Hand Image Segmentation in Video Sequence by GMM: a comparative analysis

Hebert Luchetti Ribeiro
School of Engineering at São Carlos
University of São Paulo -USP
São Carlos, São Paulo, Brazil
hebertlr@ig.com.br

Adilson Gonzaga
School of Engineering at São Carlos
University of São Paulo - USP
São Carlos, São Paulo, Brazil
adilson@sel.eesc.usp.br

Abstract

This paper describes different approaches of real-time GMM (Gaussian Mixture Method) background subtraction algorithm using video sequences for hand image segmentation. In each captured image, the segmentation takes place where pixels belonging to the hands are separated from the background based on background extraction and skin-color segmentation. A time-adaptive mixture of Gaussians is used to model the distribution of each pixel color value. For an input image, every new pixel value is checked, deciding if it matches with one of the existing Gaussians based on the distance from the mean in terms of the standard deviation. The best matching distribution parameters are updated and its weight is increased. It is assumed that the values of the background pixels have low variance and large weight. These matched pixels, considered as foreground, are compared based on skin color thresholds. The hands position and other attributes are tracked by frame. That enables us to distinguish the hand movement from the background and other objects in movement, as well as to extract the information from the movement for dynamic hand gesture recognition.

1. Introduction

This paper's main goal is to evaluate the methodologies that are able to segment hand videos aiming HCI (Human Computer Interaction) applications for real-time applications. The hand gesture recognition application based on computer vision motivated the research considering that hand gestures are a natural way for communication among people, resulting in a more natural way to interact with the computer. The hand segmentation in video images is the crucial part of the gesture recognition, because if it does not segment the object of interest properly, further analysis would be impossible. Traditionally, the background subtraction from the current scene is used for the segmentation. The distress of following this approach is to build a robust background model for complex background scenes, formed by different geometric shapes, textures and colors, illumination variations, reflections and efficiency for external and indoor environments. This paper describes the Gaussian mixture method (*GMM*) based on the work of Stauffer and Grimson [1], suitable for hand segmentation, and compares it with different approaches of the same algorithm implemented by Power and Shoonees [3] and KadewTraKuPong and Bowden [4]. An auxiliary segmentation method that builds a skin classifier is used. It explicitly defines, through a number of rules, the boundaries for skin cluster in a RGB color space [8].

2. Related Work

Stauffer and Grimson [1] used the adaptive mixture of Gaussian distributions to model each pixel from background image and a connected component algorithm to segment the foreground objects. The model presented by them [1] became very popular and used in several applications that need background estimation. However, this model is not totally adaptable to all different environment conditions considering the influence caused by shadows, reflections and foreground objects relatively static or in high traffic. Some modifications have been proposed for the original algorithm [1] to solve these matters. An interesting modification was proposed by Harville et al. [2], where, besides the color attribute, a new attribute was added to the mixture of Gaussian distribution: depth. The new attribute is extracted from a stereo binocular arrangement that allows the incorporation of the model shadows and reflections. Harville et al. [2] have also created a scene activity

measurement to manage the speed of the object that is incorporated to the background model, solving the problem when the object is almost static or in high traffic.

KadewTraKuPong and Bowden [4] presented a method that improves this adaptive background mixture model [1], reinvestigating the update equations. They use different equations at different phases. This allows the system to learn faster and more accurately, as well as to adapt to changing environments effectively. This approach updates the equations derived from *sufficient statistics* [5] and *L*-recent window formula over other approaches of McKenna *et al apud* [4]. The model begins to estimate the Gaussian mixture model using expected *sufficient statistics* update equations, then switches to *L*-recent window version when the first *L* samples are processed. The expected *sufficient statistics* update equations provide a good estimate at the beginning before all *L* samples are collected. The *L*-recent window update equations prioritize recent data, therefore the tracker can adapt to changes in the environment.

Power and Schoones [3] suggest approximations and modifications from the standard algorithm [1] to improve performance. The modification consists on replacing the probability density function value by the weight of distribution in the update equations. Other approaches different from the one mentioned previously, have been developed as an alternative to statistical models, such as Weiner filter [12], Kalman filter [14], Bayesian decision theory and principal components analysis (*PCA*) [13], and Hidden Markov Models (*HMM*) [15]. Most of these solutions present high computational costs when compared to the GMM and/or worst results for robust background estimation.

3. Background Model Estimation

The basic idea is to define a segmented region, delimiting the pixels of interest. To do so, it is necessary to model the color attribute of each pixel of an image sequence (pixel process) through an adaptive mixture of Gaussian distributions. The mixture of Gaussian distribution model is updated for each new captured observation, reducing the influence from the past observations and allowing the model adaptation according to a gradual variation on illumination. However, the Gaussian distributions represent both foreground and background. It would be necessary to define the distribution of the subsets to describe the background model. The subset definition happens at each observation, according to the associated weights of every distribution indicating the frequency that the distribution better represented the pixel. After the pixel process, the foreground pixels are submitted to skin color segmentation. The next task is to find the separated objects. First, the speckle noise is removed by morphology (*opening and closing*) [11], then connected regions are determined and grouped regions are separated into objects.

3.1. Pixel-Processing

A mixture of *K* Gaussian distributions models each pixel in the scene. The history of pixels can be defined as a temporal series from these pixel values that are vectors $\vec{X}_{i,t} = \{R_{i,t}, \quad G_{i,t}, \quad B_{i,t}\}$. For each time *t*, and every pixel $i = \{x_0, y_0\}$, the history of pixels can be represented using the equation 3.1:

$$\{\vec{X}_{i,1}, \ldots, \vec{X}_{i,t-1}\} = \{\vec{I}(x_0, y_0, j) : 1 \le j \le t-1\} \quad (3.1)$$

Where $\vec{I}$ is the frame sequence. Therefore, the pixel values are samples of some random variable $\vec{X}_{i,t}$ which includes the behavior of *K*. $\vec{X}_{i,t}$ may be 1-dimensional (*monochrome intensity*), 2-dimensional (*normalized color space or intensity-plus-range*), 3-dimensional (*color*), or D-dimensional in general (*represented as column vectors*). The probability that a certain pixel has a value of $\vec{X}_{i,t}$ at the time t can be written as shown in equation 3.2:

$$P(\vec{X}_{i,t} \mid \vec{X}_{i,1}, \ldots, \vec{X}_{i,t-1}) = \sum_{k=1}^{K} \omega_{i,t-1,k} * \eta\left(\vec{X}_{i,t}; \vec{\mu}_{i,t-1,k}, \Sigma_{i,t-1,k}\right) \quad (3.2)$$

Where η is a Gaussian probability density function, and $\omega_{i,t-1,k}$ is the weight parameter of the k_{th} Gaussian component that indicates the relative proportions of past observations modeled for every Gaussian distribution. The η_k factor is denoted the k_{th} Gaussian distribution of a mixture represented using the equation 3.3.

$$\eta\left(\vec{X}_{i,t}, \vec{\mu}_{i,t-1,k}, \Sigma_{i,t-1,k}\right) = \frac{1}{(2\pi)^{D/2} \left|\sum_{i,t-1,k}\right|^{1/2}} * \exp\left\{-\frac{1}{2}\left(\vec{X}_{i,t} - \vec{\mu}_{i,t-1,k}\right)^T \sum_{i,t-1,k} \left(\vec{X}_{i,t} - \vec{\mu}_{i,t-1,k}\right)\right\} \quad (3.3)$$

Where *D* is the *D*-dimensional from vector $\vec{X}_{i,t}$. In this case, *D* = 3 because the RGB color space was adopted. $\vec{\mu}_{i,t-1,k}$ and $\Sigma_{i,t-1,k}$ are the mean-vector and

covariance matrix values of the k_{th} Gaussian component. For computational reasons, the covariance matrix is assumed to be $\sum_{i,t-1,k} = \sigma^2_{i,t-1,k} I$. Therefore, it was established the use of the covariance matrix as $\Sigma_k = diag[\sigma_R^2\ \sigma_G^2\ \sigma_B^2]$, where $\sigma_R^2, \sigma_G^2, \sigma_B^2$ are, respectively, the variances of RGB components. This assumption allows red, green, and blue pixel values to be independent and to have the same variances. Traditionally, the *K* value is the same for every pixel, in a range between 3 and 5. When more distributions are used, the model represents more complex scenes better. However, it increases the computational costs. A Gaussian mixture characterizes the distribution of recently observed values of each pixel in the scene. A new pixel value can be represented by one of the major components of the mixture model and used to update the model. For every new observation, the mixture of Gaussian distribution used to model the observation history of each pixel must be updated. Ideally, at each time step *t*, the pixel's mixture of parameters would be re-estimated applying an exact Expectation Maximization algorithm [6] on some recent observation window, including the last one. But this is a very costly procedure, so Stauffer and Grimson [1] uses an on-line *K-means* approximation. The matching is observed through a distance calculation, normally Euclidean, and a deviation threshold parameter ***(β)***, among the current observation and the *K* Gaussian distributions. The parameter ***β*** is typically 2.5, so the boundary of the matching zone in RGB-space for η_k encompasses over 95% of the data points that would be drawn from the true Gaussian probability density. The algorithm integrates the new observed data into the background model so that every pixel value of the actual frame, $\vec{X}_{i,t}$, is checked against the existing *K* Gaussian distributions.

The pixel is considered as a matched pixel if the criterion $|Xt - \mu| \leq 2.5\sigma$ is true for all RGB channels. If a match is found for some distribution, this one is updated. In order to do that, the *K* distributions are ordered based on the value ω/σ_k and the first *B* distributions are used as a background model of the scene where *B* is estimated using the equation 3.4:

$$B = \arg_b \min\left(\sum_{K=1}^{b} \omega_{i,k} > T \right) \tag{3.4}$$

Where the threshold T is the minimum fraction of the background model. In other words, it is the minimum prior probability of the background to be in the scene. If the matched Gaussian component model is one of any *B* distributions, the model must be updated. The prior weights of the *K* distributions at time *t*, $\omega_{i,t,k}$, are adjusted as the equation 3.5.

$$\omega_{i,t,k} = (1-\alpha)\,\omega_{i,t-1,k} + \alpha M_{i,t,k} \tag{3.5}$$

Where ***α*** ($0 < \alpha \leq 1$) is the learning rate and the time constant **1/α** defines the speed at which the distributions parameters change. ***Mi,t,k*** is 1 for the matched model and 0 for the remaining models. After this approximation, the weights from both matched and unmatched models need to be renormalized. $\vec{\mu}_{i,t-1,k}$, $\sigma_{i,t-1,k}$ are parameters for unmatched distributions that keep the same value and the parameters of the distribution that match the new observation are updated using the equations 3.6 to 3.10:

$$\vec{\mu}_{i,t,k} = (1-\rho)\vec{\mu}_{i,t-1,k} + \rho X_{i,t} \tag{3.6}$$

$$\sigma^2_{R,i,t,k} = (1-\rho)\sigma^2_{R,i,t-1,k} + \rho(R_{i,t} - \mu_{R,i,t-1,k})^2 \tag{3.7}$$

$$\sigma^2_{G,i,t,k} = (1-\rho)\sigma^2_{G,i,t-1,k} + \rho(G_{i,t} - \mu_{G,i,t-1,k})^2 \tag{3.8}$$

$$\sigma^2_{B,i,t,k} = (1-\rho)\sigma^2_{B,i,t-1,k} + \rho(B_{i,t} - \mu_{B,i,t-1,k})^2 \tag{3.9}$$

$$\rho = \alpha * \eta\left(\vec{X}_{i,t}; \vec{\mu}_{i,t-1,k}, \sigma_{i,t-1,k}\right) \tag{3.10}$$

The parameter ρ is the second learning rate estimated using the equation 3.10. If none of the *K* distributions matches that pixel value, the least probable component is replaced by a new distribution with a mean equal to the current value of $\vec{X}_{i,t}$, an initially high variance, and a low weight parameter.

3.2 Skin Color Detection

After the pixel process, the results are submitted to skin color segmentation only for the pixels considered as foreground pixels. The defined metric type is determined by the skin color method. This segmentation method builds a skin classifier that defines explicitly, through a number of rules, the boundaries of skin cluster in some color space model [9]. According to Peer et al.[8], a pixel is classified as skin, in RGB color space, if some rules are accepted, as shown in equation 3.11:

$$\begin{aligned} &(R>95) \wedge (G>40) \wedge (B>20) \wedge \\ &(\max\{R,G,B\} - \min\{R,G,B\} > 15) \wedge \\ &(|R-G|>15) \wedge (R>G) \wedge (R>B) \end{aligned} \tag{3.11}$$

The main advantage of this classifier is the simplicity of skin threshold rules that builds a very fast classifier. So the goal of this skin color threshold is to

easily discriminate, in real time, the skin color and the non skin color pixels.

3.3 Noise Cleaning

When the pixel processing and the skin color threshold are complete, there is a binary image containing foreground pixels (1's) and background pixels (0's). It is essential that the resulting binary segmented image is further refined. Since a statistically significant portion of the input samples will lie in the tails of the distributions, the output image will inevitably contain small noise-generated blobs. These blobs should be eliminated through Gaussian filtering and simple area threshold that can readily remove small false blobs. The remaining blobs can then be cleaned using opening and closing morphological functions [11].

4. Methodology and Materials

Frames from video sequence are analyzed and pre-processed. A Gaussian filter with a 5x5 mask must be applied to each frame to smooth the image. A mixture of K Gaussian distributions models each pixel to subtract the background and reach the hand segmentation. The Stauffer and Grimson [1] GMM model is used to subtract the background, in the RGB color space, and it is compared to different approaches from the same algorithm implemented by Power and Shoonees [3] and KadewTraKuPong and Bowden [4].

4.1 GMM Algorithm Steps

a. **Initializing *K* Gaussians per Pixel**:

Each new Gaussian K is created with the mean, variance and weight parameters equal to the current pixel value, the initial high variance and the low initial weight, respectively (table 1).

b. **Checking the Standard Deviation Threshold:**

Check if the pixel value (Xt) is within the 2.5 standard deviation of all existing K Gaussian distributions. Then calculate the standard deviation (σ) and the mean (μ) of each existing Gaussian to check the standard deviation criterion. There are different rules according to each GMM approach:

Stauffer and Grimson [1] use the equation 4.1 and the equation 4.2 is employed in Power and Schoones [3], KadewTraKuPong and Bowden [4] methods:

$$|R-\mu_R| \le 2.5\sigma \wedge |G-\mu_G| \le 2.5\sigma \wedge |B-\mu_B| \le 2.5\sigma \quad (4.1)$$

$$\left(\frac{(R-\mu_R)}{\sigma_R}\right)^2 + \left(\frac{(G-\mu_G)}{\sigma_G}\right)^2 + \left(\frac{(B-\mu_B)}{\sigma_B}\right)^2 \le (2.5)^2 \quad (4.2)$$

c. **No Matching (*Foreground pixel*):**

No distribution was found among the existing K Gaussian distributions. The least probable distribution is replaced with a new distribution using the mean, variance and weight parameters equal to the current pixel value, the initial high variance and low initial weight, respectively. The least probable distribution is determined by the Gaussian distribution with the lowest ω/σ value.

d. **Matching Found (*Background pixel*):**

When a match is found among the existing K Gaussian distributions, the Gaussian parameters must be adjusted. The weights (ω) of all Gaussians are adjusted. The mean (μ) and the standard deviation (σ) are updated only for the matched Gaussian, while the unmatched Gaussians are not changed. The weights, means and deviations are updated using the equations 3.5 to 3.9. Where ρ is calculated using the equation 3.10 based on Stauffer and Grimson [1] and using the equation 4.3 based on Power and Schoones [3] and using the equation 4.4 based on KadewTraKuPong and Bowden [4].

$$\rho = \alpha/\omega_{i,t,k} \quad (4.3)$$

$$\rho = \alpha = \max(1/(N+1), 1/L) \quad (4.4)$$

Where N is the amount of matched distributions considered as a background, and L is the limit for the amount of matched distribution. Before the model reaches L matched distributions, the update equations consider $\alpha = 1/(N+1)$. After L distributions, the update equations consider $\alpha = 1/L$.

e. **Choosing the Background Distribution :**

After updating the parameters using the steps above, sort the Gaussians using ω/σ in descending order. Choose the first B distributions as a background model, that is, the sum of their weights (ω) is greater than T, as indicated in the equation 3.4:

If the matched distribution is one of the first B distributions, the pixel is classified as a background pixel. Whether no distribution is found among the existing K Gaussian distributions or the pixels do not match any of the first B distributions, the pixel is classified as a foreground pixel.

Table 1. Parameter values used in GMM approaches.

Method	***K***	***α***	***σ***	***ω***	***T***	***β***	***L***
[1]	3	.005	25	.05	.79	2.5	_
[3]	3	.005	30	.05	.7	2.5	_
[4]	4	_	12	.05	.7	2.5	60

4.2 Materials

The program and the tests were developed to run with the Microsoft Windows operational system. The program was implemented in Borland C++ Builder 6.0 programming language. Some OpenCV library functions [7] were used to obtain contours, image filters, image conversions, matrix operations, transpositions, image handle and images visualizations. OpenCV (*Open Source Computer Vision Library*) is a free library for both non-commercial and commercial use, with functions in C programming language and C++ classes with the most popular algorithms of computer vision and image processing. A web cam captures the video images for a 320x240 color video. The computer used for development and tests was a PC AMD Athlon 64 Processor, model 3000, with 1 GB of RAM memory and 80GB of hard disk.

5. Results

At first, the tests were implemented with adjusted parameters for each different address of the GMM algorithm. Some images of these tests, without skin color threshold, are shown in figures (1, 2) while skin color threshold images are shown in figure 3. The first column in figure 1 shows the original frames from a video sequence. The second and third columns show, respectively, the GMM segmentation based on [1] and contour extraction. Figure 2 shows the GMM segmentation and contour extraction using [3] approach on the first and second columns, and the [4] approach on the third and fourth columns. Working with the adjusted parameters for each GMM different addresses (table 1), the approach of [1] achieves a higher amount of background pixels, but it detects more background false positives than the other methods. This result produces a lot more holes and deformed contours.

The method [3] obtains better-defined shapes for the foreground pixels during the segmentation and therefore it provides better contours, but it detects more background pixels as foreground pixels in some situations. In [1], foreground objects that remain relatively static are slowly incorporated into the background model or they disappear completely. Meanwhile, in [3], they are quickly incorporated into the background model, because the simplifications on the update equations speed up the adaptive process of the algorithm.

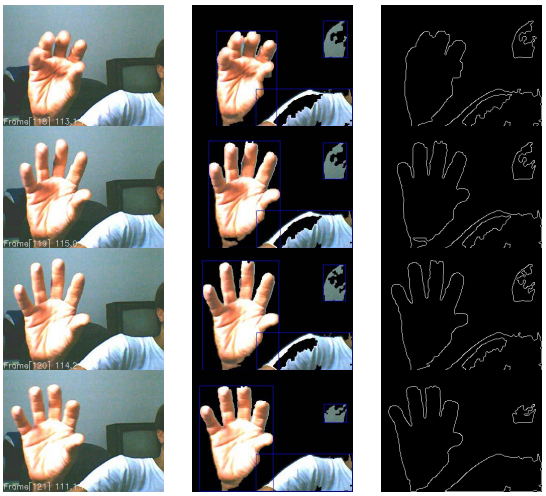

Figure 1. Frame, segmentation using [1] and contour on the first, second and third columns respectively.

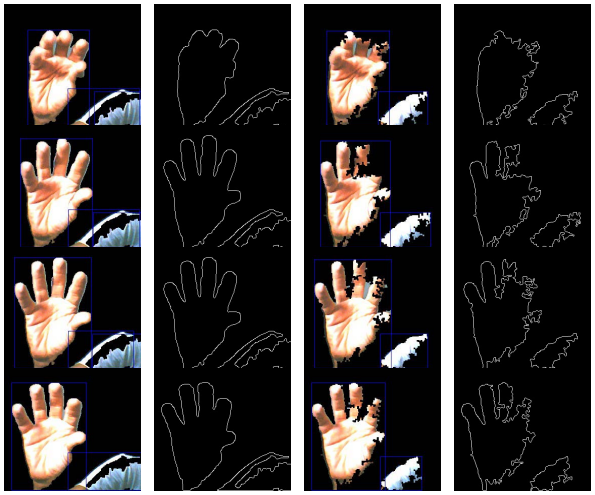

Figure 2. GMM segmentation and contour extraction using [3] on the first and second columns, respectively, and using [4] on the third and fourth columns.

The [4] results are not satisfactory, mainly because of the segmentation continuity when a lot of hand images are detected as background. Thus, this determines that it is difficult to get a good hand contour. Mistakes occur after the first *L* samples are processed when the algorithm switches to *L*-recent window version for updating equations. In this switch, the segmentation basically does not happen. Probably, the *L*-recent value is too small in our tests; according to the authors [4] it should use around 500 frames. We use 60 frames for our application that can not wait for so many frames in the learning phase, since the segmentation works in real time, and this could be

causing the problem. The same test was executed, but now using skin color threshold and figure 3 presents the original frames and the GMM segmentation processed using [1] on the second column, using [3] on the third column and using [4] on the fourth column.

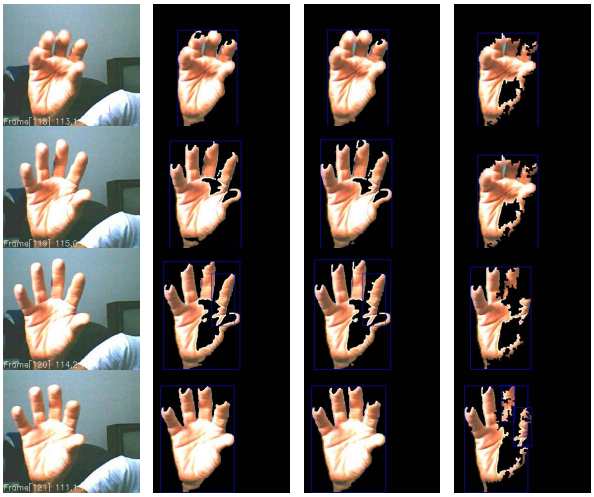

Figure 3. Original images on the first column, GMM segmentation and skin color threshold using: [1] on the second column, [3] on the third column and [4] on fourth column.

The results using skin color threshold (figure 3) are very close to those three proposals. [3] reaches a slightly better performance providing the most filled hand areas and, consequently, better contours. This threshold corrects many of the GMM segmentation mistakes for all three approaches. The skin color checking is applied only for the pixels considered as foreground. It reduces the computational cost, considering that the number of pixels to be checked is smaller than the entire image. The second phase on the test process consists of using the same parameters values presented in table 2 for the three proposals.

Table 2. The initial parameter values.

Parameters	K	α	σ	ω	T	β	L
Values	3	.005	12	.05	.79	2.5	60

Again, [3] reaches the best performance, visually in terms of segmentation (figure 4). For [1], some large hand regions are segmented and there are also traces on the way where the hand passes. For [4], a lot of background areas are incorporated to the foreground segmentation. The different results between the GMM algorithm addresses using the noise cleaning and skin color threshold and not using them need to be accentuated. The pixel process generates a large number of pixel classification mistakes originated from shadows, reflections and static objects that can be corrected or attenuated by noise cleaning and skin color threshold.

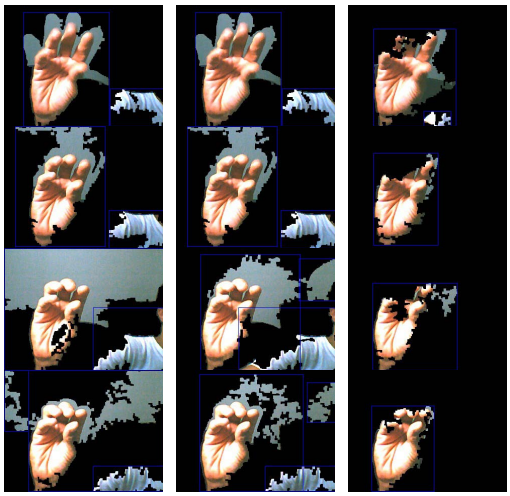

Figure 4. GMM segmentation using the same parameters values for [1] (first column), [3] (second column) and [4] (third column).

For a 320x240 color video with 9944 frames, the GMM methods [1, 3, 4] with K=3 models were tested, comparing the time processing using the same initials parameters shown on table 2. First, the GMM method without post-processing, skin threshold and tracking was used. The results are shown on the last three columns of table 3, for the three propositions [1, 3, 4].

The results presented on the first three columns of table 3 show that the frame rate reaches an average of 10.05 fps by [1], an average of 9.91 fps by [3] and an average of 10.63 fps by [4]. The GMM method test using only post-processing and tracking presents the results shown on the first, second and third columns of table 3. The frame rate reaches an average of 8.83 fps by [1], an average of 8.53 fps by [3] and an average of 9.17 fps by [4]. The last test was used the GMM method with post-processing, skin threshold and tracking. The results presented on the fourth, fifth and sixth columns of table 3 show that the frame rate reaches an average of 8.91 fps by [1], an average of 8.59 fps by [3] and an average of 9.22 fps by [4].

The results achieved by [4] have the best performance in terms of time processing, followed by [1] and [3]. The three tests show that [4] has the quickest average. Figures 5, 6 and 7 show the charts of the amount of frames per second (*fps*) indicating the performance of the three methods: without post-processing, skin threshold and tracking; with post-processing and tracking; and with post-processing, skin threshold and tracking, respectively. The test processed 9994 frames of the same video for the three proposals where the red line represents the results obtained by

[1], the blue line represents the results obtained by [3] and the green line represents the results obtained by [4].

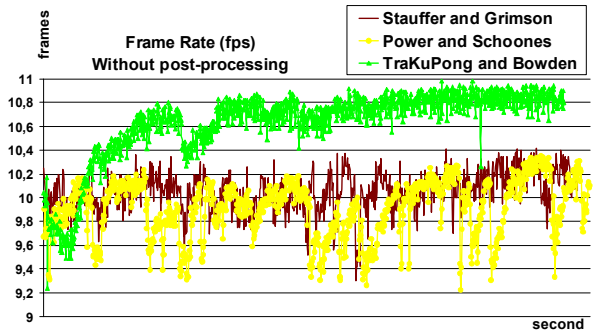

Figure 5. The amount of frames per second indicating the performance of the three methods without post-processing and skin threshold.

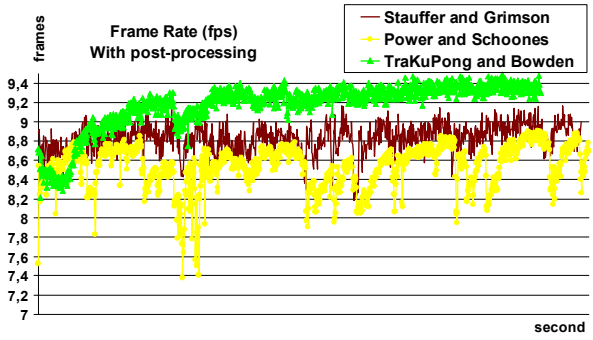

Figure 6. The amount of frames per second indicating the performance of the three methods with post-processing but without skin threshold.

These charts confirm that [4] method has the best performance in terms of time processing. It is slower than the others [1, 3] at the beginning but it becomes quicker after a number of frames are processed. This fact is due to the algorithm spending an amount of frames to the system to learn more accurately. In our test, this learning phase (sixty frames) is slower than the other methods [1, 3] in spite of the authors [4] describing that the learning phase would be faster. Most of the time, [1] approach is faster than [3]. This happens because the calculation from [1] considers that red, green, and blue pixel values are independent and have the same variances, decreasing the computational efforts. In [3], the variance is calculated for each pixel component.

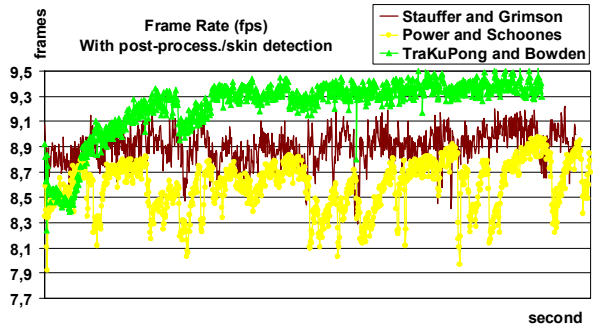

Figure 7. The amount of frames per second indicating the performance of the three methods with post-processing and skin threshold.

Table 3. Frame rates using GMM approaches on a video. The values are shown with post-processing; with post-processing, skin threshold; and without post-processing.

	Post-process			*Post-process./skin*			*No post-process.*		
GMM	[1]	[3]	[4]	[1]	[3]	[4]	[1]	[3]	[4]
Avg	8.83	8.53	9.17	8.91	8.59	9.22	10.05	9. 91	10.63
Best	9.17	8.91	9.49	9.22	8.98	9.52	10.42	10.35	10.98
Worst	8.18	7.38	8.21	8.29	7.92	8.24	9.30	9.22	9.24

6. Conclusions

This paper's main interest application is the hand segmentation as a preparation for gesture recognition. Therefore, the method [4] is not viable to this kind of application due to problems with segmentation in real time using the *L-recent* frames. The application can not wait so many frames in the learning phase due to the real time proposal, despite the chart results (figures 5, 6, 7) showing that the method [4] have the best performance in terms of time processing. An interesting fact is that the authors [4] describe that the learning phase would be faster. Comparing the three approaches during the tests using specific parameter values for each one, [3] reached the best background subtraction results, mainly due to segmentation results than in terms of time processing, since it had the worst performance. There are more homogeneous images with fewer holes determining more continuous contours on the hand images. For the tests using skin color threshold, all addresses had closer results, in terms of segmentation quality. Most of the time, [1] address is faster than [3] but visually the segmentation is not so good as the second one [2]. This happens because the calculation from [1] considers that red, green, and blue pixel values are independent and have the same variances, decreasing the computational efforts but also achieving more mistakes. In [3] the variance is calculated for each pixel component, and therefore it can obtain more accurate results.

The use of techniques with low computational cost and real time requirements are considered and adopted to emphasize the feasibility of this process for practical applications. We intend to use pixel-processing feedback by generating supervised classification, with the capacity to detect classification errors and to readjust the model of Gaussian distributions. It also intends to use the YCbCr color space that can be directly equated to the RGB color space. This color space separates the luminance from the chromatic components and it provides an attractive representation form of colors when desiring to work with a skin color modeling. The YCbCr color space has presented better results for the GMM algorithm than the RGB color space, even the normalized [2, 10].

References

[1] C. Stauffer, and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking", In: Proceedings 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), IEEE Comput. Soc. Part vol. 2, 1999.

[2] M. Harville, G. Gordon, and J. Woodfill. "Foreground segmentation using adaptive mixture models in color and depth", In: Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video, 2001.

[3] P. W. Power, and J. A. Schoones. "Understanding Background Mixture Models for Foreground Segmentation", In: Proceedings Image and Vision Computing New Zealand, University of Auckland, Auckland, New Zealand, 2002, pp. 267-271. Available on line at: http://www.is.irl.cri.nz/pubdoc/2002/JSPP2002.pdf.

[4] P. KadewTraKuPong, and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection", In: Proc. 2nd European Workshop on Advanced Video-Based Surveillance Systems, 2001. Available on line at: http://www.ee.surrey.ac.uk/Personal/R.Bowden/publications/avbs01/avbs01.pdf.

[5] N. Friedman, and S. Russell, "Image Segmentation in Video Sequences: A Probabilistic Approach", In: The Thirteenth Conference on Uncertainty in Artificial Intelligence. Brown University, Providence, Rhode Island, USA: Morgan Kaufmann Publishers, Inc., San Francisco, 1997.

[6] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM algorithm", In: J. Royal Statistical Soc., 1977, vol. 39 (Series B), pp. 1-38.

[7] Intel. OpenCV Open Source Computer Vision Library, 2006. Available on line at: http://www.intel.com/research/mrl/research/opencv/

[8] P. Peer, J. Kovac, J. and F. Solina, "Human skin colour clustering for face detection", In: submitted to EUROCON – International Conference on Computer as a Tool , 2003.

[9] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A survey on pixel-based skin color detection techniques", In: GraphiCon, Moscow, Russia, 2003.

[10] S. L. Phung, A. Bouzerdoum, and D.Chai, "A novel skin color model in ycbcr color space and its application to human face detection", In: IEEE International Conference on Image Processing (ICIP'2002), 2002, vol. 1, pp. 289–292.

[11] Gonzalez, R. C., and Woods, R. E., *Digital Image Processing*, Addison-Wesley Publishing Company, 1993.

[12] D. Koller, J. Weber, T. Huang, and J. Malik, G. Ogasawara, B. Rao and S. Russell, "Towards robust automatic traffic scene analysis in real-time", In: Proceedings of the 33rd IEEE Conference on Decision and Control (Cat. No.94CH34603). IEEE. Part vol.4, 1994.

[13] B. Henrik, B. M. Thomas, and B. M. Claus, "Real-time recognition of hand alphabet gestures using principal component analysis", In: 10th Scandinavian Conference on Image Analysis, Lappeenranta, Finland, 1997.

[14] R. G. Brown, and P. Y. C. Hwang, "Introduction to Random Signals and Applied Kalman Filtering", In: 2nd Edition, John Wiley & Sons, Inc, 1992.

[15] T. Starner, J. Weaver, and A. Pentland, "Real-time American sign language recognition using desk and wearable computer based video", In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, vol.20 (Series 12): pp. 1371–1375.