HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Computer Vision

Chapter 6: Motion detection and Tracking

Motion Detection and Tracking

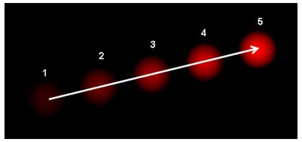
- Motion detection
- Approaches of motion detection
- Moving object tracking

Motion Detection

- Motion detection: Action of sensing physical movement in a give area
- Motion can be detected by measuring change in speed or vector of an object
- Goals of motion detection
  - Identify moving objects
  - Detection of unusual activity patterns
  - Computing trajectories of moving objects
- Applications of motion detection
  - Indoor/outdoor security
  - Real time crime detection
  - Traffic monitoring
  - Many intelligent video analysis systems are based on motion detection

Approaches to Motion Detection

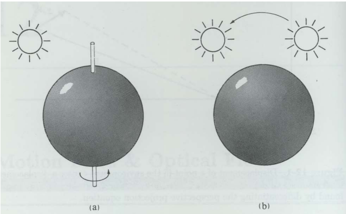
- Optical Flow
  - Compute motion within region or the frame as a whole



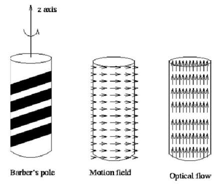
- Change detection
  - Detect objects within a scene
  - Track object across a number of frames

Optical flow vs. motion field

- Optical flow does not always correspond to motion field



Barber pole illusion



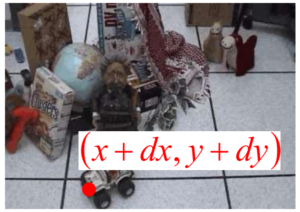
- Optical flow is an approximation of the motion field
- The error is small at points with high spatial gradient under some simplifying assumptions

Estimating optical flow

- Assume the image intensity I is constant

Time = t

Time = $t+dt$

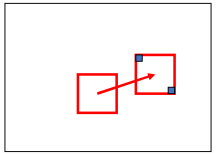


$$I_0(x, y, t) \approx I_1(x+\delta x, y+\delta y, t+\delta t)$$

Optical flow constraint

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

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t} = 0, \text{ and let } \delta t \to 0$$
$$I_x u + I_y v + I_t = 0$$
$$I_x u + I_y v = -I_t,$$
$$[I_x \quad I_y]\begin{bmatrix} u \\ v \end{bmatrix} = -I_t, \nabla I^T \mathbf{u} = \mathbf{b}; A\mathbf{u} = \mathbf{b}$$
$$E(u, v) = (I_x u + I_y v + I_t)^2$$



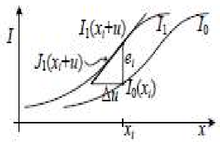
Lucas-Kanade algorithm

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

$$\|A\mathbf{u} - \mathbf{b}\|^2$$
$$\mathbf{u} = (A^T A)^{-1} A^T \mathbf{b}$$
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$\left(\sum \nabla I \cdot \nabla I^T\right) \vec{u} = -\sum \nabla I \cdot I_t$$

Matrix form

$$E(\mathbf{u}+\Delta\mathbf{u}) = \sum_i [I_1(\mathbf{x}_i+\mathbf{u}+\Delta\mathbf{u}) - I_o(\mathbf{x}_i)]^2$$
$$\approx \sum_i [I_1(\mathbf{x}_i+\mathbf{u}) + \mathbf{J}_1(\mathbf{x}_i+\mathbf{u})\Delta\mathbf{u} - I_o(\mathbf{x}_i)]^2$$
$$= \sum_i [\mathbf{J}_1(\mathbf{x}_i+\mathbf{u})\Delta\mathbf{u} + e_i]^2$$
$$\mathbf{J}_1(\mathbf{x}_i+\mathbf{u}) \approx \nabla I_1(\mathbf{x}_i+\mathbf{u}) = \left(\frac{\partial I_1}{\partial x}, \frac{\partial I_1}{\partial y}\right)(\mathbf{x}_i+\mathbf{u})$$
$$e_i = I_1(\mathbf{x}_i+\mathbf{u}) - I_o(\mathbf{x}_i)$$



$$y = f(\mathbf{x}+\Delta\mathbf{x}) \approx f(\mathbf{x}) + J(\mathbf{x})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}} H(\mathbf{x})\Delta\mathbf{x}$$

Matrix form

$$A\Delta\mathbf{u} = \mathbf{b}$$
$$A = \sum_i \mathbf{J}_1^T(\mathbf{x}_i+\mathbf{u})\mathbf{J}_1(\mathbf{x}_i+\mathbf{u})$$
$$\mathbf{b} = -\sum_i e_i \mathbf{J}_1(x_i+u)$$
$$A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}, \quad \mathbf{b} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$\mathbf{J}_1(\mathbf{x}_i+\mathbf{u}) \approx \nabla I_1(\mathbf{x}_i+\mathbf{u}) = \left(\frac{\partial I_1}{\partial x}, \frac{\partial I_1}{\partial y}\right)(\mathbf{x}_i+\mathbf{u})$$
$$e_i = I_1(\mathbf{x}_i+\mathbf{u}) - I_o(\mathbf{x}_i)$$
$$\mathbf{J}_1(\mathbf{x}_i+\mathbf{u}) \approx \mathbf{J}_0(\mathbf{x}_i)$$

Computing gradients in X-Y-T

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

likewise for I_y and I_t

The aperture problem

$$\text{Let } A = \sum \nabla I \cdot \nabla I^T, \quad \text{and} \quad \mathbf{b} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

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

Error functions

- Robust error function

$$E(\mathbf{u}) = \sum_i \rho(I(\mathbf{x}_i + \mathbf{u}) - I(\mathbf{x})), \quad \rho(x) = \frac{x^2}{1 + x^2/a^2}$$

- Spatially varying weights

$$E(\mathbf{u}) = \sum_i w_0(\mathbf{x}_i) w_1(\mathbf{x}_i + \mathbf{u}) [I(\mathbf{x}_i + \mathbf{u}) - I(\mathbf{x}_i)]^2$$

- Bias and gain: images taken with different exposure

$$I(\mathbf{x} + \mathbf{u}) = (1 + \alpha) I(\mathbf{x}) + \beta, \alpha \text{ is the gain and } \beta \text{ is the bias}$$

$$E(\mathbf{u}) = \sum_i [I(\mathbf{x}_i + \mathbf{u}) - (1 + \alpha) I(\mathbf{x}_i) - \beta]^2$$

- Correlation (and normalized cross correlation)

Horn-Schunck algorithm

- Global method with smoothness constraint to solve aperture problem
- Minimize a global energy functional with calculus of variations

$$E = \int \left((I_x u + I_y v + I_t)^2 + \alpha^2 (|\nabla u|^2 + |\nabla v|^2) \right) dx dy$$

$$\frac{\partial L}{\partial u} - \frac{\partial}{\partial x} \frac{\partial L}{\partial u_x} - \frac{\partial}{\partial y} \frac{\partial L}{\partial u_y} = 0$$

$$\frac{\partial L}{\partial v} - \frac{\partial}{\partial x} \frac{\partial L}{\partial v_x} - \frac{\partial}{\partial y} \frac{\partial L}{\partial v_y} = 0$$

where L is the integrand of the energy function

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Horn-Schunck algorithm

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta u = 0$$

$$I_y(I_x u + I_y v + I_t) - \alpha^2 \Delta v = 0$$

where $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplace operator

$$\Delta u(x, y) = \bar{u}(x, y) - u(x, y)$$

$$(I_x^2 + \alpha^2) u + I_x I_y v = \alpha^2 \bar{u} - I_x I_t$$

$$I_x I_y u + (I_y^2 + \alpha^2) v = \alpha^2 \bar{v} - I_y I_t$$

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Horn-Schunck algorithm

- Iterative scheme

$$u^{k+1} = \bar{u}^k - \frac{I_x(I_x \bar{u}^k + I_y \bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

$$v^{k+1} = \bar{v}^k - \frac{I_y(I_x \bar{u}^k + I_y \bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

- Yields high density flow
- Fill in missing information in the homogenous regions
- More sensitive to noise than local methods

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Background Subtraction

- Uses a reference background image for comparison purposes.
- Current image (containing target object) is compared to reference image pixel by pixel.
- Places where there are differences are detected and classified as moving objects.

Motivation: simple difference of two images shows moving objects

a. Original scene

b. Same scene later

Subtraction of scene a from scene b

Subtracted image with threshold of 100

Static Scene Object Detection and Tracking

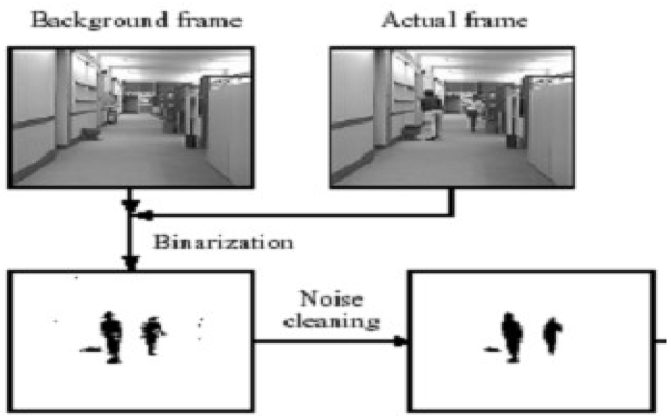
- Model the background and subtract to obtain object mask
- Filter to remove noise
- Group adjacent pixels to obtain objects
- Track objects between frames to develop trajectories

Approaches to Background Modeling

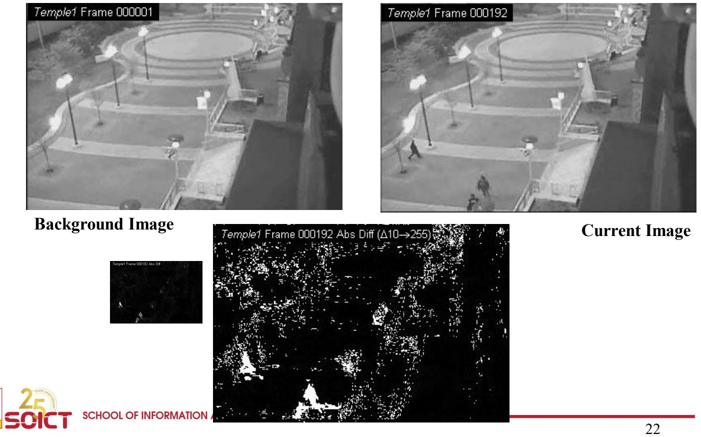
- Background Subtraction
- Statistical Methods
  (e.g., Gaussian Mixture Model, Stauffer and Grimson 2000)

Background Subtraction:

1. Construct a background image B as average of few images
2. For each actual frame I, classify individual pixels as foreground if |B-I| > T (threshold)
3. Clean noisy pixels



Background Subtraction



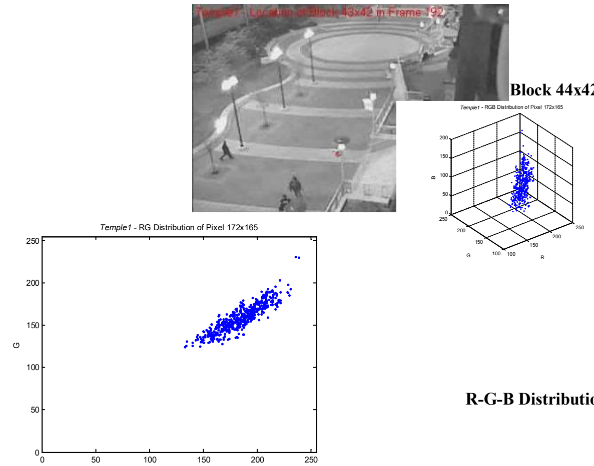
Background Image

Current Image

Statistical Methods

- Pixel statistics: average and standard deviation of color and gray level values
- Gaussian Mixture Model
  - Model the color values of a particular pixel as a mixture of Gaussians
  - Multiple adaptive Gaussians are necessary to cope with acquisition noise, lighting changes, etc.
  - Pixel values that do not fit the background distributions (Mahalanobis distance) are considered foreground

Gaussian Mixture Model



Block 44x42 Pixel 172x165

R-G-B Distribution

R-G Distribution

Detection of Moving Objects Based on Local Variation

For each block location (x,y) in the video plane

- Consider texture vectors in a symmetric window *[t-W, t+W]* at time *t*
- Compute the covariance matrix
- ***Motion measure*** is defined as the largest eigenvalue of the covariance matrix

Dynamic Distribution Learning and Outlier Detection

(1) $$\frac{f(t) - mean(t-1)}{std(t-1)} > C_1$$ **Detect Outlier**

(2) $$\frac{f(t) - mean(t-1)}{std(t-1)} < C_2$$ **Switch to a nominal state**

(3) $$mean(t) = u \cdot mean(t-1) + (1-u) \cdot f(t)$$

(4) $$std(t) = \sqrt{\sigma^2(t)}$$

Update the estimates of mean and standard deviation only when the outliers are not detected

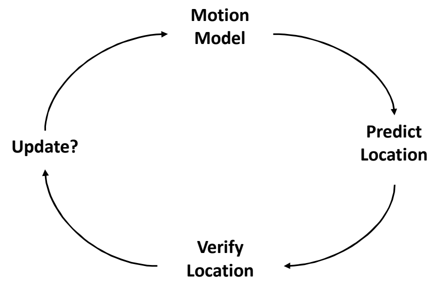
(5) $$\sigma^2(t) = u \cdot \sigma^2(t-1) + (1-u) \cdot (f(t) - mean(t-1))^2$$

Moving object tracking

- Model of object motion
- Kalman filter
- Mean Shift

Model based tracking

- Mathematical model of objects' motions:
  - position, velocity (speed, direction), acceleration
- Can predict objects' positions



Simple Motion Model

- Newton's laws

$$s(t) = s_0 + ut + \frac{1}{2} a t^2$$

- s = position
- u = velocity
- a = acceleration
  - all vector quantities
  - measured in image co-ordinates

Prediction

- Can predict position at time t knowing
  - Position
  - Velocity
  - Acceleration
- At t=0

Uncertainty

- If some error in a - Δa or u - Δu
- Then error in predicted position - Δs

$$\Delta s(t) = s_0 + \Delta u t + \frac{1}{2} \Delta a t^2$$

Verification

- Is the object at the predicted location?
  - Matching
    - How to decide if object is found
  - Search area
    - Where to look for object

Object Matching

- Compare
  - A small bitmap derived from the object vs.
  - Small regions of the image
- Matching?
  - Measure differences

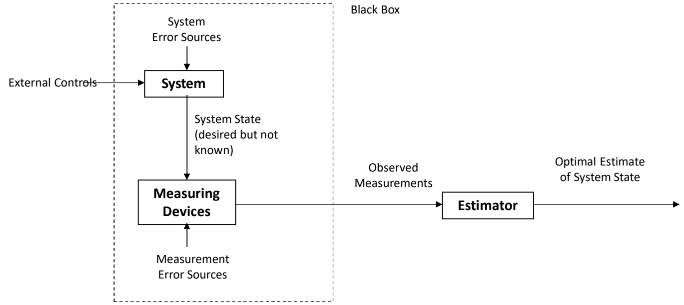
Search Area

- Uncertainty in knowledge of model parameters
  - Limited accuracy of measurement
  - Values might change between measurements
- Define an area in which object could be
  - Centred on predicted location, s ± Δs

Update the Model

- Is the object at the predicted location?
- Yes
  - No change to model
- No
  - Model needs updating
  - Kalman filter is a solution
    - Mathematically rigorous methods of using uncertain measurements to update a model

Kalman Filter



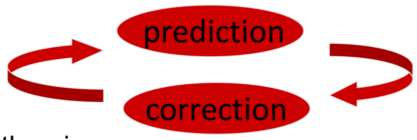
- Problem formulation
  - System state cannot be measured directly
  - Need to estimate “optimally” from measurementsh

Kalman Filter

- Recursive data processing algorithm
- Generates optimal estimate of desired quantities given the set of measurements
- Optimal?
  - For linear system and white Gaussian errors, Kalman filter is “best” estimate based on all previous measurements
  - For non-linear system optimality is ‘qualified’
- Recursive?
  - Doesn’t need to store all previous measurements and reprocess all data each time step

Kalman filter

- Matrix description of system state, model and measurement
- Progressive method



- Proper dealing with noise

Kalman filter

- Advantages of using KF in particle tracking
- Progressive method
  - No large matrices has to be inverted
- Proper dealing with system noise
- Track finding and track fitting
- Detection of outliers
- Merging track from different segments

Kalman filter assumptions

- Linear system
  - System parameters are linear function of parameters at some previous time
  - Measurements are linear function of parameters
- White Gaussian noise
  - White: uncorrelated in time
  - Gaussian: noise amplitude

$\Rightarrow$ KF is the optimal filter

Kalman filter

- Relates
  - Measurements **y**[k]
    - e.g. positions
  - System state **x**[k]
    - Position, velocity of object, etc
  - Observation matrix **H**[k]
    - Relates system state to measurements
  - Evolution matrix **A**[k]
    - Relates state of system between epochs
  - Measurement noise **n**[k]
  - Process noise **v**[k]

Mathematically

How do observations relate to model?

$$\mathbf{y}[k] = \mathbf{H}[k]\mathbf{x}[k] + \mathbf{n}[k]$$

$$\begin{bmatrix} x[k] \\ y[k] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x[k] \\ \dot{x}[k] \\ y[k] \\ \dot{y}[k] \end{bmatrix} + \mathbf{n}[k]$$

Prediction of System State

- Relates system states at epochs k and k+1

$$\hat{\mathbf{x}}[k+1 \mid k] = \mathbf{A}[k]\mathbf{x}[k \mid k] + \mathbf{v}[k]$$

$$\begin{bmatrix} \hat{x}[k+1 \mid k] \\ \hat{\dot{x}}[k+1 \mid k] \\ \hat{y}[k+1 \mid k] \\ \hat{\dot{y}}[k+1 \mid k] \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x[k \mid k] \\ \dot{x}[k \mid k] \\ y[k \mid k] \\ \dot{y}[k \mid k] \end{bmatrix} + \mathbf{v}[k]$$

Prediction of Observation

- From predicted system state, estimate what observation should occur:

$$\hat{\mathbf{y}}[k+1 \mid k] = \mathbf{H}[k]\hat{\mathbf{x}}[k+1 \mid k] + \mathbf{n}[k]$$

$$\begin{bmatrix} \hat{x}[k+1 \mid k] \\ \hat{y}[k+1 \mid k] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{x}[k+1 \mid k] \\ \hat{\dot{x}}[k+1 \mid k] \\ \hat{y}[k+1 \mid k] \\ \hat{\dot{y}}[k+1 \mid k] \end{bmatrix} + \mathbf{n}[k]$$

Updating the Model

- Predict/estimate a measurement $\hat{\mathbf{y}}[k+1|k]$
- Make a measurement $\mathbf{y}[k+1]$
- Predict state of model $\hat{\mathbf{x}}[k+1|k]$
- How does the new measurement contribute to updating the model?

Updating the Model

$$\hat{\mathbf{x}}[k+1|k+1] = \hat{\mathbf{x}}[k+1|k] + \mathbf{G}[k+1]\Delta\mathbf{y}[k+1|k]$$
$$\Delta\mathbf{y}[k+1|k] = \mathbf{y}[k+1] - \hat{\mathbf{y}}[k+1|k]$$

- G is Kalman Gain
  - Derived from A, H, v, n.

$$\mathbf{G} = \mathbf{C}[k|k]\mathbf{H}^T\left(\mathbf{H}\mathbf{C}[k|k]\mathbf{H}^T + \mathbf{n}\right)^{-1}$$
$$\mathbf{C}[k+1|k] = \mathbf{C}[k|k] - \mathbf{G}\mathbf{H}\mathbf{C}[k|k]$$
$$\mathbf{C} = \text{system covariance}$$

Example

- Tracking two corners of a minimum bounding box
- Matching using colour
- Image differencing to locate target

Condensation Tracking

- So far considered single motions
- What if movements change?
  - Bouncing ball
  - Human movements
- Use multiple models
  - plus a model selection mechanism

Selection and Tracking

- Occur simultaneously
- Maintain
  - A distribution of likely object positions plus weights
- Predict
  - Select N samples, predict locations
- Verify
  - Match predicted locations against image
  - Update distributions

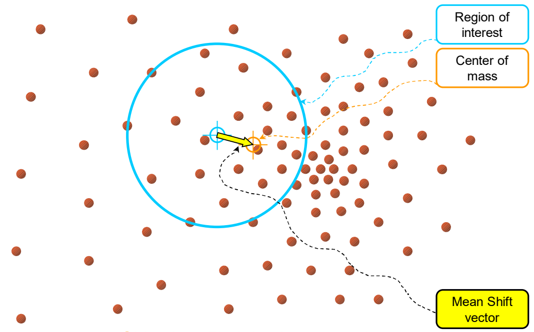
Tracking Using Hidden Markov Models

- Hidden Markov model describes
  - States occupied by a system
  - Possible transitions between states
  - Probabilities of transitions
  - How to recognise a transition

Mean-shift

- The mean-shift algorithm is an efficient approach to tracking objects whose appearance is defined by histograms. (not limited to only color)
- Motivation – to track non-rigid objects, (like a walking person), it is hard to specify an explicit 2D parametric motion model.
- Appearances of non-rigid objects can sometimes be modeled with color distributions

Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

Stolen from: *www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/**mean_shift**/**mean_shift**.ppt*

Mean Shift Vector

- Given:

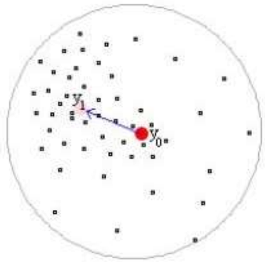
Data points and approximate location of the mean of this data:

- Task:

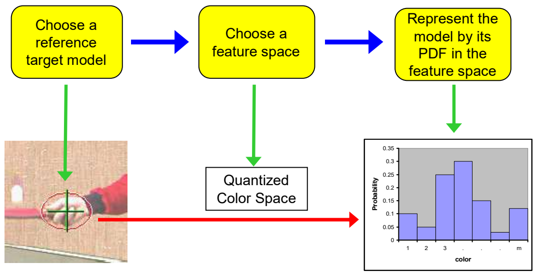
Estimate the exact location of the mean of the data by determining the shift vector from the initial mean.

Mean Shift Vector

$$M_h(\mathbf{y}) = \left[\frac{1}{n_x}\sum_{i=1}^{n_x}\mathbf{x}_i\right] - \mathbf{y}_0$$

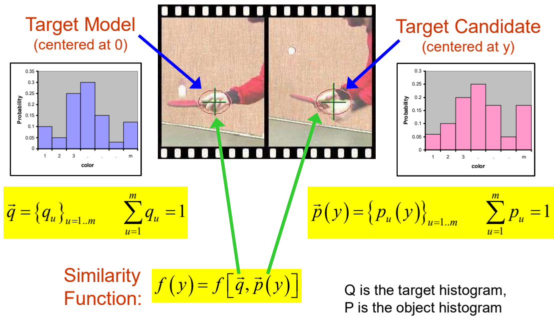


Mean-Shift Object Tracking
Target Representation



Stolen from: www.cs.wustl.edu/~pless/559/lectures/lecture22_tracking.ppt

Mean-Shift Object Tracking
PDF Representation

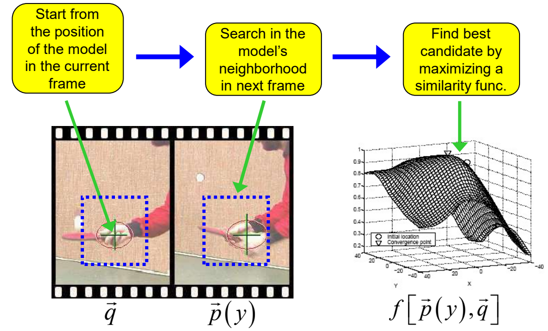


Stolen from: www.cs.wustl.edu/~pless/559/lectures/lecture22_tracking.ppt

Q is the target histogram,
P is the object histogram
(depends on location y)

Mean-Shift Object Tracking

Target Localization Algorithm



Stolen from: www.cs.wustl.edu/~pless/559/lectures/lecture22_tracking.ppt

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Mean Shift

- Mean-Shift in tracking task:
  - ➢ track the motion of a cluster of interesting features.
- 1. choose the feature distribution to represent an object (e.g., color + texture),
- 2. start the mean-shift window over the feature distribution generated by the object
- 3. finally compute the chosen feature distribution over the next video frame
  - Starting from the current window location, the mean-shift algorithm will find the new peak or mode of the feature distribution, which (presumably) is centered over the object that produced the color and texture in the first place.
  - In this way, the mean-shift window tracks the movement of the object frame by frame.

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict