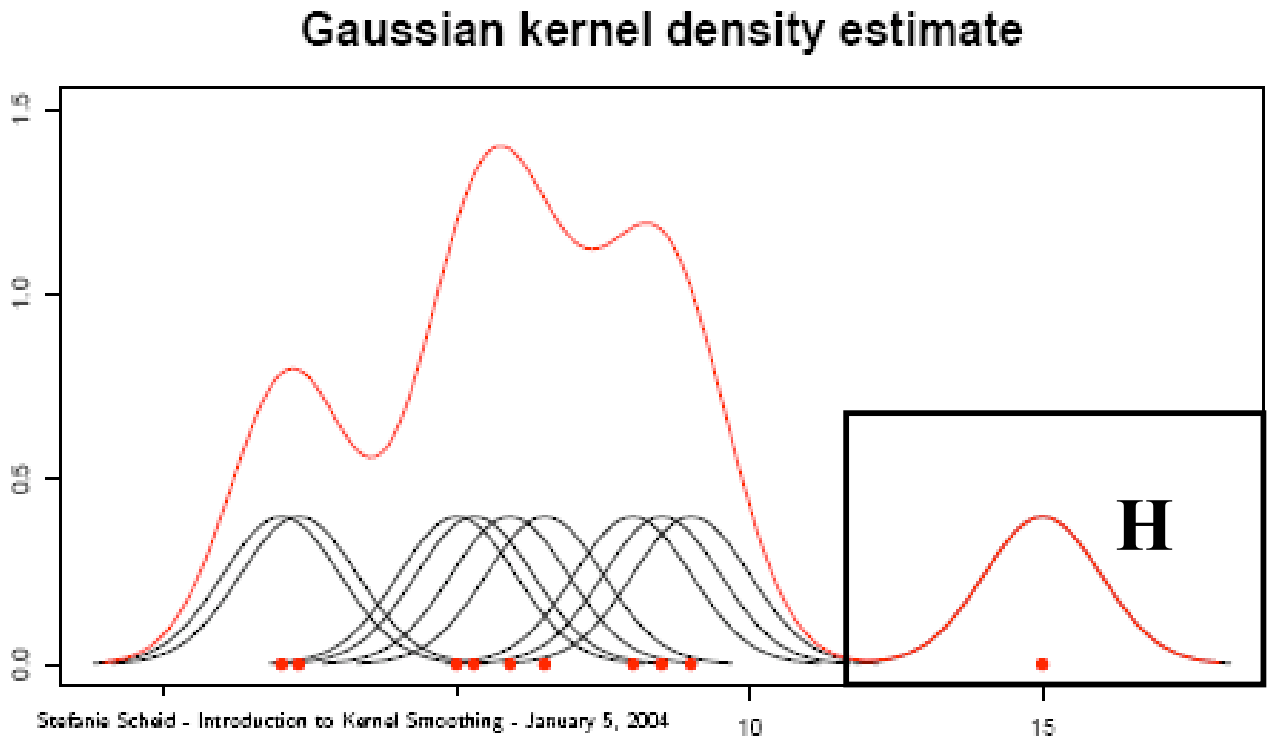
More on Mean-shift

R.Collins, CSE, PSU

CSE598G Spring 2006

Recall: Kernel Density Estimation

Given a set of data samples x_i; $i=1...n$
Convolve with a kernel function H to generate a smooth function $f(x)$
Equivalent to superposition of multiple kernels centered at each data point



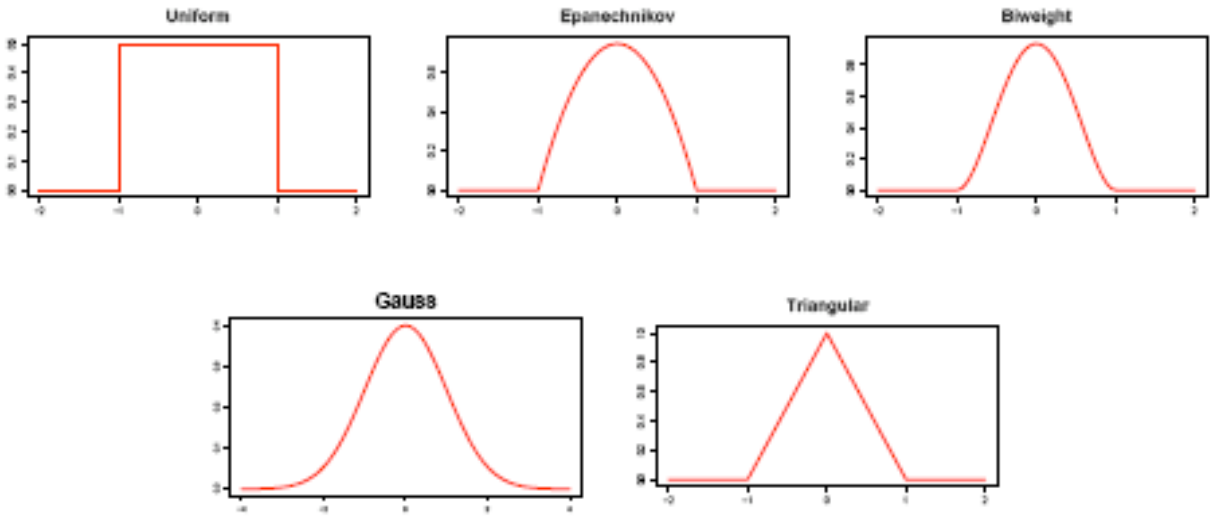
Recall: Kernel Density Estimation

For kernel H with bandwidth h, estimated function value *f* at location *x* is

$$f(x) = \sum H(x_i - x)$$

superposition of kernels centered at x_i

some sample kernels:

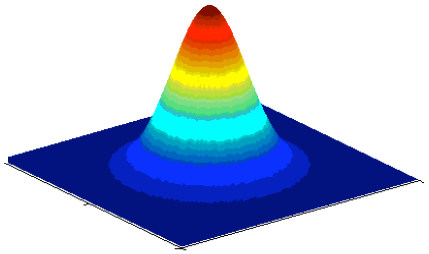


Stefanie Scheid - Introduction to Kernel Smoothing - January 5, 2004

Robert Collins
CSE598G

Radially Symmetric Kernels

Height at point is function only of distance from center



Can be written in terms of a 1D Profile function that is is a function of the radius (we will use squared radius below)

$$H(x_i - x) \equiv h(\|x_i - x\|^2)$$

$$\equiv h((x_i - x)^T (x_i - x))$$

$$\equiv h(r) \qquad r(x) = (x_i - x)^T (x_i - x)$$

Robert Collins
CSE598G

Kernel-Shadow Pairs

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

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

Examples

Epanichnikov $h(r) = 1 - r \quad h'(r) = -1 \Rightarrow k(r) = 1$ **Flat**

Biweight $h(r) = (1-r)^2 \quad h'(r) = -2(1-r) \Rightarrow k(r) = 1 - r$ **Epanichnikov**

Gaussian $h(r) = e^{-\beta r} \quad h'(r) = -\beta e^{-\beta r} \Rightarrow k(r) = e^{-\beta r}$ **Gaussian**

self-replicating!

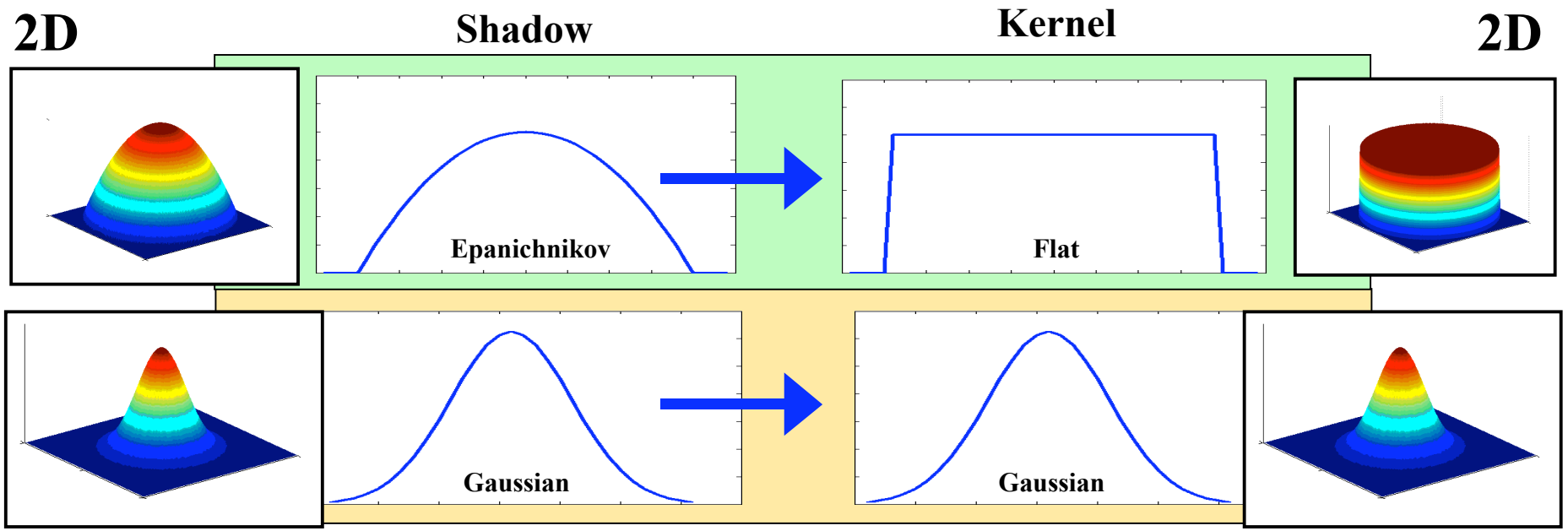
Robert Collins
CSE598G

Kernel-Shadow Pairs

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

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

Examples



Robert Collins
CSE598G

Mean-Shift and Gradient Ascent

We will derive an explicit equation relating the mean-shift procedure using kernel K with gradient ascent on the KDE surface formed by using the shadow kernel H.

$$\text{KDE} \quad f(x) = \sum H(x_i - x)$$

$$= \sum h(\|x_i - x\|^2)$$ **rewrite using profile function**

gradient of KDE

$$\nabla f(x) = \nabla \sum h(\|x_i - x\|^2) = \sum \nabla h(\|x_i - x\|^2)$$

Robert Collins
CSE598G

Mean-Shift and Gradient Ascent

$$\nabla f(x) = \nabla \sum h(\|x_i - x\|^2) = \sum \nabla h(\|x_i - x\|^2)$$

Sidebar derivation:

$$\nabla h = \frac{\partial h}{\partial x} = \frac{\partial h(r)}{\partial x}$$

change of variables

$$r(x) = (x_i - x)^T (x_i - x)$$

$$= \frac{\partial h}{\partial r} \frac{\partial r(x)}{\partial x}$$

chain rule

$$= h'(r) \frac{\partial}{\partial x} (x_i^T x_i - 2x^T x_i + x^T x)$$

$$= h'(r) [-2x_i + 2x]$$

$$= -2h'(r)(x_i - x)$$

$$= -2h'(\|x_i - x\|^2)(x_i - x)$$

change variables back

Robert Collins
CSE598G

Mean-Shift and Gradient Ascent

We will derive an explicit equation relating the mean-shift procedure using kernel K with gradient ascent on the KDE surface formed by using the shadow kernel H.

KDE $f(x) = \sum H(x_i - x)$

$= \sum h(\|x_i - x\|^2)$ **rewrite using profile function**

gradient of KDE

$\nabla f(x) = \sum \nabla h(\|x_i - x\|^2)$

$= -2 \sum h'(\|x_i - x\|^2)(x_i - x)$ **change of vars + chain rule**

$= c \sum k(\|x_i - x\|^2)(x_i - x)$ **definition of kernel shadow pairs h'(r) = - c k(r)**

$= c \sum K(x_i - x)(x_i - x)$ **rewrite as kernel K**

Robert Collins
CSE598G

Mean-Shift and Gradient Ascent

cont.

$$\nabla f(x) = c \sum K(x_i - x)(x_i - x)$$

$$\frac{\nabla f(x)}{c} = \sum K(x_i - x) x_i - \left[\sum K(x_i - x)\right] x$$

x this does not depend on i, so can come out of the summation

call this p(x).
It is another KDE, using kernel K instead of H

$$= \left[\sum K(x_i - x) x_i - p(x)\, x\right] \left[\frac{p(x)}{p(x)}\right]$$

this is just 1 !

$$\frac{\nabla f(x)}{c\, p(x)} = \frac{\sum K(x_i - x) x_i}{\sum K(x_i - x)} - x$$

this is the relationship we wanted to derive.

Robert Collins
CSE598G

Mean-Shift and Gradient Ascent

cont.

weighted center of mass

$$\frac{\nabla f(x)}{c\,p(x)} = \frac{\sum K(x_i - x)\,x_i}{\sum K(x_i - x)} - x$$

mean shift vector

$$\Big/ \quad \frac{\nabla \sum H(x_i - x)}{c \sum K(x_i - x)}$$

- same direction as gradient of KDE_H
- step size is inversely proportional to KDE_K
- mean-shift performs gradient ascent with adaptive step size

Robert Collins
CSE598G

Generalizing to Weighted Points

Let each point sample x_i have an associated <u>nonnegative</u> weight $w(x_i)$

Can rederive the equations with $w(x_i)$ factors in them:

$$f(x) = \sum H(x_i - x)\, w(x_i)$$

KDE using shadow kernel H

$$p(x) = \sum K(x_i - x)\, w(x_i)$$

KDE using kernel K

$$\frac{\nabla f(x)}{c\, p(x)} = \frac{\sum K(x_i - x)\, w(x_i)\, x_i}{\sum K(x_i - x)\, w(x_i)} - x$$

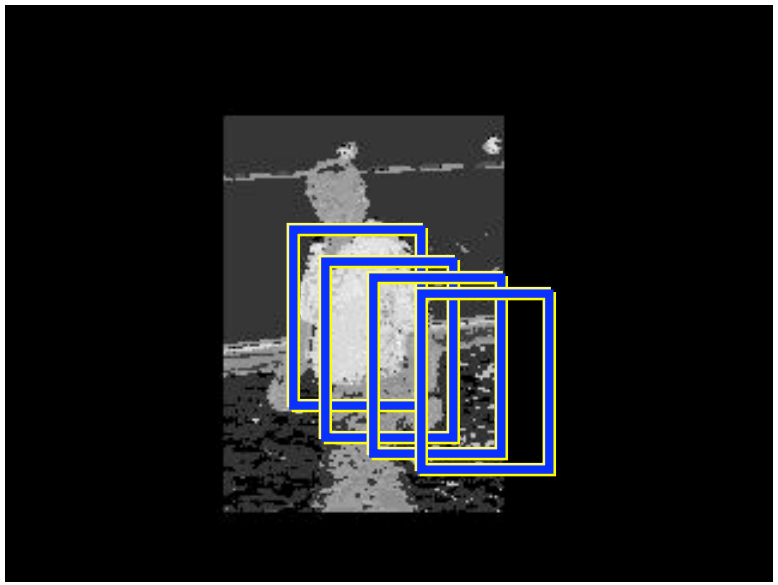
mean shift vector is still a gradient ascent process

This is important for running on images. Since pixels form a lattice, spatial density of samples is fixed, so need a weight to denote sample density at each point.

Robert Collins
CSE598G

Mean-Shift Tracking

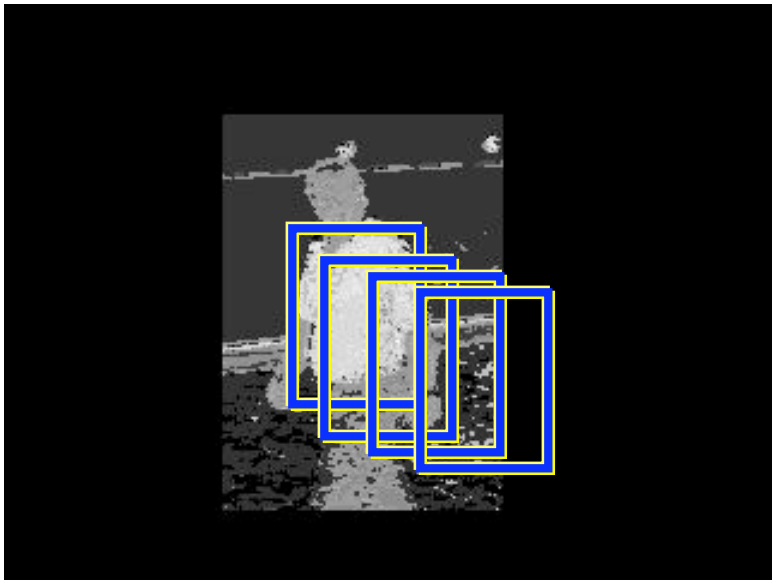
Let pixels form a uniform grid of data points, each with a weight (pixel value) proportional to the "likelihood" that the pixel is on the object we want to track. Perform standard mean-shift algorithm using this weighted set of points.

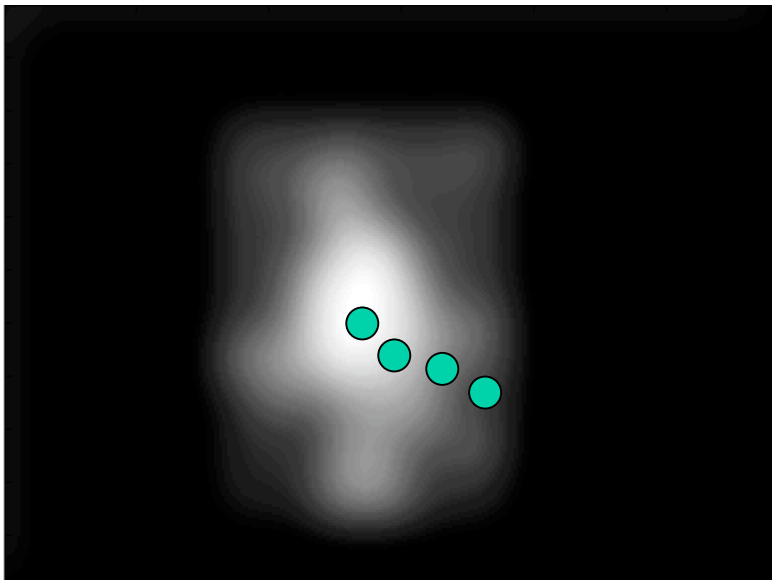


$$\Delta x = \frac{\sum_i K(x_i - x)\, w(x_i)\, (x_i - x)}{\sum_i K(x_i - x)\, w(x_i)}$$

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.





Computational savings: only have to compute convolution values at the relatively small number of points you visit on the path to the mode.

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

Claim: these weight images are all the mean-shift algorithm "sees", whether they be explicitly computed (e.g. Bradski) or implicitly computed (e.g. Comaniciu, Ramesh and Meer).

Explicit Weight Images

histogram backprojection

histogram is an empirical estimate of p(color | object) = p(c | o)

Bayes rule says p(o | c) = p(c | o) p(o) / p(c)

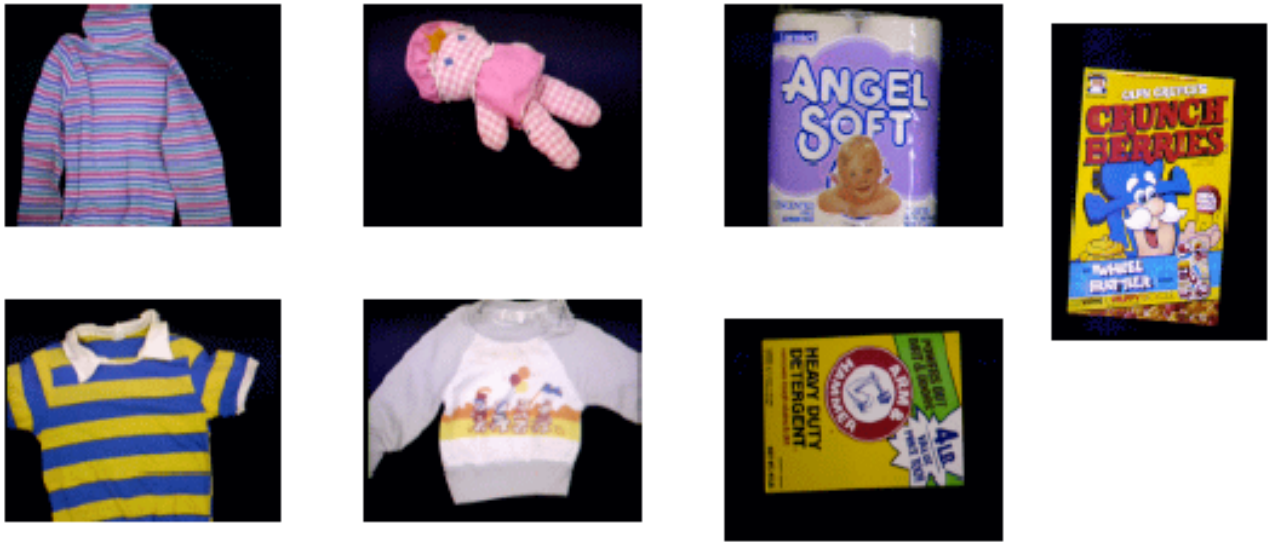
Simplistic approx: assume (p(o)/p(c)) is constant. Then p(o|c) = p(c|o).

Use histogram h as a lookup table to set pixel values in the weight image. (if pixel maps to bucket i in histogram h, set weight for that pixel to h(i)

Sidebar: Swain and Ballard, 1991

Using color histograms for recognition.

- Works surprisingly well
- In the first paper (1991), 66 objects could be recognized almost without errors



[Swain & Ballard, 1991]

Swain and Ballard

- Histogram backprojection:
  - „Where in the image are the colors being looked for?"
  - Query: object with histogram M
  - Given: image with histogram I
  - Compute the „ratio histogram":

$$R_i = \min\left(\frac{M_i}{I_i}, 1\right)$$

    - R reveals how important an object color is, relative to the current image.
    - This value is projected back into the image (i.e. the image values are replaced by the values of R that they index).
    - The result image is convolved with a circular mask the size of the target object.
    - Peaks in the convolved image indicate detected objects.

does this sound familiar?

Swain and Ballard

Object Localization

- Example result after backprojection
  - Looking for blue pullover...

Note: relationship between recognition and tracking. This will come up again later.

[Swain & Ballard, 1991]

Sidebar: Jones and Rehg, 2002

"Statistical Color Models with Application to Skin Detection", M. J. Jones and J. M. Rehg, *Int. J. of Computer Vision*, 46(1):81-96, Jan 2002

General Idea:

- **Learn skin color distribution from examples**
- **Learn distributions of skin and nonskin color**
- **Nonparametric distributions: color histograms**
- **Bayesian classification of skin pixels**

Robert Collins
CSE598G

Learning from Examples

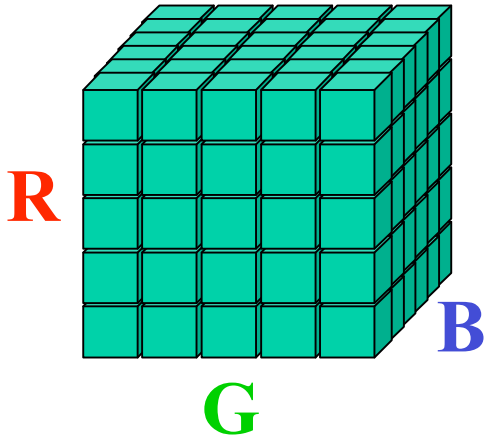
First, have your grad student hand label thousands of images from the web

$$P(\text{rgb} \mid \text{skin}) = \frac{\text{number of times rgb seen for a skin pixel}}{\text{total number of skin pixels seen}}$$

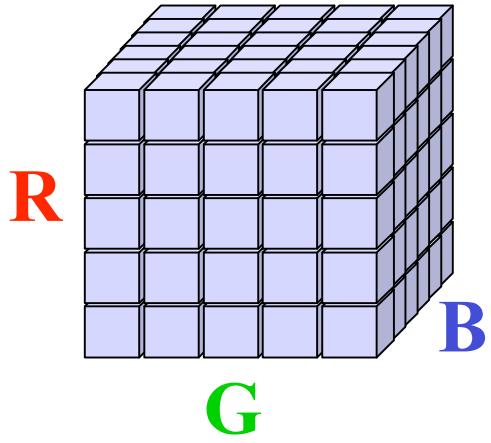
$$P(\text{rgb} \mid \text{not skin}) = \frac{\text{number of times rgb seen for a non-skin pixel}}{\text{total number of non-skin pixels seen}}$$

These statistics stored in two 32x32x32 RGB histograms

Skin histogram

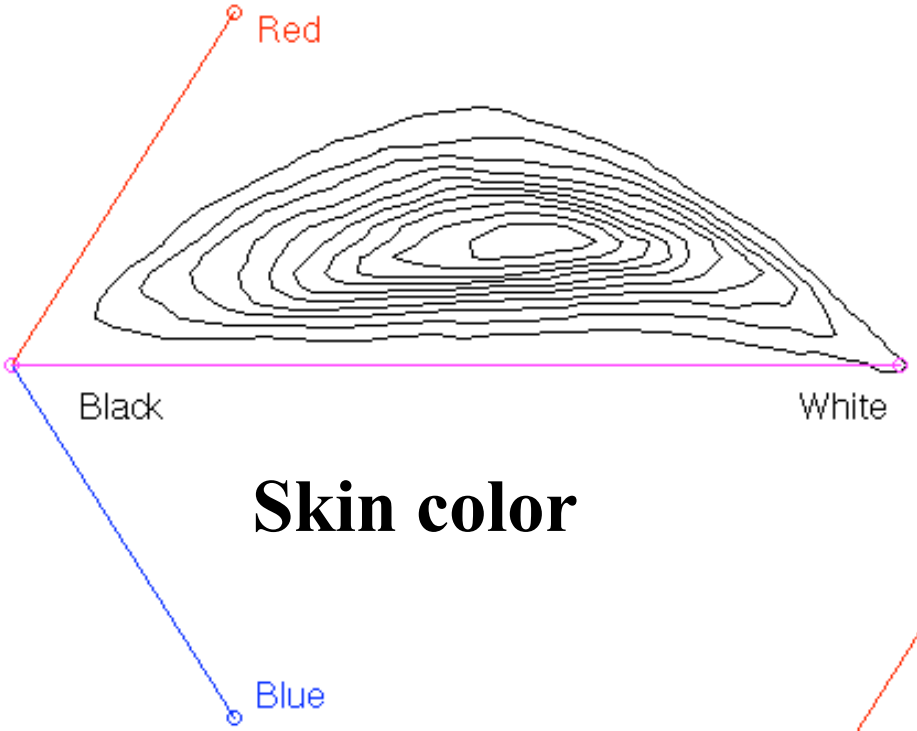


Non-Skin histogram



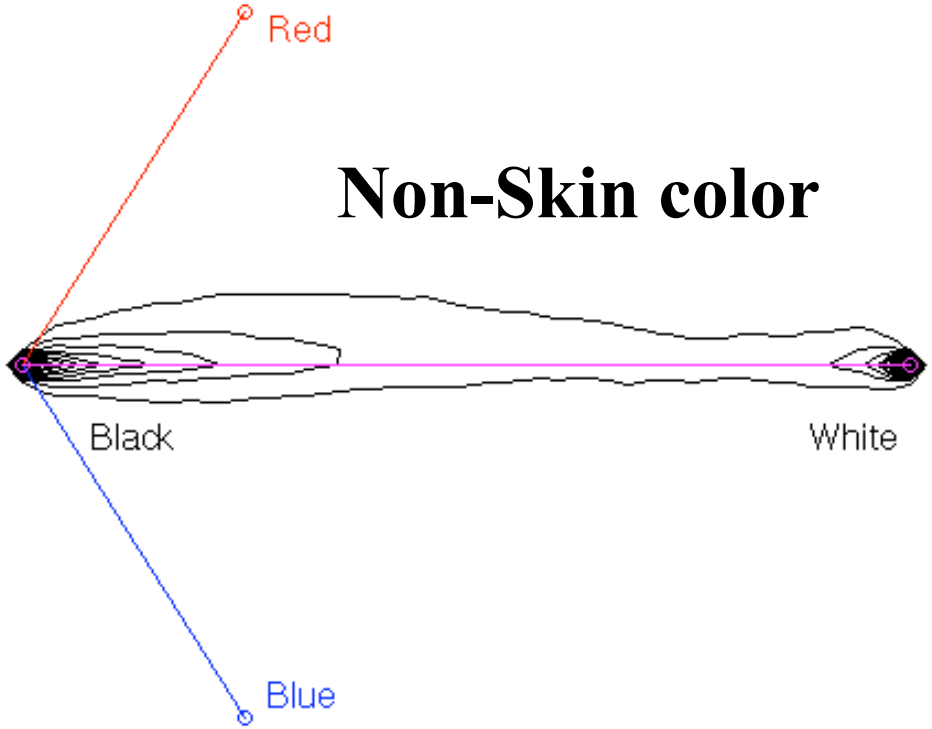
Robert Collins
CSE598G

Learned Distributions



Skin color

Non-Skin color



Robert Collins
CSE598G

Likelihood Ratio

$$\text{Label a pixel skin if } \frac{P(rgb \mid skin)}{P(rgb \mid not\ skin)} > \Theta$$

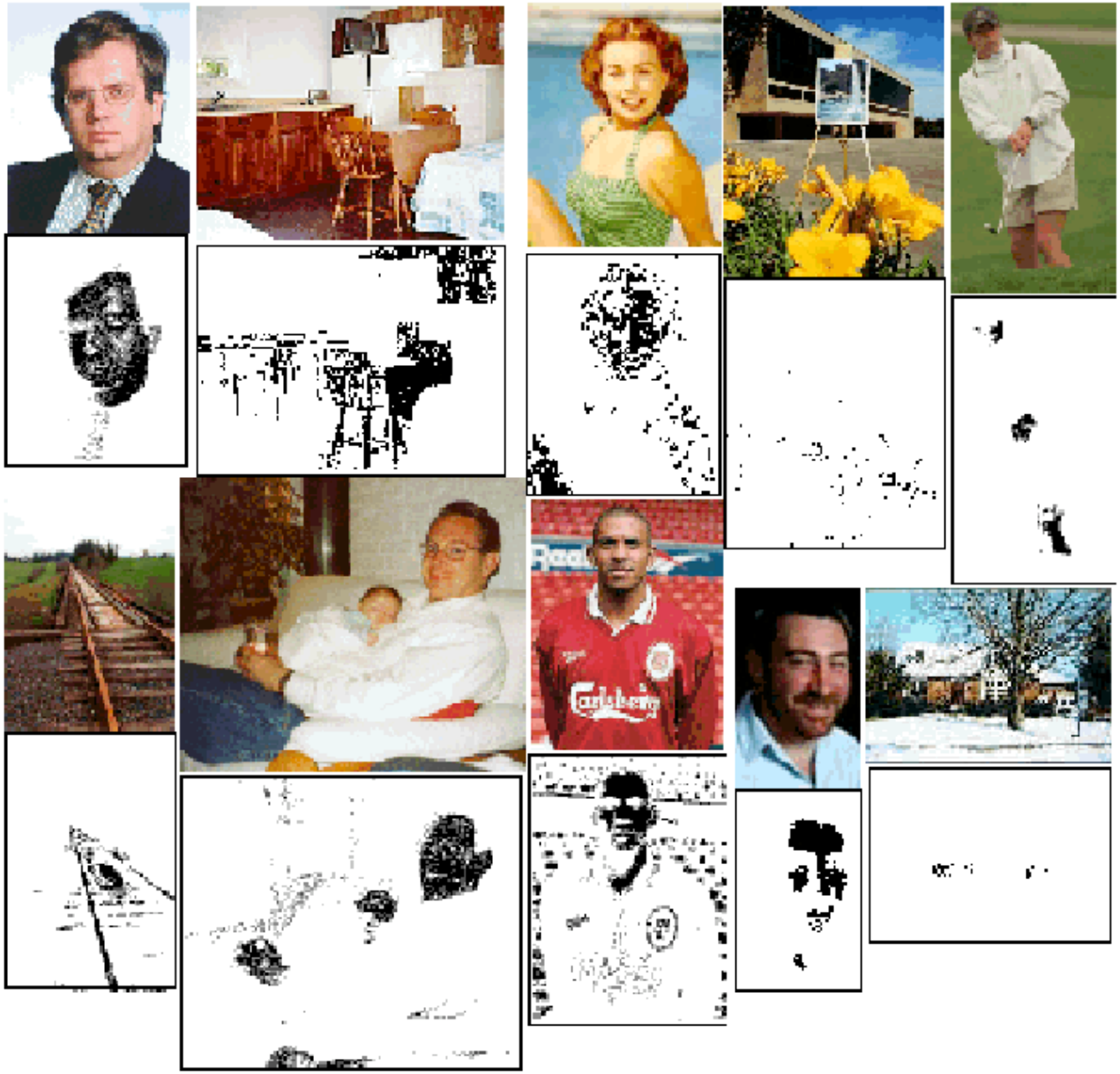
$$\Theta = \frac{(\text{cost of false positive})\ P(\text{ seeing not skin})}{(\text{cost of false negative})\ P(\text{ seeing skin})}$$

$$0 <= \Theta <= 1$$

Robert Collins
CSE598G

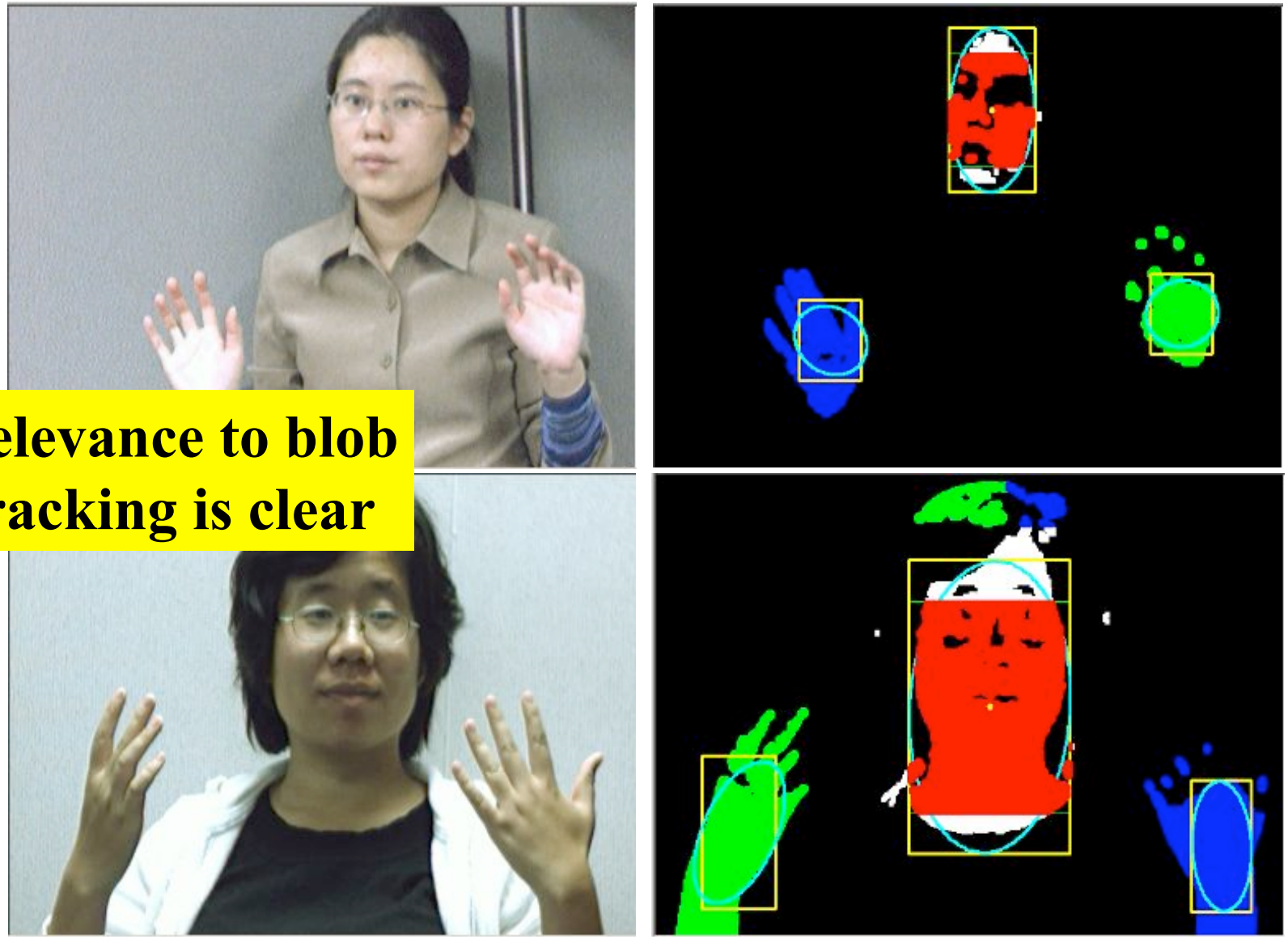
Sample Pixel Classifications

$\Theta = .4$



Sample Application: HCI

Haiying Guan, Matthew Turk, UCSB



relevance to blob tracking is clear

Implicit Weight Images

Sometimes the weight image is not explicitly created. An example is Comaniciu, Ramesh and Meer. Here, the "weight image" is embedded in the procedure (taking derivatives of bhattacharyya coeff with respect to image location of the window of samples).

Interesting point: their weight image changes between iterations of mean-shift, as compared to iterating to convergence on an explicit weight image!

Comaniciu et.al.

Color Histogram Representation:

target model: $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1...m}$ $\quad \sum_{u=1}^{m} \hat{q}_u = 1$

target candidate: $\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1...m}$ $\quad \sum_{u=1}^{m} \hat{p}_u = 1\,.$

Distance between histograms measured by:

$$d(\mathbf{y}) = \sqrt{1 - \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]}\,,$$

note: this is a function of window of location y

where

$$\hat{\rho}(\mathbf{y}) \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u}\,,$$

Bhattacharyya coefficient

Comaniciu et.al.

the histograms are computed via Parzen estimation:

$$\hat{q}_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i^\star\|^2)\delta\left[b(\mathbf{x}_i^\star) - u\right] ,$$

$$\hat{p}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right)\delta\left[b(\mathbf{x}_i) - u\right] ,$$

where k is some radially symmetric smoothing kernel (profile)

This is Important!

- **interpolates histograms in off-lattice locations**
- **makes histogram p_u differentiable wrt to y**

Comaniciu et.al.

via Taylor series expansion about current window location y_0:

this does not depend on y

so just need to maximize this.
Note: it is a KDE!!!!

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u} + \frac{C_h}{2}\sum_{i=1}^{n_h} w_i k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)$$

where $$w_i = \sum_{u=1}^{m}\sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}}\delta\left[b(\mathbf{x}_i) - u\right] .$$

find mode of second term by mean-shift iterations:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h}\mathbf{x}_i w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{x}_i}{h}\right\|^2\right)} \quad \text{where } g(x) = -k'(x).$$

Comaniciu et.al.

At each iteration:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}$$

which is just standard mean-shift on (implicit) weight image w_i

Let's look at the weight terms more closely. For each pixel x_i:

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}}\, \delta\left[b(\mathbf{x}_i) - u\right] .$$

This term is only 1 once in the summation

If pixel x_i's value maps to histogram bucket B,
then $w_i = \text{sqrt}(q_B / p_B(y_0))$

Comaniciu et.al.

So if model histogram is $q_1, q_2, ..., q_m$

and current data histogram is $p_1, p_2, ..., p_m$

form weights $q_1/p_1, q_2/p_2, \ldots, q_m/p_m$

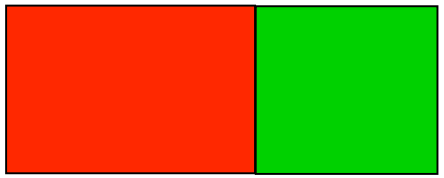
and then do "histogram backprojection" of these values into the image to get the weight image w_i

note to self:
Swain and Ballard

also note, $p_1, p_2, ..., p_m$ changes at each iteration, and therefore so does the weight image w_i

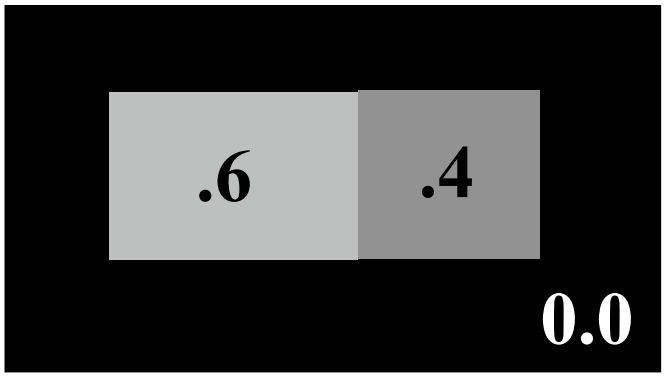
Qualitative Intuition

Assume some object that
is 60% red and 40%green

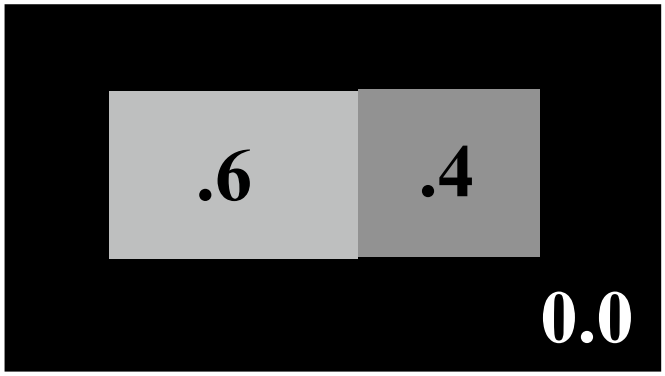


q1 = .6, q2 = .4, qi = 0 for all other i

If we just did histogram backprojection of these likelihood values (ala Bradski), we would get a weight image



Qualitative Intuition



Mean shift does a weighted center of mass computation at each interation

Mean shift window will be biased towards the region of red pixels, since they have higher weight

Robert Collins
CSE598G

Qualitative Intuition

Now use Comaniciu et.al.'s weights
Let's say the data histogram is perfectly located

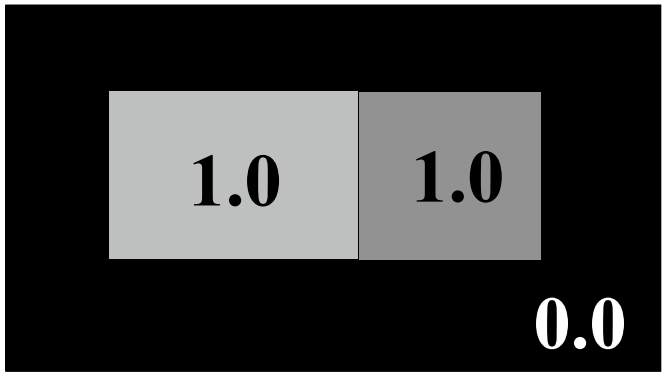
$$q1 = .6,\ \ q2 = .4,\ \ qi = 0 \text{ for all other } i$$

say something about 0 values

$$p1 = .6,\ \ p2 = .4,\ \ pi = 0 \text{ for all other } i$$

$$w1 = \text{sqrt}(.6/.6),\ \ w2 = \text{sqrt}(.4/.4),\ \ wi = 0 \text{ for all other } i$$

Resulting weight image:



This is the indicator function image we always hope for!

Qualitative Intuition

Say we have too little percentage of red in data hist

$$q1 = .6,\ \ q2 = .4,\ \ qi = 0 \text{ for all other } i$$

$$p1 = .5,\ \ p2 = .5,\ \ pi = 0 \text{ for all other } i$$

$$w1 = \text{sqrt}(.6/.5),\ \ w2 = \text{sqrt}(.4/.5),\ \ wi = 0 \text{ for all other } i$$

$$>1 \qquad\qquad <1$$

So red pixels will be favored in center of mass computation, hopefully causing more of them to be included at the next iteration.

Qualitative Intuition

Say we have very little percentage of red in data hist

$$q1 = .6,\ \ q2 = .4,\ \ qi = 0 \text{ for all other } i$$

$$p1 = .2,\ \ p2 = .8,\ \ pi = 0 \text{ for all other } i$$

$$w1 = \text{sqrt}(.6/.2),\ \ w2 = \text{sqrt}(.4/.8),\ \ wi = 0 \text{ for all other } i$$

$$>>1 \qquad\qquad <<1$$

So red pixels will be even more highly weighted relative to green pixels.

Qualitative Intuition

Say we have too much percentage of red in data hist

$$q1 = .6,\ \ q2 = .4,\ \ qi = 0 \text{ for all other } i$$

$$p1 = .7,\ \ p2 = .3,\ \ pi = 0 \text{ for all other } i$$

$$w1 = \text{sqrt}(.6/.7),\ \ w2 = \text{sqrt}(.4/.3),\ \ wi = 0 \text{ for all other } i$$

$$<1 \qquad\qquad >1$$

So green pixels will now be favored.

Other Features

We've only talked about color, but of course we could use histograms of other features like edge orientations (or any filter-bank response)

However, one point I've been trying to make today is that we don't need histograms at all. We just need a way to label pixels with the likelihood that they belong to the object (rather than the background).

That is, we just need to be able to specify a weight image, either explicitly or implicitly.

Discuss using mean-shift to find modes in correlation surfaces

Explicit is easy. Implicit??

Brief overview of our two "bleeding edge" topics

Choose discussion leaders.