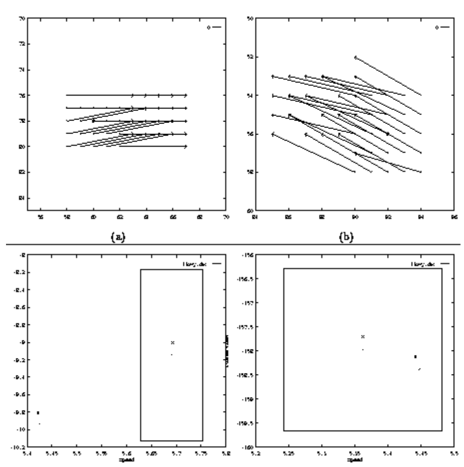
Lecture-16

Computing Motion Trajectories

http://www.cs.ucf.edu/~vision/papers/shah/93/SRT93.pdf

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)

$$\mathbf{Z}(k) = \mathbf{Z}(k-1) + K(k)(Y(k) - H(k)\mathbf{Z}(k-1))$$

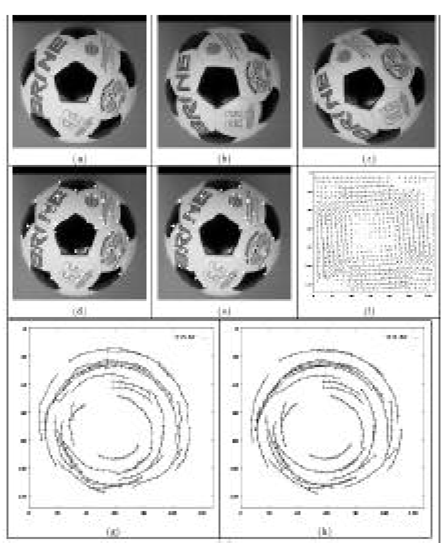
$$K(k) = P(k-1)H^T(k)\ (W^T + H\ \ P(k-1)H^T(k))^{-1}$$

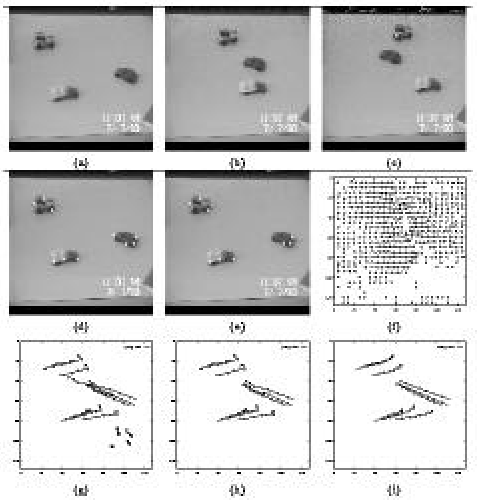
$$P(k) = (I - K(k)H(k))P(k-1)$$

$$Y(k) = -f^T(\mathbf{Z}(\mathbf{k}-\mathbf{1}), \mathbf{y}) + \frac{\partial f}{\partial \mathbf{Z}}\mathbf{Z}(k-1)$$

$$H(k) = \frac{\partial f}{\partial \mathbf{Z}}$$

$$W = \frac{\partial f}{\partial \mathbf{y}} \mathrm{A}^T \frac{\partial f}{\partial \mathbf{y}}^T$$





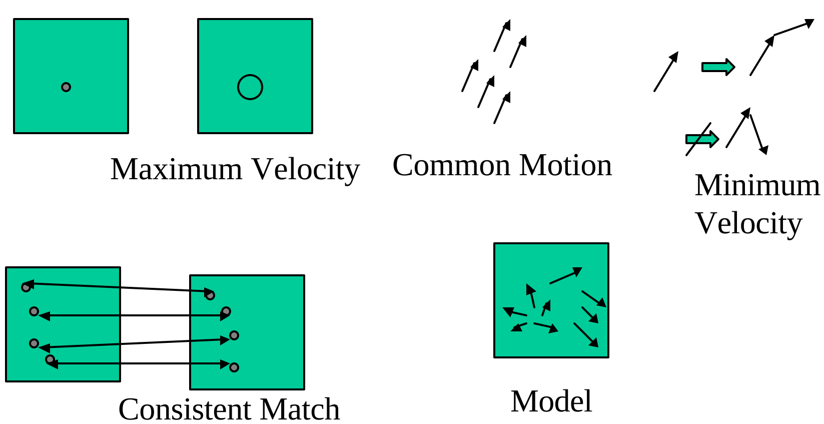
Point Correspondence

- Given *n* video frames taken at different time instants and *m* points in each frame, the motion correspondence problem deals with a mapping of a point in one frame to another point in the second frame such that no two points map onto the same point

Key Points

- Constraints→ Cost Function
- Algorithm→ Minimize the cost function

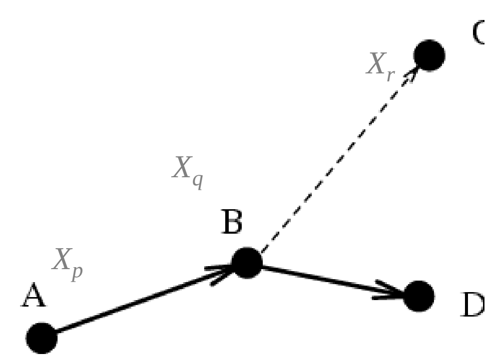
Constraints



Proximal Uniformity Constraint

- Most objects in the real world follow smooth paths and cover small distance in a small time.
  - Given a location of point in a frame, its location in the next fame lies in the proximity of its previous location.
  - The resulting trajectories are smooth and uniform.

Proximal Uniformity Constraint



Proximal Uniformity Constraint

Establish correspondence by minimizing:

$$d(X_p^{k-1}, X_q^k, X_r^{k+1}) = \frac{\| \overline{X_p^{k-1} X_q^k} - \overline{X_q^k X_r^{k+1}} \|}{\sum_{x=1}^{m}\sum_{z=1}^{m} | \overline{X_x^{k-1} X_y^k} - \overline{X_y^k X_z^{k+1}} \|} + \frac{\| \overline{X_q^k X_r^{k+1}} \|}{\sum_{x=1}^{m}\sum_{z=1}^{m} \| \overline{X_y^k X_z^{k+1}} \|}$$

Initial correspondence is known, for each x in the denominator of the first term y is known.

Greedy Algorithm

- For k=2 to n-1 do
- Construct M, an mxm matrix, with the points from k-th frame along the rows and points from (k+1)-th frame along the columns. Let

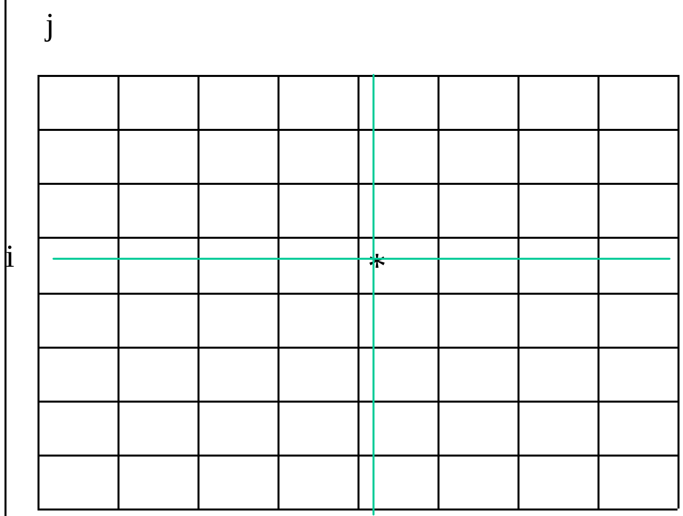
$$M[i, j] = d(X_p^{k-1}, X_q^k, X_r^{k+1})$$

when

- for a=1 to m do
  - Identify the minimum element $[i, l_i]$ in each row i of M
  - Compute *priority matrix*, B, such that $B[i, l_i] = \sum_{j=1, j\neq l_i}^{m} M[i, j] + \sum_{k=1, k\neq i}^{m} M[k, l_i]$ for each *i*.
  - Select $[i, l_i]$ pair with highest *priority* value and make $f^k(i) = l_i$
  - Mask row *i* and column l_i from M.

http://www.cs.ucf.edu/~vision/papers/PAPER3.PDF

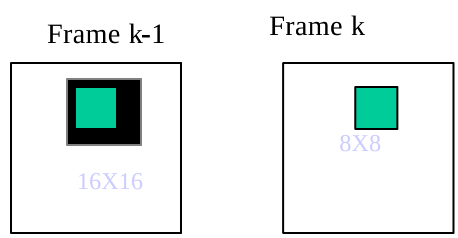
M[i,j]



$$M = \begin{bmatrix} .6 & .3 \\ .7 & .2 \end{bmatrix}$$

$$B = \begin{bmatrix} .8 \\ 1 \end{bmatrix}$$

Block Matching



Origin is at bottom right corner

Block Matching

- For each 8X8 block, centered around pixel (x,y) in frame k, B_k
  - Obtain 16X16 block in frame k-1, enclosing (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)

$$(u(x,y),v(x,y)) = \arg\min_{\substack{u=0\ldots-8\\ v=0\ldots8}} \sum_{i=0}^{-7}\sum_{j=0}^{7}\left(f_k(x+i,y+j)-f_{k-1}(x+i+u,y+j+v)\right)^2$$

Minimum Absolute Difference (MAD)

$$(u(x,y), v(x,y)) = \arg\min_{\substack{u=0\ldots-8 \\ v=0\ldots8}} \sum_{i=0}^{-7}\sum_{j=0}^{7} \left| \left(f_k(x+i, y+j) - f_{k-1}(x+i+u, y+j+v) \right) \right|$$

Maximum Matching Pixel Count (MPC)

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

$$(u(x,y), v(x,y)) = \arg\max_{\substack{u=0\ldots-8 \\ v=0\ldots8}} \sum_{i=0}^{-7}\sum_{j=0}^{7} T(x+i, y+j; u,v)$$

Cross Correlation

$$(u,v) = \arg\max_{\substack{u=0\ldots-8\\ v=0\ldots8}} \sum_{i=0}^{-7}\sum_{j=0}^{7}\left(f_k(x+i,y+j)).(f_{k-1}(x+i+u,y+j+v)\right)$$

Normalized Correlation

$$(u,v) = \arg\max_{\substack{u=0\ldots-8\\ v=0\ldots8}} \frac{\sum_{i=0}^{j=-7}\sum_{j=0}^{7}\left(f_k(x+i,y+j)).(f_{k-1}(x+i+u,y+j+v)\right)}{\sqrt{\sum_{i=0}^{-7}\sum_{j=0}^{7} f_{k-1}(x+i+u,y+j+v).f_{k-1}(x+i+u,y+j+v)}}$$

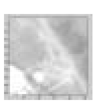
Mutual Correlation

$$(u,v) = \arg\max_{\substack{u=0\ldots-8 \\ v=0\ldots 8}} \frac{1}{64\, s_1 s_2} \sum_{i=0}^{-7} \sum_{j=0}^{7} \left(f_k(x+i, y+j) - m_1 \right).\left(f_{k-1}(x+i+u,\, y+j+v) - m_2 \right)$$

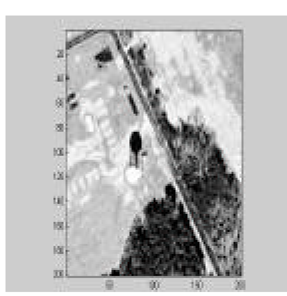
Sigma and mu are standard deviation and mean of patch-1 and patch-2 respectively

Correlation Surface

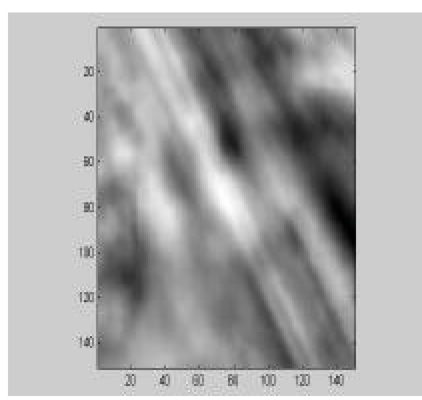
$$C(u,y) = \sum_{i=0}^{-7} \sum_{j=0}^{7} \left(f_k(x+i, y+j) \right).\left(f_{k-1}(x+i+u, y+j+v) \right)$$



mission



reference



Correlation surface

Correlation Using FFT

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

$$c(u,v)=f(x,y)\otimes g(x,y)$$

Fourier transform

convolution

$$C(w_1,w_2)=F(w_1,w_2).G(w_1,w_2)$$

Append smaller patch with zeros to make it equal to bigger patch
Find FFT of reference (*f(x,y)* and mission (*g(x,y)*) patches
Multiply two FFTs
Find inverse FFT of the product, this will give you a correlation surface

Phase Correlation

$$c(x,y)=f(x,y)\otimes g(x,y)$$

$$C(w_1,w_2)=F(w_1,w_2).G(w_1,w_2)$$

$$\tilde{C}(w_1,w_2)=\frac{F(w_1,w_2).G(w_1,w_2)}{|F(w_1,w_2).G(w_1,w_2)|}$$

Assume

$$f(u,v)=g(x+u,y+v)$$

Then

$$F(w_1,w_2)=G(w_1,w_2)e^{-(i(w_1u+w_2v))}$$

Now

$$\tilde{C}(w_1,w_2)=e^{-(i(w_1u+w_2v))}$$

$$c(x,y)=\boldsymbol{d}(x-u,y-v)$$

Issue with Correlation

- Patch Size
- Search Area
- How many peaks

Change Detection

Main Points

- Detect pixels which are changing due to motion of objects.
- Not necessarily measure motion (optical flow), only detect motion.
- A set of connected pixels which are changing may correspond to moving object.

Picture Difference

$$D_i(x,y) = \begin{cases} 1 & if \quad DP(x,y) > T \\ 0 & \ldots\ldots otherwise \end{cases}$$

$$DP(x,y) = | f_i(x,y) - f_{i-1}(x,y) |$$

$$DP(x,y) = \sum_{i=-m}^{m} \sum_{j=-m}^{m} | f_i(x+i, y+j) - f_{i-1}(x+i, y+j) |$$

$$DP(x,y) = \sum_{i=-m}^{m} \sum_{i=-m}^{m} \sum_{k=-m}^{m} | f_i(x+i, y+j) - f_{i+k}(x+i, y+j) |$$

Background Image

• The first image of a sequence without any moving objects, is background image.

• Median filter

$$B(x, y) = median(f_1(x, y), \ldots, f_n(x, y))$$

PFINDER

Pentland

Pfinder

- Segment a human from an arbitrary complex background.
- It only works for single person situations.
- All approaches based on background modeling work only for fixed cameras.

Algorithm

- Learn background model by watching 30 second video
- Detect moving object by measuring deviations from background model
- Segment moving blob into smaller blobs by minimizing covariance of a blob
- Predict position of a blob in the next frame using Kalman filter
- Assign each pixel in the new frame to a class with max likelihood.
- Update background and blob statistics

Learning Background Image

- Each pixel in the background has associated mean color value and a covariance matrix.
- The color distribution for each pixel is described by a Gaussian.
- YUV color space is used.

Detecting Moving Objects

- After background model has been learned, Pfinder watches for large deviations from the model.
- Deviations are measured in terms of Mahalanobis distance in color.
- If the distance is sufficient then the process of building a blob model is started.

Detecting Moving Objects

- For each of k blob in the image, log-likelihood is computed

$$d_k = -.5(y-m_k)^T K_k^{-1}(y-m_k) - .5\ln|K_k| - .5m\ln(2\pi)$$

- Log likelihood values are used to classify pixels

$$s(x,y) = \arg\max_k (d_k(x,y))$$

Updating

- The statistical model for the background is updated.

$$K^t = E[(y-m^t)(y-m^t)^T]$$
$$m^t = (1-a)m^{t-1} + ay$$

- The statistics of each blob (mean and covariance) are re-computed.