

Mean-Shift Blob Tracking through Scale Space

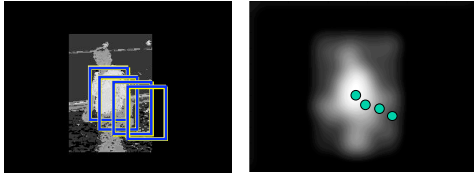
Robert Collins, CVPR '03

Abstract

- Mean-shift tracking
- Choosing scale of kernel is an issue
- Scale-space feature selection provides inspiration
- Perform mean-shift with scale-space kernel to optimize for blob location and scale

Nice Property

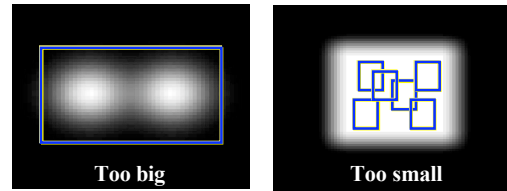
Running mean-shift with kernel K on weight image w is equivalent to performing gradient ascent in a (virtual) image formed by convolving w with some "shadow" kernel H .



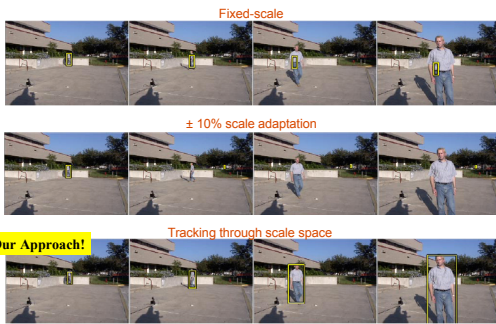
$$\Delta x = \frac{\sum_a K(a-x) w(a)}{\sum_a K(a-x) w(a)} \rightarrow -c \nabla_x [\sum_a H(a-x) w(a)]$$

Size Does Matter!

Mean-shift is related to kernel density estimation, aka Parzen estimation, so choosing correct scale of the mean-shift kernel is important.



Size Does Matter



Some Approaches to Size Selection

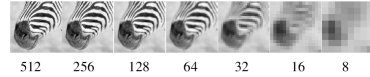
- Choose one scale and stick with it.
- Bradski's CAMSHIFT tracker computes principal axes and scales from the second moment matrix of the blob. Assumes one blob, little clutter.
- CRM adapt window size by +/- 10% and evaluate using Battacharyya coefficient. Although this does stop the window from growing 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.



Scale-Space Theory

Scale Space

Basic idea: different scales are appropriate for describing different objects in the image, and we may not know the correct scale/size ahead of time.



Scale Selection

Scale Selection Principle (T. Lindeberg):

In the absence of other evidence, assume that a scale level, at which some (possibly non-linear) combination of normalized derivatives assumes a local maximum over scales, can be treated as reflecting a characteristic length of a corresponding structure in the data.



What are normalized derivatives?: Example (using 2nd order derivatives):

$$\sigma^{n+m} \frac{\partial^{(n+m)} f}{\partial x^n \partial x^m} \quad \sigma^2 \nabla^2 f = \sigma^2 \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)$$

“Laplacian” operator.

Laplacian Operator and LoG

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

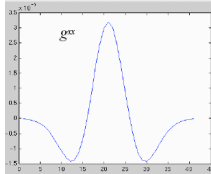
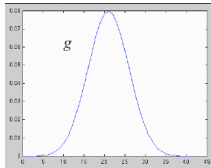
Just another linear filter.

$$\nabla^2 (f(x, y) \otimes G(x, y)) = \nabla^2 G(x, y) \otimes f(x, y)$$

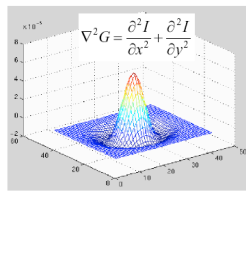
Laplacian of
Gaussian-filtered image

Laplacian of Gaussian (LoG)
-filtered image

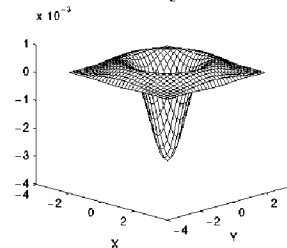
LoG Operator



Second derivatives:
Laplacian



$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$$



Robert Collins
CSE598G

Approximating LoG with DoG

LoG can be approximate by a Difference of two Gaussians (DoG) at different scales

$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$ Best approximation when: $\sigma_2/\sigma_1 = 1,6$
but more convenient if: $\sigma_1 = \frac{\sigma}{\sqrt{2}}, \sigma_2 = \sqrt{2}\sigma$

We will come back to DoG later

Robert Collins
CSE598G

Local Scale Space Maxima

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

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

DnormL is a normalized Laplacian of Gaussian operator $\sigma^2 \text{LoG}_\sigma$

Example for blob detection

Robert Collins
CSE598G

Extrema in Space and Scale

Robert Collins
CSE598G

Example: Blob Detection

Robert Collins
CSE598G

Why Normalized Derivatives

Let image $f(x)$ contain a blob with scale σ , centered at x_0 , so that $L(x_0; \sigma)$ is a local maximum in the scale space representation of f . Consider a new image $g(x) = f(x/s)$ that is just a spatially scaled version of f . We would then expect $L(x_0/s; \sigma)$ to be a local maxima in the scale space representation of g , which it is. However, the magnitudes of the two filter responses are different, which makes it hard to compare feature descriptions across different scales. Assuming without loss of generality that the blob is centered at zero, and performing a change of variables $y = sx$ on $L(0; \sigma)$ we find that

Laplacian of Gaussian (LOG)

$$L(0; \sigma) = \sum_y \frac{2\sigma^2 - \|y\|^2}{2\pi\sigma^4} e^{-\frac{\|y\|^2}{2\sigma^2}} f(x) \quad (10)$$

$$= \sum_y \frac{2\sigma^2 - \|y/s\|^2}{2\pi\sigma^4} e^{-\frac{\|y/s\|^2}{2\sigma^2}} f(y/s) \frac{1}{s^2} \quad (11)$$

$$= s^2 \sum_y \frac{2(\sigma/s)^2 - \|y\|^2}{2\pi(\sigma/s)^4} e^{-\frac{\|y\|^2}{2(\sigma/s)^2}} g(y) \quad (12)$$

$$= s^2 L(0; \sigma/s) \quad (13)$$

Amplitude of LOG response decreases with greater smoothing

Robert Collins
CSE598G

Interesting Observation

If we approximate the LOG by a Difference of Gaussian (DOG) filter we do not have to normalize to achieve constant amplitude across scale.

$$DOG(x; \sigma) = \frac{1}{2\pi\sigma^2/1,6} e^{-\frac{\|x\|^2}{2\sigma^2/1,6}} - \frac{1}{2\pi\sigma^2(1,6)} e^{-\frac{\|x\|^2}{2\sigma^2(1,6)}} \quad (14)$$

Why does the LOG filter have to be normalized to give an invariant response across scales, while its approximation, the DOG filter, does not? The DOG operator does not approximate the LOG in the traditional sense of $DOG(x) \approx LOG(x)$. Rather, the two functions are approximately equal up to a scale factor, such that $DOG(x)/LOG(x) \approx \text{constant}$. Specifically, by considering the values of the two functions at the origin $x = 0$, it is easy to see that

$$DOG(x; \sigma) \approx 0,4875 \sigma^2 LOG(x; \sigma)$$

Another Explanation

The relationship between D and $\sigma^2 \nabla^2 G$ can be understood from the heat diffusion equation (parameterized in terms of σ rather than the more usual $t = \sigma^2$):

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G.$$

From this, we see that $\nabla^2 G$ can be computed from the finite difference approximation to $\partial G / \partial \sigma$, using the difference of nearby scales at $k\sigma$ and σ :

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

and therefore,

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G.$$

Lowe, IJCV 2004 (Sift key paper)

Anyhow...

Scale space theory says we should look for modes in a DoG - filtered image volume. Let's just think of the spatial dimensions for now

We want to look for modes in DoG-filtered image, meaning a weight image convolved with a DoG filter.

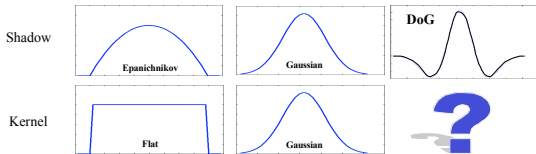
Insight: if we view DoG filter as a shadow kernel, we could use mean-shift to find the modes. Of course, we'd have to figure out what mean-shift kernel corresponds to a shadow kernel that is a DoG.

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



$$h'(r) = -c k(r)$$

Kernel related to DoG Shadow

shadow

$$DOG(x; \sigma) = \frac{1}{2\pi\sigma^2/1.6} e^{-\frac{|x|^2}{2\sigma^2/1.6}} - \frac{1}{2\pi\sigma^2(1.6)} e^{-\frac{|x|^2}{2\sigma^2(1.6)}}$$

$$= G(x; x_0, \sigma_1) - G(x; x_0, \sigma_2) \quad \text{where } \sigma_1 = \sigma/\text{sqrt}(1.6) \\ \sigma_2 = \sigma^*\text{sqrt}(1.6)$$

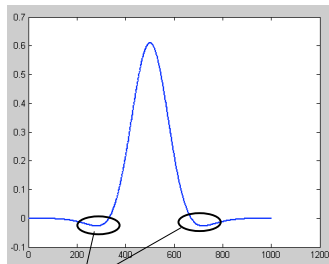
kernel

$$K_x(x, s) = G(x; x_0, \sigma_1) / \sigma_1^2 - G(x; x_0, \sigma_2) / \sigma_2^2$$

$$h'(r) = -c k(r)$$

Kernel related to DoG Shadow

$$K_x(x, s) = G(x; x_0, \sigma_1) / \sigma_1^2 - G(x; x_0, \sigma_2) / \sigma_2^2$$



some values are negative. Is this a problem?

Umm... Yes it is

Dealing with Negative Weights

Show little demo with neg weights

mean-shift will sometimes converge to a valley rather than a peak.

The behavior is sometimes even stranger than that (step size becomes way too big and you end up in another part of the function).

Why we might want negative weights

Given an n-bucket histogram $\{m_i | i=1, \dots, n\}$ and data histogram $\{d_i | i=1, \dots, n\}$, CRM suggest measuring similarity using the Battacharyya Coefficient

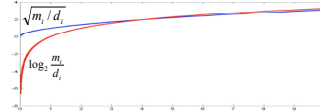
$$\rho = \sum_{i=1}^n \sqrt{m_i \times d_i}$$

They use the mean-shift algorithm to climb the spatial gradient of this function by weighting each pixel falling into bucket i the term at right

$$w_i = \sqrt{m_i / d_i}$$

Note the similarity to the likelihood ratio function

$$w_i \approx \log_2 \frac{m_i}{d_i}$$



Why we might want negative weights

$$w_i \approx \log_2 \frac{m_i}{d_i} \quad \text{Using the likelihood ratio makes sense probabilistically.}$$

For example: using mean-shift with uniform kernel on weights that are likelihood ratios:

$$\sum_a \log \frac{m_i}{d_i}$$

would then be equivalent to using KL divergence to measure difference between model m and data d histograms.

$$\sum_a \log \frac{m_i}{d_i} = n \sum_{i=1}^n d_i \log \frac{m_i}{d_i}$$

sum over pixels sum over buckets with value i
(note, n*d_i pixels have value i)

Analysis: Scaling the Weights

recall: mean shift offset

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

what if w(a) is scaled to c*w(a)?

$$\frac{\sum_a K(a-x) \cancel{c} w(a)(a-x)}{\sum_a K(a-x) \cancel{c} w(a)}$$

So mean shift is invariant to scaled weights

Analysis: Adding a Constant

what if we add a constant to get w(a)+c?

$$\begin{aligned} \Delta_x &= \frac{\sum K(a-x)(w(a)+c)(a-x)}{\sum K(a-x)(w(a)+c)} \\ &= \frac{\sum K(a-x)w(a)(a-x) + \cancel{c} \sum K(a-x)(a-x)}{\sum K(a-x)(w(a)+c)} \quad K(x) = K(-x) \\ &= \frac{\sum K(a-x)w(a)(a-x)}{\sum K(a-x)(w(a)+c)} \end{aligned}$$

So mean shift is not invariant to an added constant

This is annoying!

Adding a Constant

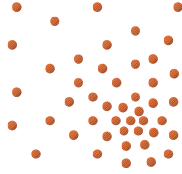
result: It isn't a good idea to just add a large positive number to our weights to make sure they stay positive.

show little demo again, adding a constant.

Another Interpretation of Mean-shift Offset

Thinking of offset as a weighted center of mass doesn't make sense for negative weights.

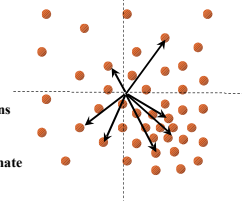
$$\Delta x = \frac{\sum_a \overset{\text{weight}}{K(a-x)} \overset{\text{point}}{w(a)(a-x)}}{\sum_a K(a-x) w(a)}$$



Another Interpretation of Mean-shift Offset

Think of each offset as a vector, which has a direction and magnitude.

$$\Delta x = \frac{\sum_a \overset{\text{vector}}{K(a-x) w(a) (a-x)}}{\sum_a K(a-x) w(a)}$$



Note, a negative weight now just means a vector in the opposite direction.

Interpret mean shift offset as an estimate of the "average" vector.

Note: numerator interpreted as sum of directions and magnitudes
But denominator should just be sum of magnitudes (which should all be positive)

Absolute Value in Denominator

We now see how to modify the mean-shift equation (1) to make sense for negative weights. The numerator of that equation votes for both the magnitude and direction of point-wise offset vectors, so the negative weights should stay. However, the denominator normalizes by the overall total magnitude of the votes, and therefore we must sum only the magnitude (the absolute value) of each term. The modified equation is

$$\Delta x = \frac{\sum_a K(a-x) w(a) (a-x)}{\sum_a |K(a-x) w(a)|} \quad (7)$$

With this modification, mean-shift remains well-behaved on images that contain negative pixel values.

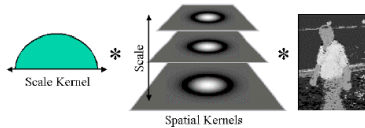
or does it?

back to the demo

There can be oscillations when there are negative weights.
I'm not sure what to do about that.

Outline of Scale-Space Mean Shift

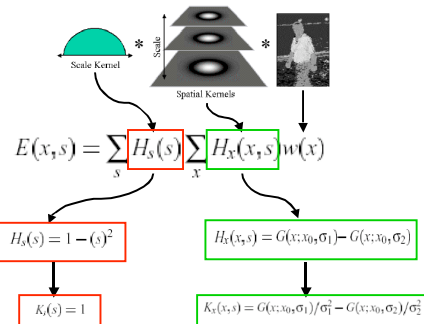
General Idea: build a "designer" shadow kernel that generates the desired DOG scale space when convolved with weight image $w(x)$.



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



Mean-Shift through Scale Space

- 1) Input weight image $w(a)$ with current location x_0 and scale s_0
- 2) Holding s fixed, perform spatial mean-shift using equation

$$\Delta_x = \frac{\sum_s H_s(s) \sum_a k_s(a - x_0, s) w(a) (a - x_0)}{\sum_s H_s(s) \sum_a [k_s(a - x_0, s) w(a)]}$$

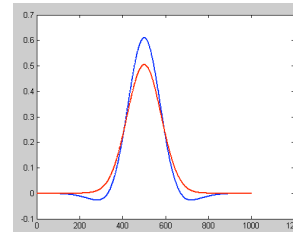
- 3) Let x be the location computed from step 2. Holding x fixed, perform mean-shift along the scale axis using equation

$$s' = \frac{\sum_s \sum_a H_s(x, s) w(a) s}{\sum_s \sum_a H_s(x, s) w(a)}$$

- 4) Repeat steps 2 and 3 until convergence.

Second Thoughts

Rather than being strictly correct about the kernel K , note that it is approximately Gaussian.



blue: Kernel associated with shadow kernel of DoG with sigma σ

red: Gaussian kernel with sigma $\sigma/\sqrt{1.6}$

so why not avoid issues with negative kernel by just using a Gaussian to find the spatial mode?

scaledemo.m

interleave Gaussian spatial mode finding with 1D DoG mode finding.

Summary

- Mean-shift tracking
- Choosing scale of kernel is an issue
- Scale-space feature selection provides inspiration
- Perform mean-shift with scale-space kernel to optimize for blob location and scale

Contributions

- Natural mechanism for choosing scale WITHIN mean-shift framework
- Building “designer” kernels for efficient hill-climbing on (implicitly-defined) convolution surfaces