Computer Vision
and
Machine Learning
(Motion and tracking)

Bhabatosh Chanda

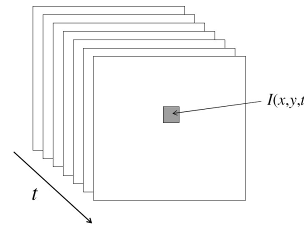
chanda@isical.ac.in

Jason Corso (jjcorso), web.eecs.umich.edu/~jjcorso/t/598F14

1

Video as frame sequence

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



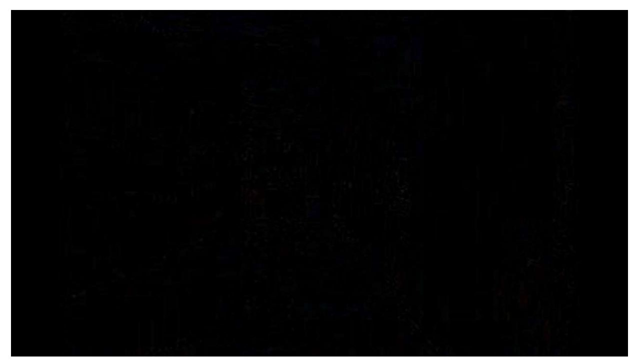
- We assume *a video clip* starts at *t=0* and frames are Δt apart.

2

Original video

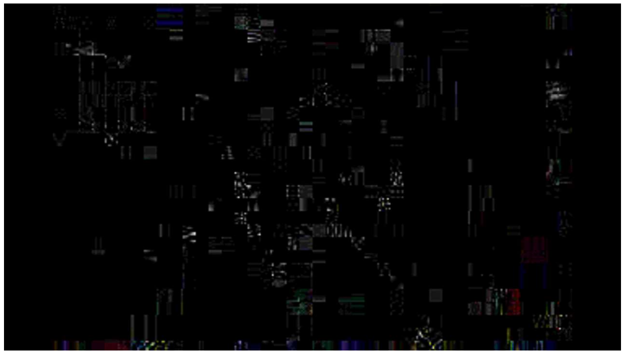
3

Subtracted from reference



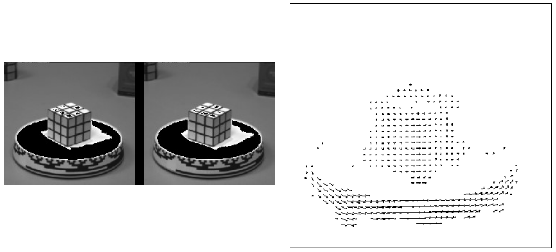
4

Subtracted pair-wise



Motion field

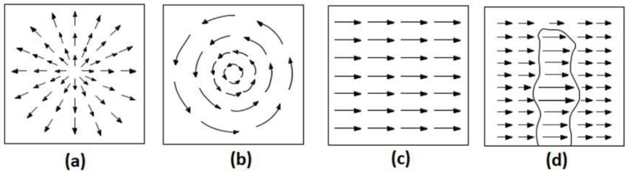
- The motion field is the projection of the 3D scene motion onto the 2D image plane.



Motion field

- The actual relative motion between objects in 3D scene and the camera is 3 dimensional.
  - Motion will have horizontal (X), vertical (Y), and depth (Z) components, in general.
- We can project these 3D motions onto 2D plane to get a two-dimensional *Motion field.*
- Motion field is the *projection of the actual 3D motion* in the scene onto the image plane.
- Motion Field is what we actually **need to estimate** for applications.

Motion field: examples



(a) Translation perpendicular to a image plane.

(b) Rotation about axis perpendicular to image plane.

(c) Translation parallel to image plane at constant distance.

(d) Nearer objects show larger translation parallel to surface.

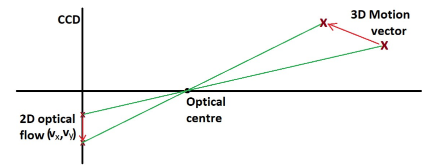
Motion field and Optical flow

- Optical flow is the apparent motion of brightness patterns between 2 frames in an image sequence
  - Why does brightness pattern change between frames?
- Assuming that illumination does not change:
  - changes are due to the RELATIVE MOTION between the scene and the camera and there are 3 possibilities:
    - Camera still, moving scene
    - Moving camera, still scene
    - Moving camera, moving scene
- *Optical Flow* is what we ***can estimate from image*** Sequences.

9

Motion field and Optical flow

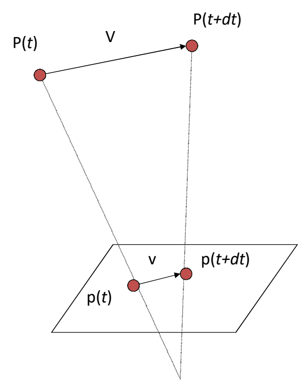
- Motion Field = Projection of real world 3D motion onto 2D plane.
- Optical Flow Field = Motion of brightness pattern present in 2D image!



10

Motion field and Optical flow

- $\boldsymbol{P}(t) = (X(t), Y(t), Z(t))$ is a moving 3D point
- Vel. of 3D point: $\boldsymbol{V} = d\boldsymbol{P}/dt$
- $\boldsymbol{p}(t) = (x(t), y(t))$ is the projection of $\boldsymbol{P}$ in the image
- Apparent velocity $\boldsymbol{v}$ in the image: given by components $v_x = dx/dt$ and $v_y = dy/dt$
- These components (v_x, vy) are known as the *optical flow* of the image.



11

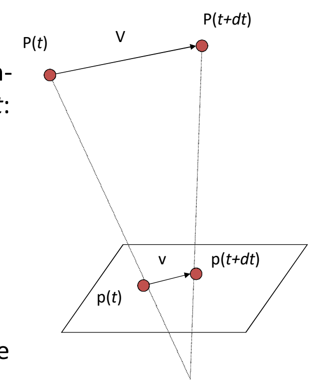
Motion field and Optical flow

To find image velocity $\boldsymbol{v}$, differentiate $\boldsymbol{p} = (x, y)$ with respect to t:

$$x = f\frac{X}{Z} \qquad v_x = f\frac{ZV_x - V_z X}{Z^2} = \frac{fV_x - V_z x}{Z}$$

$$y = f\frac{Y}{Z} \qquad v_y = \frac{fV_y - V_z y}{Z}$$

Image motion is a function of both the 3D motion ($\boldsymbol{V}$) and the depth of the 3D point (Z)



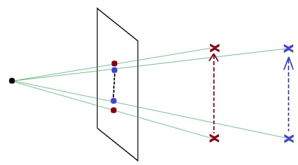
12

Motion field and Optical flow

- Pure translation: $\boldsymbol{V}$ is constant everywhere

$$v_x = \frac{fV_x - V_z x}{Z}, \quad v_y = \frac{fV_y - V_z y}{Z} \quad \Big\} \quad \mathbf{v} = \frac{1}{Z}(\mathbf{v}_0 - V_z \mathbf{x}), \text{ where } \mathbf{v}_0 = (fV_x, fV_y)$$

- The Magnitude of the motion vectors is inversely proportional to the depth Z.



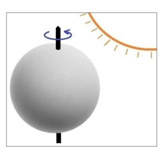
13

Optical flow

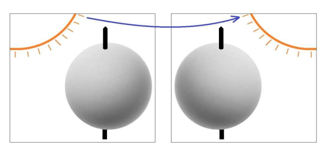
- Definition: Optical flow is the *apparent* motion of brightness patterns in the image.
- Ideally, optical flow would be the same as or proportional to the motion field.
- Frequently works, but not always.
- *Have to be careful:* apparent motion can be caused by lighting changes without any actual motion.

14

Optical Flow vs. Motion Field



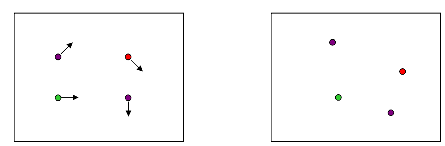
A smooth sphere is rotating under constant illumination. Thus the optical flow field is zero, but the motion field is Not.



A fixed sphere is illuminated by a moving source—the shading of the image changes. Thus the motion field is zero, but the optical flow field is not.

15

Estimating optical flow

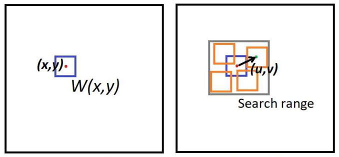


$I(x,y,t-1)$ $\quad$ $I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field $u(x,y)$ and $v(x,y)$ between them.
- Key assumptions
  - Brightness constancy: Projection of the same point looks the same in every frame.
  - Small motion: Points do not move very far.
  - Spatial coherence: Points move like their neighbors.

16

Discrete search to Optical flow



I(x,y,t-1) *I(x,y,t)*

- Given window W*(x,y,t-1)*, find best matching window in *I(x,y,t)*.
- Minimize SSD or SAD of pixels in window over second image

$$min_{(u,v)} = \sum_{(x,y)\in W} |I(x,y,t-1) - I(x+u,y+v,t)|^2$$

- search over specified range of *(u,v)* values called **search range**
- Displacement of best matched window gives *(u,v)*

17

Discrete search to Optical flow

18

Discrete search to Optical flow

19

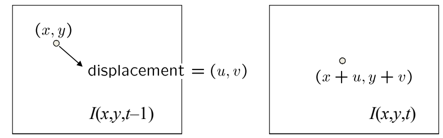
Discrete search to Optical flow

20

Discrete search to Optical flow

21

The brightness constancy constraint



- Displacement vector (u,v) is space dependent.
- This suggests that
  - horizontal comp. of displacement vector at (x,y) = u(x,y)
  - vertical comp. of displacement vector at (x,y) = v(x,y)
- Brightness Constancy Equation:

$$I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$$

for all *(x,y)*.

22

The brightness constancy constraint

- Suppose for δt time interval displace of point *(x,y)* is given by *(δx, δy)*
- Brightness Constancy Equation, then, becomes:

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

- Replacing the right-side by Taylor expansion:

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

- Ignoring the higher order terms (H.O.T.):

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

- This is called ***2D motion constraint equation***.

23

The brightness constancy constraint

- Rewriting ***2D motion constraint equation***

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

as

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

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

- Denoting spatial intensity gradient by $\nabla I = (I_x, I_y)$ and velocity vector by $\vec{v} = (u, v)$ 2D-motion constraint equation becomes

$$(I_x, I_y) \cdot (u, v) = -I_t \quad \text{or} \quad \nabla I \cdot \vec{v} = -I_t$$

- So at a pixel one equation with two unknowns (u,v).

24

The brightness constancy constraint

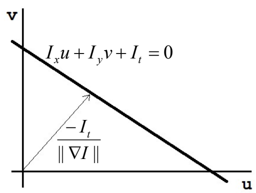
- At a single pixel we get a line:

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

$$\nabla I^T(x,y,t)\vec{v} = -I_t$$

where

$$\nabla I(x,y,t) = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \text{ and } \vec{\mathrm{v}} = \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix}$$

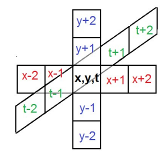


Aperture problem:

We get at most "Normal Flow" – with one point we can only detect movement perpendicular to the brightness gradient.

Multi-dimensional differentiation

Simoncelli (1994) proposed the following filter for computing multidimensional derivatives:

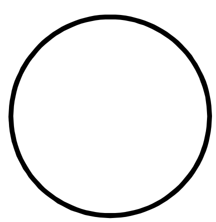


Index	p5	d5
-2	0.036	-0.108
-1	0.249	-0.283
0	0.431	0.0
1	0.249	0.283
2	0.036	0.108

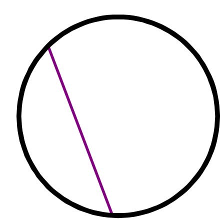
To compute I_x

- Convolve 5 frames *I(.,.,t-2), I(.,.,t-1), I(.,.,t), I(.,.,t-1) and I(.,.,t-2)* with *'p5'* to get a new image $I^t(.,.,t)$
- Convolve $I^t(.,.,t)$ with *p5* along *y*-direction to get $I_y^t(.,.,t)$
- Convolve $I_y^t(.,.,t)$ with d5 to get derivative I_x along *x*-direction.

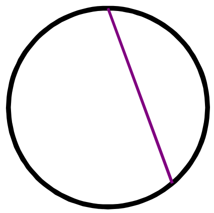
The aperture problem



The aperture problem

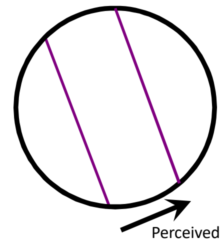


The aperture problem



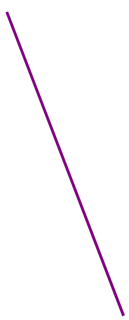
29

The aperture problem



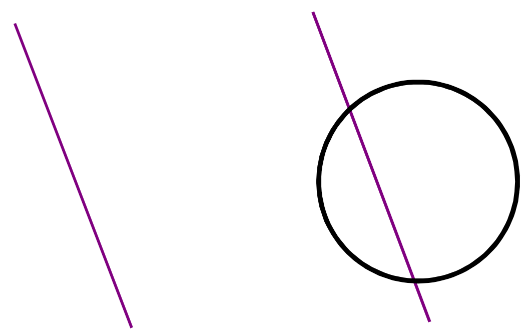
30

The aperture problem



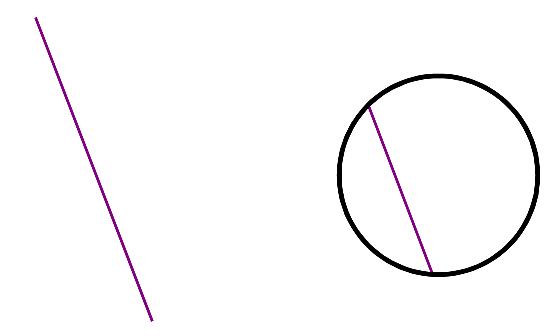
31

The aperture problem

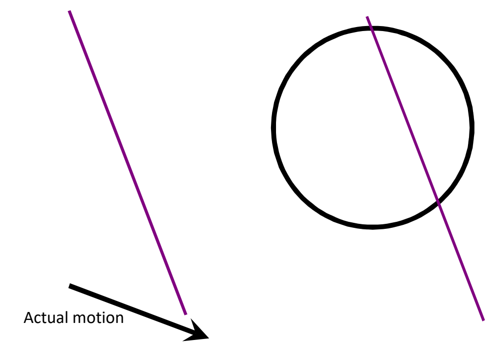


32

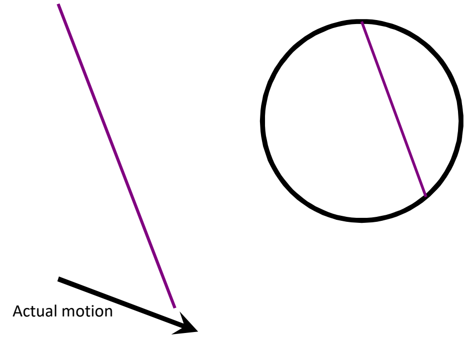
The aperture problem



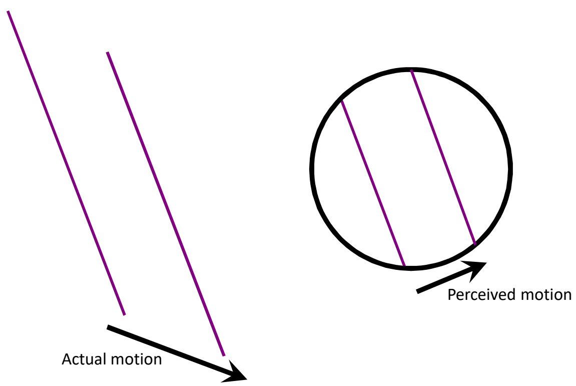
The aperture problem



The aperture problem



The aperture problem



Lucas and Kanade OF algorithm

- Consider

$$(I_x, I_y)\cdot(u,v) = -I_t \quad \text{or} \quad \nabla I \cdot \vec{v} = \text{-}I_t$$

- How to get more equations for a pixel?
- Let the velocity or optical flow (u,v) is smooth (neighborhood coherency.
  - That means over a small neighbourhood (u,v) is uniform.
- 2D optical flow may be estimated by local least-squares
- Modeling weighted least-squares fit of local first order motion constraint over a neighbourhood Ω

$$J(\vec{v}) = \sum_{(x,y)\in\Omega} W^2(x,y)\,|\nabla I(x,y,t)\cdot\vec{v} + I_t(x,y,t)|^2$$

37

Lucas and Kanade OF algorithm

- In matrix-vector notation squared sum may be written as

$$J(\vec{v}) = \left\| \begin{bmatrix} w(p_1) & & & \\ & w(p_2) & & \\ & & \ddots & \\ & & & w(p_n) \end{bmatrix} \left(\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ I_x(p_3) & I_y(p_3) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ -I_t(p_3) \\ \vdots \\ -I_t(p_n) \end{bmatrix} \right) \right\|^2$$

where *n* is the no. of pixels in the neighborhood and *p=(x,y)*

- This may be written as

$$\begin{aligned} J(\vec{v}) &= \| W(A\vec{v} - \vec{b}) \|^2 \\ &= (\vec{v}^T A^T - \vec{b}^T) W^T W (A\vec{v} - \vec{b}) \end{aligned}$$

38

Lucas and Kanade OF algorithm

- Expanding the expression we get

$$J(\vec{v}) = \vec{v}^T A^T W^T W A \vec{v} - \vec{v}^T A^T W^T W \vec{b} - \vec{b}^T W^T W A \vec{v} + \vec{b}^T W^T W \vec{b}$$

- Taking derivative w.r.t. $\vec{v}$ and equating to zero vector:

$$\frac{\partial J(\vec{v})}{\partial \vec{v}} = \vec{0} = 2A^T W^2 A\vec{v} - A^T W^2 \vec{b} - A^T W^2 \vec{b} + \vec{0}$$

- On solving we get

$$\vec{v} = (A^T W^2 A)^{-1} A^T W^2 \vec{b}$$

where

$$A^T W^2 A = \begin{bmatrix} \sum w^2(x,y) I_x^2(x,y) & \sum w^2(x,y) I_x(x,y) I_y(x,y) \\ \sum w^2(x,y) I_x(x,y) I_y(x,y) & \sum w^2(x,y) I_y^2(x,y) \end{bmatrix}$$

39

Horn and Schunck OF algorithm

- Motion constraint equation is combined with global smoothness of estimated velocity field *(u,v).*
- minimizing:

$$\begin{aligned} E(\vec{v}) &= (I_x u + I_y v + I_t)^2 + \lambda^2[(\nabla u)^2 + (\nabla v)^2] \\ &= (I_x u + I_y v + I_t)^2 + \lambda^2[(u - \bar{u})^2 + (v - \bar{v})^2] \end{aligned}$$

- Differentiating with respect to u and v and equating to zero:

$$\begin{aligned} (I_x^2 + \lambda^2)u + I_x I_y v &= \lambda^2 \bar{u} - I_x I_t \\ I_x I_y u + (I_y^2 + \lambda^2) v &= \lambda^2 \bar{v} - I_y I_t \end{aligned}$$

- Average $\bar{u}$ and $\bar{v}$ are computed over a region around (x, y).

40

Horn and Schunck OF algorithm

- Solving the equations (by Gauss-Seidel method)

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

- we get

$$u = \overline{u} - I_x \frac{I_x\overline{u} + I_y\overline{v} + I_t}{\lambda^2 + I_x^2 + I_y^2}$$
$$v = \overline{v} - I_x \frac{I_x\overline{u} + I_y\overline{v} + I_t}{\lambda^2 + I_x^2 + I_y^2}$$

41

Iterative algorithm for computing OF

- Set *k=0*
- Initialize all $u^k(x,y)$ and $v^k(x,y)$ with 0
- Until some error measure is satisfied, do

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

42

Optical flow: Examples


43

Optical flow: Examples

44

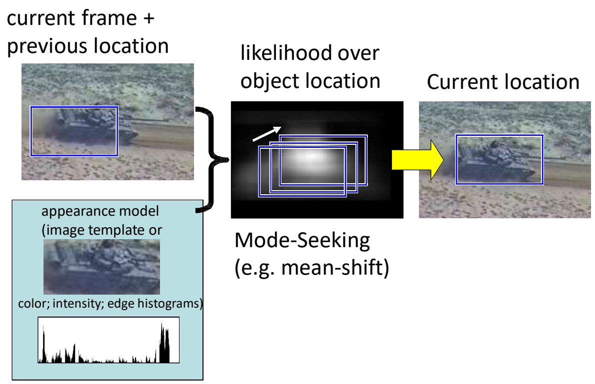
Tracking moving object: Example

Tracking moving object: Example

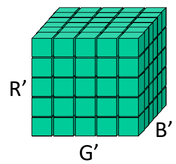
Tracking: Motivation

- 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 may be modeled with color distributions or PDF.

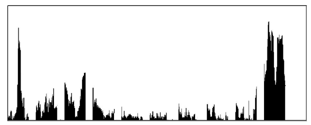
Appearance-Based tracking



Appearance via colour histogram



Discretize



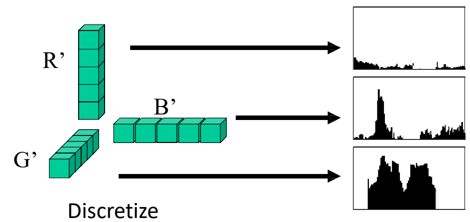
Color distribution (1D histogram normalized to have unit weight)

Original histogram size: (2^8 bits)³

R' = R << (8 - bits)
G' = G << (8 - bits)
B' = B << (8 - bits)

Example: 4-bit encoding of R, G and B channels yields a histogram of size 16*16*16 = 4096.

Appearance via colour histogram



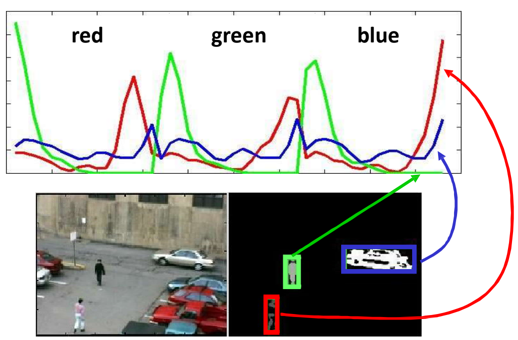
Discretize

R' = R << (8 - bits)
G' = G << (8 - bits)
B' = B << (8 - bits)

Original histogram size: (2^8 bits)x3

Example: 4-bit encoding of R, G and B channels yields a histogram of size 16+16+16 = 48.

Color Histogram Example



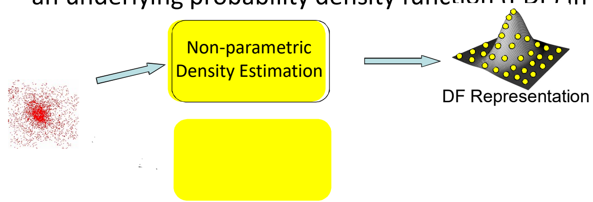
Mean-Shift tracking

- The mean-shift algorithm is an efficient approach to tracking objects whose appearance is defined by color.
  - not limited to only color, however. Could also use edge orientations, texture, motion

What is mean shift?

<u>A tool for</u>:

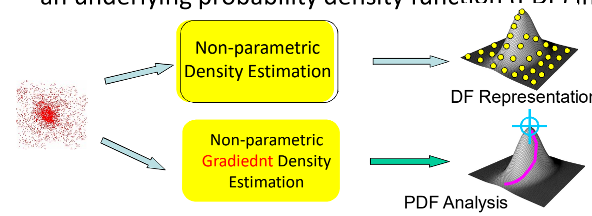
- Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^N



What is mean shift?

<u>A tool for</u>:

- Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^N

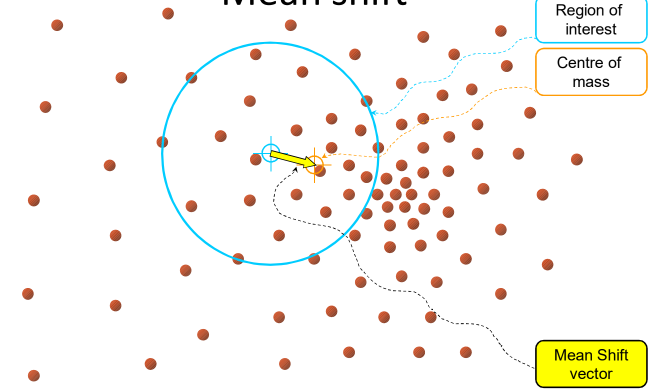


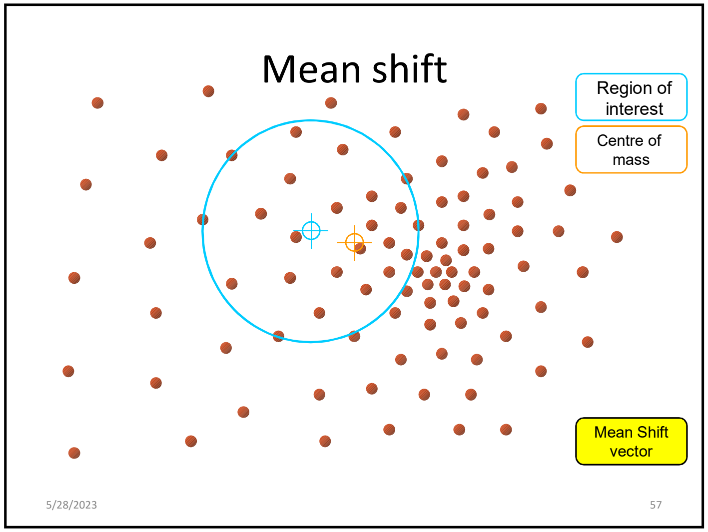
PDF in feature space: (i) Color space , (ii) Scale space, and (iii) Actually any feature space you can conceive

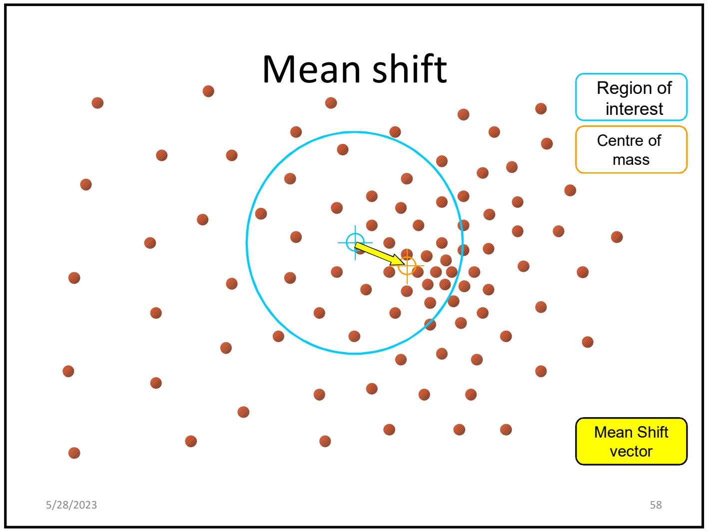
Feature space: example

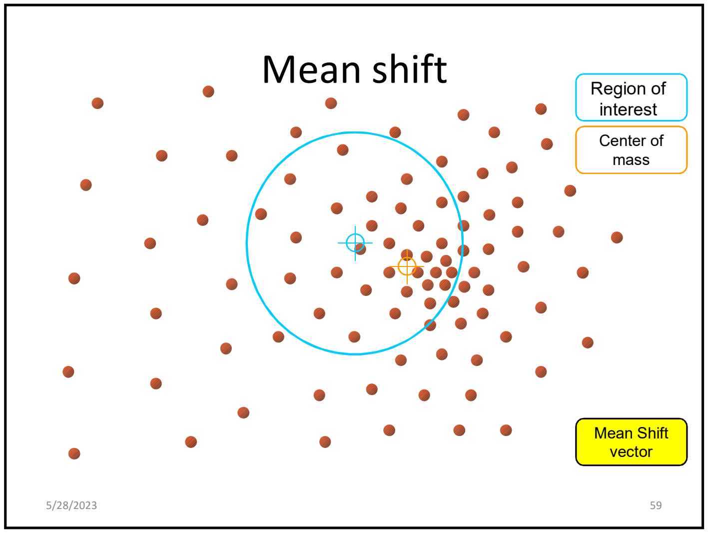
- Feature may be described as histogram of
  - R, G, B triplets (or a subset of these)
  - H, S, V triplets (or a subset of these)
  - Texture etc.
- Each point of feature space may be
  - Similarity between model and test histogram
  - Histogram back-projection values *(assumes model histogram is unimodal)*

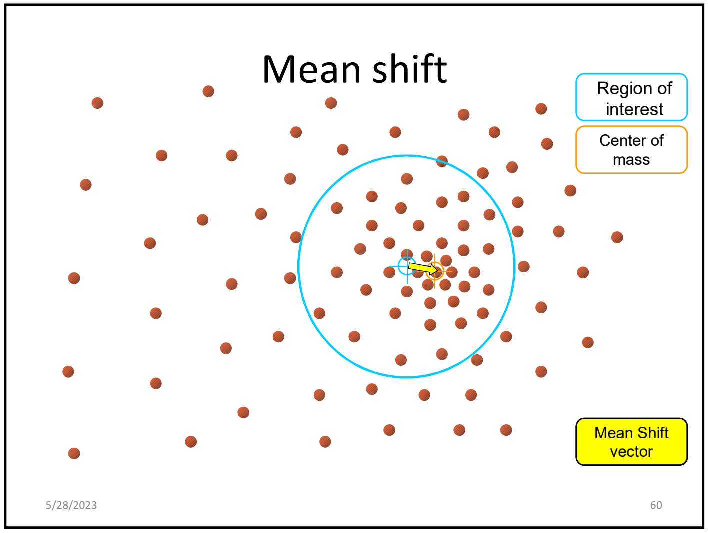
Mean shift



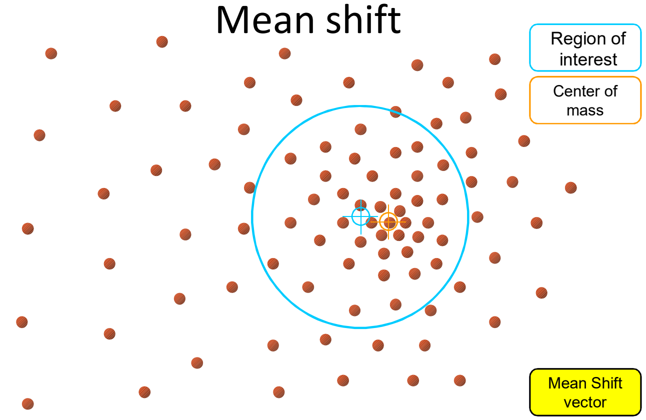

Mean shift
Region of interest
Centre of mass
Mean Shift vector
5/28/2023
57


Mean shift
Region of interest
Centre of mass
Mean Shift vector
5/28/2023
58

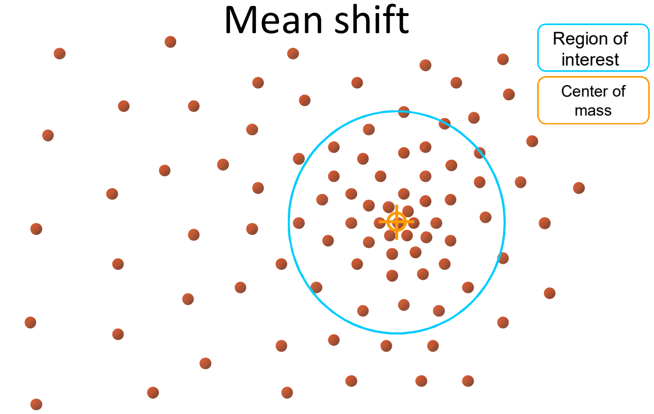

Mean shift
Region of interest
Center of mass
Mean Shift vector
5/28/2023
59


Mean shift
Region of interest
Center of mass
Mean Shift vector
5/28/2023
60

Mean shift



Mean shift



Computing the mean shift

Simple Mean Shift procedure:

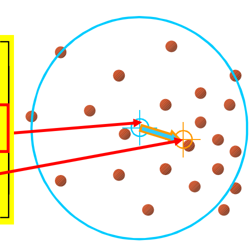
- Compute mean shift vector
- Translate the Kernel window by **m(x)**

Computing the mean shift

Simple Mean Shift procedure:

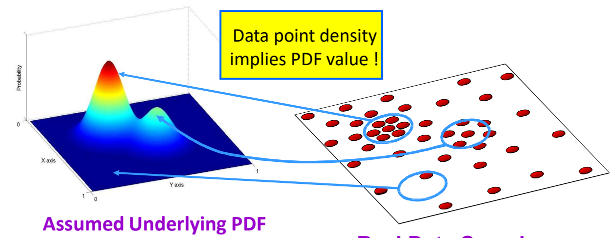
- Compute mean shift vector
- Translate the Kernel window by **m(x)**

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)} - \mathbf{x} \right]$$



Non-Parametric density estimation

Assumption: The data points that are sampled from a PDF.

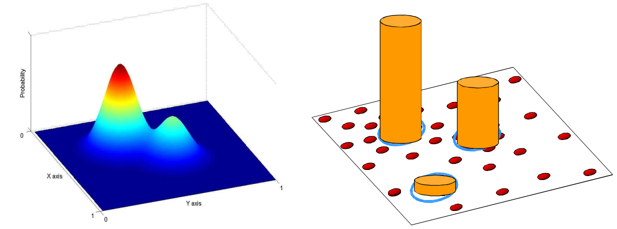


Assumed Underlying PDF

Real Data Samples

65

Non-Parametric density estimation

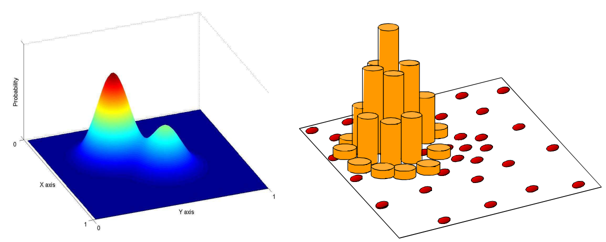


Assumed Underlying PDF

Real Data Samples

66

Non-Parametric density estimation



Assumed Underlying PDF

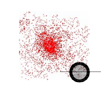
Real Data Samples

67

Kernel Density Estimation

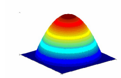
(Various Kernels)

- A function of some finite number of data points $x_1, x_2, x_3, \dots, x_n$



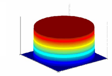
$$P(x) = \sum_{i=1}^{n} K(x - x_i)$$

- Epanechnikov kernel



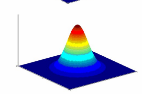
$$K_E(x) = \begin{cases} c(1/||x||^2) & ||x|| \le 1 \\ 0 & otherwise \end{cases}$$

- Uniform kernel



$$K_U(x) = \begin{cases} c & ||x|| \le 1 \\ 0 & otherwise \end{cases}$$

- Normal kernel



$$K_N(x) = c \cdot \exp(-\frac{1}{h}||x||^2)$$

68

Kernel and Profile

- Radially symmetric kernel:

$$K(x) = ck(||x||^2)$$

Profile

- Example:

$$P(x) = \frac{1}{n}\sum_{i=1}^{n} K(x - x_i) = \frac{1}{n}c\sum_{i=1}^{n} k(||x - x_i||^2)$$

Kernel Density Estimation

- Reconsider

$$P(x) = \frac{1}{n}c\sum_{i=1}^{n} k(||x - x_i||^2)$$

- Taking derivative

$$\nabla P(x) = \frac{1}{n}c\sum_{i=1}^{n} \nabla k(||x - x_i||^2)$$

$$\nabla P(x) = \frac{1}{n}2c\sum_{i=1}^{n} (x - x_i)k'(||x - x_i||^2)$$

$$= \frac{1}{n}2c\sum_{i=1}^{n} (x_i - x)g(||x - x_i||^2)$$

$$= \frac{1}{n}2c\sum_{i=1}^{n} x_i g(||x - x_i||^2) - \frac{1}{n}2c\sum_{i=1}^{n} xg||x - x_i||^2)$$

Kernel Density Estimation

- Rewriting:

$$\nabla P(x) = \frac{1}{n}2c\sum_{i=1}^{n} x_i g(||x - x_i||^2) - \frac{1}{n}2c\sum_{i=1}^{n} xg(||x - x_i||^2)$$

$$= \frac{1}{n}2c\sum_{i=1}^{n} x_i g(||x - x_i||^2) - \frac{1}{n}2cx\sum_{i=1}^{n} g(||x - x_i||^2)$$

$$= \frac{1}{n}2c\sum_{i=1}^{n} g(||x - x_i||^2)\left[\frac{\frac{1}{n}2c\sum_{i=1}^{n} x_i g(||x - x_i||^2)}{\frac{1}{n}2c\sum_{i=1}^{n} g(||x - x_i||^2)} - x\right]$$

- Put $g(||x - x_i||^2) \rightarrow g_i$

Computing mean shift

- Finally, we obtain

$$\nabla P(x) = \frac{2c}{n}\sum_{i=1}^{n} g_i\left[\frac{\sum_{i=1}^{n} x_i g_i}{\sum_{i=1}^{n} g_i} - x\right]$$

- OR

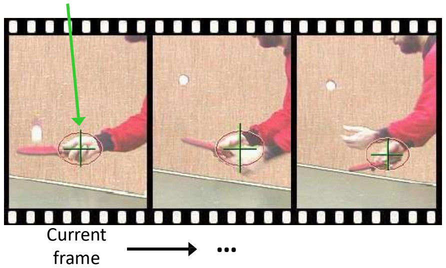
$$\nabla P(x) = \left(\frac{2c}{n}\sum_{i=1}^{n} g_i\right)m(x)$$

- Thus mean shift

$$m(x) = \frac{\nabla P(x)}{\frac{2c}{n}\sum_{i=1}^{n} g_i}$$

Mean shift object tracking

Choose a reference model in the current frame → Choose a feature space → Represent the model in the chosen feature space



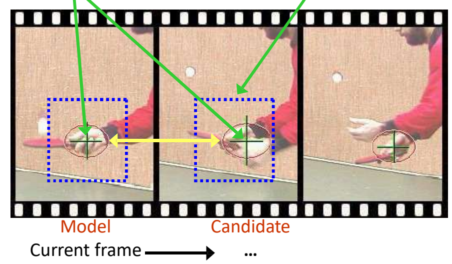
Mean shift object tracking

Start from the position of the model in the current frame

Search in the model’s neighborhood in the next frame

Find best candidate by maximizing a similarity func

Repeat the same process in the next pair of frames



Thank you !

Any question?