

Using Multiple Views To Resolve Human Body Tracking Ambiguities

Alex Leykin, Florin Cutzu and Mihran Tuceryan
Computer Science Department
Indiana University
Bloomington, IN 47405-7104, USA
oleykin@cs.indiana.edu
florin@cs.indiana.edu tuceryan@iupui.edu

Abstract

This paper outlines the theoretical background and presents a new approach to human body tracking with monocular static camera. A novel “view-based representation” is introduced at the feature extraction stage. We show that ambiguities in correspondence, such as the ones that occur as the result of occlusion, can be resolved by using this approach. In particular, we store color information for each object in a vector of views, where the number of elements is determined online, using unsupervised clustering followed by the cluster validity assessment. Based on this representation a tracking system was developed. The preliminary results presented show the discriminative potential of the proposed system.

1 Introduction

The majority of modern tracking algorithms consist of primarily three stages [10]. First, the objects of interest or *foreground* have to be separated from noise or *background*. In case of a static or motionless camera this can be done by creating a certain background model either a priori or during the run time and, consequently, subtracting this model from each frame of the tracking sequence. Second, blobs of distinct shape, potentially corresponding to the objects being tracked have to be extracted from the resulting scene and feature vectors are to be built descriptive of every blob. The last, third stage, is concerned with matching each blob to the object (human body in our case) over time and space.

In this work we have concentrated on the second, representational stage. We approach it by treating visual features that describe each moving blob as two distinct subsets. The first set - *view independent* features, are invariant of the position and orientation of the human body as well as the illumination of the scene. For the second set contains *view-based* features, where each object is thought of as having multiple views. The tracker keeps the information about each view separately.

Having accumulated information about each view, the program can compute matching of the model with the blob to each view independently and to choose the best match as the current view. Implementation of multiple views resolves ambiguities while matching

blobs in each frame to the human body objects in the system. In particular the system recognizes humans correctly after being fully occluded by another moving person (Figure 3).

In the following section of this paper we discuss the techniques for representation and feature extraction existing in the tracking literature. Following that, we outline the three stages of our tracking algorithm with the stress on the multiple-view representation in sections 3.2 and 3.3. Finally, we discuss the results obtained by using our tracker, draw conclusions and outline several prospects for the future work.

2 Related work

One of the forms of view-based object representations in computer vision literature, is eigen spaces. In such a layout as an object undergoes affine transformations along the period of time it is matched to a set of the eigen views. These schemes can be used to track primarily rigid objects [2], where the model for each object is acquired upfront, during the learning stage. Although the eigenspace accumulated the information about objects' color, shape and texture, it is rather difficult to utilize it for real-time tracking as these three can change independently during tracking time. Specifically, various scaling approaches have to be applied to bring the size of the object into correspondence with that of the eigenview. Allowing for these constraints, eigen-view modeling has been used successfully primarily for face tracking [4, 13].

More implementation-oriented works have utilized simplistic feature extraction techniques. For example, in [8, 17] each object's color and position are recorded and no feature model is built. In complex tracking situations, this kind of blending of the views can result in a number of ambiguities, for instance, when all aforementioned features are identical for two objects undergoing a split event after occlusion.

In [3, 1] the tracking feature is a distribution of colors represented by a color histogram, which is compared with a histogram of colors observed within the current region of interest. This region of gaussian shape is obtained most of the times by some form of EM algorithm. The method proved to be very productive and tolerant to partial occlusions. It was demonstrated primarily in tracking single moving objects (e.g. human faces) in less restricted environment (moving camera), but it was not subjected to rigorous testing on video sequences with multiple moving actors.

Dramatically different approach is to build a 3-D model of the object being tracked [16, 6]. This is a graphically and computationally intense approach with more than one high-resolution camera required. Therefore, most of the such algorithms operate in very limited environment with no occlusions. Human body is modeled by stick figures or combinations of blobs, where color does not play a crucial discriminatory role, for the authors are after recognizing human activity, not tracking.

3 Method

3.1 Background subtraction

To subtract the background we have employed an adaptive illumination-invariant method that operates in HSV space. To discriminate between the moving objects and the static

background, we have exploited the notion of chromaticity and brightness distortion to isolate highlights and shadows from the actual moving objects (see [7]).

3.1.1 Building the model

Each pixel i of the background was modeled by a 4-tuple $\langle \mu_i, \sigma_i, \gamma_i, \beta_i \rangle$, where μ_i is the expected color value, σ_i is the standard deviation of color value, γ_i is the variation of the brightness distortion, and β_i is the variation of the chromaticity distortion of the i -th pixel. Let $I_i = [I_H(i), I_S(i), I_V(i)]$ be i -th pixel in the current frame. If the color and brightness distortion are denoted by C_i and B_i , then the system is a set of equations [Eq. 1- 6]:

$$\mu_i = [\mu_H(i), \mu_S(i), \mu_V(i)] \quad (1)$$

$$\sigma_i = [\sigma_H(i), \sigma_S(i), \sigma_V(i)] \quad (2)$$

$$C_i = \sqrt{\left(\frac{I_S(i)^2 + \mu_S(i)^2}{\sigma_S(i)^2} \right) - \frac{2 * I_S(i) * \mu_S(i)}{\sigma_S(i)^2} * \cos \left(\frac{I_H(i) - \mu_H(i)}{\sigma_H(i)} \right)} \quad (3)$$

$$B_i = \frac{|I_V(i) - \mu_V(i)|}{\sigma_V(i)} \quad (4)$$

$$\gamma_i = \frac{\sum_{n=1}^N C_n(i)}{N} \quad (5)$$

$$\beta_i = \frac{\sum_{n=1}^N B_n(i)}{N} \quad (6)$$

The variations of color and brightness distortion in eq. 5, 6 are obtained as the averages of C_i and B_i over N video sequence frames. The background model in this system was initialized over a small number of frames (seconds). After that an adaptive process was used to update background model, where the values from [Eq. 1-6] were accumulated as running aggregates.

3.1.2 Subtracting the model

When the background model is initialized, for each input frame I we mark the pixel I_i as belonging to the foreground if:

$$\frac{C_i}{\gamma_i} > T_C \text{ and } \frac{B_i}{\beta_i} < T_B \quad (7)$$

where T_C is the threshold to ensure a sufficient level of color distortion, given the history of the distortion at pixel i ; and T_B is the threshold to ensure that extremely dark foreground pixels, for which C_i is always small (due to their proximity to the origin of the color space) get marked as a foreground.

The output of the subtraction process is a binary map image, with *ones* designating a foreground object and *zeros* marking the background. We subsequently subject the binary image to the standard contour extraction algorithm, which produces a vector B of *blobs*. Each blob is a moving area in the video frame that is a potential candidate for tracking process (see Fig. 1).

3.2 View-based vs. view independent features

As an object moves through the field of the camera view, the appearance of the object may undergo significant changes due to a number of factors. The translation of the object across the image plane may change the size of the object as well as the aspect currently observed by the camera. The rotation of the object itself radically changes the view, with respect to its color, texture and shape. For non-rigid objects, such as human bodies each rotation can be accompanied by a complex shape transformation as in walking or sitting down and standing up. The most complicated, perhaps, is the change of object's color due to the variations of incident light cast by other objects and varying distance from light sources.

With observations above in mind we came to the conclusion that while tracking non-rigid objects for long periods of time one has to consider two types of features that can be available as the descriptors for each object: *view independent* and *view-based* features. We define *view* as the visual appearance of the tracked object as the result of its particular location with respect to the camera.

The view independent features considered in our study - object position, velocity and dimensions - form a 6-tuple $\langle p_x, p_y, v_x, v_y, d_x, d_y \rangle$. The position is determined as the coordinates of the the center of mass of the blob contour. Velocity, is obtained as a discrete differential between p_x and p_y in the current and previous frames. The dimensions are the width and height of the rectangle bounding the blob. Being view independent, these features do not undergo any significant changes with the change of the object's orientation (assuming human upright posture), neither they are dependent on the variations in illumination.

Perhaps, the characteristic of a moving human that varies the most as the person changes orientation is color. We have observed significant changes in color histogram as a single person rotates in one place in front of the camera. These changes are predominantly due to two factors: differences in color of the clothing in front and at the back (especially, when wearing an unbuttoned coat or shirt) and different proportions of facial and hair color in different views (e.g. in the side view there may be more skin color, because of the exposed arms). Therefore, the appearance of each human body with respect to color can be conveniently encoded by a vector of *normalized color signatures*

For each human body model we have computed a histogram vector $H = [h_1, h_2, \dots, h_K]$, each representing a different view of the person. Each histogram h is a two-dimensional histogram in H and S space with 32 bins in each dimension. The number of bins was chosen empirically to provide enough discrimination between different average hue and saturation, yet at the same time not to overload the system with redundant color information.

By looking at the 2-D color histogram in figure 2 one can notice the sparse character of this matrix. Therefore a variable length normalized color signature s was created for each h as follows:

$$A = \sum_{x=1, y=1}^{32} h_{xy} \quad (8)$$

$$\text{if } h_k > A * \delta \text{ then } s_i = \frac{h_k}{A} \quad (9)$$

where δ determines how many percent of the total color distribution must a color represent in order to be in the signature. This number was empirically chosen to be 0.01. Usually from the 32x32 histogram this would yield a color signature of length 5 to 10 elements (see Fig. 2), significantly reducing the redundancy. The signature of the object which is less diverse in color simply contains fewer elements. The resulting vector S served as the final color descriptor of each view.

3.3 Clustering of The Views

The question arises, how many views are there for each object? The easiest implementation would be to represent each human body with the fixed number of views. This technique, however, will create unnecessary redundancy for the objects undergoing smaller changes in appearance. Another reasoning is to implement a simple thresholding scheme to create new views, i.e for each object, while matching to the histogram of a blob \hat{h} , create a new view if:

$$\hat{h} > h_k, \forall k = \overline{1, K} \quad (10)$$

Thus the creation of a new view will depend solely on the choice of the threshold. A low threshold will result in overabundance of insignificant views created, a high one may prevent useful views from being created.

To surmount the shortcomings mentioned above, we deal with the creation of new views as with a clustering problem. Let $J = [j_1, j_2, \dots, j_L]$ be a vector of histogram observations, then each element j_l represents a sample in a 16-dimensional space (the number of histogram bins). We applied standard k-means clustering algorithm to assign each of L samples to one of K clusters. The clusters were then re-labeled to correspond their counterpart from the previous step. We used a variant of the *Hungarian method* for bipartite graph matching to perform re-labeling (see [11]).

Initial clustering of the views was performed only after the object has been in the scene for several seconds, to let the view samples constitute a set substantially representative of the objects color properties. Then re-clustering was performed every 100 frames, to ensure that the discriminatory power of the views does not suffer from the bad choice of random initial centroids.

Clustering has a computational cost polynomially increasing with the number of samples. To overcome it, we have kept the number of samples constant once it reached a certain level by withdrawing older samples and pushing new ones into vector J . This especially makes sense, since the color descriptors of the object may become outdated as it moves to the different environment or changes its configuration (e.g a person, taking off his coat). Thus the length of vector J was maintained at the level, sufficient to accommodate 10 seconds of video (approx. 300 elements).

3.3.1 Distance measures

As the distance measure between two histograms therefore between the cluster centroids and sample points in each cluster we used the earth movers distance (*EMD*) [15]. We have chosen *EMD* as the distance measure because it takes into account colors in bins and can be operating on 2-dimensional nature of histograms and does not suffer from quantization problems. There is one more feature of *EMD* that was specifically useful in the case of

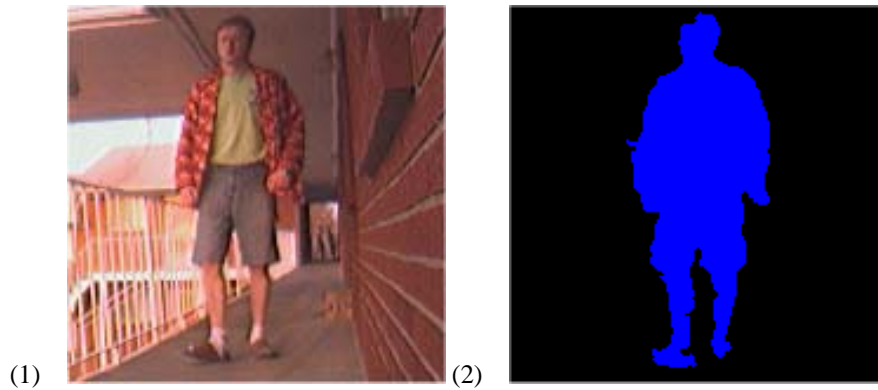


Figure 1: (1) original image (2) blob detection

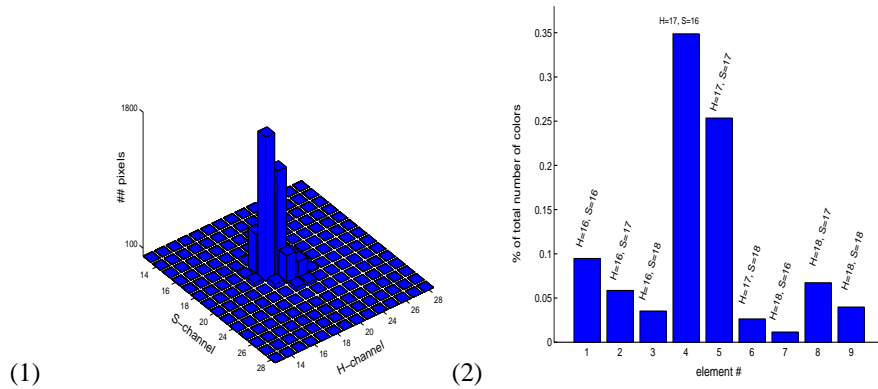


Figure 2: (1) center cut from the 32x32 histogram in HS-space (2) normalized color signature (the values of H and S above each bar represent bin coordinates in HS space)



Figure 3: Select frames from tracking sequence. Two moving objects merge into a single blob (frame 502) and split (frame 542). The objects are color coded with BLUE corresponding to the first object and GREEN to the second one. The correct labels are restored after the merge-split sequence

the histograms that change over time. Let us consider a histogram of a human body at a frame f and the histogram of the same body at the frame $f + 1$:

If the object undergoes a slight shift in hue or saturation values due the fluctuations in ambient lighting, the histogram bins will shift as well. Now, computing a simple square distance would result in the difference between h_f and $h_{(f+1)}$ significantly higher than the actual perceptual change taking place in the image. The use of EMD allows to avoid excessively strict punishment for such shift both in hue and in saturation space. To form the input suitable for EMD algorithm, each of the histograms was converted to a variable length *normalized color signature* as described in section 3.2.

Cluster centroids are usually computed as the average of all the points that belong to the cluster. To do this, however, we need an *addition operation* that has not been defined for two signatures of different length. To overcome this problem, for each view we accumulate the histograms from each view sample assigned to this view. This way normalized color signature can be computed from aggregate view histogram at all times.

3.3.2 Clustering tendency

In many cases, such as people dressed in the uniformly colored clothing, a single view is sufficient to grasp all the color information. Thus as the first step of the clustering process it is necessary to determine if the view information is predisposed to have clusters and if not then do not proceed with the clustering process. To determine the clustering tendency we used one of the nearest neighbor measures method [9, 12].

To answer the question if the data should remain as one big cluster or be further split into smaller clusters we perform the k-means clustering with $K = 2$ as described above. Then, for the resulting distribution we compute:

$$I = \frac{1}{L} \sum_{i=1}^L f_{\beta}(j_i), \quad (11)$$

where $f_{\beta}(j_i)$ is the fraction of the β nearest neighbors of j which have the same label as j . This value I ranges from 0 to 1 and will be the lowest when a clustering breaks apart a compact group of data because the values of $f_{\beta}(j_i)$ will be small along the borderline between two clusters. If the two clusters are well separated in space, I will be close to 1. We have chosen the empirical threshold of $I > 60$ to be adequate to proceed with multiple view/clusters.

3.3.3 Cluster validity

In fixed-cluster algorithms, the number of clusters is specified a priori, and the clusters are formed from random initial conditions. Our clustering algorithm iterates for number of clusters K ranging from 1 to 6. At each iteration the validity of clustering was assessed using Dunn's validity index [5]:

$$D = \min_{1 \leq i \leq n} \left\{ \min_{1 \leq j \leq n} \left\{ \frac{\min_{\forall c_i \in C_i, c_j \in C_j} d(c_i, c_j)}{\max_{1 \leq k \leq n} d^0(c_k)} \right\} \right\}, \quad (12)$$

where $d(c_i, c_j)$ is the distance between clusters C_i and C_j (inter cluster distance), $d^0(c_k)$ - intra cluster distance of cluster c_k and n is the number of clusters.

We select K which maximizes D thus maximizing inter cluster spread and minimizing intra cluster variance.

3.4 Tracking

As described in section 3.1 at each frame our system is presented with a vector of blobs - $B = [B_1, B_2, \dots, B_M]$. As the result of the tracking algorithm applied at the previous frames the system stores the state of each object being tracked in the vector $O = [O_1, O_2, \dots, O_N]$. Each element from B is described by a vector of features, namely a 7-tuple $\langle \hat{p}_x, \hat{p}_y, \hat{v}_x, \hat{v}_y, \hat{d}_x, \hat{d}_y, \hat{h} \rangle$, where h is the histogram of the blob at the current frame. Each element from O is described by a 7-tuple $\langle p_x, p_y, v_x, v_y, d_x, d_y, H \rangle$, where H is a vector of objects' views. As the next step the global distance matrix D is computed as follows:

$$D_{ij} = D(B_i, O_j), \quad (13)$$

where D - is the distance from blob i to the object j . D is obtained as a weighted sum of the distance of each of 6 pairs of view independent features, plus the distance from h to the closest view in H :

$$\begin{aligned} D(B_i, O_j) = & w_p * [d(\hat{p}_x, p_x) + d(\hat{p}_y, p_y)] + \\ & + w_v * [d(\hat{v}_x, v_x) + d(\hat{v}_y, v_y)] + \\ & + w_d * [d(\hat{d}_x, d_x) + d(\hat{d}_y, d_y)] + \\ & + w_h * \min_{\forall h_i \in H} [d(\hat{h}, h_i)] \end{aligned} \quad (14)$$

The weights w_p , w_v , w_d and w_h were determined empirically and are based on the matching scenario. There are several scenarios to consider with respect to M .

Most of the time during the tracking process the number of blobs M detected is equal to the number of objects in the system $M = N$, when no full occlusions have occurred. In this case the object position information is the most important, therefore, utilizing the motion smoothness assumption we assign a greater value to w_p . According to the linear motion velocity model the v_x and v_y will remain constant and the size of the object will change only slightly. We set w_v and w_d to be higher values correspondingly.

Another type of matching we perform in case when either $M > N$ or $N > M$, which reflects either objects disappearing due to occlusions (merge event) or objects appearing due to a split event. Note that we did not consider objects exiting from the scene and reappearing later. The occlusions are caused either by the background object or by another moving person. For later, both *merging* bodies are assigned to the same blob and are tracked as one until the split event occurs. The blob size becomes irrelevant ($w_d = 0$) and values of w_p and w_v are also reduced to minimum: the tracking is performed mostly based on the color information. In this scenario multiple views serve to disambiguate between two objects' details.

Subsequently, to matrix D we apply the algorithm for close to optimal object-to-blob matching outlined by Rangarajan and Shah that runs in $O(\max(M, N))$. The algorithm tries to minimize the cost of local and global matches and the same time. For more details see [14].

4 Experimental results

The real-time tracking system in place processes the input from a DirectX video stream. This is a video-file or a camera, connected via the Fire-Wire port. The system performed at approximately 10 Hz on input frames 320 by 240 pixels on a AMD K7 1GHz processor. The tracker was tested on the sample sequences with up to 3 moving subjects both indoors and outdoors and in the free-tracking experiment with up to five arbitrary moving subjects.

The algorithm demonstrated the steady ability to handle merge-split events for the subjects with similar clothing and to maintain the number of the objects in the scene constant once they enter. Tracking through longer periods of time with subjects re-entering poses a different problem and will be addressed in future work.

5 Conclusions and future work

In this paper we presented a new algorithm for tracking moving objects in the situation where these objects are occluded. The key elements presented were the adaptive background model, utilization of multiple views for each object to store a more detailed and exact representation of each human body. Additionally, procedures to minimize the cost of local and global matching have been employed. The algorithm allows to address the situation of occlusions effectively and to detect merging and splitting events. It provides the flexibility to the system necessary for tracking in real-life environments. The testing sessions demonstrated the stable and high-quality performance of the program. Each object in the video sequence has been identified and tracked during several merge-split events. Although the algorithm is fully implemented and can be used in various applications (such as security applications), there are several areas of future work and development. Thus, improvements can be made to make the background subtraction method truly adaptive. Currently, if a moving object enters a scene but then stops it will be gradually blended into the background model. By determining the type of the object we can either exclude it from further background model accumulation process (e.g. a human) or treat it as a part of the background (e.g. a briefcase).

Another characteristic which can be improved is related to our notion of a view. Current system is designed around (although is not limited to) the human subjects walking in the upright posture. We are currently working on broadening the definition of view to include the transitions in objects shape, such as sitting down and standing up.

As one of the future developments of this work we plan to incorporate human body shape descriptors, e.g. *Hu-moments*, as one of the view-based features.

References

- [1] An em-like algorithm for color-histogram-based object tracking. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2004.
- [2] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *In Proc. of European Conference on Computer Vision*, 1996.

- [3] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *In Proc. of IEEE Conf. Computer Vision and Pattern Recognition*, 2000.
- [4] Fernando de la Torre, Shaogang Gong, and Stephen McKenna. View alignment with dynamically updated affine tracking. *In Proc. of IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.
- [5] J. C. Dunn. Well-separated cluster and optimal fuzzy partition. *J. Cybern*, 1974.
- [6] D.M. Gavrila and L.S. Davis. Towards 3-D Model-based Tracking and Recognition of Human Movement. *Int. Workshop on Face and Gesture Recognition*, 1995.
- [7] T. Horprasert, D. Harwood, and L. Davis. A statistical approach for real-time robust background subtraction and shadow detection. *In Proc. of International Conference on Computer Vision*, 1999.
- [8] Stephen Intille, James W. Davis, and Aaron F. Bobick. Real-time closed world tracking. *In Proc. of Computer Vision and Patter Recognition*, 1997.
- [9] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall Advanced Reference Series, 1988.
- [10] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 2001.
- [11] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimiztion, Algorithms and Complexity, chap. Weighted Matching*, p. 248. Prentice-Hall, 1982.
- [12] R. Pauwels and G. Frederix. Non-parametric clustering for image segmentation and grouping. *Computer Vision and Image Understanding*, 1999.
- [13] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. *In Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1994.
- [14] K. Rangarajan and M. Shah. Establishing motion correspondenc. *Comp. Vis., Graph., and Img. Proc.*, 1991.
- [15] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000.
- [16] G. Shakhnarovich, L. Lee, and T. Darrell. Integrated face and gait recognition from multiple views. *In Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2001.
- [17] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997.