are Layers?

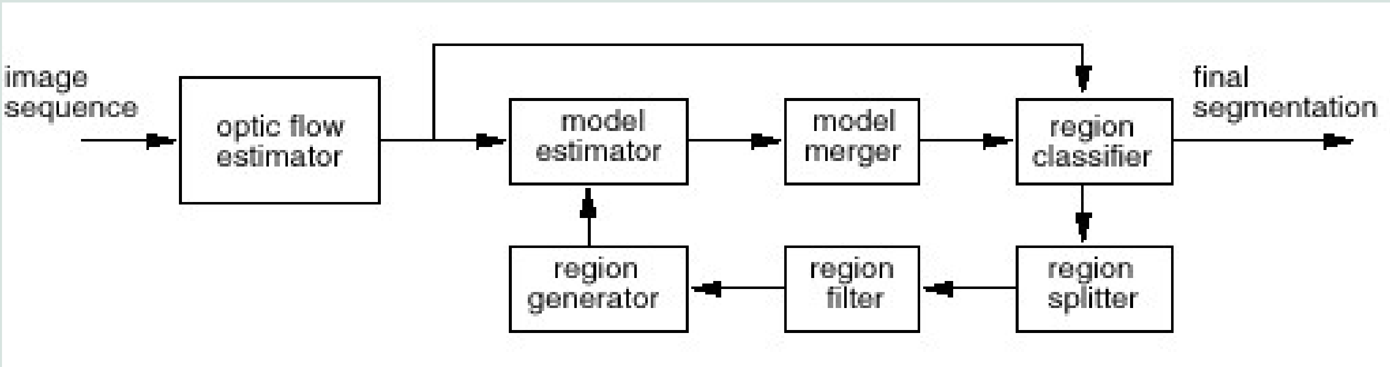
- intensities
- alphas
- velocities



[Wang & Adelson, 1994;
Darrell & Pentland 1991]

How to Estimate the Layers?

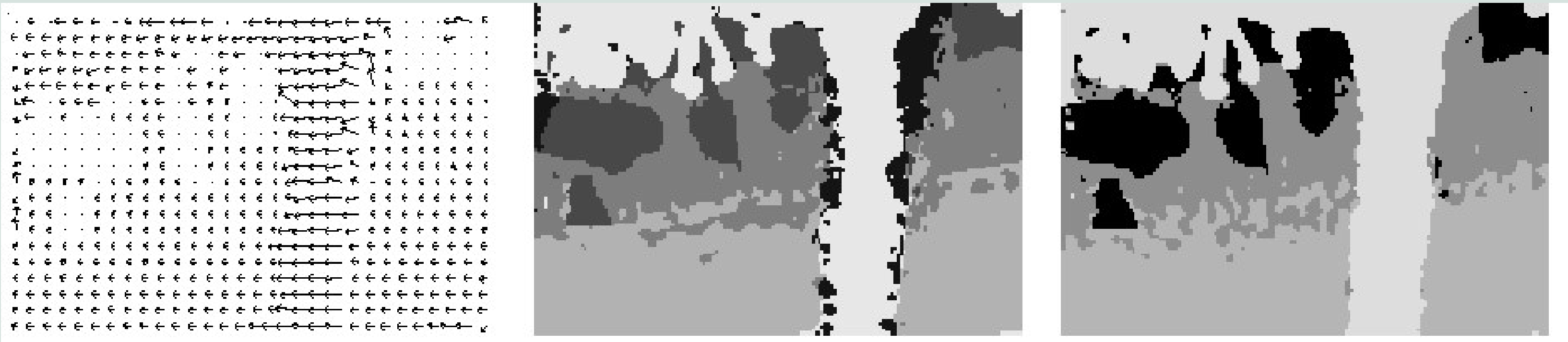
1. Compute coarse-to-fine flow
2. Estimate affine motion in blocks (regression)
3. Cluster with *k-means*
4. Assign pixels to best fitting affine region
5. Re-estimate affine motions in each region…

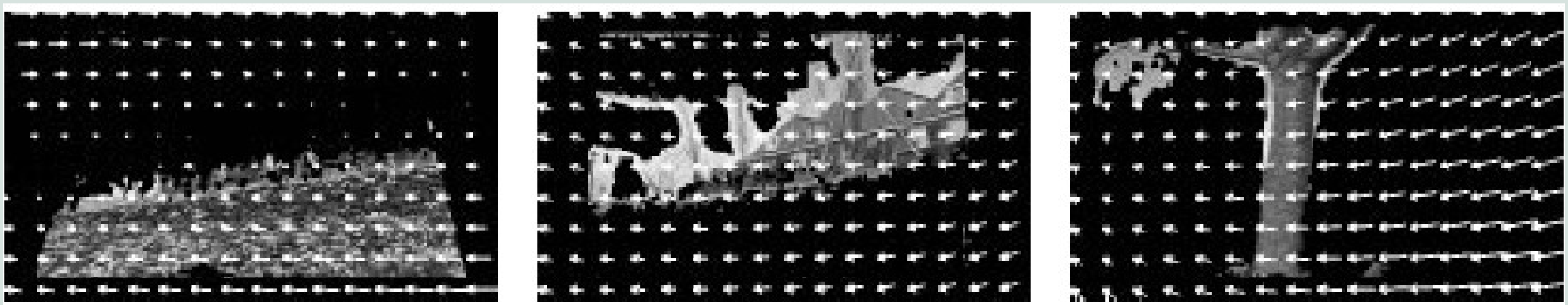


Layer Synthesis

- For each layer:
  - Stabilize the sequence with the affine motion
  - Compute median value at each pixel
  - Determine occlusion relationships

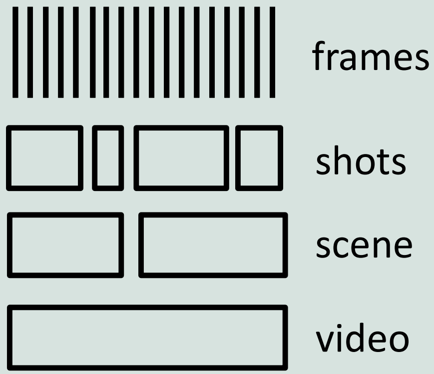
Results





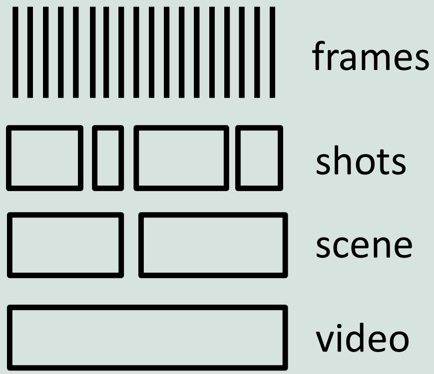
Partitioning of Video Data

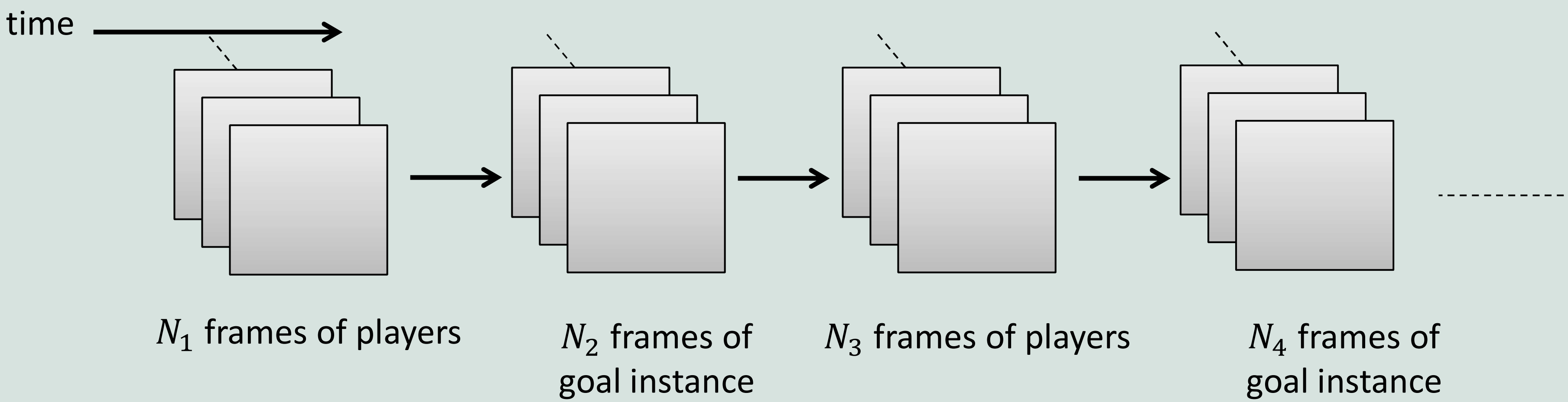
- Frames
  - Equally spaced images with a specific time resolution
- Shots
  - Frames that were recorded by a camera without changing its position or direction
- Scene
  - Semantically related group of shots
- Video
  - Unpartitioned image data over time



Key Frame Detection

- Extracts important and meaningful frames in a video sequence
- Determines «cut» images where similarity between frames is lost
- Small changes in the frames needs to be ignored.
- Also known as temporal segmentation
  - useful in video summarization
    - e.g., determining the goal instances in a football match





Key Frame Detection: Pixel Comparison

- For gray-level images average difference between consecutive frames

$$D(t, t+1) = \sum_{x=1}^{X} \sum_{y=1}^{Y} \frac{|I_t(x, y) - I_{t+1}(x, y)|}{X.Y}$$

- For color images average difference between consecutive frames

$$D(t, t+1) = \sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{c} \frac{|I_t(x, y, c) - I_{t+1}(x, y, c)|}{X.Y}$$

Key Frame Detection: Pixel Comparison

- Differences only above a threshold of T_1 can be counted

$$DP(t, t+1, x, y) = \begin{cases} 1 \quad if \quad |I_t(x, y) - I_{t+1}(x, y)| > T_1 \\ 0 \quad otherwise \end{cases}$$

- Key frame is then determined using a second threshold, T_2

$$D(t, t+1) = \sum_{x=1}^{X} \sum_{y=1}^{Y} \frac{DP(t, t+1, x, y)}{X.Y}, if D(t, t+1) > T_2$$

Key Frame Detection: Block-based Comparison I

- Block-wise evaluation may be effective for camera and object movements
- Difference likelihoods of λ_k only above a threshold of T_1 can be counted

$$DP(t, t+1, k) = \begin{cases} 1 & if \quad \lambda_k > T_1 \\ 0 & otherwise \end{cases}$$

$$\lambda_k = \frac{\left[\frac{{\sigma^2}_{k,t} + {\sigma^2}_{k,t+1}}{2} + \frac{\left(\mu_{k,t} - \mu_{k,t+1}\right)^2}{2}\right]^2}{{\sigma^2}_{k,t} . {\sigma^2}_{k,t+1}}$$

  - $\mu_{k,t}$ is average value for block k at time t
  - ${\sigma^2}_{k,t}$ is variance of block k at time t

Key Frame Detection: Block-based Comparison II

- Key frame is then determined using a second threshold, T_2
  - c_k is coefficient for block k.
  - If central blocks are considered more significant c_k can be higher for central blocks.

$$D(t, t+1) = \sum_{k=1}^{B:\#\ of\ blocks} c_k . DP(t, t+1, k)\,,$$

$$if \quad D(t, t+1) > T_2$$

Key Frame Detection: Histogram Comparison

- Global histogram comparison
  - H is normalized between 0 and 1.
  - h is number of gray levels in general.

$$D(t, t+1) = \sum_{j=0}^{h-1} |H_t(j) - H_{t+1}(j)|,$$

- Key frame is found when total histogram difference is above T_1

$$if \quad D(t, t+1) > T_1$$

Key Frame Detection: Histogram Comparison

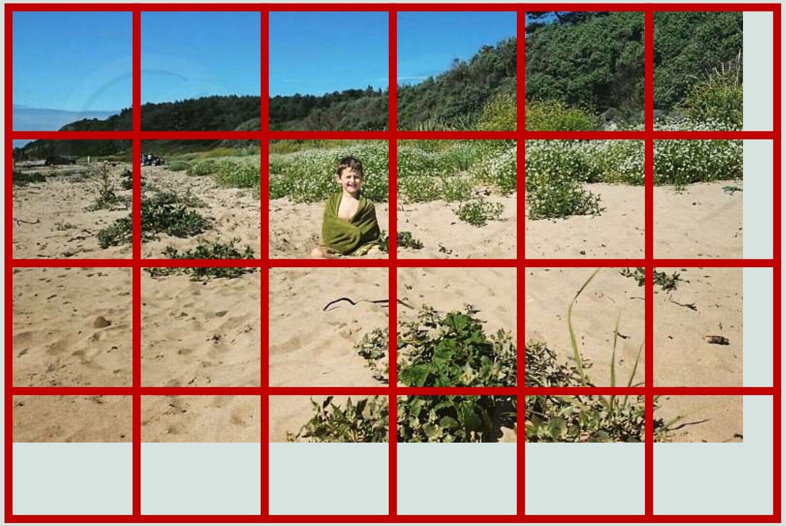
- Local histogram comparison
  - used when local changes is of interest
  - H is normalized between 0 and 1
  - h is number of gray levels in general
  - B is number of blocks in the frame

$$DP(t, t+1, k) = \sum_{j=0}^{h-1} |H_t(j,k) - H_{t+1}(j,k)|$$

$$D(t, t+1) = \sum_{k=1}^{B} DP(t, t+1, k)$$

- Key frame is found when total local histogram difference is above T_1

$$if \quad D(t, t+1) > T_1$$

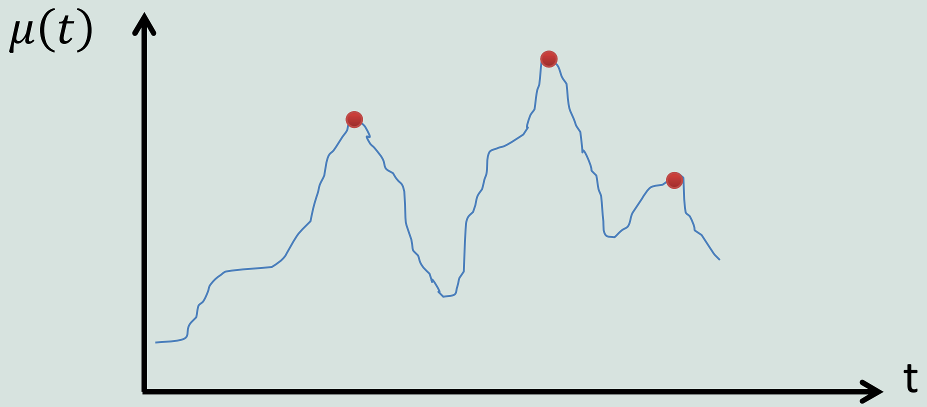


Key Frame Detection: Motion-based Comparison

- Comparison can be alternatively performed using
  - Edge information
  - Flow information
- Flow information is already measured between t and t + 1

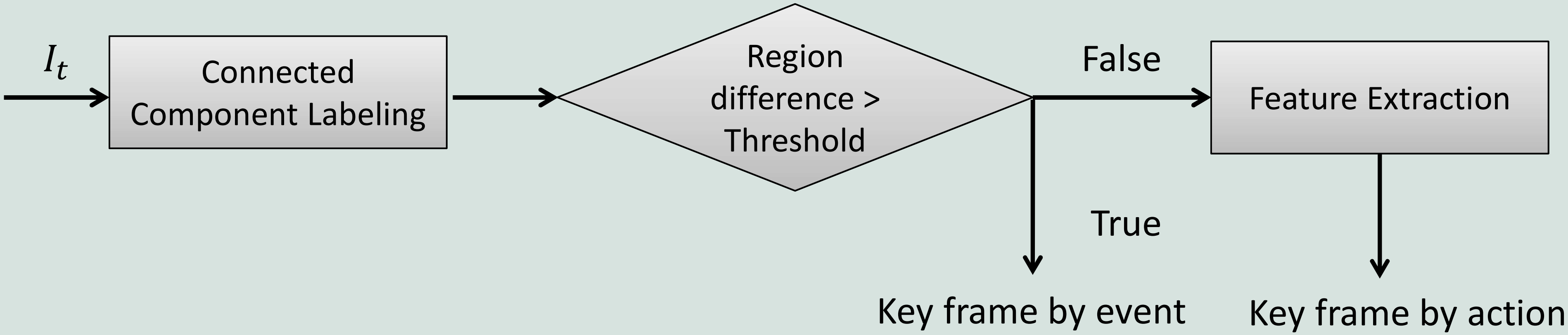
$$\mu(t) = \sum_{x=1}^{X} \sum_{y=1}^{Y} |F_{horizontal}(x, y, t) + F_{vertical}(x, y, t)|$$

- Then local maxima in $\mu(t)$ is searched



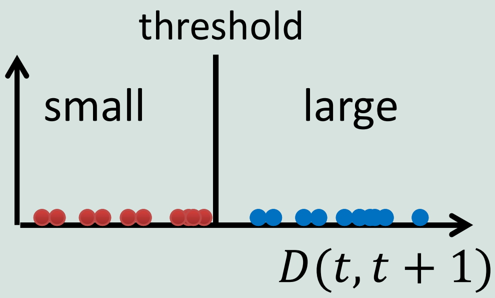
Key Frame Detection: Object-based Comparison

- Different methods can be combined to detect key frames by action or key frames by event



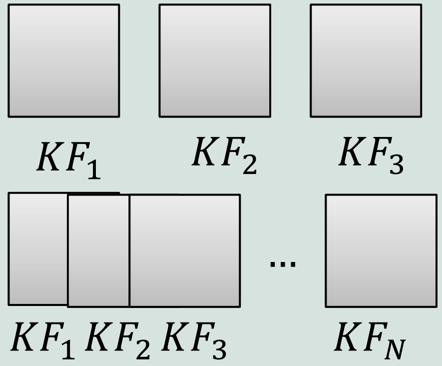
Key Frame Detection: Computing Threshold Values

- Random or guessed threshold values may lead to problems for different videos
- An algorithm to determine threshold values
  1. Obtain a set of differences $D(t, t+1)$ between frames
  2. Group the differences into two as large and small
  3. Apply k-means to find the threshold
     - Choose two centers in the set of differences
     - Cluster all differences (samples) according to the chosen centers
     - Re-compute cluster centers
     - Continue a number of times or until no difference (sample) changes its cluster
     - Threshold is the smallest value of the cluster with larger center



μ_1	μ_2	
20	800	
40	500	
50	$\boxed{400}$	threshold
60	700	
70	650	

Key Frame Detection: Displaying Key Frames



- Key frames are a series of still images
  - Each key frame is a frame
- Still images can be montaged for synopsis for almost static camera views
- Montage of still images
  1. Choose one key frame as reference
  2. Compute transformations of all key frames to reference
  3. Warp the key frames to reference
  4. Use most frequent or mean intensity value of all warped images for each coordinate
- The resulting image is known as panaroma view or synopsis mosaics