Computer Vision

James L. Crowley

M2R MoSIG — Fall Semester

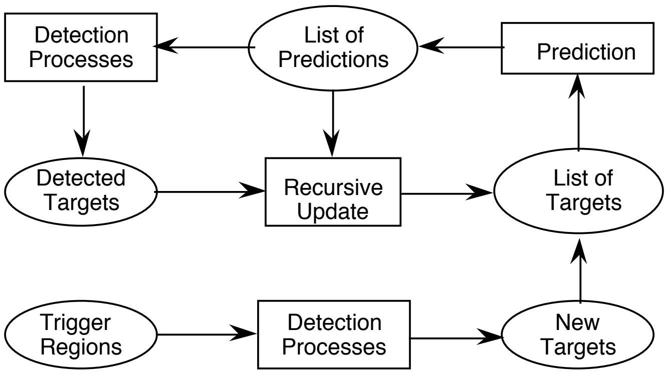
Lesson 6 - Practical — 19 Nov 2020

Bayesian Detection and Tracking

Outline

Bayesian Detection and Tracking

A Bayesian Tracker is a cyclic process composed of the cycles predict, detect and update.



Tracking
1) Focusing computing resources
2) Improves reliability by compensating for lost detections
3) Improves precision by avoiding distractors
4) Makes it possible to estimate motion derivatives.

Pixel Level Target Detection

A pixel level detector estimates the likelihood that a pixel belongs to a target. Many possible estimation processes can be used. Popular pixel level detectors include:

1) Background Difference Subtraction
2) Ratio of Color Histograms
3) Ratio of Receptive field histograms
4) Motion (temporal image differences).

It is sometimes possible to combine 2 or more of these techniques!

One of my doctoral students demonstrated reliable face detection with histograms of color-opponent receptive fields. Another student combined a Bayesian tracker with face detection using a Cascade of Linear Detectors learned with AdaBoost.

Face Detection with a Ratio of Histograms of color pixel

Last time we saw that we could calculate the probability that a pixel belonged to a target class using a lookup table constructed from a ratio of color histograms.

We allocate two tables $h(\vec{X})$ and $h_T(\vec{X})$ where $h_T(\vec{X})$ is the histogram of colors for target pixels and $h(\vec{X})$ is the histogram of colors for all pixels.

Training data is M color pixels $\{\vec{X}_m\}$ for which we have an indicator function $y(\vec{X}_m)$.
The data contains M_T pixels that are target (skin color) pixels:

The histogram of all colors is constructed with $\forall_{m=1}^{M}: h(\vec{X}_m) \leftarrow h(\vec{X}_m)+1$
And the histogram of skin color pixels is constructed using a ground truth function $y(\vec{X}_m)$ that returns P is $\vec{X}_m$ is skin else N.

$$\forall_{m=1}^{M}: \text{ IF } y(\vec{X}_m) = P \text{ THEN } h_T(\vec{X}_m) \leftarrow h_T(\vec{X}_m)+1;\ M_T \leftarrow M_T+1;$$

The probabilities for obtaining a color vector are

$$P(\vec{X}) = \frac{1}{M}h(\vec{X}) \text{ and } P(\vec{X} \mid Skin) = \frac{1}{M_T}h_T(\vec{X})$$

The probability that a pixel belongs to the target class is: $P(Skin) = \frac{M_T}{M}$

From this we can show that the probability that a pixel is skin, is simply the ratio of the two histograms.

$$P(Skin \mid \vec{X}) = \frac{P(\vec{X} \mid Skin)P(Skin)}{P(\vec{X})} = \frac{\frac{1}{M_T}h_T(\vec{X}) \cdot \frac{M_T}{M}}{\frac{1}{M}h(\vec{X})} = \frac{h_T(\vec{X})}{h(\vec{X})}$$

We can use this to compute a lookup table $L_{Skin}(\vec{X}) = \frac{h_T(\vec{X})}{h(\vec{X})}$

if $h(\vec{X}) = 0$ then $h_{Skin}(\vec{X}) = 0$ because $h_{Skin}(\vec{X})$ is a subset of $h(\vec{X})$.
we will need to test this case to avoid an error from divide by 0.

If we ASSUME that a new image, *P(i,j)*, has similar illumination and color composition then we can use this technique to assign a probability to each pixel by table lookup.

The result is an image in which each pixel is a probability *S(i,j)* that the pixel *(i,j)* belongs to class skin.

$$S(i,j) = L_{Skin}(P(i,j))$$

<Karl Schwertd Skin Tracking Demo - 1995>

This method can be generalised to ANY vector of features.
For example, the appearance of a neighborhood given by the receptive field vector.

$$\vec{A}(i,j,\sigma_i) = P(i,j) * (G_x, G_y, G_{xx}, G_{xy}, G_{yy}) \text{ at some } \sigma_i$$

Given M samples of pixels $S=\{\vec{A}_m\}$ with a subset $T \subset S$ composed of M_T target pixels.

We construct a probability image for the target, t(i,j) as

$$T(i,j) = p(\text{target} \mid \vec{A}(i,j)) = \frac{h_T(\vec{A}(i,j))}{h(\vec{A}(i,j))}$$

We must, however assure that the number of sample $M_k >> Q$ the number of cells in the histogram. For D features and a quantification of N levels per feature $Q=N^D$

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.

A typical representation for a Gaussian blob is $\vec{B} = \begin{pmatrix} x \\ y \\ l \\ w \\ \theta \end{pmatrix}$ with confidence CF

Position *(x, y)* is estimated from the center of gravity. Spatial extent (length, *l,* width, *w*, and Orientation, *θ*) are estimated from the principal components of the second moment (covariance).

The confidence factor at time t, CF_t, is estimated recursively using a weighted sum of the previous confidence and the sum of the detection probability pixels in the ROI.

Bayesian Tracking

Target Blobs may be detected in an entry region, or by a scanning window detector or by random positioning of a detection window.

Given a ROI (t,l,b,r) of detected pixels *T(i,j)*

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

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

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

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

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

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

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

Principle components (λ_1, λ_2) determine the length, l, width , w, and orientation, θ.

$$R\Sigma R^T = \Lambda = \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix} \quad \text{where } R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

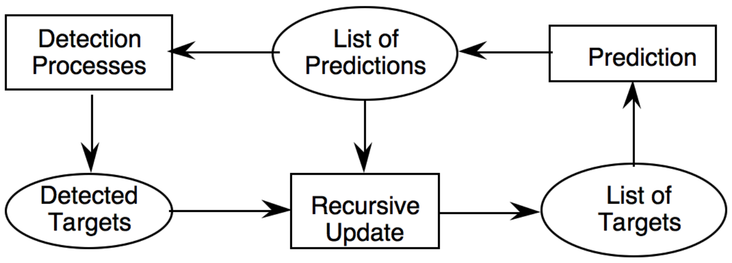
$l = \lambda_1$, $w = \lambda_2$ and Orientation, $\theta = \frac{R_{11}}{R_{21}} = \frac{\cos(\theta)}{\sin(\theta)}$

It is convenient to retain the first and second moment vectors, $\vec{\mu} = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$ and

$\Sigma = \begin{pmatrix} \sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2 \end{pmatrix}$, as we will need these to predict the ROI for the next image.

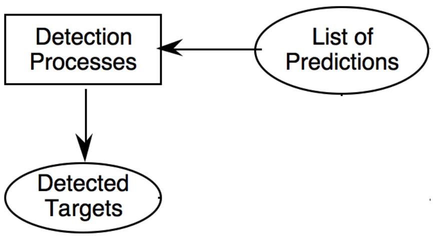
Bayesian Tracking

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



Temporal Prediction

We use the current position and spatial extent to predict a region of interest for the next image.



However, the scene evolves. Targets move with unknown velocity and acceleration. To compensate we enlarge the spatial extent of the target with an uncertainty, expressed as a covariance of possible positions. The uncertainty captures the possible loss of information during the time from the most recent observation.

Let us represent the position and spatial extent of the blob, $\vec{B}_t$ at time t as: with $\hat{\mu}_t$, $\hat{\Sigma}_t$. We will use these to predict a position and spatial extent for the ROI at time *t+1t* as: $\vec{\mu}^*_{t+1}$, Σ^*_{t+1}

The Zeroeth order (constant position) Tracker

In the absence of a model for target movement, we use the observed position at time t to predict the position at time t+1. This is referred to as a zeroth order model.

$$\vec{\mu}^*_{t+1} \leftarrow \hat{\mu}_t$$

The target may move with an unknown velocity and acceleration. We can model the loss in precision of the blob position as an error covariance, Q. This covariance can be learned as the movements (Δx, Δy) of targets from one frame to another of targets in the training set of image sequences.

$$Q = \begin{pmatrix} \sigma^2_{\Delta x} & \sigma^2_{\Delta x \Delta y} \\ \sigma^2_{\Delta y \Delta x} & \sigma^2_{\Delta y} \end{pmatrix}$$

For each cycle, the error covariance is added to the blob covariance to provide the spatial extent of the predicted ROI.

$$\Sigma^*_{t+1} = \hat{\Sigma}_t + Q$$

This sum of covariance results in a new larger covariance. We use the predicted Σ^*_{t+1} to determine the spatial extent for the ROI in the next image.

Computing the ROI from a predicted covariance

We can use principle components analysis to compute the bounding box from the new predicted position $\vec{\mu}^*_t$ and covariance, Σ^*_t. The principle components of Σ^*_t tell us the length and orientation of the major axis of Σ^*_t. The dimensions of the ROI are determined from the eigenvectors.

$$\Delta i = \max\{\text{abs}(\lambda_1 \, Cos(\theta), \text{abs}(\lambda_2 \, Sin(\theta)\}$$
$$\Delta j = \max\{\text{abs}(\lambda_1 \, Sin(\theta), \text{abs}(\lambda_2 \, Cos(\theta)\}$$

We regardless of the angle, θ, the bounding upper left corner, (*l, t*) of the bounding box is: $l = \mu_i - \Delta i \quad t = \mu_j - \Delta j$

The lower right corner, (*b, r*) is : $r = \mu_i + \Delta i \quad b = \mu_j + \Delta j$

Weighting the ROI by a Gaussian Mask.

The likelihood of detection can be improved by doubling the size of the ROI and to weighting the ROI with a Gaussian mask. In this case, the ROI becomes:

$$l = \mu_i - 2 \cdot abs(\Delta i) \qquad t = \mu_j - 2 \cdot abs(\Delta j)$$
$$r = \mu_i + 2 \cdot abs(\Delta i) \qquad b = \mu_j + 2 \cdot abs(\Delta j)$$

Detected target pixels within the enlarged ROI (*l,t,r,b*) in the ROI are then weighted with:

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

This has the effect of reducing the influence of detected pixels that are further from the predicted position. We then compute the new target position and spatial extent within the enlarged ROI as described above.

Note that this is NOT done for initial target detection in "entry" regions.

Estimating target confidence

Targets can disappear due to occlusion or other errors. As explained above, we estimate the confidence in the detection, CF, as the average detection probability within the ROI.

$$CF = \frac{S}{(b-t)(r-l)} \qquad \text{where} \quad S = \sum_{i=l}^{r} \sum_{j=t}^{b} T(i,j)$$

When the target is initially detected, the confidence factor for the blob is initialized from the confidence factor of the detection.

$$CF_0 = CF$$

Subsequently, during tracking, the detection confidence can be used to recursively update the confidence for the tracked blob, CF_t, with an update factor α.

$$CF_t = \alpha \cdot \frac{S}{(b-t)(r-l)} + (1-\alpha) \cdot CF_{t-\Delta t}$$

We test the resulting

$$\text{if } CF_t \leq CF_{min} \quad \text{Target Lost .}$$

CF_{min} is the minimum required average probability per pixel to detect a target.
In this case the target is removed.