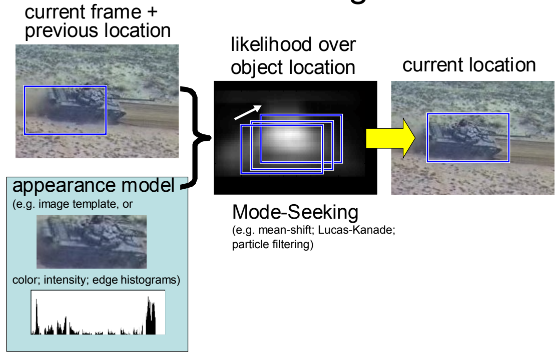
Mean-shift Tracking

R.Collins, CSE, PSU
CSE598G Spring 2006

Appearance-Based Tracking



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

- 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

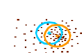
Mean Shift Theory

Credits: Many Slides Borrowed from

www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/mean_shift/mean_shift.ppt

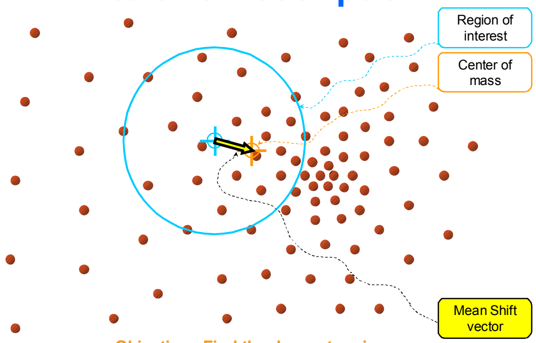
Mean Shift
Theory and Applications

Yaron Ukrainitz & Bernard Sarel
weizmann institute
Advanced topics in computer vision



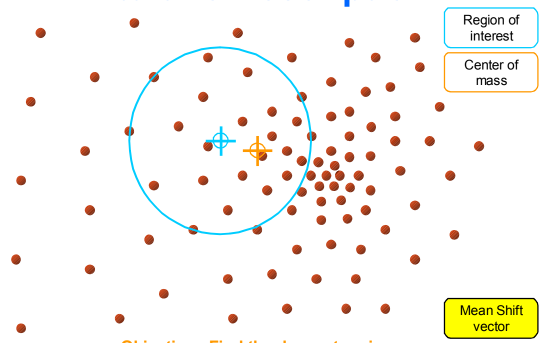
even the slides on my own work! --Bob

Intuitive Description



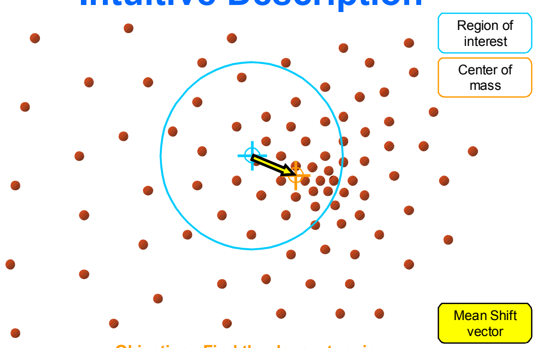
Objective : Find the densest region

Intuitive Description



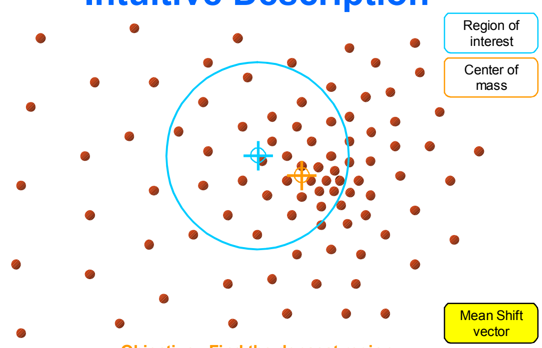
Objective : Find the densest region

Intuitive Description



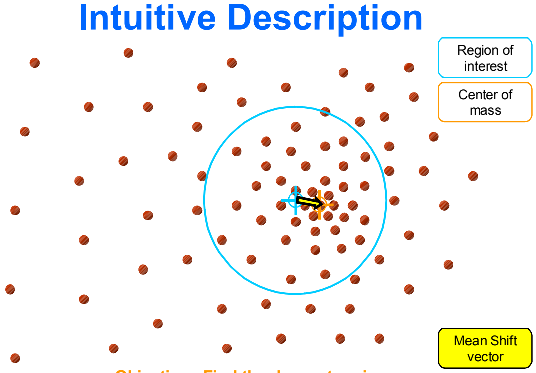
Objective : Find the densest region

Intuitive Description



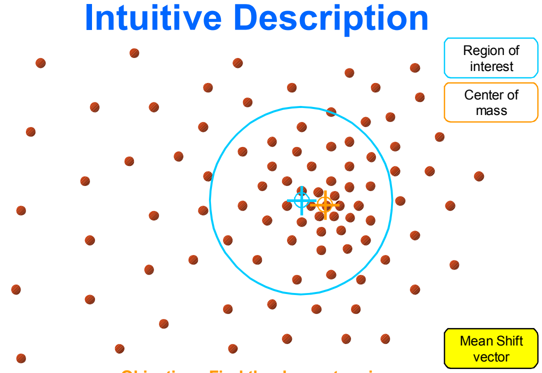
Objective : Find the densest region

Intuitive Description



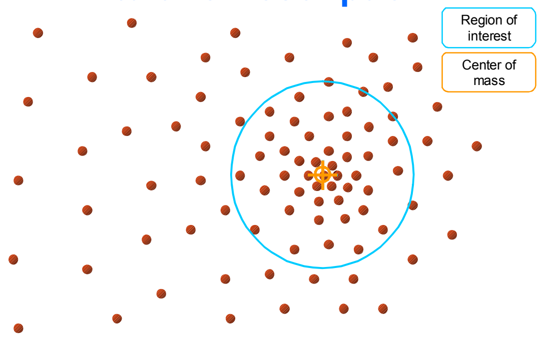
Objective : Find the densest region

Intuitive Description



Objective : Find the densest region

Intuitive Description

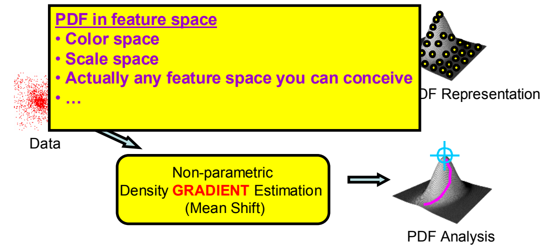


Objective : Find the densest region

What is Mean Shift ?

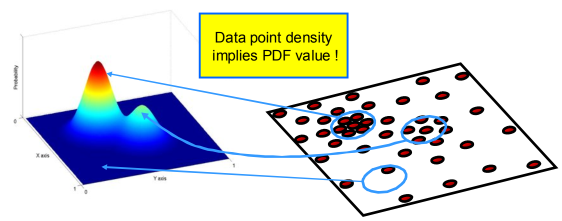
A tool for:
Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^N

PDF in feature space
- Color space
- Scale space
- Actually any feature space you can conceive
- …



Non-Parametric Density Estimation

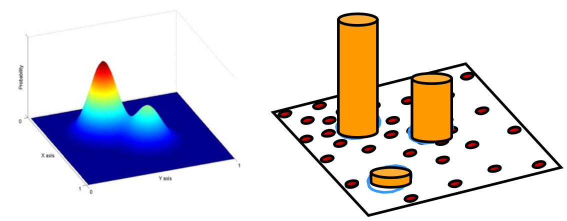
Assumption : The data points are sampled from an underlying PDF



Assumed Underlying PDF

Real Data Samples

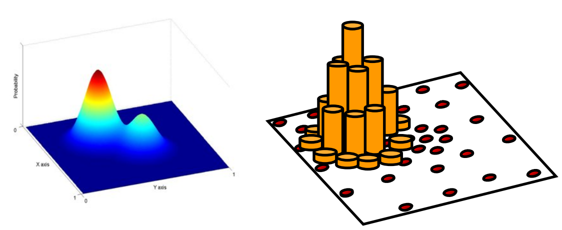
Non-Parametric Density Estimation



Assumed Underlying PDF

Real Data Samples

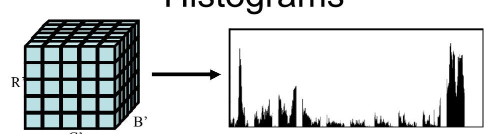
Non-Parametric Density Estimation



Assumed Underlying PDF

Real Data Samples

Appearance via Color Histograms



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

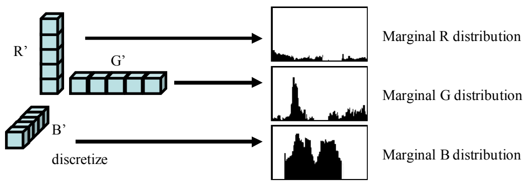
R' = R << (8 - nbits)
G' = G << (8 - nbits)
B' = B << (8-nbits)

Total histogram size is (2^(8-nbits))^3

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

Smaller Color Histograms

Histogram information can be much much smaller if we are willing to accept a loss in color resolvability.

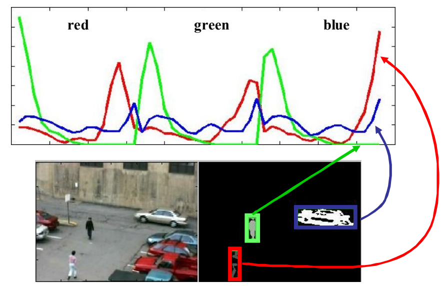


R' = R << (8 - nbits)
G' = G << (8 - nbits)
B' = B << (8-nbits)

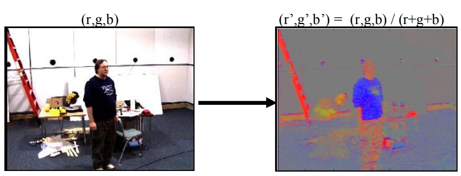
Total histogram size is 3*(2^(8-nbits))

example, 4-bit encoding of R,G and B channels yields a histogram of size 3*16 = 48.

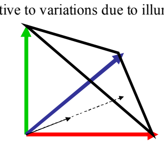
Color Histogram Example



Normalized Color



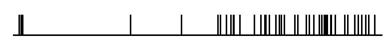
Normalized color divides out pixel luminance (brightness), leaving behind only chromaticity (color) information. The result is less sensitive to variations due to illumination/shading.



Intro to Parzen Estimation

(Aka Kernel Density Estimation)

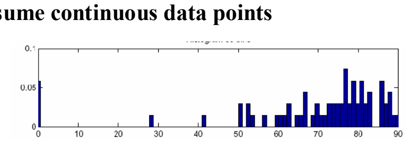
Mathematical model of how histograms are formed
Assume continuous data points



Parzen Estimation

(Aka Kernel Density Estimation)

Mathematical model of how histograms are formed
Assume continuous data points



Convolve with box filter of width w (e.g. [1 1 1])
Take samples of result, with spacing w
Resulting value at point u represents count of data points falling in range u-w/2 to u+w/2

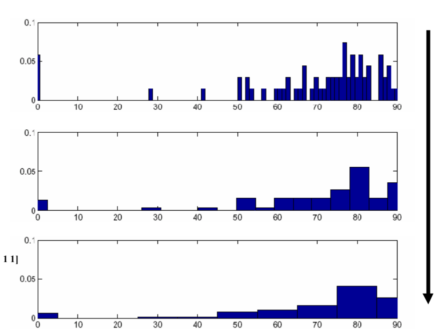
Example Histograms

Box filter

[1 1 1]

[1 1 1 1 1 1 1]

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]



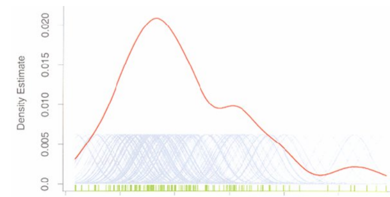
Increased smoothing

Why Formulate it This Way?

- Generalize from box filter to other filters (for example Gaussian)
- Gaussian acts as a smoothing filter.

Kernel Density Estimation

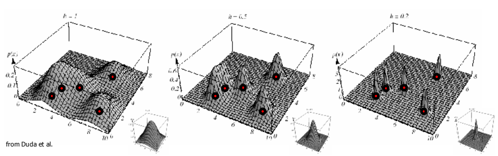
- **Parzen windows**: Approximate probability density by estimating local density of points (same idea as a histogram)
  - Convolve points with window/kernel function (e.g., Gaussian) using scale parameter (e.g., sigma)



from Hastie *et al.*

Density Estimation at Different Scales

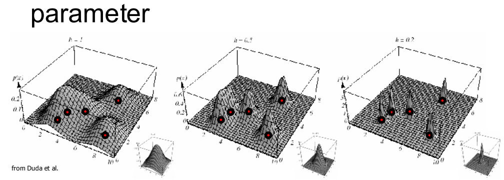
- Example: Density estimates for 5 data points with differently-scaled kernels
- Scale influences accuracy vs. generality (overfitting)



from Duda et al.

Smoothing Scale Selection

- Unresolved issue: how to pick the scale (sigma value) of the smoothing filter
- Answer for now: a user-settable parameter



from Duda et al.

Kernel Density Estimation

Parzen Windows - General Framework

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

A function of some finite number of data points $x_1 \ldots x_n$

Data

Kernel Properties:

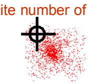
- Normalized $\int_{R^d} K(\mathbf{x})d\mathbf{x} = 1$
- Symmetric $\int_{R^d} \mathbf{x}K(\mathbf{x})d\mathbf{x} = 0$
- Exponential weight decay $\lim_{\|\mathbf{x}\|\to\infty} \|\mathbf{x}\|^d K(\mathbf{x}) = 0$
- ??? $\int_{R^d} \mathbf{x}\mathbf{x}^T K(\mathbf{x})d\mathbf{x} = c\mathbf{I}$

Kernel Density Estimation

Parzen Windows - Function Forms

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

A function of some finite number of data points $x_1 \ldots x_n$



Data

In practice one uses the forms:

$$K(\mathbf{x}) = c\prod_{i=1}^{d} k(x_i) \quad \text{or} \quad K(\mathbf{x}) = ck(\|\mathbf{x}\|)$$

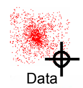
Same function on each dimension

Function of vector length only

Kernel Density Estimation
Various Kernels

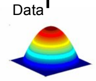
$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x}-\mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$

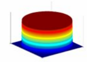


Examples:

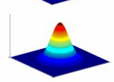
- Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1-\|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Uniform Kernel $K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



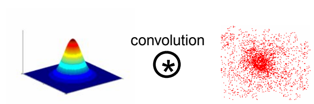
- Normal Kernel $K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$



Key Idea:

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

Superposition of kernels, centered at each data point is equivalent to convolving the data points with the kernel.



Kernel Density Estimation
$\underbrace{\text{}}_{}$ Gradient

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x}-\mathbf{x}_i)$$

Give up estimating the PDF !
Estimate **ONLY** the gradient

Using the Kernel form: $K(\mathbf{x}-\mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$

Size of window

We get :

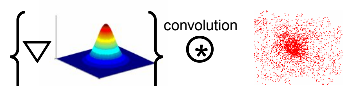
$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \cdot \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

$g(\mathbf{x}) = -k'(\mathbf{x})$

Key Idea:

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x}-\mathbf{x}_i)$$

Gradient of superposition of kernels, centered at each data point is equivalent to convolving the data points with gradient of the kernel.



Kernel Density Estimation
Computing The Mean Shift
Gradient

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \cdot \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

$g(\mathbf{x}) = -k'(\mathbf{x})$

Computing The Mean Shift

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \cdot \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

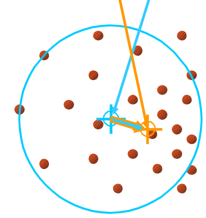
Yet another Kernel density estimation !

Simple Mean Shift procedure:

- Compute mean shift vector

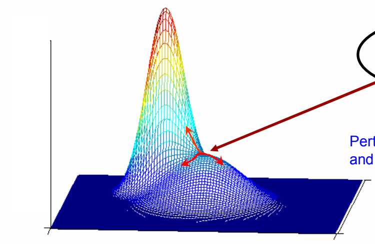
$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)} - \mathbf{x}\right]$$

- Translate the Kernel window by $\mathbf{m(x)}$



$g(\mathbf{x}) = -k'(\mathbf{x})$

Mean Shift Mode Detection

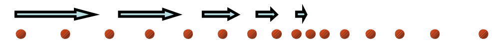


What happens if we reach a saddle point ?

Perturb the mode position and check if we return back

Updated Mean Shift Procedure:
- Find all modes using the Simple Mean Shift Procedure
- Prune modes by perturbing them (find saddle points and plateaus)
- Prune nearby – take highest mode in the window

Mean Shift Properties

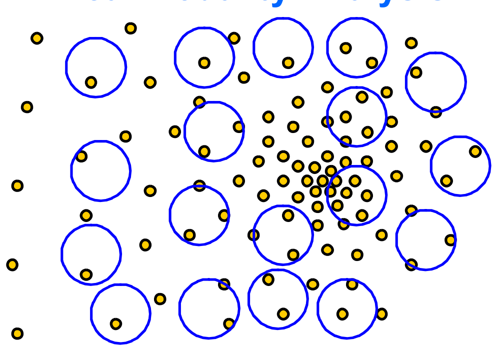


- Automatic convergence speed – the mean shift vector size depends on the gradient itself.
- Near maxima, the steps are small and refined

} **Adaptive** Gradient Ascent

- Convergence is guaranteed for infinitesimal steps only → infinitely convergent, (therefore set a lower bound)
- For Uniform Kernel ( ), convergence is achieved in a finite number of steps
- Normal Kernel ( ) exhibits a smooth trajectory, but is slower than Uniform Kernel ( ).

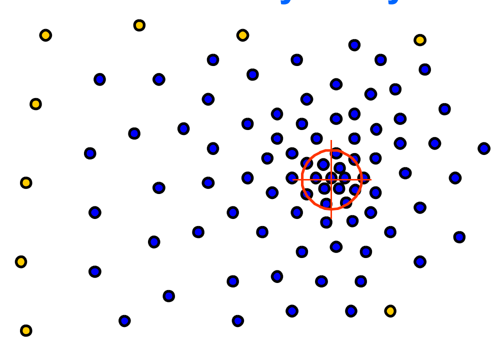
Real Modality Analysis



Tessellate the space with windows

Run the procedure in parallel

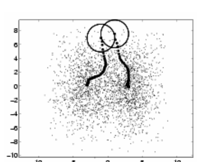
Real Modality Analysis



The **blue** data points were traversed by the windows towards the mode

Real Modality Analysis

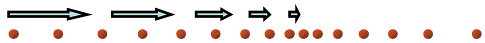
An example



Window tracks signify the steepest ascent directions

Adaptive Mean Shift

Mean Shift Strengths & Weaknesses



Strengths :

- Application independent tool
- Suitable for real data analysis
- Does not assume any prior shape (e.g. elliptical) on data clusters
- Can handle arbitrary feature spaces
- Only ONE parameter to choose
- *h* (window size) has a physical meaning, unlike K-Means

Weaknesses :

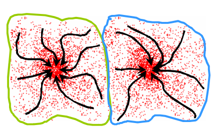
- The window size (bandwidth selection) is not trivial
- Inappropriate window size can cause modes to be merged, or generate additional “shallow” modes ➔ Use adaptive window size

Mean Shift Applications

Clustering

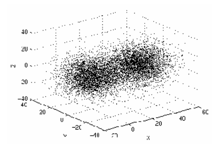
Cluster : All data points in the ***attraction basin*** of a mode

Attraction basin : the region for which all trajectories lead to the same mode



Mean Shift : A robust Approach Toward Feature Space Analysis, by Comaniciu, Meer

Clustering
Synthetic Examples

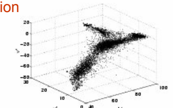


Simple Modal Structures

Complex Modal Structures

Clustering
Real Example

Feature space:
L*u*v representation



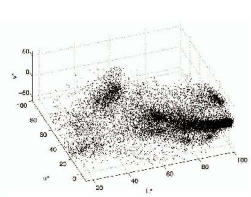
(a)

Initial window enters

M

pruning

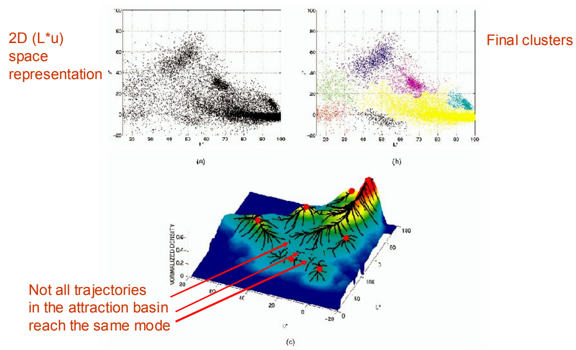
Clustering
Real Example



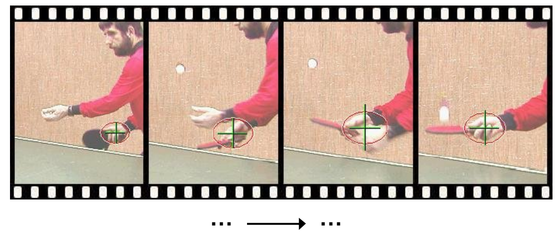
L*u*v space representation

Clustering

Real Example



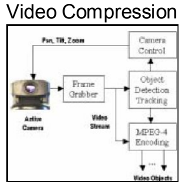
Non-Rigid Object Tracking



Kernel Based Object Tracking, by Comaniciu, Ramesh, Meer (CRM)

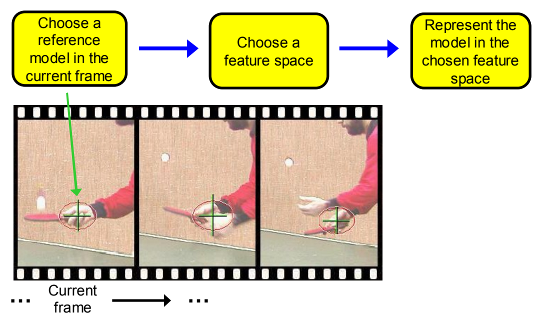
Non-Rigid Object Tracking

Real-Time

Surveillance	Driver Assistance	Object-Based Video Compression
		

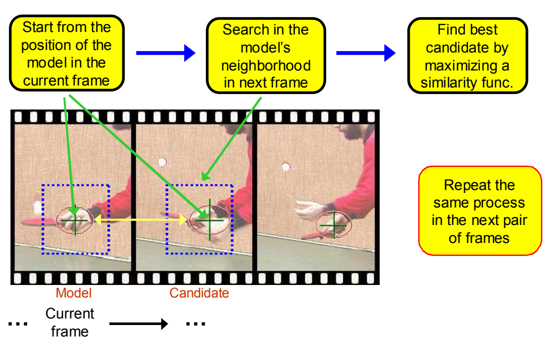
Mean-Shift Object Tracking

General Framework: Target Representation



Mean-Shift Object Tracking

General Framework: Target Localization



Using Mean-Shift for Tracking in Color Images

Two approaches:

1) Create a color "likelihood" image, with pixels weighted by similarity to the desired color (best for unicolored objects)

2) Represent color distribution with a histogram. Use mean-shift to find region that has most similar distribution of colors.

Mean-shift on Weight Images

Ideally, we want an indicator function that returns 1 for pixels on the object we are tracking, and 0 for all other pixels

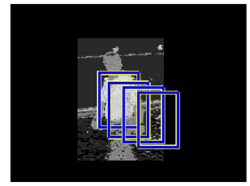
Instead, we compute likelihood maps where the value at a pixel is proportional to the likelihood that the pixel comes from the object we are tracking.

Computation of likelihood can be based on
- color
- texture
- shape (boundary)
- predicted location

Note: So far, we have described mean-shift as operating over a set of point samples...

Mean-Shift Tracking

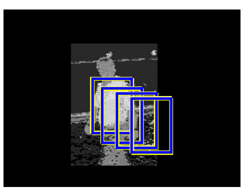
Let pixels form a uniform grid of data points, each with a weight (pixel value) proportional to the "likelihood" that the pixel is on the object we want to track. Perform standard mean-shift algorithm using this weighted set of points.

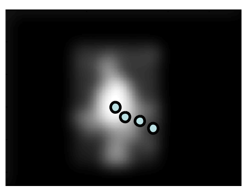


$$\Delta x = \frac{\sum_a K(a-x)\, w(a)\, (a-x)}{\sum_a K(a-x)\, w(a)}$$

Nice Property

Running mean-shift with kernel K on weight image w is equivalent to performing gradient ascent in a (virtual) image formed by convolving w with some "shadow" kernel H.





Note: mode we are looking for is mode of location (x,y) likelihood, NOT mode of the color distribution!

Kernel-Shadow Pairs

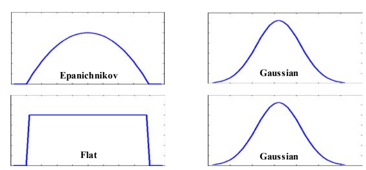
Given a convolution kernel H, what is the corresponding mean-shift kernel K?
Perform change of variables $r = ||a-x||^2$
Rewrite $H(a-x) \Rightarrow h(||a-x||^2) \Rightarrow h(r)$.

Then kernel K must satisfy **$h'(r) = - c\, k(r)$**

Examples

Shadow

Kernel



Using Mean-Shift on Color Models

Two approaches:

1) Create a color "likelihood" image, with pixels weighted by similarity to the desired color (best for unicolored objects)

2) Represent color distribution with a histogram. Use mean-shift to find region that has most similar distribution of colors.

High-Level Overview

Spatial smoothing of similarity function by introducing a spatial kernel (Gaussian, box filter)

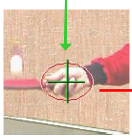
Take derivative of similarity with respect to colors. This tells what colors we need more/less of to make current hist more similar to reference hist.

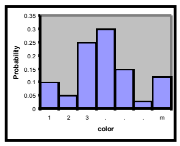
Result is weighted mean shift we used before. However, the color weights are now computed "on-the-fly", and change from one iteration to the next.

Mean-Shift Object Tracking
Target Representation

Choose a reference target model → Choose a feature space → Represent the model by its PDF in the feature space

Quantized Color Space

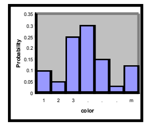


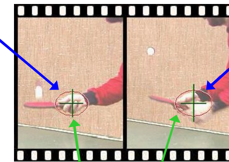


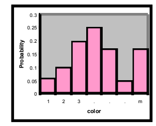
Mean-Shift Object Tracking
PDF Representation

Target Model (centered at 0)

Target Candidate (centered at y)







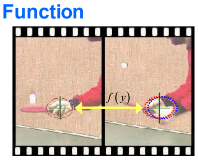
$\vec{q} = \{q_u\}_{u=1..m} \qquad \sum_{u=1}^{m} q_u = 1$

$\vec{p}(y) = \{p_u(y)\}_{u=1..m} \qquad \sum_{u=1}^{m} p_u = 1$

Similarity Function: $f(y) = f\left[\vec{q}, \vec{p}(y)\right]$

Mean-Shift Object Tracking
Smoothness of Similarity Function

Similarity Function: $f(y) = f\left[\vec{p}(y), \vec{q}\right]$



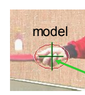
Problem: Target is represented by color info only → Spatial info is lost → Large similarity variations for adjacent locations →

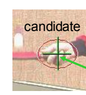
f is not smooth → Gradient-based optimizations are not robust

Solution: Mask the target with an isotropic kernel in the spatial domain → *f*(*y*) becomes smooth in *y*

Mean-Shift Object Tracking
Finding the PDF of the target model

$\{x_i\}_{i=1..n}$ Target pixel locations





$k(x)$ A differentiable, isotropic, convex, monotonically decreasing kernel
- Peripheral pixels are affected by occlusion and background interference

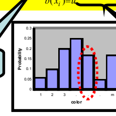
$b(x)$ The color bin index (1..*m*) of pixel *x*

Probability of feature u in model

$q_u = C \sum_{b(x_i)=u} k\left(\left\| x_i \right\|^2\right)$

Normalization factor

Pixel weight

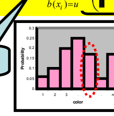


Probability of feature u in candidate

$p_u(y) = C_h \sum_{b(x_i)=u} k\left(\left\| \frac{y - x_i}{h} \right\|^2\right)$

Normalization factor

Pixel weight



Mean-Shift Object Tracking
Similarity Function

Target model: $\vec{q} = (q_1, \ldots, q_m)$

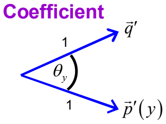
Target candidate: $\vec{p}(y) = (p_1(y), \ldots, p_m(y))$

Similarity function: $f(y) = f\left[\vec{p}(y), \vec{q}\right] = ?$

The Bhattacharyya Coefficient

$\vec{q}' = \left(\sqrt{q_1}, \ldots, \sqrt{q_m}\right)$

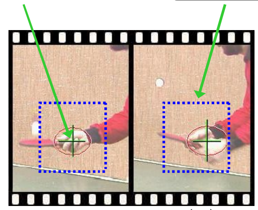
$\vec{p}'(y) = \left(\sqrt{p_1(y)}, \ldots, \sqrt{p_m(y)}\right)$



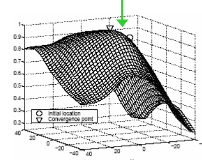
$$f(y) = \cos\theta_y = \frac{p'(y)^T q'}{\left\| p'(y) \right\| \cdot \left\| q' \right\|} = \sum_{u=1}^{m} \sqrt{p_u(y) q_u}$$

Mean-Shift Object Tracking
Target Localization Algorithm

Start from the position of the model in the current frame → Search in the model's neighborhood in next frame → Find best candidate by maximizing a similarity func.



$\vec{q} \qquad \vec{p}(y)$



$f\left[\vec{p}(y), \vec{q}\right]$

Mean-Shift Object Tracking
Approximating the Similarity Function

$$f(y)=\sum_{u=1}^{m}\sqrt{p_u(y)\,q_u}$$

Model location: y_0

Candidate location: y

Linear approx. (around y_0)

$$f(y)\approx\underbrace{\frac{1}{2}\sum_{u=1}^{m}\sqrt{p_u(y_0)\,q_u}}_{\text{Independent of } y}+\frac{1}{2}\sum_{u=1}^{m}\boxed{p_u(y)}\boxed{\sqrt{\frac{q_u}{p_u(y_0)}}}$$

$$p_u(y)=C_h\sum_{b(x_i)=u}k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$$

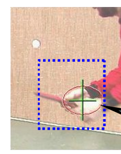
$$\frac{C_h}{2}\sum_{i=1}^{n}\boxed{w_i}\,k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$$

Density estimate! (as a function of y)

Mean-Shift Object Tracking
Maximizing the Similarity Function

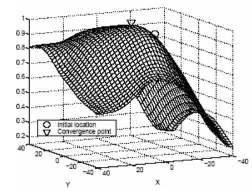
The mode of $\frac{C_h}{2}\sum_{i=1}^{n}w_i k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ = sought maximum

Important Assumption:



The target representation provides sufficient discrimination

One mode in the searched neighborhood



Mean-Shift Object Tracking
Applying Mean-Shift

The mode of $\frac{C_h}{2}\sum_{i=1}^{n}w_i k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ = sought maximum

Original Mean-Shift: Find mode of $c\sum_{i=1}^{n}k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ using $y_1=\dfrac{\sum_{i=1}^{n}x_i g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n}g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}$

Extended Mean-Shift: Find mode of $c\sum_{i=1}^{n}\boxed{w_i}k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ using $y_1=\dfrac{\sum_{i=1}^{n}x_i\boxed{w_i}g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n}\boxed{w_i}g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}$

Mean-Shift Object Tracking
About Kernels and Profiles

A special class of radially symmetric kernels: $K(x)=ck\left(\|x\|^2\right)$

The profile of kernel K

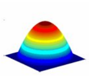
$$k'(x)=-g(x)$$

Extended Mean-Shift: Find mode of $c\sum_{i=1}^{n}w_i k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ using $y_1=\dfrac{\sum_{i=1}^{n}x_i w_i g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n}w_i g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}$

Mean-Shift Object Tracking
Choosing the Kernel

A special class of radially symmetric kernels: $K(x)=ck\left(\|x\|^2\right)$

Epanechnikov kernel:



$$k(x)=\begin{cases}1-x & \text{if } \|x\|\le 1\\ 0 & \text{otherwise}\end{cases}$$

Uniform kernel:

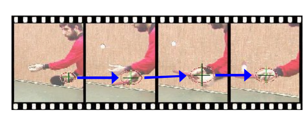
$$g(x)=-k'(x)=\begin{cases}1 & \text{if } \|x\|\le 1\\ 0 & \text{otherwise}\end{cases}$$

$$y_1=\frac{\sum_{i=1}^{n}x_i w_i g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n}w_i g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}\longrightarrow y_1=\frac{\sum_{i=1}^{n}x_i w_i}{\sum_{i=1}^{n}w_i}$$

Mean-Shift Object Tracking
Adaptive Scale

Problem:

The scale of the target changes in time → The scale (h) of the kernel must be adapted



Solution:

Run localization 3 times with different h → Choose h that achieves maximum similarity

Mean-Shift Object Tracking
Results

From Comaniciu, Ramesh, Meer

Feature space: 16×16×16 quantized RGB
Target: manually selected on 1st frame
Average mean-shift iterations: 4

Mean-Shift Object Tracking
Results

Partial occlusion

Distraction

Motion blur

Mean-Shift Object Tracking
Results

From Comaniciu, Ramesh, Meer

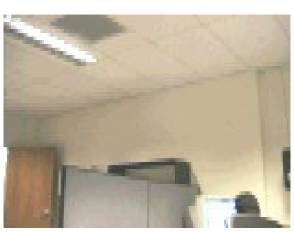
Mean-Shift Object Tracking
Results

From Comaniciu, Ramesh, Meer

Feature space: 128×128 quantized RG

Mean-Shift Object Tracking
Results

The man himself…



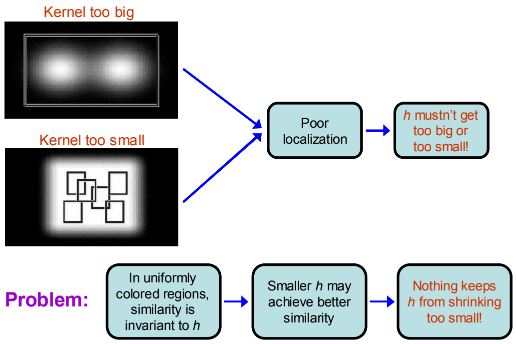
From Comaniciu, Ramesh, Meer

Feature space: 128×128 quantized RG

Handling Scale Changes

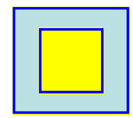
Mean-Shift Object Tracking

The Scale Selection Problem



Some Approaches to Size Selection

- Choose one scale and stick with it.
- Bradski's CAMSHIFT tracker computes principal axes and scales from the second moment matrix of the blob. Assumes one blob, little clutter.
- CRM adapt window size by +/- 10% and evaluate using Battacharyya coefficient. Although this does stop the window from growing too big, it is not sufficient to keep the window from shrinking too much.
- Comaniciu's variable bandwidth methods. Computationally complex.
- Rasmussen and Hager: add a border of pixels around the window, and require that pixels in the window should look like the object, while pixels in the border should not.



Center-surround

Tracking Through Scale Space

Motivation

Mean-shift Blob Tracking through Scale Space, by R. Collins

Scale Space Feature Selection

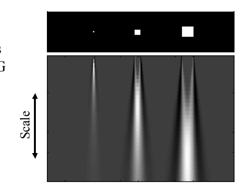
Form a resolution scale space by convolving image with Gaussians of increasing variance.

$$L(\cdot;\ t) = g(\cdot;\ t) * f(\cdot)$$

Lindeberg proposes that the natural scale for describing a feature is the scale at which a normalized differential operator for detecting that feature achieves a local maximum both spatially and in scale.

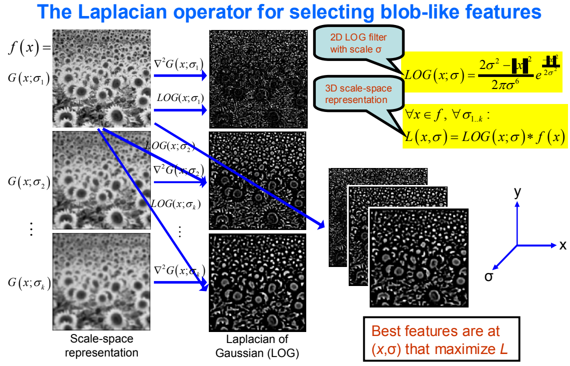
$$\begin{cases} (\nabla(\mathcal{D}_{norm}L))(x_0;\ t_0) = 0, \\ (\partial_t(\mathcal{D}_{norm}L))(x_0;\ t_0) = 0. \end{cases}$$

For blob detection, the Laplacian operator is used, leading to a search for modes in a LOG scale space. (Actually, we approximate the LOG operator by DOG).



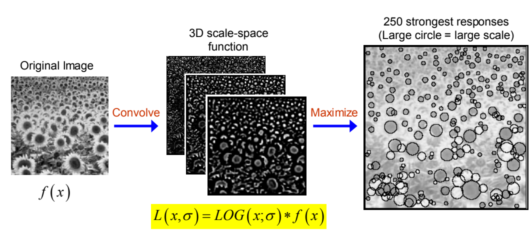
Lindeberg's Theory

The Laplacian operator for selecting blob-like features



Lindeberg's Theory

Multi-Scale Feature Selection Process



Tracking Through Scale Space

Approximating LOG using DOG

$$LOG(x;\sigma) \approx DOG(x;\sigma) = G(x;\sigma) - G(x;1.6\sigma)$$

2D LOG filter with scale σ

2D DOG filter with scale σ

2D Gaussian with μ=0 and scale σ

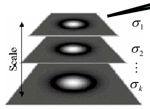
2D Gaussian with μ=0 and scale 1.6σ

Why DOG?

- Gaussian pyramids are created faster
- Gaussian can be used as a mean-shift kernel

3D spatial kernel

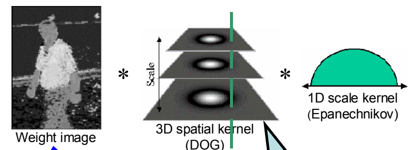
DOG filters at multiple scales

$K(x,\sigma) =$ 

Scale-space filter bank

Tracking Through Scale Space

Using Lindeberg's Theory



Weight image

3D spatial kernel (DOG)

1D scale kernel (Epanechnikov)

$= E(x,\sigma)$

3D scale-space representation

Centered at current location and scale

Modes are blobs in the scale-space neighborhood

Need a mean-shift procedure that finds local modes in $E(x,\sigma)$

Recall:

Model: $\vec{q} = (q_1, \ldots, q_m)$ at y_0

Candidate: $\vec{p}(y) = (p_1(y), \ldots, p_m(y))$

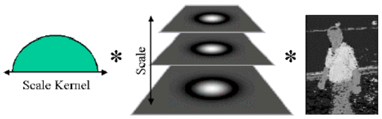
Color bin: $b(x)$

Pixel weight: $w(x) = \sqrt{\dfrac{q_{b(x)}}{p_{b(x)}(y_0)}}$

The likelihood that each candidate pixel belongs to the target

Outline of Approach

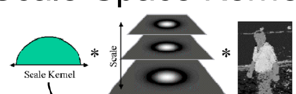
General Idea: build a "designer" shadow kernel that generates the desired DOG scale space when convolved with weight image w(x).



Change variables, and take derivatives of the shadow kernel to find corresponding mean-shift kernels using the relationship shown earlier.

Given an initial estimate (x_0, s_0), apply the mean-shift algorithm to find the nearest local mode in scale space. Note that, using mean-shift, we DO NOT have to explicitly generate the scale space.

Scale-Space Kernel



$$E(x,s) = \sum_s \boxed{H_s(s)} \sum_x \boxed{H_x(x,s)}\, w(x)$$

$$\boxed{H_s(s) = 1 - (s)^2} \qquad \boxed{H_x(x,s) = G(x;x_0,\sigma_1) - G(x;x_0,\sigma_2)}$$

$$\boxed{K_s(s) = 1} \qquad \boxed{K_x(x,s) = G(x;x_0,\sigma_1)/\sigma_1^2 - G(x;x_0,\sigma_2)/\sigma_2^2}$$

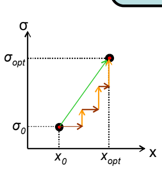
Tracking Through Scale Space

Applying Mean-Shift

Use interleaved spatial/scale mean-shift

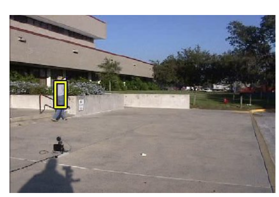
Spatial stage: Fix σ and look for the best *x*

Scale stage: Fix x and look for the best σ

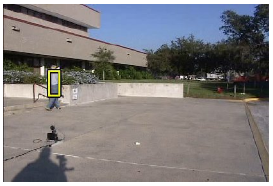


Iterate stages until convergence of *x* and σ

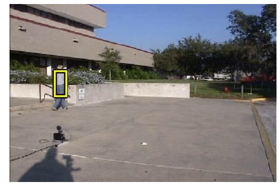
Sample Results: No Scaling



Sample Results: +/- 10% rule



Sample Results: Scale-Space



Tracking Through Scale Space

Results

Fixed-scale

± 10% scale adaptation

Tracking through scale space