Mean-Shift Object Tracking

Comaniciu, D.  Ramesh, V.  Meer, P.  *Kernel-based object tracking*, PAMI, May 2003
Non-Rigid Object Tracking

Many Slides from: Yaron Ukrainitz & Bernard Sarel
Mean-Shift Object Tracking

General Framework: Target Representation

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

Current frame
Mean-Shift Object Tracking

General Framework: Target Localization

1. Start from the position of the model in the current frame.
2. Search in the model’s neighborhood in the next frame.
3. Find the best candidate by maximizing a similarity function.
4. Repeat the same process in the next pair of frames.
Using MS for Tracking Objects

• Two Approaches
  – Create a color “likelihood” image, with pixels weighted by similarity to the desired color (best for unicolored objects)
  – Represent color distribution with a histogram. Use mean-shift to find region that has most similar distribution of colors.
Mean-shift on Weight Images

• 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.
Meanshift 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.

\[ \Delta x = \frac{\sum_a K(a-x) w(a) (a-x)}{\sum_a K(a-x) w(a)} \]
Seeking Mode of Location

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

• Two Approaches
  – Create a color “likelihood” image, with pixels weighted by similarity to the desired color (best for unicolored objects)
  – Represent color distribution with a histogram. Use mean-shift to find region that has most similar distribution of colors.

*(Kernel Based Object Tracking, by Comaniciu, Ramesh, Meer, PAMI 2003)*
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

Probability

1 2 3 . . . m

color

Kernel Based Object Tracking, by Comaniniu, Ramesh, Meer
Mean-Shift Object Tracking

PDF Representation

Target Model
(centered at 0)

Target Candidate
(centered at y)

\[ \vec{q} = \{ q_u \}_{u=1}^m \quad \sum_{u=1}^m q_u = 1 \]

\[ \vec{p}(y) = \{ p_u(y) \}_{u=1}^m \quad \sum_{u=1}^m p_u = 1 \]

Similarity Function:

\[ f(y) = f[\vec{q}, \vec{p}(y)] \]
Mean-Shift Object Tracking

Smoothness of Similarity Function

Similarity Function: \( f(y) = f[\tilde{p}(y), \tilde{q}] \)

Problem:
- Target is represented by color info only
- Spatial info is lost
- Large similarity variations for adjacent locations
- \( f(y) \) 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

\[
\{x_i\}_{i=1..n} \quad \text{Target pixel locations}
\]

\[ k(x) \] 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**

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

**Probability of feature \( u \) in candidate**

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

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

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

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

\[ \bar{q} \quad \bar{p}(y) \]

\[ f \left[ \bar{p}(y), \bar{q} \right] \]
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 \( y_0 \))

\[ f(y) \approx \frac{1}{2} \sum_{u=1}^{m} \sqrt{p_u(y_0)q_u} + \frac{1}{2} \sum_{u=1}^{m} p_u(y) \sqrt{\frac{q_u}{p_u(y_0)}} \]

Independent of \( y \)

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

Density estimate! (as a function of \( y \))

\[ C_h \sum_{i=1}^{n} w_ik \left( \frac{\| y - x_i \|^2}{h} \right) \]
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

\[
\frac{C_h}{2} \sum_{i=1}^{n} w_i k \left( \frac{y-x_i}{h} \right)^2
\]

= sought maximum

**Original Mean-Shift:**

Find mode of

\[
c \sum_{i=1}^{n} k \left( \frac{y-x_i}{h} \right)^2
\]

using

\[
y_1 = \frac{\sum_{i=1}^{n} x_i g \left( \frac{y_0-x_i}{h} \right)^2}{\sum_{i=1}^{n} g \left( \frac{y_0-x_i}{h} \right)^2}
\]

**Extended Mean-Shift:**

Find mode of

\[
c \sum_{i=1}^{n} w_i k \left( \frac{y-x_i}{h} \right)^2
\]

using

\[
y_1 = \frac{\sum_{i=1}^{n} x_i w_i g \left( \frac{y_0-x_i}{h} \right)^2}{\sum_{i=1}^{n} w_i g \left( \frac{y_0-x_i}{h} \right)^2}
\]
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 using

\[ y_1 = \frac{\sum_{i=1}^{n} x_i w_i g \left( \left\| \frac{y_0 - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} w_i g \left( \left\| \frac{y_0 - x_i}{h} \right\|^2 \right)} \]
Mean-Shift Object Tracking
Choosing the Kernel

A special class of radially symmetric kernels:

\[ K(x) = ck\left(\|x\|^2\right) \]

Epanechnikov kernel:

\[ k(x) = \begin{cases} 1-x & \text{if } \|x\| \leq 1 \\ 0 & \text{otherwise} \end{cases} \]

Uniform kernel:

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

\[ y_1 = \frac{\sum_{i=1}^{n} x_i w_i g\left(\frac{y_0 - x_i}{h}\right)}{\sum_{i=1}^{n} w_i g\left(\frac{y_0 - x_i}{h}\right)} \]

\[ y_1 = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} \]
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

Feature space: $16 \times 16 \times 16$ quantized RGB
Target: manually selected on 1st frame
Average mean-shift iterations: 4
Mean Shift Object Tracking

Advantages:

- Gradient Ascent
- Invariance to
  - Clutter & Camera Motion
  - Pose changes
  - Very fast
Qualitative Intuition

- Consider an object 60% red & 40% green
  \[ q_1 = 0.6 \text{ and } q_2 = 0.4, \]
  \[ q_i = 0 \text{ for all other } i \]

- MS performs weighted centre of mass computation at each iteration
- MS window will be biased towards red region, since they have higher weights.
Qualitative Intuition

- Now we will use Comaniciu et al. ’s weights
  \[ p_1 = 0.6 \text{ and } p_2 = 0.4, \quad p_i = 0 \text{ for all other } i \]

Weights

\[
\sqrt{\frac{q_m}{p_u(y_0)}}
\]
Qualitative Intuition

- Now we will use Comaniciu et al.’s weights
  \[ p_1 = 0.4 \text{ and } p_2 = 0.6, \quad p_i = 0 \text{ for all other } i \]

Weights

\[
\sqrt{\frac{q_m}{p_u(y_0)}}
\]
Qualitative Intuition

- Now we will use Comaniciu et al. ’s weights
  \( p_1 = 0.8 \) and \( p_2 = 0.2 \), \( p_i = 0 \) for all other \( i \)

Weights

\[
\sqrt{\frac{q_m}{p_u(y_0)}}
\]
Kernel Based Tracking

- Maximize similarity between target and candidate models in feature space.
- Models based on weighted histogram
- Weight obtained using a kernel ‘$K$’
  - Higher weight to central object pixels
- Gradient ascent to find the local maximum
Kernel Object Tracking
Bhattacharyya Coefficient

Bhattacharya coefficient map

\[ f(y_0) = \sum_{u=1}^{m} \sqrt{P_u(y_0)q_u} \]
Tracking in Feature Space

Target

Current Frame

Next Frame
Tracking in Feature Space

\[
\text{Weight} = \sqrt{\frac{q_m}{p_u(y_0)}}
\]
Tracking in Feature Space

Weight = \frac{q_m}{\sqrt{p_u(y_0)}}
Tracking in Feature Space

Weight = \sqrt{\frac{q_m}{p_u(y_0)}}
Tracking in Feature Space

\[ x_3 = \text{COM} \]

Weight = \[ \sqrt{\frac{q_m}{p_u(y_0)}} \]
Tracking Through Scale Space

Motivation

Spatial localization for several scales

Previous method

Simultaneous localization in space and scale

This method

Mean-shift Blob Tracking through Scale Space, by R. Collins
Lindeberg’s Theory
Selecting the best scale for describing image features

Scale-space representation  Differential operator applied  50 strongest responses
Lindeberg’s Theory

The Laplacian operator for selecting blob-like features

\[ f(x) = G(x; \sigma) \]

\[ \nabla^2 G(x; \sigma) \]

\[ LOG(x; \sigma) \]

\[ \forall x \in f, \forall \sigma_{1..k} : L(x, \sigma) = LOG(x; \sigma) \ast f(x) \]

2D LOG filter with scale \( \sigma \)

3D scale-space representation

Best features are at \((x, \sigma)\) that maximize \(L\)
Lindeberg’s Theory
Multi-Scale Feature Selection Process

Original Image

Convolve

3D scale-space function

Maximize

$\text{Convolve} \rightarrow \text{3D scale-space function} \rightarrow \text{Maximize}$

$L(x, \sigma) = \text{LOG}(x; \sigma) * f(x)$

250 strongest responses
(Large circle = large scale)
Tracking Through Scale Space
Approximating LOG using DOG

\[ \text{LOG}(x; \sigma) \approx \text{DOG}(x; \sigma) = G(x; \sigma) - G(x; 1.6\sigma) \]

- 2D LOG filter with scale \( \sigma \)
- 2D DOG filter with scale \( \sigma \)
- 2D Gaussian with \( \mu=0 \) and scale \( \sigma \)
- 2D Gaussian with \( \mu=0 \) and scale 1.6\( \sigma \)

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

3D spatial kernel

\[ K(x, \sigma) = \scalebox{0.8}{\text{SCALE-SPACE FILTER BANK}} \]

DOG filters at multiple scales
Tracking Through Scale Space
Using Lindeberg’s Theory

Weight image

Recall:
- Model: \( \tilde{q} = (q_1, \ldots, q_m) \) at \( y_0 \)
- Candidate: \( \tilde{p}(y) = (p_1(y), \ldots, p_m(y)) \)
- Color bin: \( b(x) \)
- Pixel weight: \( w(x) = \sqrt{\frac{q_{b(x)}}{p_{b(x)}(y_0)}} \)

Centered at current location and scale

1D scale kernel (Epanechnikov)

3D scale-space representation

Modes are blobs in the scale-space neighborhood

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

The likelihood that each candidate pixel belongs to the target
Tracking Through Scale Space

Example

Image of 3 blobs

A slice through the 3D scale-space representation
Tracking Through Scale Space
Applying Mean-Shift

Use interleaved spatial/scale mean-shift

Spatial stage:
Fix $\sigma$ and look for the best $x$

Scale stage:
Fix $x$ and look for the best $\sigma$

Iterate stages until convergence of $x$ and $\sigma$
Tracking Through Scale Space

Results

Fixed-scale

± 10% scale adaptation

Tracking through scale space
Robust Tracking

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background Noise</td>
<td>FG/BG separation</td>
</tr>
<tr>
<td>Partial occlusion</td>
<td>Fragments</td>
</tr>
<tr>
<td>Background clutter Weight</td>
<td>Background based</td>
</tr>
<tr>
<td>Appearance changes</td>
<td>Model Update</td>
</tr>
<tr>
<td>Scale change</td>
<td>Marginal Histogram</td>
</tr>
</tbody>
</table>


Foreground separation

To accurately model the object using histogram of pixels in object and background window leading to robustness to clutter.

\[ L(x_i) = \log \frac{\max(h_o(x_i), \varepsilon)}{\max(h_b(x_i), \varepsilon)} \]

Thresholding \( L(x_i) \) classifies each pixel as foreground / background.
Fragment based tracking

Voting & Re-initialization

- Select the best-tracked fragment (Bhattacharyya coefficient)
- Object position taken from this fragment
- Move all other fragments back according to the initial relative position

\[ x_{obj} = x_J - \Delta_J \]
\[ J = \arg \max_f \left[ \rho(p_{f,t}(y), q_{f,t}) \right] \]
\[ x_f = x_{obj} + \Delta_f \quad (f = 1...F, f \neq J) \]
Results

Partial Occlusion

Kernel Tracking

Proposed Approach
Model Updation

• Commonly used: Forgetting factor

• Updating: Replace the target if
  – Percentage of foreground pixels > T1
  – Similarity of fragment with previous frame > T2

• Faster & no tuning required

• Use of fragments
  – All fragments may not be updated together

• Assumes that there is no abrupt change of illumination compared to the frame rate (25fps)
Results

Model update to tackle Illumination change

Kernel Tracking

Proposed Approach
Foreground tracking

Use the foreground separation

Each histogram bin is assigned a weight

\[ \lambda_u = \max \left( \frac{1}{1 + \exp \left( - \frac{L_u - a}{b} \right)} , 0.1 \right) \]

New model defined by

\[ p_u = \frac{1}{C} \sum_{i=1}^{n} k(x_i) \lambda_u \delta[b(x_i - u)] \]
Results

Background clutter

- Kernel Tracking
- Proposed Approach
Tracking with scale changes

Color histograms – unsuited for this

Search Window

Target Object
80% Blue
20% Green

Correct Scale
Similarity : 0.80

Incorrect Scale
Similarity : 0.89

Incorrect Scale
Similarity : 0.89
Tracking with scale changes

Incorporating edge information
Convolution with horizontal and vertical Prewitt operators
Joint histogram of horizontal and vertical edge images

Non uniform bin sizes
Histogram equalization
Each bin has roughly the same number of pixels

Joint edge & color histogram
One bin corresponding to a particular RGB value, vertical & horizontal response

Problems
Marginal histograms

Joint Histograms

– Number of bins increases exponentially
– Similarity measure drops quickly
– Speed, memory constraints

Marginal Densities?
Maximize the product

\[ \rho = \rho_1 \times \rho_2 \]

\[ \rho_1(y_0) = \sum_u \sqrt{p_u(y_0)q_u} \quad \rho_2(y_0) = \sum_m \sqrt{s_m(y_0)t_m} \]

\( \rho_1 \) and \( \rho_2 \): Similarity Coeff. using marginal densities
Marginal Mean Shift

We need to maximize

\[ \rho(y_0) = \left[ \sum_u \sqrt{p_u(y_0)q_u} \right] \left[ \sum_m \sqrt{s_m(y_0)t_m} \right] \]

Taking Taylor Series expansion about \( y_0 \)

\[ \rho(y) = \frac{\rho_1(y_0)}{2} \sum_m s_m(y) \sqrt{\frac{t_m}{s_m(y_0)}} + \frac{\rho_2(y_0)}{2} \sum_u p_u(y) \sqrt{\frac{q_m}{p_u(y_0)}} \]

Substituting for \( p_u(y) \) and \( s_m(y) \)

\[ \rho(y) = \sum_x w_i K(.) \]

We once again have a weighted KDE

where

\[ w_i = \frac{C_2 \rho_1(y_0)w_i^2}{2} + \frac{C_1 \rho_2(y_0)w_i^1}{2} \]
Marginal Mean Shift

Target, Candidate models

Bhattacharya Coefficient (To be maximized)

Pixel weight (for tracking)

\[ q, p \]

\[ \{q_1q_2\}, \{p_1p_2\} \]

\[ \rho \]

\[ \rho_1 \times \rho_2 \]

\[ C_2 \rho_1 w_i^2 + \frac{C_1 \rho_2 w_i^1}{2} \]

Sum of individual weights!!

- Over-all mean shift vector is obtained by weighing each of the partial mean shift vectors with the other Bhattacharyya coefficient.
- Hence one coefficient being low suppresses the other shift vector so that it can be maximized first.
Results

Scale Change & occlusion