Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis

Fatih Porikli Oncel Tuzel
Mitsubishi Electric Research Labs,
Murray Hill, NJ 07974, USA

Abstract

We present an automatic, real-time human tracking and observation system. Robustness and speed are the two major bottlenecks of the existing approaches. We improve upon the robustness and speed of the current state-of-art by integrating a mean-shift based model update technique with an adaptive change detection method. We also provide optimal solutions for several other stages including illumination compensation, skin color detection, shadow removal, morphological filtering, event analysis of a tracking system. In addition, we introduce a novel background refresh mechanism. Thus, the proposed framework is capable of handling shortcomings of template and correspondence based tracking approaches. The results with the ICVS-PETS data sets show the effectiveness of the algorithm.

1. Introduction

Accurate object segmentation and tracking under the constraint of low computational complexity presents a challenge. A typical detection system is built by finding regions in motion, eliminating shadows and noise, constructing and tracking objects in video.

Background Subtraction The most common approach for discriminating a moving object from the rest for stationary camera setup is background subtraction. Basically, background detection approaches can be classified as non-adaptive and adaptive methods. Manual selection, pixel-wise voting, and mean value search algorithms are among the non-adaptive methods. Adaptive methods include averaging images over time, alpha-blending [5], Kalman filtering, Gaussian mixture models (GMM), etc. Although averaging and alpha blending are simple and fast, they are not effective for scenes with many moving objects particularly if they move slowly. Besides, they cannot handle multi-modal backgrounds. They recover slowly when an object occupies the scene at the initialization phase. Pixel-wise voting among the accumulated images may handle some of the recovery problems, however it becomes computationally very expensive with the increasing number of images. Kalman filtering approaches such as in [7] may only provide some partial solution. An adaptive multi-class statistical model for the background and foreground is proposed in [10]. However, this systems is sensitive to initialization inaccuracies. A widely accepted solution for multi-model backgrounds that uses a mixture of Gaussian models is presented in [4]. A similar approach with minor differences on update and initialization mechanisms that are formulated as an EM framework in [3]. The Gaussian model based approaches have capability of dealing with illumination changes. Also, repetitive variations are learned. For the model numbers higher than three, this method becomes too slow to be practical. A major shortcoming of all the above background methods is that they lack an adaptive mechanism to control the update frequency.

Shadow Removal Shadows cause serious problems such as merging of objects, distortion of color histogram of objects, shape deformations, false identifications. There are many proposed methods to find shadows. Some approaches prefer the HSV color space analysis since a shadow cast on a background does not change significantly its hue [2]. Other works exploit saturation information considering shadows often lower the saturation of the pixels. In [6], a pixel is classified into one of the four categories, foreground, background, shadow, highlight, depending on the distortion of the brightness and the distortion of the chrominance of the difference. The main limitation of these methods is that they do not adapt to different types of shadow, e.g. ambient, spot. In [8] the shadow detection is provided by verifying three criteria: the presence of a darker uniform region; the presence of a high difference in luminance with respect to reference; and the presence of static and moving edges. However, these assumptions are difficult to justify in general. They require several predefined parameters.

Object Tracking Generally speaking, tracking of objects can be done either by back-tracking or by forward-tracking. The back-tracking based approach segments foreground regions in the current image and then establishes the correspondence of regions between the previous image. The forward-tracking approach estimates the positions of the regions in the current frame using the segmentation result obtained for the previous image. For establishing cor-

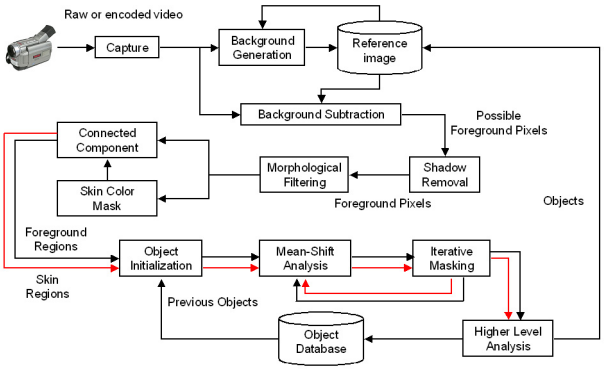


Figure 1: The flow diagram of the object tracking.

respondence, several object templates are utilized [5], [10], [9]. The limitation of this approach is a single template is not sufficient for wide variety of applications, e.g. human tracking and traffic monitoring require different templates. A possible forward-tracking technique is mean-shift analysis. Mean-shift is a nonparametric density gradient estimator. In [1], it is employed to derive the object candidate that is the most similar to a given model while predicting the next object location. This method provides accurate localization, and it is computationally fast. However, the mean-shift tracker is not automatic since it requires initial model properties, i.e. object boundary, etc.

In this paper, we have integrated a model-based background subtraction into a mean-shift based forward tracking mechanism. We combined these methods to accomplish an automatic and robust tracker that can handle high resolution color video in real-time. We also address other main difficulties, i.e. managing sudden illumination changes in the scene, overcasts, shadows, and correspondence problems. We find human face and arms by applying a skin color mask which is formulated in the RGB space by offline training. As shown in Fig.1, our method generates a reference image using pixel-wise mixture models, finds changed part of image by background subtraction, removes shadows by analyzing color and spatial properties of pixels, determines objects, and tracks them in the consecutive frames. In addition, we developed geometry and motion based rules to detect a set of events such as hand raise, hand touching head, number of heads and arms, etc., for human objects.

The paper is organized as follows. Section 2 discusses the generation of adaptive reference image, detection of skin color pixels, and computation of change detection scores. Section 3 explains the proposed shadow removal algorithm. Section 4 describes the construction of objects by connected components. Section 5 explains the fusion of change detection scores and mean shift based tracking and gives tracking examples.

2. Background Generation & Update

In background subtraction, the current image is compared to a reference image to detect the changed pixels. Thus, the first problem is how to determine the reference image. During tracking, the reference image should be compensated according to the lighting condition of the scene. Thus, the second problem is how to make the reference image adaptable to illumination changes. The third problem is how frequent the adaptation should be.

The reference image is constructed by utilizing pixel-wise mixture of models to support multi-model backgrounds. We model the history of a color channel each pixel by a mixture of Gaussian distributions. Let the color value of a pixel be denoted by $I(p)$. We define the probability of observing the current pixel color value for a single channel at frame t as

$$P(I(p),t) = \sum_{n}^{N} w_n(t) g(I(p), \mu_n(t), \sigma_n^2(t)) \qquad (1)$$

where N is the number of distributions, $w_n(t)$ is the weight of the n^{th} Gaussian in the mixture, $\mu_n(t)$ and $\sigma_n^2(t)$ is the mean value and the variance of the n^{th} Gaussian in the mixture at frame t respectively, and g is a Gaussian probability density function as

$$g(I(p), \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[I(p)-\mu]^2}{\sigma^2}} \qquad (2)$$

In the above formulation, the color channels are assumed to be independent from each other due to the computational reasons. Another option is to use a covariance matrix, however this formulation had minimal improvement on the results we observed although the model update mechanism became computationally very demanding. A flow diagram of the background generation is shown in Fig. 2.

Initially, the variances of all models are set to a high value, and their weights to a small value, since none of the background models has sufficient statistics to be a confident prediction. The reference image B is updated by comparing the current pixel with the existing K Gaussian distributions. In case the color value of the current pixel is similar to the mean value of a distribution, it is marked as a match. The distance threshold is set to 2.5σ to include the 95% of the color values which form the model. If none of the K distributions $(K < N)$ matches the current pixel value, a new distribution is initialized. In case of $K = N$, the distribution with the highest variance (lowest confidence) is replaced with a distribution with the current pixels value as its mean value, and a large initial variance. The means and variances of the matched distributions are updated using a learning coefficient as

$$\mu_n(t) = [1-\alpha]\mu_n(t-1) + \alpha I(p) \qquad (3)$$
$$\sigma_n^2(t) = [1-\alpha]\sigma_n^2(t-1) + \alpha[\mu_n(t) - I(p)]^2 \qquad (4)$$

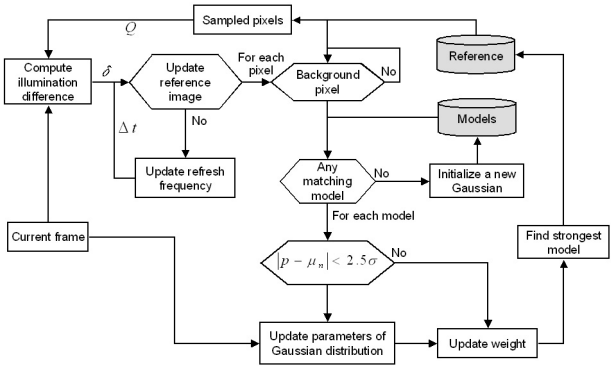


Figure 2: The background update mechanism.

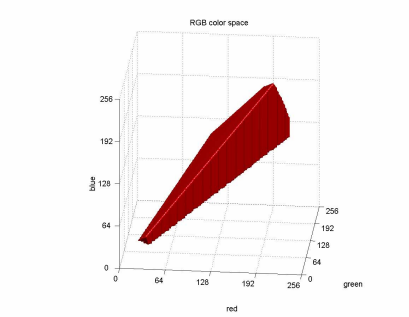


Figure 3: Simplified skin colors in the RGB space.

and the weights of the existing K distributions $w_n(t)$ $n = 1,..,K$ are adjusted as

$$w_n(t) = \begin{cases} (1-\alpha)w_n(t-1)+\alpha & |\mu_n(t)-I(p)| < 2.5\sigma_n(t) \\ (1-\alpha)w_n(t-1) & |\mu_n(t)-I(p)| \geq 2.5\sigma_n(t) \end{cases}$$

where $I(p)$ is the current color value of a pixel. To determine the value of a reference image pixel at a channel, the Gaussian model with the smallest variance (highest confidence) and the highest weight is selected.

The learning coefficient α serves as a parameter that controls the rate of the adaptation of the reference image to the current frame. With the higher the illumination change between the frames, the larger values of the learning coefficient is selected, thus the background image is influenced more by the current image. However, such high values may cause a stopped object to erode in the background as well.

As the answer of the third question, the reference image is updated if only a lighting change occurs in the scene. In fact, there is no need to update the Gaussian models that already have very low variance (high confidence). Instead of blindly updating the background at certain frequencies as in [4], we detect illumination changes and control the updating mechanism accordingly.

For this propose, we compute an illumination change score $\lambda(t)$ for a set of randomly selected pixels that do not correspond to an object in the previous frame

$$\lambda(t) = \left| 1 - \sum_{q \in Q} \frac{|B(q)||I(q)|\cos\theta}{|B(q)|} \right| \qquad (5)$$

where θ is the angle between the pixels color vector $I(q)$ and background color vector, and Q is the pixel set described above. In case the value of the illumination change score $\lambda(t)$ is larger than a threshold the learning parameter is adjusted as

$$\alpha = 0.01 + \frac{\lambda(t)}{c} \qquad \tau_1 < \lambda(t) \qquad (6)$$

where c is the number of pixels in the set Q. The value of τ_1 is determined empirically, and it controls the agility of the update mechanism depending on the application. The thresholds are set empirically. We adapt the period Δt_m of the update mechanism as shown in Fig. 4

$$\Delta t_m = \begin{cases} \Delta t_{max} & \lambda(t) \leq \tau_0 \\ \min(\Delta t_{m-1}+1, \Delta t_{max}) & \tau_0 < \lambda(t) \leq \tau_1 \\ 1 & \tau_1 < \lambda(t) \end{cases} \qquad (7)$$

where m is the total number of updates after initialization, $\tau_0 < \tau_1$, and Δt_{max} is a the maximum number of frames that system waits before an update. We observed that this parameter depends on the amount of the image noise.

2.1 Skin Color Extraction

In addition to the foreground-background separation, the foreground pixels are further classified as skin color pixels or non-skin color pixels. We used a skin color mask $S(c)$ that gives the likelihood of a color vector corresponding to human skin. To obtain this mask, several manually segmented human face and body images from various races are mapped into the RGB space, and the number of pixels for each color vector is counted. Gaussian density models $G(c^*, c)$ are assigned according to the number of pixels

$$G(c^*, n^*, c) = n^* \exp\left(-\frac{|c-c^*|}{\gamma}\right) \qquad (8)$$

where n^* is the number of skin color pixels corresponding to the color vector c^*, and γ is a density constant. This parameter can be a variable of color vectors, i.e. $\gamma(c, c^*)$, by establishing a perceptual similarity of the colors. However, the RGB color space is not suitable for obtaining a perceptual similarity matrix. The skin color mask is obtained by

$$S(c) = \sum_{c_i \in V} G(c_i) \qquad (9)$$

where V is a 3D volume around the color vector. We used a $(3\times3\times3)$ cube as the volume V in the simulations.

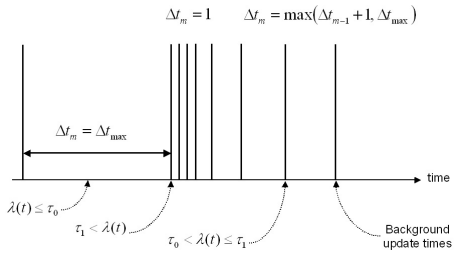
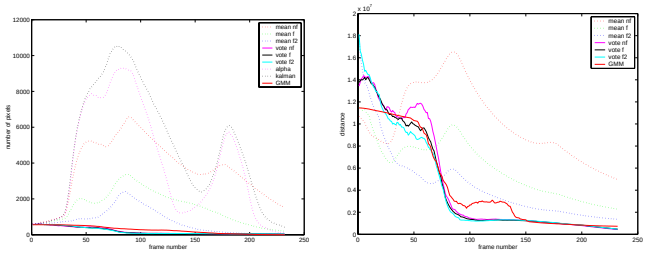


Figure 4: The update coefficients.



(a) (b)

Figure 5: (a) Number of outlier pixels T_{err}. (c) Averaged difference Δ_{err}.

We have also fitted a parametric shape to the simplified mask such that $S(c) = 1$ if the following conditions are satisfied; $|\arctan(\frac{b}{r}) - \frac{\pi}{4}| < \frac{\pi}{8}$, $|\arctan(\frac{g}{r}) - \frac{\pi}{6}| < \frac{\pi}{18}$, $|\arctan(\frac{b}{g}) - \frac{\pi}{5}| < \frac{\pi}{15}$, and $0.1 < r + g + b < 0.9$ as shown in Fig. 3.

2.2 Change Detection

Since the background is adapted only when an illumination change is detected, major reduction in overall computation is achieved while still taking advantages of Gaussian mixture model approach. The color difference $d(p)$ between the current image and the reference image is defined as

$$d(p) = |B(p) - I(p)| w_n^*(t,p) \tag{10}$$

where $w_n^*(t,p)$ stand for the weight of the best model corresponding to pixel p. If this distance is bigger than a color threshold, it is marked as a possible foreground pixel. There should be a lower limit for the model variance since it directly effects the weight. Otherwise, in case the best background model has very low variance, a competing model may be initialized which is very close to it due to the fact that 2.5σ becomes small in return.

The color difference is multiplied by the skin color mask $d(p) \cdot S(c_p)$ to determine the human skin color pixels.

2.3 Comparison of Background Algorithms

We tested the proposed method and several other background generation methods including the mean computation, voting, alpha-blending, Kalman filtering, and the conventional GMM. We evaluated the quality of the generated backgrounds using a ground-truth background, which is generated by voting among the selected 300 frames. We observed that this background is accurate enough to be assigned as the ground-truth for the specific input sequence.

We computed two error scores to measure frame differences. The first score is the color difference between the ground-truth and the current background

$$\Delta_{err} = \left(\sum_p [\mathcal{B}(p)_t - \mathcal{G}(p)]^2 \right)^{0.5} \tag{11}$$

where $\mathcal{G}$ is the ground-truth. The second score T_{err} counts the number of pixels at the current background that do not match with the ground-truth. For the mean computation and voting methods, we removed the moving regions to minimize blurring of the background. We estimated moving regions simply by taking frame-wise difference between the current frame t and the previous frame $(t-1)$. We also tested using longer time laps such as $(t-5)$. Fig.5 shows the computed error scores T_{err} and Δ_{err} for a test sequence in which the scene was initially occluded with two objects. Our tests with other sequences gave similar results.

We observed that the GMM method has the best stability. It was able to handle the moving regions without disturbing the background, and its accuracy improves with each update. The adaptation performance of the Kalman filtering depends on the preset learning parameters. Besides, it cannot handle both slow and fast moving regions and background changes at the same time. The mean computation method causes blurring for multi-modal backgrounds, and ghost effects for moving regions. The alpha-blending is very sensitive to the noise.

In terms of computational complexity, the alpha-blending method is the fastest method; it takes less than 1 msec/frame to estimate a 320×240 color background on a 1.8Ghz machine. The GMM method with 3 models reaches 17 msec/frame, however its performance rapidly decreases for the higher number of models and becomes very slow if more than 5 models are used. The Kalman filtering performs mediocre around 25 msecs/frame since it suffers from the pixel-wise update mechanism. The mean computation method also requires intensive operations, it takes 30 msecs/frame for a 100 frames window. The voting method is impracticable for large number of frames, e.g. it spends several seconds for a 100 frames window.

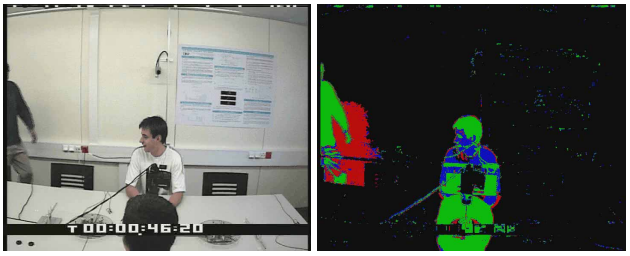


Figure 6: Original image and change detection result. Red pixels are detected as shadow, and green pixels as foreground. Blue pixels will be reevaluated in the iterative loop.

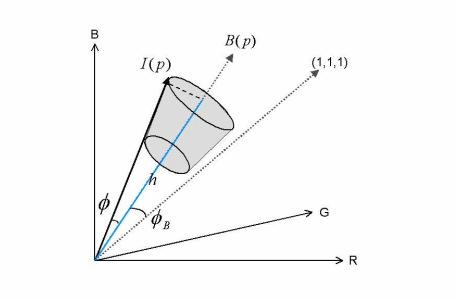


Figure 7: Shadow is defined as a conic volume.

3. Shadow Removal

After background subtraction, we find the possible foreground pixels that their color values are different from the reference image. However, not all the pixels change their color values because they correspond to objects. Shadow pixels are also detected as different from the background, thus the background subtraction gives us pixels correspond to objects as well as shadows. Likelihood of being a shadow pixel is evaluated iteratively by observing the color space attributes and local spatial properties as shown in Fig. 6.

At the first stage, we determine whether a pixel is a possible shadow pixel or not by evaluating the different components of color variation. We carried out our evaluations in the RGB color space. We assume that shadow decreases the luminance and changes the saturation, yet it does not affect the hue. The projection of the color vector to the background color vector gives us the luminance change h

$$h = |I(p)| \cos \phi \tag{12}$$

where ϕ is the angle between $B(p)$ and $I(p)$. Note that, the color values are normalized to $[0, 1]$ range. We define a luminance ratio as $r = |B(p)|/h$. We compute a second angle ϕ_B between the $B(p)$ and the white color $(1, 1, 1)$. For each possible foreground pixel obtained, we apply the following test and classify the pixel as a shadow pixel if it satisfies both of the conditions

$$\phi < \min(\phi_B, \phi_0) \ , \qquad r_1 < r < r_2 \tag{13}$$

where ϕ_0 is the maximum angle separation, $r_1 < r_2$ determines maximum allowed darkness and brightness respectively. Thus, we define the shadow as a conic around the background color vector in the color space (Fig.7). Those pixels that satisfy the above conditions are marked as possible shadow pixels, the rest remains as possible foreground.

At the second stage, we refine the shadow pixels by evaluating their local neighborhood. If the illumination ratio of two shadow pixels are not similar than they assigned as unclassified. Then, inside a window the number of foreground F, shadow S, and unclassified pixels U are counted for the center pixel, and following rules are applied iteratively: $(F > U) \wedge (F > S) \rightarrow F$, $(S > U) \wedge (S > F) \rightarrow S$, and else U. The shadow removal mechanism is proved to be effective and adjustable to the different lighting conditions.

4. Pixels to Objects

After shadow removal, we have the binary image of foreground pixels that corresponds to the objects. The next task is to find the separate objects. To accomplish this, we first remove speckle noise by morphology, then determine connected regions, and group regions into separate objects for both foreground pixels and skin color pixels separately.

To speed up the filtering process, we developed a fast erosion-dilation filter. Since we have a binary foreground-background map, we transfer each 32 horizontal pixel values into a 4-byte integer number. By shifting right and left, and applying logical inclusion operation to the upper and lower rows, we apply morphological dilation. In the second pass, logical exclusion is applied instead of inclusion similarly to erode the image. Thus, we achieve 32 times faster filtering by taking the advantage of the architecture.

We apply connected component analysis to determine the connected regions of the foreground pixels after filtering. We implemented it as a graph-based single-pass algorithm instead of a conventional two-pass approach. During the connected component analysis, we detect the connected regions, compute the total number of pixels of a connected region, the center of mass, the outer box coordinates, and also an inner box coordinates. The inner box includes 90% of the pixels by starting from the pixels close to the center of mass. Using an inner box, it is possible to differentiate the convex regions from the elongated ones. A convex region has a small ratio of the outer box to inner box coordinate ratio, whereas an elongated region has larger values.

We employ a rule-based decision process that initializes an object by observing the properties of the connected components. We use the above properties of regions to merge

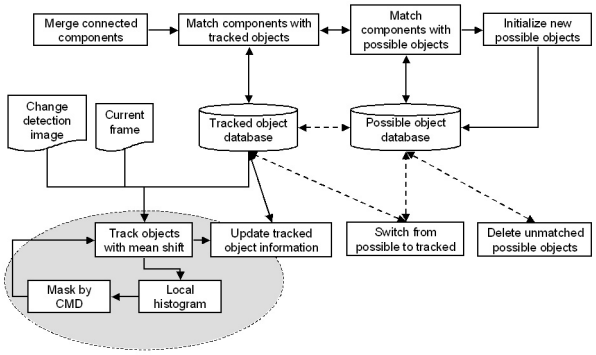


Figure 8: The fusion stages are marked in the ellipse.

them together. If the inner boxes of two regions are overlapping they are assigned to the same object. Otherwise, if their outer boxes are overlaying and the overlapping area is comparable to the area of regions they are assigned to the same object. For each group of merged regions, an object such that its status is set to "possible" is initialized, and a single outer box is fitted for it. This box and the regions construct the initial object. The possible objects are upgraded to regular objects after they consistently appear. We measured the consistency score as the number of consecutive frames where the possible object is detected. The current implementation assumes 8 frames as the consistency threshold. This score decreases if the object cannot be matched in the next frame. If the score becomes less than the threshold the status of the object is degraded, and if becomes zero, the object is deleted from the current object list.

Note that, we do not label each connected component and its pixels by the object number it was assigned. The connected component structure is preserved. On the other hand, an object keeps the record of its constituent connected components. Thus, the object regions are not necessarily be connected to each other. This improves the tracking accuracy under shadow or other difficult conditions that objects can be divided into multiple small parts. Objects become very close or disappear behind each other causes one type of occlusion. In such cases, the update process of the object color histograms (as explained in the following section) and their consistency scores are frozen, an overall box is created for the occluded region, and possible partitions of the connected components within this box is observed in the following frames. As soon as one of the partitions is qualified as a possible object as defined above, it is compared with the frozen histograms also its speed, direction with the occluded objects properties. Even multiple object occlusion situations, the system was able to track the objects after they separated. The second type of occlusion, i.e. an object suddenly disappears outside of an entry/exit region, is handled by using the consistency score. The Fig. 8 shows a detailed flow diagram of the decision process. Once a regular object is initialized, it is tracked by using color histograms.

5 Fusion of GMM & Mean-Shift

The mean-shift tracker can find the correct localization of a region in the following frame using the region's characteristic distributions, e.g. color histogram and change detection mask. However, it cannot initialize the region by itself.

We provide the initial regions and their properties by background subtraction. During this process, we specify the color distribution histogram of a region. Then, using color histograms, we achieve to track objects by computing the highest gradient direction. Let $\mathbf{h}_1 = \{h_1(n)\}$ be a given discrete histogram density, and $\mathbf{h}_2(y) = \{h_2(y,n)\}$ is the target discrete histogram density for the location y such that $\sum_n^N h_1(n) = 1$ and $\sum_n^N h_2(y,n) = 1$. The initial color histogram $\mathbf{h}_1$ is computed from the color vectors $I(p_i)$ where $i = 1,..R$ of the current object as

$$h_1(n) = \frac{1}{C_1} \sum_i^R k(p_i)\delta[I(p_i) - n] \tag{14}$$

where normalization constant C_1 is the sum of values of the kernel function $k()$, i.e. $C_1 = \sum_i^R k(p_i)$, and $\delta(.)$ is impulse function. Similarly, the target histogram becomes

$$h_2(y,n) = \frac{1}{C_2} \sum_i^R k(y,p_i)\delta[I(p_i) - n]. \tag{15}$$

The target histogram location estimation problem is formulated as the derivation of estimate that maximizes the Bayes error associated with the given and target distributions. For this purpose, the discrete sample estimate of the Bhattacharya coefficient is utilized since it is nearly optimal and imposes a metric structure. This estimate and the distance between two histograms are defined

$$\rho[\mathbf{h}_1, \mathbf{h}_2(y)] = \sum_n^N \sqrt{\mathbf{h}_1, \mathbf{h}_2(y)} \tag{16}$$

$$d(y) = \sqrt{1 - \rho[\mathbf{h}_1, \mathbf{h}_2(y)]}. \tag{17}$$

Our aim is to minimize the above distance as a function of y in the neighborhood of a given location y_0 by exploiting the mean shift iterations. Using Taylor expansion around the $\mathbf{h}_2(y_0)$, the Bhattacharya coefficient $\rho[\mathbf{h}_1, \mathbf{h}_2(y)]$ is approximated as

$$\approx \frac{1}{2} \sum_n^N \sqrt{\mathbf{h}_1, \mathbf{h}_2(y_0)} + \frac{1}{2C_2} \sum_i^R \beta_i k(y,p_i) \tag{18}$$

where

$$\beta_i = \sum_n^N \delta[I(p_i) - n] \sqrt{\frac{\mathbf{h}_1}{\mathbf{h}_2(y_0)}} \tag{19}$$

Since only the second term in the above equation is dependent y, it has to be maximized to minimize the distance. This term corresponds to the density estimate using kernel profile k at y location with weights β_i.

Instead of using a predetermined kernel profile as in [1], we adapt the kernel such that the pixels that are detected as foreground will have higher weights. Using foreground information prevents background pixels from causing the estimation process to deviate. We do not allow a background pixel, which has similar color value to the initial object, to change the maximum gradient direction. The kernel $k(y, p_i)$ is defined as

$$k(y, p_i) = \frac{d(p_i)}{C_d} \tag{20}$$

where the normalization constant C_d represent the maximum possible color distance. The maximization process is iterated by given the current object histogram $\mathbf{h}_1(n)$ extracted from the previous frame:

1. Compute the histogram $\mathbf{h}_2(y_0)$ in the current frame, calculate $\rho[\mathbf{h}_1, \mathbf{h}2(y_0)] = \sum_n^N \sqrt{\mathbf{h}_1, h_2(y_0)}$,
2. Compute the weights β_i, $i = 1, ..., R$,
3. Derive the new location y_1 by mean-shift,

$$y_1 = \frac{\sum_i^R p_i \beta_i k(p_i)}{\sum_i^R \beta_i k(p_i)} \tag{21}$$

   Update the target histogram $\mathbf{h}_2(y_1)$, and calculate $\rho[\mathbf{h}_1, \mathbf{h}2(y_1)]$ as above,
4. Stop if $|\rho[\mathbf{h}_1, \mathbf{h}2(y_1)| < \epsilon$, else $y_1 \rightarrow y_0$, go to step 1.

Each object is tracked as explained above. Then the new kernel scale is computed by alpha blending the previous kernel scale and the outer box of the object in the current frame. The background subtraction may be computed for only a respectively small portion of the image which corresponds to the neighborhood of the objects and entry/exit regions, thus computational load is reduced further.

After obtaining objects for both skin color regions and for all foreground regions, the human heads are determined by heuristics based on the relative position of the skin color objects with respect to the foreground objects, and sizes and aspect ratios of the skin color regions. First, separate human objects are detected using foreground and skin color objects. Then, the head regions are assigned. Thus, a human object includes a head region, several other skin regions for arms, hand, etc. and body regions from foreground objects. The events, such as hand raise and touching to head, are formulated as the specific motion and position of the skin color objects grouped under a human object. For example, hand raise is defined as skin region, which does not correspond to a head, moves upward, reaches same level as the its corresponding head region, and moves downward. Depending to the detection task, several other movements and gestures can also be detected since the object detection provides accurate objects. We are currently developing the movement detection part.

6. Results and Discussion

We implemented the tracking system on a PC platform with a 1.8Ghz computer. The processing time is 27 milliseconds on average, ranging from $18 \sim 43$ milliseconds for 320×240 color video, and around 100 milliseconds for 720×576 color video. Further improvement is possible by optimizing the code. The number of objects the algorithm can handle at the same time is not limited.

We tested our method for the ICVS-PETS dataset scenario-A, camera-1 setup. Fig. 10 shows the detection results. We produced a 30 frames/sec AVI video from the camera-1 JPEG frames. We did not fine tune any GMM, skin color mask, change detection, and shadow removal parameter. We did not define any specific entry/exit regions in the image either. Since we use the size of the objects, we set the minimum and maximum sizes for the head objects. This was the only adaptation.

Our test with the camera-1 sequence showed that the proposed method accurately detects and tracks heads. In the given result images (Fig. 10), we assigned a unique color code for each tracked skin color object. For the test sequence, each head kept its unique color code. The same color head regions in the result images corresponds to the same human object. Each separate skin color object has shown in its own box. We proved that the segmented regions are accurately match with the ground-truth.

We also present movement detection result in Fig. 9. We show that our detection results are very close to the ground-truth. We accurately counted the number and frames of the leftmost speakers (Daniel) hand raises, and the number of visible heads. We also manage to identify the frames where the speaker in the middle (Fabian) touches his head.

Fusion: The major novelty of our proposal is that we make the semi-automatic mean-shift tracker completely automatic using an improved GMM background subtraction method. This is not a straightforward integration because we did not treat each component as a separate stage i.e. doing background subtraction first and then tracking each object by mean-shift. Our tests showed such an approach prone to the initialization errors and it can lose objects easily due to the intrinsic properties of mean-shift if the

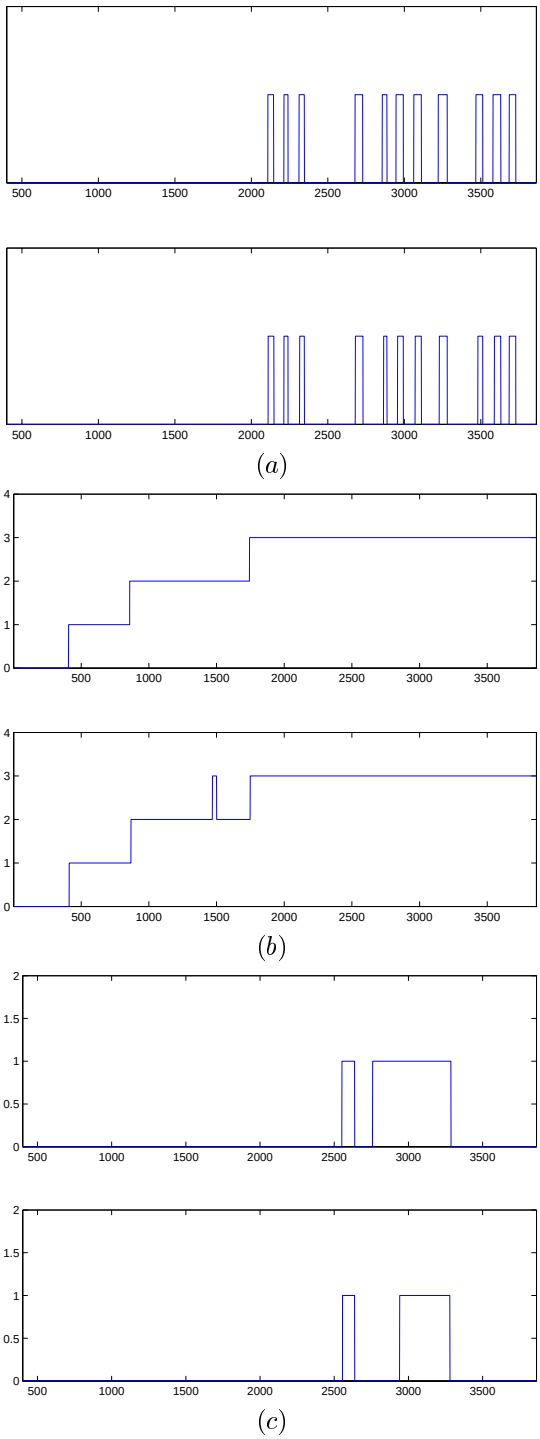


Figure 9: (a) Frames where the leftmost speaker (Daniel) raises his arm. (b) Number of heads facing the camera. We also detect another speaker (Pierre) shortly around frame 1500. (c) Frames where the middle speaker (Fabian) touches his head. (Upper rows: ground-truth, lower rows: results of our method) *Dataset is courtesy of ICVS-PETS.*

background color is close to the objects color distribution. We modified the model update and scaling mechanisms of the original mean-shift tracker such that the mean-shift tracker uses both color histogram and background map at the same time. Furthermore, instead of depending on the preset model scaling parameters of the mean-shift, we use the GMM results to update the object shape and properties between the frames.

Background Generation: We used the Stauffer's GMM background approach. Their method updates the Gaussian models at a preset frame period that and does not have a mechanism to adapt the learning parameter. We improved the adaptation performance of the original GMM by observing the amount of illumination change in the background and updating a second learning coefficient accordingly. This improvement significantly reduces the computational load by minimizing unnecessary model updates.

Shadow Removal: Our shadow detection is inspired by [6]. However, we improved their method by adapting not only to the luminance difference but also to the saturation difference. We defined the shadow color range as a conic-cylinder around the background color vector. We also tested the shadow color range against our dual-layer neural net training algorithm. Our experiments show we have higher detection accuracy than the methods compared in [11].

Sensitivity: To adapt the certain shadow detection and background subtraction parameters of our proposed, we run several sequences and evaluated their results. Once these parameters are set, we did not change their values later. We made the magnitude of the shadow density (dark, light, etc) as an adjustable parameter, however for the test sequences we used (ICVS-PETS, PETS2002, ETRI) we did not need to tune this parameter either. There was no fine-tuning of any other system parameter for the results. Thus, we believe that our parameter adaptation methods are working accurately, and the performance of our tracker is robust.

The performance of the background adaptation and mean-shift analysis based object tracking method is comparable with the state-of-art, and it is fully automatic. It does not have the intrinsic shortcomings of the template-matching approaches such as resolution, pose, and illumination dependencies. As a result of accurate and robust detection, the objects can be further analyzed to determine the gestures.

References

[1] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift, IEEE Conf. Computer Vision and Pattern Recognition, South Carolina, Vol. 2, 2000.

[2] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting objects, shadows and ghosts in video streams by exploiting color and motion information", Proc. of IEEE CIAP, 2001.

Figure 10: Tracking results for camera-1. Note that the heads are tracked accurately, i.e. each head has a unique color code.

[3] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," In Proc. of the Thirteenth Conf. on Uncertainty in Artificial Intelligence, 1997.

[4] C. Stauffer and W.Grimson, "Adaptive background mixture models for real-time tracking", Proc. IEEE CVPR, 1999

[5] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: Who? when? where? what? a real-time system for detecting and tracking people". Proc. of IEEE ICAFGR, 1998.

[6] T. Horprasert, D. Harwood, and L. Davis, "A statistical approach for real-time robust background subt. and shadow detection", Proc. of IEEE ICCV Frame-rate Workshop, 1999.

[7] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using Kalman-filtering". Proc. of ICAM, 1995.

[8] J. Stauder, R. Mech, and J. Ostermann, "Detection of moving cast shadows for object segmentation", IEEE Transactions on Multimedia, Vol 1, 1999.

[9] M. Yamada, K. Ebihara, and J. Ohya, "A new robust real-time method for extracting human silhouettes from color images", Proc. of IEEE ICAFGR, 1998.

[10] C.R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body". IEEE Trans. on PAMI, Vol 19, 1998.

[11] A. Prati, I. Mikic, M. Trivedi, "Shadow detection algorithms for traffic flow analysis", ICITS, 2001