# Probabilistic Detection and Tracking at High Frame Rates Using Affine Warping

Aleš Ude[1,2] Christopher G. Atkeson[1,3]

[1]ATR, Human Information Science Laboratories, Department 3
2-2-2 Hikaridai, Seika-cho
Soraku-gun, 619-0288, Japan

[2]Jožef Stefan Institute
Dept. of Autom., Biocyb. and Robotics, Jamova 39
1000 Ljubljana, Slovenia

[3]Carnegie Mellon University
Robotics Institute
5000 Forbes Avenue, Pittsburgh
PA, 15213, USA

## Abstract

*This paper addresses two vital issues that can affect real-time operation of a visual tracking system: the realization of an effective subsampling policy and the real-time initialization of the tracking algorithm. We propose to use affine warping to subsample the images selectively only in those regions that contain too much data for real-time operation. The automatic detection of objects of interest in images captured by a moving camera is based on random search which enables us to set all thresholds automatically without any user support. Using these methods, we implemented a probabilistic tracker that can detect and track up to 10 objects at 60 Hz on a dual processor 933 MHz Pentium III PC.*

## 1. Introduction

We are interested in interaction between people and humanoids. The ability to observe humans and their actions using a set of cameras mounted on a humanoid robot's head is a necessary prerequisite towards this end [5]. We would greatly benefit from a system that can detect and track multiple objects in images acquired from cameras in motion at high frame (field) rates, e. g. at 60 Hz, which is the highest rate we can get from a standard, interlaced NTSC camera.

Probabilistic "blob trackers", often based on the maximization of some sort of a likelihood function, have become increasingly popular in recent years. A number of blob trackers using various modalities such as color histograms, Gaussian color mixtures, intensity gradients, depth, optical flow or a combination of these modalities have been proposed up to now [1, 2, 3, 4, 5, 6]. This paper describes two techniques that improve real-time operation of such systems on standard PCs: the selection of an effective subsampling policy and the real-time detection of objects of interest.

Due to space limitations we only briefly describe the implemented tracking framework in the rest of this section. The main topics of this paper, i. e. the implementation of a subsampling policy and the real-time initialization of the tracking algorithm, are described in Section 2 and 3. This is followed by experimental results and discussion.

### 1.1. Probabilistic Framework

We represent the observed environment by a number of random processes (blobs). Let's denote the probability that a pixel located at $\boldsymbol{u}$ having color intensity $I_{\boldsymbol{u}}$ was generated by the process $\Theta_k,\ k = 1, \ldots, K$, by $\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_k)$. We also introduce two additional processes: the optional background process $\Theta_{K+1}$, which describes the stationary background (useful only for fixed cameras), and the outlier process $\Theta_0$, which models the data not captured by other processes. Assuming that every pixel stems from one of the mutually exclusive processes $\Theta_k,\ k = 0, \ldots, K+1$, we can write the probability that color $I_{\boldsymbol{u}}$ was observed at location $\boldsymbol{u}$ using the total probability law

$$\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\boldsymbol{\Theta}) = \sum_{k=0}^{K+1} \omega_k \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_k), \tag{1}$$

where $\omega_k$ is a prior probability to observe the process $\Theta_k$, $\sum_{k=0}^{K+1} \omega_k = 1$, and $\boldsymbol{\Theta} = \{\Theta_0, \Theta_1, \ldots, \Theta_{K+1}\}$. Under these assumptions, the posterior probability that pixel $\boldsymbol{u}$ stems from the $l$-th process is given by the Bayes' rule

$$p_{\boldsymbol{u},l} = \frac{\omega_l \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_l)}{\sum_{k=0}^{K+1} \omega_k \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_k)}. \tag{2}$$

Ignoring the correlation of assigning neighboring pixels to processes, the overall probability can be approximated by

$$\mathrm{P}(\boldsymbol{I}) = \mathrm{P}(\boldsymbol{I}|\boldsymbol{\Theta}) = \prod_{\boldsymbol{u}} \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\boldsymbol{\Theta}). \tag{3}$$

At each time step, we would like to determine $(\Theta_1, \ldots, \Theta_K, \omega_0, \omega_1, \ldots, \omega_{K+1})$ so that likelihood (3) is maximized. Instead of maximizing criterion (3) directly, it is often easier to minimize its negative logarithm (log-likelihood).

Before we can minimize the log-likelihood, we must decide how to model the process distributions $\Theta_k$. Our approach uses shape and color properties to evaluate the probability that a pixel was generated by one of these processes. Assuming that these two properties are independent of each other, we have

$$\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_l) \sim \mathrm{p}(I_{\boldsymbol{u}}|\Theta_l)\mathrm{p}(\boldsymbol{u}|\Theta_l). \tag{4}$$

In many cases, for example when tracking body parts, the 2-D shape of the tracked objects is roughly ellipsoidal and can be approximated by the center of the object's image $\boldsymbol{x}_l$ and by the covariance matrix $\boldsymbol{\Sigma}_l$ of pixels contained in it. Thus the shape part of the probability that pixel $\boldsymbol{u}$ belongs to the $l$-th blob can be characterized by a Gaussian distribution

$$\mathrm{p}(\boldsymbol{u}|\Theta_l) = \frac{1}{2\pi\sqrt{\det(\boldsymbol{\Sigma}_l)}} \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_l)\boldsymbol{\Sigma}_l^{-1}(\boldsymbol{x}-\boldsymbol{x}_l)). \tag{5}$$

For the object's color probability, we assume that it can be modeled by a Gaussian mixture model

$$\mathrm{p}(I_{\boldsymbol{u}}|\Theta_l) = \sum_{k=1}^{K_l} \omega_{l,k}\,\mathrm{p}(I_{\boldsymbol{u}}|\overline{I_{l,k}}, \boldsymbol{\Gamma_{l,k}}), \tag{6}$$

where $\sum_{k=1}^{K_l} \omega_{l,k} = 1$ and

$$\begin{aligned}\mathrm{p}(I_{\boldsymbol{u}}|\overline{I_{l,k}}, \boldsymbol{\Gamma}_{l,k}) \;&=\; \frac{1}{\sqrt{(2\pi)^{2 \text{ or } 3}\det(\boldsymbol{\Gamma}_{l,k})}}\cdot \\ &\exp(-\frac{1}{2}(I_{\boldsymbol{u}} - \overline{I_{l,k}})\boldsymbol{\Gamma}_{l,k}^{-1}(I_{\boldsymbol{u}} - \overline{I_{l,k}})).\end{aligned} \tag{7}$$

The outlier process is modeled by a fixed uniform distribution and the background colors are modeled by unimodal Gaussians at each image pixel.

The blob and background colors are kept constant in the current version of the tracker. They are learnt off-line. Thus at each tracking step we need to maximize (3) over shape parameters $\{(\boldsymbol{x}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^{K}$ and mixture probabilities $\{\omega_k\}_{k=0}^{K+1}$. A good iterative approach is provided by an EM-algorithm, in which this is done by first calculating the posterior probabilities $p_{\boldsymbol{u},l}$ (given by Eq. (2), (4), (5), (7)) using the current estimate for $\{\Theta_k\}$ and $\{\omega_k\}$ (the expectation step) and then estimating the parameters $\{(\boldsymbol{x}_k, \boldsymbol{\Sigma}_k)\}$ and $\{\omega_k\}$ as if $p_{\boldsymbol{u},l}$ were constants independent of them (the maximization step). The maximization step consists of calculating the weighted mean and covariances of image pixels with $p_{\boldsymbol{u},l}$ being used as weights and of the reestimation of $\{\omega_k\}$. This process is repeated until convergence.

## 2. Subsampling through Affine Warping

Complex visual routines such as the tracker described above necessitate a substantial amount of processing at each image pixel and therefore cannot be applied to whole images in real-time. Since it is usually not feasible to simplify the processing at each image pixel without making the resulting tracker less reliable, the alternative approach taken by many practical tracking systems is to apply techniques such as windowing, masking and subsampling to reduce the amount of information that needs to be processed.

To ensure that the processing time needed to process each of the blobs is approximately the same, it is desirable to subsample a region around the current blob location to a window of constant size so that the same number of pixels needs to be processed for each blob. The problem is that the size of a blob can change as it moves towards or away from the camera, making a fixed window size inappropriate.

A common feature of blob trackers such as the ones described in [1, 2, 3, 4, 5, 6] and Section 1.1 is that they approximate the shape of tracked objects by the second order statistics of pixels that are probabilistically classified as "blob pixels". By computing the eigenvalue decomposition of the associated covariance matrices we can estimate the extent of the blobs along their major and minor axes, i. e. calculate the location and shape of ellipses enclosing the blob pixels. As the lengths of both axes can differ significantly, it makes sense to subsample the image along the principal blob directions instead of image coordinate axes and to apply a different scaling factor along each of these directions considering the length of the corresponding axis.

Subsampling along the principal directions can be implemented by applying the following transformations: (1) translate the blob so that its center is aligned with the origin of the image, (2) rotate the blob so that its principal directions are aligned with the coordinate axes, (3) scale the blob so that its major and minor axis are shorter than the sides of a predefined window, (4) translate the blob so that its center is aligned with the center of the new window. The resulting mapping in homogeneous coordinates is given by the following affine transformation:

$$\begin{aligned}\boldsymbol{A}_k^i &= \begin{bmatrix} 1 & 0 & \frac{w}{2} \\ 0 & 1 & \frac{w}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{w}{2a_k^i s} & 0 & 0 \\ 0 & \frac{w}{2b_k^i s} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}(\theta_k^i)^T & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 & 0 & l^i - u_k^i \\ 0 & 1 & -v_k^i \\ 0 & 0 & 1 \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix},\end{aligned} \tag{8}$$

where $\boldsymbol{u}_k^i = [u_k^i, v_k^i]^T$ and $\theta_k^i$ are the position and orientation of the $k$-th blob at $i$-th measurement time, $a_k^i$ and $b_k^i$ are the half lengths of its major and minor axis, $w$ is the predefined size of the window onto which we map the region around the blob and $s$ is a scale factor specifying how much smaller than the target window should the mapped blob be. This is necessary to ensure that the mapped region really contains the tracked object because we do not know the exact blob parameters in the next image field in advance. The blob parameters $(u_k^i, v_k^i, \theta_k^i, a_k^i, b_k^i)$ are estimated by a prediction process. In our system, the prediction of blob parameters is based on a discrete 2nd order dynamical system

$$x_i = ax_{i-1} + bx_{i-2} + e_i, \tag{9}$$

where $x_i$ is one of $(u_k^i, v_k^i, \theta_k^i, a_k^i, b_k^i)$ and $e_i$ is the system noise. Factors 2 and $l_i$ that appear in the last matrix of Eq. (8) ($l_i$ is 0 or 1 depending on which field we work with)

**Figure 1.** Warping three regions around the tracked body parts. The original image overlayed with the estimated blobs (displayed as crosses) is also shown.

account for the fact that we process image fields and not frames whereas the blob parameters are estimated in a full frame coordinate system.

If a tracked blob is sufficiently small so that we can assume that at the next measurement time it will stay within the window of size $w$ centered at $\boldsymbol{u}_k^i$, i. e. if $2 * b_k^i \leq w/s$, then we do not need to warp the input image but simply copy the quadratic region of interest onto the target window. In this way we ensure that we do not scale blobs that are already small enough to be processed in real-time.

The process of geometrically transforming the input image by the affine mapping given in Eq. (8) is know as *affine warping*. Since matrices $\boldsymbol{A}_k^i$ are invertible, we can implement the affine warping by parsing through the pixels of the output window, which is often smaller than the region of interest containing the blob, and by applying the inverse mapping ${\boldsymbol{A}_k^i}^{-1}$ to each of the pixels in this window. We estimate the associated color intensities at these positions either by a nearest neighbor or linear interpolation. The transformed pixel positions are also stored because our tracker needs them for the estimation of the next blob position.

Trackers such as the one described in Section 1.1 can then be applied to the warped images (see Fig. 1). The expectation phase consists of evaluating conditional probabilities (2) at all pixels of the warped image. Substantial processing time savings can be achieved in this way without losing accuracy because we reduce the amount of information only in those directions in which we have too much data. The maximization phase can also be carried out efficiently using the pixel values stored when warping the original image. Since the size of the warped image is fixed, we can ensure that the processing time per blob is approximately constant and thus guarantee the real-time operation of the system.

### 2.1. Parallel Processing

To further reduce the computation time, we implemented a parallel version of the tracker on a dual processor PC using the multithreading facility provided by Windows 2000 operating system. Our algorithm can be parallelized at different levels, but it quickly turned out that fine level parallelism does not result in processing time savings because of the incurred overhead. For this reason, we parallelized the algorithm at a rather coarse level by splitting the blobs in two groups and starting one thread per blob group.

The denominator of Eq. (2) makes the blobs aware of each other. This makes it necessary to synchronize both threads to ensure that all probabilities $\mathrm{p}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_l)$ are evaluated before calculating the posterior probabilities $p_{\boldsymbol{u},l}$ from Eq. (2). The problem is that these probabilities are evaluated in the warped spaces and not in the original image space. Therefore we need to rewarp the calculated probability maps. The affine mapping that needs to be applied to warp the probability map of blob $\Theta_{l_1}$ onto the probability map of blob $\Theta_{l_2}$ is given by ${\boldsymbol{A}_{l_1}^i}^{-1} * \boldsymbol{A}_{l_2}^i$. The rewarping operation needs to be carried out only when the regions of interest associated with both blobs intersect.

## 3. Automatic blob detection via random search

Automatic detection of objects of interest and the subsequent initialization of the tracker is a very necessary part of every practical tracking system. As we are interested in dynamic scenes captured by cameras in motion, it is necessary that the detection algorithm runs as fast or faster than the tracking algorithm. It is useless to come up with a result after a long analysis of one image because the object of interest or cameras might move to a different location before the processing is finished. In addition, a practical system should not expect from a user to set various parameters for different scenes and objects because this is tedious. The ground knowledge for our system is provided by color and shape probability distributions. As it is time consuming to search for ellipsoidal objects in an image, we use color only as ground knowledge to initialize the tracker.

Based on color, the probability that a pixel belongs to the $l$-th blob is given by Eq. (6). Since we do not have any information about the initial state of the blobs, we randomly select their shapes and locations in the image. The shape parameters are varied in a controlled way so that 2-D sizes of the generated blobs remain within prespecified limits. To achieve real-time operation, we warp a region of interest around each of the blobs onto a window of fixed size as described in Section 2. Color probabilities (6) are then estimated at each pixel of the warped image. If the sum of all probabilities within the window exceeds a certain threshold, i. e.

$$\mathrm{f}(\Theta_l) = \sum_{\boldsymbol{u}} \mathrm{p}(I_{\boldsymbol{u}}|\Theta_l) > r_l, \tag{10}$$

the region is deemed interesting and the tracker is started using the associated, randomly selected blob parameters.

**Figure 2.** Successful tracking in various situations.

| $E(f(\Theta_l))$ | $\max_{t_i} f(\Theta_l)$ | $E(f(\Theta_l))$ | $\max_{t_i} f(\Theta_l)$ |
|---|---|---|---|
| 1.925e-003 | 1.246e-001 | 6.975e-003 | 1.433e-001 |
| 3.956e-004 | 2.206e-002 | 2.329e-005 | 1.044e-003 |
| 5.121e-004 | 3.229e-002 | 1.944e-004 | 2.162e-002 |
| 4.852e-003 | 5.599e-001 | 1.245e-003 | 2.718e-002 |

**Table 1.** Average and maximum sum of color probabilities for 8 blobs over a period of five seconds

It is not possible to select thresholds $r_l$ in advance because they depend on lighting conditions, the variability of color within the object, and the quality of a color model. For example, randomly searching through a video stream for 5 seconds (300 images) while looking for 8 objects of different colors resulted in the average and maximum probability sums that are shown in Table 1. Analyzing these results it becomes clear that we cannot set a single threshold that would account for all possible situations. Therefore we break the initialization process in two phases. First we explore the video stream for a sufficient period of time (typically for 5 seconds) and sample the sums from Eq. (10). We can then set a threshold for each of the sought objects to

$$r_l = (1-\lambda)\mathrm{E}(f(\Theta_l)) + \lambda \max_{t_i} f(\Theta_l),\ 0 < \lambda < 1. \quad (11)$$

$\lambda$ was typically set to 0.67 in our experiments.

In the second phase we restart the random search. The $l$-th object is deemed found once the automatically selected threshold $r_l$ is exceeded. If the tracker loses the object, we restart the initialization process either with the first or with the second phase depending on the application. Hence our system is also able to recover from failure.

## 4. Experimental Results and Discussion

The first four images in the upper row of Fig. 2 show stereo hand and head tracking from a robot camera in motion. This output was used to realize the oculomotor behavior of smooth pursuit and real-time mimicking of hand motion. Simultaneous tracking of five objects is presented in the rightmost image. At 60 Hz, we were able to track in real-time at most 10 objects on a dual processor Pentium III PC, with the window size $w$ for affine warping set to 41.

The lower row shows that our tracker works reliably also under varying illumination. Unlike other approaches, we used multiple color models taken under various lighting conditions to solve this problem. The sum of color probabilities from Eq. (10) was used as a criterion to select the color model that best explains the current situation. Three separate color models were used in this example. In this way we can avoid the pitfalls of on-line adaptation of color models, although a combination of both, i. e. multiple models and on-line color adaptation, is probably the best solution. Such an implementation was made possible by the proposed subsampling policy which enabled us to evaluate a number of color models at each pixel in real-time.

**Acknowledgment:** Support for Chris Atkeson was also provided by US National Science Foundation Award IIS-9711770.

## References

[1] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, pp. 569–574, San Juan, Puerto Rico, 1997.

[2] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition, Vol. 2*, pp. 142–149, Hilton Head, South Carolina, 2000.

[3] N. Jojic, M. Turk, and T. S. Huang. Tracking self-occluding articulated objects in dense disparity maps. In *Proc. 7th Int. Conf. Computer Vision*, pp. 123–130, Kerkyra, Greece, 1999.

[4] S. J. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 17(3-4):225–231, March 1999.

[5] A. Ude, T. Shibata, and C. G. Atkeson. Real-time visual system for interaction with a humanoid robot. *Robotics and Autonomous Systems*, 37(2-3):115–125, November 2001.

[6] Y. Wu and T. S. Huang. A co-inference approach to robust visual tracking. In *Proc. Eight Int. Conf. Computer Vision, Vol. II*, pp. 26–33, Vancouver, Canada, 2001.