Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 - MMIS — Fall Semester 2016
Lesson 2 — 12 October 2016

# Face Detection using Color Histograms

**Outline**

## Notation

| | |
|---|---|
| $x_d$ | A feature. An observed or measured value. |
| $\vec{X}$ | A vector of D features. |
| D | The number of dimensions for the vector $\vec{X}$ |
| $\vec{y}$ | The true class for an observation $\vec{X}$ |
| $\{\vec{X}_m\}$ $\{y_m\}$ | Training samples for learning. |
| $y_m$ | An annotation (or ground truth) function |
| $R(\vec{X}_m)$ | The Detection function or Recognition function $R(\vec{X}_m) \in \{P, N\}$ |
| $M$ | The number of training samples. |
| $M_T$ | The number of training samples in the target class |
| $h(\vec{X})$ | A multidimensional histogram of $\vec{X}$ |
| Q | The number of cells in a histogram |

## 1. Two-Class Pattern Detectors

A pattern detector is a classifier with K=2.

Class k=1: The target pattern.

Class k=2: Everything else.

Pattern detectors are used in computer vision, for example to detect faces, road signs, publicity logos, or other patterns of interest. They are also used in signal communications, data mining and many other domains.

The pattern detector is learned as a detection function $g(\vec{X})$ followed by a decision rule, *d()*. The detection function is learned from a set of training data composed of M sample observations $\{\vec{X}_m\}$ where each sample observation is labeled with an indicator variable $\{y_m\}$

$y_m =$ P or Positive for examples of the target pattern (class 1)

$y_m =$ N or Negative for all other examples (class 2)

A ratio of histograms can provides a discriminant function as :

$$g(\vec{X}) = p(C_1 \mid \vec{X}) = \frac{h_1(\vec{X})}{h_1(\vec{X}) + h_2(\vec{X})} = \frac{h_1(\vec{X})}{h(\vec{X})}$$

where $h_1(\vec{X})$ is the histogram of training samples for the target class
and $h(\vec{X})$ is the histogram of ALL training samples.

To simplify the algebra, we can create a lookup table

$$L(\vec{X}) = \frac{h_1(\vec{X})}{h(\vec{X})}$$

This gives a discriminant function of

$$g(\vec{X}) = L(\vec{X})$$

Observations for which $g(\vec{X}) > 0.5$ are estimated to be members of the target class. This will be called POSITIVE or P.

Observations for which $g(\vec{X}) \le 0.5$ are estimated to be members of the background. This will be called NEGATIVE or N.

We can encode this as a decision function to define our detection function $R(\vec{X}_m)$

$$R(\vec{X}) = d(g(\vec{X})) = \begin{cases} P & \text{if } g(\vec{X}) > 0.5 \\ N & \text{if } g(\vec{X}) \le 0.5 \end{cases}$$

For training we need ground truth (annotation). For each training sample the annotation or ground truth tells us the real class $y_m$

$$y_m = \begin{cases} P & \vec{X}_m \in \text{Target - Class} \\ \text{N} & \text{otherwise} \end{cases}$$

The Classification can be TRUE or FALSE.

if $R(\vec{X}_m) = y_m$ then T else F

This gives

$R(\vec{X}_m) = y_m$ AND $R(\vec{X}_m) = P$ is a TRUE POSITIVE or TP
$R(\vec{X}_m) \ne y_m$ AND $R(\vec{X}_m) = P$ is a FALSE POSITIVE or FP
$R(\vec{X}_m) \ne y_m$ AND $R(\vec{X}_m) = N$ is a FALSE NEGATIVE or FN
$R(\vec{X}_m) = y_m$ AND $R(\vec{X}_m) = N$ is a TRUE NEGATIVE or TN

To better understand the detector we need a tool to explore the trade-off between making false detections (false positives) and missed detections (false negatives). The Receiver Operating Characteristic (ROC) provides such a tool

# 2. Performance Evaluation Metrics

## ROC Curves

Two-class classifiers have long been used for signal detection problems in communications and have been used to demonstrate optimality for signal detection methods. The quality metric that is used is the Receiver Operating Characteristic (ROC) curve. This curve can be used to describe or compare any method for signal or pattern detection.

The ROC curve is generated by adding a variable Bias term to a discriminant function.

$$R(\vec{X}) = d(g(\vec{X}) + B)$$

and plotting the rate of true positive detection vs false positive detection where $R(\vec{X}_m)$ is the classifier as in lesson 1. As the bias term, B, is swept through a range of values, it changes the ratio of true positive detection to false positives.

For a ratio of histograms, $g(\vec{X}_m)$ is a probability ranging from 0 to 1.
B can range from less than $-0.5$ to more than $+0.5$.
When $B \leq -0.5$ all detections will be Negative.
When $B > +0.5$ all detections will be Positive.
Between $-0.5$ and $+0.5$ $R(\vec{X})$ will give a mix of TP, TN, FP and FN.

The bias term, B, can act as an adjustable gain that sets the sensitivity of the detector. The bias term allows us to trade False Positives for False Negatives.

The resulting curve is called a Receiver Operating Characteristics (ROC) curve.
The ROC plots True Positive Rate (TPR) against False Positive Rate (FNR) as a function of B for the training data $\{\vec{X}_m\}$, $\{y_m\}$.

### True Positives and False Positives

For each training sample, the detection as either Positive (P) or Negative (N)

IF $g(\vec{X}_m)+B > 0.5$ THEN P else N

The detection can be TRUE (T) or FALSE (F) depending on the indicator variable $y_m$

IF $y_m = R(\vec{X}_m)$ THEN T else F

Combining these two values, any detection can be a True Positive (TP), False Positive (FP), True Negative (TN) or False Negative (FN).

For the M samples of the training data $\{\vec{X}_m\}$, $\{y_m\}$ we can define:

#P as the number of Positives,
#N as the number of Negatives,
#T as the number of True and
#F as the number of False,

From this we can define:

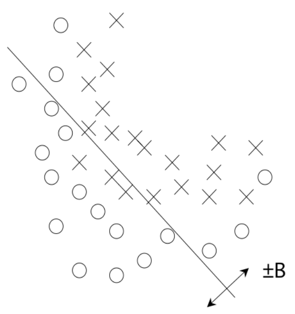
#TP as the number of True Positives,
#FP as the number of False Positives,
#TN as the number of True Negative,
#FN as the number of False Negatives.

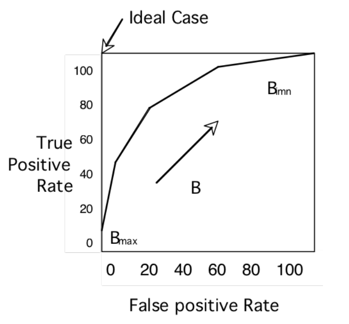
Note that #P = #TP + #FN
And #N = #FP+ #TN

The True Positive Rate (TPR) is $TPR = \frac{\#TP}{\#P} = \frac{\#TP}{\#TP + \#FN}$

The False Positive Rate (FPR) is $FPR = \frac{\#FP}{\#N} = \frac{\#FP}{\#FP + \#TN}$

The ROC plots the TPR against the FPR as a bias B is swept through a range of values.





When B is less than –0.5, all the samples are detected as N, and both the TPR and FPR are 0. As B increases both the TPR and FPR increase. Normally TPR should rise monotonically with FPR. If TPR and FPR are equal, then the detector is no better than chance.

The closer the curve approaches the upper left corner, the better the detector.

| | | $y_m = R(\vec{X}_m)$ | |
|---|---|---|---|
| $d(g(\vec{X}_m) + B > 0.5)$ | | T | F |
| | P | True Positive (TP) | False Positive (FP) |
| | N | True Negative (TN) | False Negative (FN) |

## Precision and Recall

**Precision**, also called Positive Predictive Value(PPV) is the fraction of retrieved instances that are relevant to the problem.

$$PP = \frac{TP}{TP + FP}$$

A perfect precision score (PPV=1.0) means that every result retrieved by a search was relevant, but says nothing about whether all relevant documents were retrieved.

**Recall**, also known as sensitivity (S), hit rate, and True Positive Rate (TPR) is the fraction of relevant instances that are retrieved.

$$S = TPR = \frac{TP}{T} = \frac{TP}{TP + FN}$$

A perfect recall score (TPR=1.0) means that all relevant documents were retrieved by the search, but says nothing about how many irrelevant documents were also retrieved.

Both precision and recall are therefore based on an understanding and measure of relevance. In our case, "relevance" corresponds to "True".
Precision answers the question "How many of the Positive Elements are True ?"
Recall answers the question "How many of the True elements are Positive"?

In many domains, there is an inverse relationship between precision and recall. It is possible to increase one at the cost of reducing the other.

## F-Measure

The F-measures combine precision and recall into a single value. The F measures measure the effectiveness of retrieval with respect to a user who attaches 2 times as much importance to recall as precision.

The $F_1$ score weights recall higher than precision.

**$F_1$ Score**:

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

The F1 score is the harmonic mean of precision and sensitivity.
This is the geometric mean divided by the arithmetic mean.

**Accuracy:**

**Accuracy** is the fraction of test cases that are correctly classified (T).

$$ACC = \frac{T}{M} = \frac{TP + TN}{M}$$

where M is the quantity of test data.

Note that the terms Accuracy and Precision have a very different meaning in Measurement theory. In measurement theory, accuracy is the average distance from a true value, while precision is a measure of the reproducibility for the measurement.

**Matthews Correlation Coefficient**

The Matthews correlation coefficient is a measure of the quality of binary (two-class) classifications. This measure was proposed by the biochemist Brian W. Matthews in 1975.

MCC takes into account true and false positives and negatives and is generally regarded as a balanced measure that can be used even if the classes are of very different sizes.

The MCC is in essence a correlation coefficient between the observed and predicted binary classifications

MCC results a value between +1 and -1, where +1 represents a perfect prediction, 0 no better than random prediction and −1 indicates total disagreement between prediction and observation.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The original formula given by matthews was:

M = Total quanitity of test data:

$$M = TN + TP + FN + FP$$

$$S = \frac{TP + FN}{M}$$

$$P = \frac{TP + FP}{M}$$

$$MCC = \frac{TP/M - S \cdot P}{\sqrt{PS(1-S)(1-P)}}$$

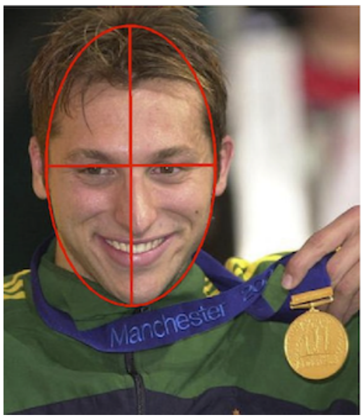
## 3. Detecting Skin Pixels in Color histograms

A ratio of color histograms can be used to construct a simple detector for skin pixels in images. Color skin pixels can be used to detect faces, hands and other skin colored regions in images.

Consider the images in the FDDB (Face Detection Data Set and Benchmark) data base maintained at UMASS: http://vis-www.cs.umass.edu/fddb/
This data-base was constructed for face detection and not for skin detection.
Face regions have been hand-labeled as both boxes and ellipses.

All images are RGB with each pixel containing 3 colors: Red, Green and Blue.

We will use the ellipses as ground truth for skin regions. This will create errors in the training data. A typical image with and annotated face regions as an ellipse looks like:



Note that there are skin pixels that are NOT in the ellipse (hand, ears, neck etc), and there are non-skin pixels that ARE in the face (hair, teeth, etc). This will lead to minor errors in the detection. Your job will be to measure the impact of these errors in building a face detector using detection of skin pixels.

Annotations of face regions are represented as an elliptical region, denoted by a 6-tuple ($r_a$, $r_b$, $\theta$, $c_x$, $c_y$, 1) where $r_a$ and $r_b$ refer to the half-length of the major and minor axes, $\theta$ is the angle of the major axis with the horizontal axis, and $c_x$ and $c_y$ are the column and row image coordinates of the center of this ellipse.

```
Ellipse Data:
2002/07/24/big/img_82
1
59.268600 35.142400 1.502079 149.366900 59.365500  1
```

the standard form of an ellipse with a major axis along the horizontal (x) axis is:

$$\frac{\left(x-c_x\right)^2}{r_a^2}+\frac{\left(y-c_y\right)^2}{r_b^2}=1$$

when rotated by θ:

$$\frac{\left((x-c_x)\cos(\theta)+(y-c_y)\sin(\theta)\right)^2}{r_a^2}+\frac{\left((x-c_x)\sin(\theta)+(y-c_y)\cos(\theta)\right)^2}{r_b^2}=1$$

## Histograms as an Estimation of Probability of Color

Assume that we have a color image, where each pixel *(i,j)* is a color vector, $\vec{p}(i,j)$, composed of 3 integers between 0 and 255 representing Red, Green and Blue.

$$\vec{p}(i,j) = \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

We can build a color histogram of the image by counting the number of times that each unique value of (R, G, B) occurs in the image. To do this we allocate a table h(R, G, B) of 256 x 256 x 256 cells, with each cell initially set to zero.
If each color is represented by 8 bits, then the table h(R, G, B) has $Q=(2^8)^3$ cells. $Q=(2^8)^3=256^3 = 2^{24} = 2^4 \cdot 2^{20}$ = 16 Meg Cells.

Using our rule of 8 samples per cell, we need $2^3 x 2^4 = 2^7$ = 128 Meg Pixels.
If the training data is composed of 128x128 images, then how many images do we need for h(X)?

128 x 128 pixels = $2^7 \text{ x } 2^7 = 2^{14}$ = 16 K pixels. Thus we need 8000 images !
What can we do?

Often we can reduce both the number, D, of features and the number of values, N, for each feature.

For example, for many color images, N=32 color values are sufficient to detect objects. We simply divide each color R, G, B by 8.

R' = Trunc(R/8), G'=Trunc(G/8), B'=Trunc(B/8).

This reduces our histogram to $8x8x8=(2^3)^3 = 2^9=512$ cells

We can also use our knowledge of physics to look for features that are "invariant".

## Color Skin Detection using Luminance and Chrominance.

Luminance captures surface orientation (3D shape) while Chrominance is a signature for object pigment (identity). Thus it is convenient to transform the (RGB) color pixels into a color space that separates Luminance from Chrominance.

$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} \Leftarrow \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Normalizing out luminance provides a popular space for skin detection: the (r,g) space.

Luminance: L= R+G+B

Chrominance : $r = c_1 = \frac{R}{R+G+B}$ $g = c_2 = \frac{G}{R+G+B}$

These are often called "r" and "g" in the literature. The (r, g) space is often used to detect skin colored pixels. It is common to normalize r and g to natural numbers coded with N values between 0 and N – 1 by :

$$r = trunc\left(N \cdot \frac{R}{R+G+B}\right) \qquad g = trunc\left(N \cdot \frac{G}{R+G+B}\right)$$

Skin pigment is generally always the same chrominance value. Luminance can change with pigment density, and skin surface orientation, but chrominance will remain invariant. Thus we can use (r,g) as an invariant color signature for detecting skin in at each pixel. From experience, N = 32 color values seems to work well for skin, but larger values of N may work better with some data sets.

Suppose we have a set of K training images $\{\vec{p}(i,j)\}$ of size RxC where each pixel is an RGB color vector. This gives a total of M=NxRxC color pixels. Suppose that $M_T$ of these are labeled as skin pixels.

We allocate two table : *h(r,g)* and $h_{skin}(r,g)$ of size N x N.

```
For all i,j,k in the training set  $\{\vec{p}_k(i,j)\}$
BEGIN
    $r = trunc\left(N \cdot \frac{R}{R+G+B}\right)$    $g = trunc\left(N \cdot \frac{G}{R+G+B}\right)$
    $h(r,g) = h(r,g) + 1$
    IF the pixel $\vec{p}_k(i,j)$ is $\in$ skin region THEN
        $h_{skin}(r,g) = h_{skin}(r,g) + 1$ and $M_{skin} = M_{skin} + 1$
END
```

As before, we can obtain a lookup table $L_{skin}(r,g)$ that gives the probability that a pixel is skin.

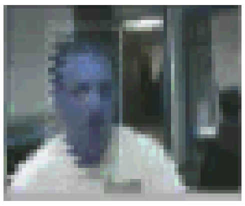
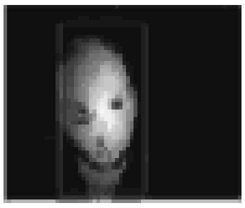
$$L_{skin}(r,g) = \frac{h_{skin}(r,g)}{h(r,g)}$$

Given a new RGB image $\vec{p}(i,j)$ for each pixel *(i,j)*

$$r = trunc\left(N \cdot \frac{R}{R+G+B}\right) \qquad g = trunc\left(N \cdot \frac{G}{R+G+B}\right)$$

$$g(\vec{p}(i,j)) = L_{skin}(r(i,j), g(i,j))$$

We can display the resulting probabilities as a gray scale image, by multiplying by each value by 256. White is high probability. Black is low probability.

(images from a Bayesian skin tracking in real time - 1993)

We can improve the detection by grouping adjacent skin pixels into skin "blobs" and tracking skin blobs over time.

## 4. Bayesian Tracking of Gaussian Blobs

Rather than represent a skin region as a collection of pixels, we can calculate a Gaussian Blob. A "Blob" represents a region of an image. Gaussian blobs express a region in terms of moments.

Assume of image of probabilities of the detection of a target: *T(i,j),* where for each pixel:

$$T(i,j) = L_{skin}(r(i,j), g(i,j))$$

The zeroth moment of the probabilities is the mass (sum of probabilities). Average mass represents confidence.

The first moment gives is the center of gravity. This is the "position" of the blob.

The second moment is the covariance. This gives size and orientation.

We typically enclose the blob in some rectangular Region of Interest (ROI) in order to avoid "distraction" by neighboring blobs. The ROI is obtained by some form of estimation or <u>a priori knowledge</u>. In continuous operation the ROI be provided by tracking.

Let us represent the ROI as a rectangle : (t,l,b,r)

t - "top" - first row of the ROI.
l - "left" - first column of the ROI.
b - "bottom" - last row of the ROI
r - "right" -last column of the ROI.

(t,l,b,r) can be seen as a bounding box, expressed by opposite corners (l,t), (r,b)
We will compute the moments within this ROI (bounding box).

### Moment Calculations for Blobs

Given a target probability image *T(i,j)* and a ROI (t,l,b,r):

Sum: $$S = \sum_{i=l}^{r} \sum_{j=t}^{b} T(i,j)$$

We can estimate the "confidence" as the average detection probability:

Confidence: $$CF = \frac{S}{(b-t)(r-l)}$$

**First moments:**

$$x = \mu_i = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} T(i,j)\cdot i$$
$$y = \mu_j = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} T(i,j)\cdot j$$

Position is the center of gravity: $(\mu_i, \mu_j)$

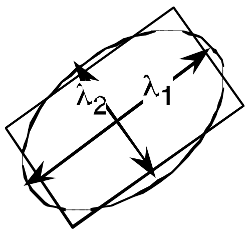
We will use this as the position of the blob.

**Second Moments**:

$$\sigma_i^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} T(i,j)\cdot (i-\mu_i)^2$$
$$\sigma_j^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} T(i,j)\cdot (j-\mu_j)^2$$
$$\sigma_{ij}^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} T(i,j)\cdot (i-\mu_i)\cdot(j-\mu_j)$$

These compose a covariance matrix: $C = \begin{pmatrix}\sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2\end{pmatrix}$

The principle components ($\lambda_1$, $\lambda_2$) determine the length and width.
The principle direction determines the orientation of the length.
We can discover these by principle components analysis.



$$RCR^T = \Lambda = \begin{pmatrix}\lambda_1^2 & 0 \\ 0 & \lambda_2^2\end{pmatrix}$$

where

$$R = \begin{pmatrix}\cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta)\end{pmatrix}$$

The length to width ratio, $\lambda_1/\lambda_2$, is an invariant for shape.
The angle θ is a "Covariant" for orientation.

We can use the "eigenvalues", or characteristic values, $\lambda_1$, $\lambda_2$, to define the "width and height" of the blob:

For example:

$w=\lambda_1$, $h=\lambda_2$

This suggests a "feature vector" for the blob: $\vec{X} = \begin{pmatrix} x \\ y \\ w \\ h \\ \theta \end{pmatrix}$

where $x=\mu_i$, $y=\mu_j$, $w=\lambda_1$, $h=\lambda_2$ and $CF = \frac{S}{(b-t)(r-l)}$

The confidence (CF) can be seen as the "Likelihood" that the model for the blob is correct.

Tracking allows us to continually update an estimate for the features of the Blob, even if the blob is temporarily lost to occlusion or noise.

The tracked object is often referred to as a "target". The vector $\vec{X}$ provides the "model" for the target blob:

Blob model: $\vec{X} = \begin{pmatrix} x \\ y \\ w \\ h \\ \theta \end{pmatrix}$ along with CF. (Confidence)

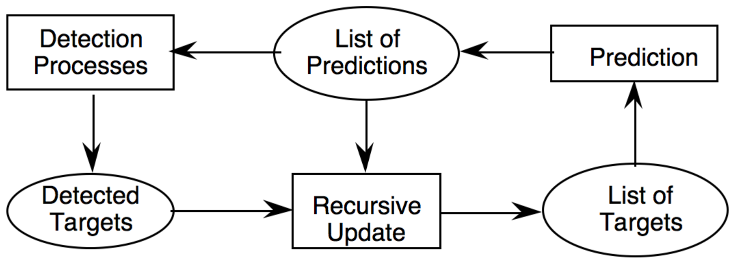
The precision of the blob can be represented by a covariance matrix

$$P = \begin{pmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 & \sigma_{xw}^2 & \sigma_{xh}^2 & \sigma_{x\theta}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 & \sigma_{yw}^2 & \sigma_{yh}^2 & \sigma_{y\theta}^2 \\ \sigma_{wx}^2 & \sigma_{wy}^2 & \sigma_{ww}^2 & \sigma_{wh}^2 & \sigma_{w\theta}^2 \\ \sigma_{hx}^2 & \sigma_{hy}^2 & \sigma_{hw}^2 & \sigma_{hh}^2 & \sigma_{h\theta}^2 \\ \sigma_{\theta x}^2 & \sigma_{\theta y}^2 & \sigma_{\theta w}^2 & \sigma_{\theta h}^2 & \sigma_{\theta\theta}^2 \end{pmatrix}$$

**Bayesian Tracking (extra material)**

(extra material - not essential for the course)

A Bayesian tracker is a recursive estimator, composed of three phases: Predict, Detect, Update.



Detection can be provided by detecting the blob using color statistics within a target "Region of Interest" given by a bounding box centered on a previous position. The size of this box is determined by the estimated size of the blob enlarged by the uncertainty $P$.

The Gaussian window is the previous covariance for the blob, enlarged by some "uncertainty" covariance. The uncertainty captures the possible loss of information during the time from the most recent observation.

Our Gaussian blob is

Position: $\vec{\mu}_t = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$ Size : $C_t = \begin{pmatrix} \sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2 \end{pmatrix}$ along with $CF_t$.

where the second moment of the detected pixels, C, was used to compute to determine the width, height and orientation:

$$RCR^T = \Lambda = \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix} = \begin{pmatrix} h^2 & 0 \\ 0 & w^2 \end{pmatrix}$$

so

$$C = R^T \Lambda R = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} h^2 & 0 \\ 0 & w^2 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Let us define the estimated blob at time t as: $\hat{\mu}_t$, $\hat{C}_t$

Let us define the predicted feature vector at time t as: $\vec{\mu}_t^*$, $C_t^*$

We will compute the estimated blob from by multiplying the detected pixels by a Gaussian mask determined from the predicted blob. The Covariance is multiplied by 2 to offset the fact that we will use mask to estimate a new covariance.

Gaussian Mask: $G(\vec{\mu}_t^*, 2C_t^*)$

Detected target pixels:

$$t(i,j) \leftarrow L(\vec{c}(i,j)) \cdot e^{-\frac{1}{2}\left(\begin{pmatrix} i \\ j \end{pmatrix} - \begin{pmatrix} \mu_i \\ \mu_i \end{pmatrix}\right)^T 2C_t^{*-1}\left(\begin{pmatrix} i \\ j \end{pmatrix} - \begin{pmatrix} \mu_i \\ \mu_i \end{pmatrix}\right)}$$

where L() is our lookup table for the ratio of histograms.
This is a form of “robust” estimation that uses the Gaussian mask to reject “outlying” detections.

We then estimate the new position and covariance as before:

**First moments:** $\mu_i = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j) \cdot i$ $\mu_j = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j) \cdot j$

**Second Moments:**

$$\sigma_i^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j) \cdot (i-\mu_i)^2$$

$$\sigma_j^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j) \cdot (j-\mu_j)^2$$

$$\sigma_{ij}^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j) \cdot (i-\mu_i) \cdot (j-\mu_j)$$

Position: $\hat{\vec{\mu}}_t = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$ Size : $\hat{C}_t = \begin{pmatrix} \sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2 \end{pmatrix}$

For this we must predict the new position, detect (observe) the blob, and the update the estimate.

The following is a “zero-th” order Kalman filter. This is the simplest (almost trivial) case of a Bayesian tracker. A first order Kalman filters estimates parameters and their derivatives. The math is more complex but the principles are the same.