# Real-Time Detecting and Labeling of Human Body

Yang Yang, C. S. Chua, Y. K. Ho, and Ren Ying
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore 639798
eyangyang@ntu.edu.sg

## Abstract

*This paper proposes a real time method for detecting multiple moving persons and labeling their body parts. This method is characterized by simplicity in realization and good accuracy in detecting body parts of moving persons. Three procedures are included when using this method: detecting moving regions, finding out faces and torsos, detecting legs. Several innovative techniques are employed through the whole detecting process. An adaptive threshold selection technique is utilized to binarize the edge-difference image. Also a fast and effective Line-Scan Clustering(LSC) algorithm is developed to carry out tasks of removing noise and locating moving regions simultaneously. In detecting legs an accumulated histogram, which is obtained by adding together a sequence of bounding boxes of leg regions, is used to determine the color of legs. Using these novel techniques fast and accurate detection and labeling of moving persons has been achieved. In order to testify the proposed method a great number of experiments have been conducted and excellent results are obtained.*

*Keywords: real time vision, detecting and labeling of human body, motion segmentation, image subtraction, clustering analysis, accumulated histogram.*

## 1. Introduction

Detecting and tracking of whole-body human is very useful in many practical applications such as indoor scene surveillance, virtual reality interfaces and traffic management. Labeling of the body parts can be employed to interpret and recognize activities of human.

Up to today many algorithms have been developed. One class of detecting algorithms can be identified as algorithms with modeled background scene, which assumes the camera and background are fixed. For example, Ismail Haritaoglu, et al., [2] present a real time system for detecting and tracking people, $W^4$. They model the background scene by representing each pixel by three values: its minimum and maximum intensity values, the maximum intensity difference between consecutive frames observed during this training period. These values are estimated over several seconds of video and are updated periodically for those parts of the scene that $W^4$ determines to contain no foreground objects. Christopher Richard, et al., [11] describe a system for detecting and tracking a person, $Pfinder$. They model the scene surrounding the human as a texture surface; each point on the texture surface is associated with a mean color value and a distribution about that mean. The color distribution of each pixel is modeled with the Gaussian described by a full covariance matrix. To accomplish learning the scene before locating people in a scene $Pfinder$ begins by acquiring a sequence of video frames that do not contain a person. Typically this sequence is relatively long, a second or more. $Pfinder$ assumes there is a single person in the image. Another class of detecting algorithms are mainly based on the evaluation of motion information. T. Olson and F. Brill [7] present a general purpose system for moving object detection and event recognition where moving objects are detected using change detection and tracked using first-order prediction and nearest neighbor matching. However, as soon as there is other motion in the image besides human beings, this approach has severe difficulties. It is worth to mention that recently color has been applied to the detecting and tracking of moving objects. B. Heisele, et al. [3] use clusters of consistent color to track moving objects. However, an initial manual labeling step is required. There is another group of tracking algorithms based on active contours [5], which require the initial contour of the objects before tracking.

With the ultimate aim of building a complete system of detecting, tracking multiple moving persons and recognizing their activities in real time, this paper is concentrated on detecting moving persons and labeling their body parts, while tracking and recognizing activities are beyond the scope of this paper. The process of the proposed method in this paper includes three procedures. First we develop an improved fast image subtraction method to give the approximate regions of moving objects using two frames. Then color information is used to detect the face and torso within

these regions. Finally Legs are found out through analyzing the accumulated histogram, which is the accumulation of a sequence of bounding boxes of leg regions.

This paper is organized as follows: Section 2 describes the location of moving regions. The detection of face, torso and legs are illustrated in Section 3 and 4 respectively. Experimental results are given in Section 5.

## 2. Moving region location

### 2.1. Background

Motion segmentation is a well studied problem in computer vision. There are two popular methods used to extract moving objects: optical flow computation and image subtraction.

The optical flow can be segmented to identify different moving objects in the image. Two approaches are commonly used to compute the disparity map between two frames. The continuous approach (or gradient approach) uses spatio-temporal variation following the famous motion constraint equation (1) [4] which states that the intensity $I$ at pixel $(x, y)$ at time $t$ should be equal to the intensity at pixel $(x + \triangle x, y + \triangle y)$ at time $t + \triangle t$ due to the motion of the pixel during the time interval $\triangle t$,

$$I(x, y, t) = I(x + \triangle x, y + \triangle y, t + \triangle t) \qquad (1)$$

Gradient-based approaches of optical flow computation can only handle small displacements and are time consuming. The discrete approach (or feature-based approach) for computing the optical flow consists of matching features extracted from both frames. This technique is equivalent to the stereo matching technique which has not been solved well.

Another approach for motion segmentation based on image subtraction is very simple. The image subtraction using three frames [1] can detect only one moving object. We consider the image subtraction using two frames from an image sequence as more than one moving person may occur in our scenes. After thresholding the difference image of the two frames, pixels belonging to the static background should not be extracted. Unfortunately, pixels that belong to parts of the object that overlap between the two frames will not be extracted either and the location of the moving object in both frames will be extracted. Generally the threshold used in binarizing the difference image is empirically chosen [6] [1], which cannot always obtain good results since the environment such as lighting condition will vary with time. Due to inaccurate background compensation and various kinds of noises, false motion (we call it noise also) occurs in the binary difference image. Morphological filter is often applied to noise removal, however, its size will greatly influence the result obtained. Generally, finding a satisfactory combination of erosion and dilation steps is quite difficult

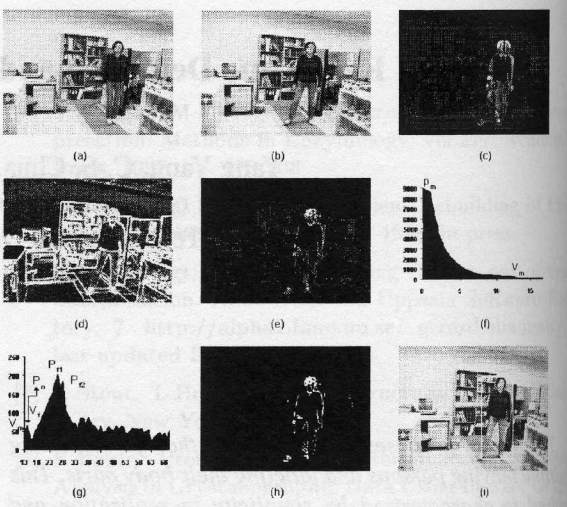


**Figure 1. Moving regions location. (a),(b) Two continuous frames; (c) Gray difference image; (d) Edge image; (e) Edge-difference image $D_e$; (f) and (g) Histogram of the edge-difference image; (h) Binary edge-difference image; (i) Bounding bo x of the moving object detected using LSC on the binary edge-difference image.**

and no fixed combination works well. After erosion and dilation, the connected component labeling is used to obtain the connected moving region. However, the target pixels in the binary difference image cannot be guaranteed to be connected. In this section we develop an improved, fast image subtraction method to give the approximate location of moving objects.

### 2.2. IFIS method (Improv ed fast image subtraction method)

The RGB color image is converted to HSI (hue, saturation and intensity ) space. The *hue* or *tone* is the pure color of the light. It is the dominant color we see. *Saturation* indicates how much white light is mixed with the color (the tint). *Saturation* of 0 means no color: we see white or gray. The *intensity*, which indicates how much total light there is (the shade). In this part color information is not useful, so only the *intensity* image is considered. Firstly the gray difference image between the previous frame $f_{t-1}$ and the current frame $f_t$ is computed without motion compensation (if the platform is stationary) or with motion compensation (if the platform is moving). For example, Fig. 1(a)and (b) are two continuous frames, Fig. 1(c) is the gray difference image of Fig. 1(a) and (b). We can see from the difference image that the location of the moving object in both frames

is highlighted and will be extracted if we detect change directly from the difference image. In order to extract just the edges of the moving object in the current frame we calculate an edge image (using Soble edge detector) of the current frame $f_t$. Fig. 1(d) is the edge image of Fig. 1(b). Then we multiply the gray difference image and the gray edge image pixel by pixel, and the resultant image is a new image called gray edge-difference image, $D_e$. In the gray edge-difference image $D_e$ only the moving edges (or the edges of moving objects) are highlighted which can be seen in Fig. 1(e). The gray edge-difference image, $D_e$, needs to be thresholded to keep only locations of significant change. However, the traditional thresholding techniques [8] [9] [10] cannot give satisfactory results. In this section, we describe a threshold selection algorithm to adaptively obtain the threshold in the gray edge-difference image, $D_e$.

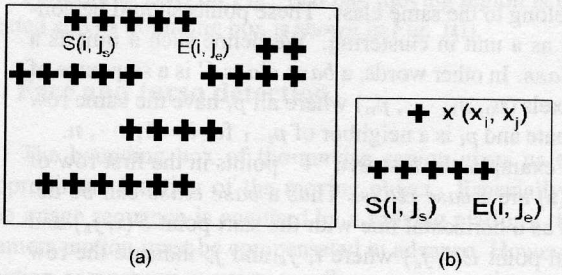


**Figure 2. Simulated binary image and *base class*. (a) A simulated binary image; (b) A *base class* can be described as a ho rizontalline with the start point $S(i, j_s)$ and terminal point $E(i, j_e)$.**

### 2.2.1 Adaptive threshold selection

In the edge-difference image, $D_e$ ( see Fig. 1(e)), we define all the pixels on the edge of the moving objects as foreground pixels. The rest pixels are named background pixels. Obviously in $D_e$ (Fig. 1(e)) the amount of foreground pixels is remarkable smaller than that of background pixels, while the gray level of foreground pixels is much higher compared with that of the background pixels. We can also using the histogram $H_{D_e}$ of $D_e$ to depict these features as follows:

1) In order to make the histogram of Fig. 1(e) be seen clearly we divide the whole histogram into two partial histograms. In these histograms the horizontal axis is the gray level and the vertical axis represents the number of points. Fig. 1(f) shows the partial histogram within the interval [0 15]. Fig. 1(g) is the partial histogram within the interval [13 70] where the vertical coordinate is exaggerated. We do not draw the histogram after the gray level 70 because the number of points is nearly to 0.

2) The highest peak $P_m$ in Fig. 1(f) corresponds to the background pixels because most of pixels in the gray edge-difference image (Fig. 1(e)) are background pixels. In Fig. 1(f) the highest peak $P_m$ exceeds the given coordinate limitation of the vertical axis.

3) The foreground pixels must distribute on the right side of the right valley $V_m$ of peak $P_m$ (see Fig. 1(g)).

4) The narrower and lower peak such as peak $P_n$ in Fig. 1(g) is normally formed due to noise or external disturbances and should not be considered.

5) Two close peaks such as $P_{f1}$ and $P_{f2}$ in Fig. 1(g) can be treated as one peak.

In accordance with the features listed above we can find out the threshold for binarizing the edge-difference image, $D_e$, from the histogram. Firstly the histogram is smoothed so that the narrower and lower peak will be removed and the close peaks will be combined. Secondly from all the peaks on the right side of the valley $V_m$ we find out the first peak, $P_f$ (after smoothing of the histogram the peak $P_n$ is removed and the two peaks $P_{f1}$ and $P_{f2}$ are combined into one peak $P_f$). Finally we obtain the left valley $V_f$ of the peak $P_f$. The gray level corresponding to the valley $V_f$ is selected as the threshold.

The gray edge-difference image, $D_e$, is binarized using the threshold and the product is a binary edge-difference image. The binary edge-difference image of Fig. 1(e) is shown in Fig. 1(h). Due to inaccurate background compensation and various kinds of noises, false motion (we call it noise also) occurs in the binary edge-difference image. Morphological filter is often applied to noise removal, however, its size will greatly influence the result obtained. Generally, finding a satisfactory combination of erosion and dilation steps is quite difficult and no fixed combination works well. Moreover erosion and dilation are time consuming. After noise removal the connected component labeling is used to obtain the connected moving region. However, the foreground pixels in the binary edge-difference image is not guaranteed to be connected. Here a fast and effective Line-Scan Clustering(LSC) algorithm is presented to carry out tasks of removing noise and locating moving regions simultaneously.

### 2.2.2 LSC algorithms

#### A) Basic ideas

We use the Euclidean distance as the similarity measure. A class is represented as a set of target points in a binary image. For any point in one class, at least one point in the same class can be found and the Euclidean distance between them is less than a given distance threshold, $\delta_H$. We find some particular properties in a binary image and take them into consideration in the algorithm designing. Fig. 2(a) draws a simulated binary image where a "+" indicates a target point. We investigate the following properties in binary images:

(1) In one row, continuous neighboring target points

must belong to the same class. These points should be considered as a unit in clustering. We define such a unit as a *base class*. In other words, a *base class* $C$ is a sequence of "+"-pixels $(p_0, p_1, \cdots, p_n)$ where all $p_i$ have the same row coordinate and $p_i$ is a neighbor of $p_{i-1}$ for $i = 1, \cdots, n$.

For example, the first five "+" points in the first row of Fig. 2(a) are a *base class*. Thus a *base class* can be described as a horizontal line with the start point $S(i, j_s)$ and terminal point $E(i, j_e)$ where $i$, $j_s$ and $j_e$ indicate the row coordinate, the column coordinate of the start point and the terminal point of this *base class*, respectively (referred to Fig. 2(b)). We employ a *base class* as a minimum unit in clustering.

(2) When a *base class* ($S(i, j_s)$, $E(i, j_e)$) is obtained, we need to determine whether it can merge with other previous classes. If the distance threshold is $\delta_H$, then only those previous classes which contain at least one point whose column coordinate falls into the interval $[j_s - \delta_H, j_e + \delta_H]$ need to be considered. We call this limitation as $inter - column\ limitation$.

Because if no such points exist in the previous class, the minimum inter-column distance between the *base class* and the previous class is larger than $\delta_H$, the Euclidean distance between them is surely larger than $\delta_H$. The reason is as follows:

if no such points exist in the previous class, for any a point $P(x, y)$ in the previous class, its column coordinate $y$ will be:

$$y < j_1 = j_s - \delta_H \ or \ y > j_2 = j_e + \delta_H. \quad (2)$$

The Euclidean distance $\delta$ between $P(x, y)$ and any a point $Q(i, j)$ in the base class is:

$$\delta = ||P - Q|| = \sqrt{(j - y)^2 + (i - x)^2}. \quad (3)$$

Since $Q(i, j)$ belongs to the base class ($S(i, j_s)$, $E(i, j_e)$), $j$ will be:

$$j_s \leqslant j \leqslant j_e. \quad (4)$$

From equation 2 and 4 we can derive that $|j - y| > \delta_H$. From equation 3 we know $\delta > \delta_H$, which means the distance between a *base class* and a previous class which does not satisfy the $inter - column\ limitation$ will be greater than $\delta_H$. Hence when checking whether a *base class* can merge with other previous classes, we do not need check the previous class which does not satisfy the $inter - column\ limitation$.

Moreover, not all the previous classes which satisfy $inter - column\ limitation$ must be checked, but only the class which lately visits the $j - th$ column where $j \in [j_s - \delta_H, j_e + \delta_H]$ needs to be checked. That is because the inter-row distance between this previous class and the *base class* is the minimum and the Euclidean distance between them is the minimum. We call this limitation as $lately - visiting\ limitation$.

(3) Even a previous class satisfies both the $inter - column\ limitation$ and $lately - visiting\ limitation$, not all the points in this previous class must be involved in the distance calculation when computing the distance between the previous class and the *base class* ($S(i, j_s)$, $E(i, j_e)$). If we set $p = j_s - \delta_H$, $q = j_e + \delta_H$, only points of the previous class fall into the set $X$ where $X = \{(x_i, x_j) | x_j \in [p, q]$, $x_i$ is the row coordinate of the *base class* that visits the column $x_j$ lately $\}$, will be considered, because the inter-column distance between the *bass class* and the point in the previous class not in $X$ is larger than $\delta_H$.

Since a *base class* can be described as a horizontal line the computing of the distance between a point and a *base class* can be simplified as a Euclidean distance of two points by virtue of the following definition (referred to Fig. 2(b)):

Definition: The distance, $\delta$, between a point $x(x_i, x_j)$ and a *base class* ($S(i, j_s)$, $E(i, j_e)$) can be expressed as:

$$\delta = \begin{cases} ||S - x|| = \sqrt{(j_s - x_j)^2 + (i - x_i)^2} & \text{if}(x_j < j_s), \\ ||E - x|| = \sqrt{(j_e - x_j)^2 + (i - x_i)^2} & \text{if}(x_j > j_e), \\ |i - x_i| & \text{otherwise.} \end{cases} \quad (5)$$

**B) Algorithm description**

Before describing LSC algorithm we introduce two data structures. These data structures help us to implement this algorithm. The data structure of a *base class* $C$ called $BASECLASS$ is represented as a 5-tuple consisting the fields BEGINX, BEGINY, ENDY, LABEL, and COUNT. BEGINX(C)is the row coordinate of $C$. BEGINY(C) is the column coordinate of the start point of C while ENDY(C) is the column coordinate of the terminal point of $C$. LABEL(C) contains the class label of $C$. COUNT(C) records the number of points in $C$. In order to make the classes merging operation easier and record the classes information more convenient, we add another 5 fields into the structure of $BASECLASS$, they are TAG, BRANCH, PARENT, LEFTSON, and RIGHTSIB. Therefore a class not only a *base class* can be represented by this structure. In fact, a class constitutes a tree on which each node is represented by a $BASECLASS$. TAG(C) indicates whether the node $C$ is the root of the tree where node $C$ locates. If $C$ is the root TAG(C) is set to $'t'$ otherwise $'f'$. BRANCH(C) accumulates the number of nodes in tree $C$. PARENT(C), LEFTSON(C) and RIGHTSIB(C) indicate the parent, left son and right sibling of $C$ respectively.

Another very simple but important data structure we introduced is $ColumnArray$ which contains two fields: $RowCoordinate$ and $ClassLabel$. We define a $ColumnArray$ type array $vol$ which is an array as large as the width of the binary image. For $vol[j]$, the field of $RowCoordinate$ records the row coordinate of the class that lately visits the $j-th$ column and the field of $ClassLabel$ records the label of the class that lately visits the $j-th$ column.

LSC algorithm scans the image only once, and it merges classes whenever their distance is less than or equal to $\delta_H$ and no repeated merging operation is done. This feature greatly reduces the running time of the clustering algorithm. How to do that?

First, the label of each node $C$ in a class is referred to the label of the root of the tree where node $C$ locates. When determining whether two classes belong to the same class or not LSC compares the labels of the roots of these two classes. Hence there will not be any confusion in the labels of different $base\ classes$ or classes.

Second, when merging two classes LSC algorithm will check the branches of the trees corresponding to these two classes and merge the tree with less branches into the tree with more branches. This will avoid producing unstable trees and will reduce the time of tree searching.

Now LSC algorithm can be described as follows:

Step 1: Let a variable, $kindnum$, represent the label of the current $base\ class$ and set $kindnum = 0$.

Step 2: Scan the image row by row to yield the current $base\ class$ $(S(i, j_s),\ E(i, j_e))$ and give it a temporal class label $TempLabel$ with initial value $kindnum$.

Step 3: Take a point $x_j$ ( $x_j \in X = \{(RowCoordinate(vol[j]), j) \mid j \in [p, q]\ where\ p = j_s - \delta_H, q = j_e + \delta_H\ \}$ and calculate the distance, $\delta$, between $x_j$ and the current $base\ class$ by Eq. (5). If $\delta \leq \delta_H$, merge the current $base\ class$ (or the current class if the current $base\ class$ has been merged into a previous class ) with the previous class and modify $TempLabel$ accordingly.

Step 4: Let $X - x_j \rightarrow X$. If $X \neq \emptyset$, then go to step 3.

Step 5: Modify corresponding contents of the record array, $vol$, according to the label of the current $base\ class$, and the column and row coordinates that the current $base\ class$ spans. If the current $base\ class$ cannot be merged into any previous class ($TempLabel = kindnum$), then the current $base\ class$ is a new class, set its class label be $TempLabel$, and update $kindnum$, namely, $kindnum = kindnum + 1$.

Step 6: If the image scanning is finished, then the algorithm is over; else go to step 2.

LSC algorithm not only produces the number of classes automatically but also saves considerable computational time because it scans the whole image only once. Also the $base\ class$, not the single point, is considered as the minimum unit when clustering.

Performing LSC on Fig. 1(h) one moving region is detected and its bounding box is shown in Fig. 1(i).

## 3. Face and torso detection

The bounding box of the moving region gives us the approximate location of the moving object. Especially if the image sequence is acquired by a moving platform, the camera motion must be compensated in advance. However, motion compensation using an affine or perspective model is not error free and includes false motion in the edge-difference image, thus the bounding box will be bigger than the actual bounding box of the moving person. In order to determine whether the moving object is a person and detect more accurate information of the person we use color to detect the face in the bounding box of the moving object since color is an important feature of human skin. Once a face is located in the bounding box, a person is verified present; otherwise it will not be considered any more.

### 3.1. Color model

We want a color model to detect skin color pixels in a changing environment. Processing color is relatively robust to changes in viewpoint, scale, shading, and distinguishes relatively well in complex (cluttered) backgrounds. However, the color of a face is influenced by many factors such as ambient light, clothes, noise, object movement, camera and others. Even under the same lighting conditions, background colors such as colored clothes may influence skin-color appearance. Moreover, human skin color differs from person to person. Therefore most color-based systems are sensitive to changes in viewing environment. The general color models, such as r-g and CIE-xy, need two primary color components to describe one color, which means two variants must be controlled simultaneously. We select the $HSI$ color space because "$hue$" means $color$ and "$hue$" itself can be used to describe one color. For instance "$hue = 60°$" is representative of "yellow" color. The varying range of "$hue$" of skin color can be estimated in different environments when assuming the natural color of the skin is fixed. For example, the skin of Chinese descent can be identified with a natural color of "yellow" even though the saturation of "yellow" may vary within a relatively wide range. Hence we describe the face color distribution by a one-dimension Gaussian model N($\mu$,$\beta^2$), where $\mu$ and $\beta^2$ represent the mean value and variance of "hue" respectively. For a pixel $i$ with value $hue_i$ the Mahalanobis distance from $hue_i$ to $\mu$ is defined as

$$D_m{}^2 = (hue_i - \mu)^2 / \beta^2 \qquad (6)$$

The larger $D_m$ is, the lower the probability that the pixel belongs to the skin color. Equation (6) can be used to identify skin-colored pixels.

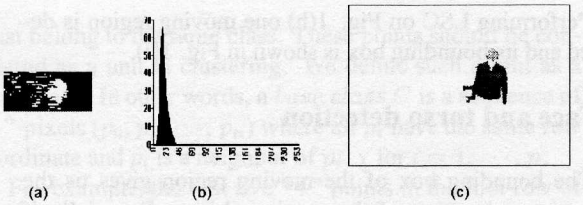


**Figure 3. Face and torso detection. (a) Skin-color labeling on one fifth of the bounding box of the moving region; (b) Smoothed histogram of a rectangle region $R_s$ on the torso; (c) Detected face and torso.**

Such a color model can be built up from a set of face images. In order to detect the skin-color pixels in a changing environment, we reset $\beta$ to a higher value, $\beta_h$. We detect skin-color pixels using this color model on one fifth of the region of the bounding box of the moving region. Fig. 3(a) is a skin-color labeled region where white pixel means skin-color pixel.

### 3.2. Facedetection and torso detection

LSC algorithm is performed on the skin-color labeled region. Clusters with a certain number of points greater than a given value are taken as face candidates. Since faces in image sequences are often small, we cannot use facial features such as the eyes, nose and mouth to identify the face. We use the shape features to further verify the face. Let $l_1$ and $l_2$ represent the length and width of the bounding rectangle of the face candidate, respectively. The face should satisfy the two limitations: 1)The ratio $(l_1/l_2)$ is in a predefined range; 2)The ratio $(area/(l_1 \times l_2))$ is greater than a given value. Although these are not adequate to verify that a skin-colored region is a face, in real world there is little chance that a moving object with a skin-colored face in the head position is not a person.

Once a face is detected we try to detect the torso in the region below the face of the bounding box of the moving region. Assuming the color of a person's clothes is normally distributed, we select a rectangle region $R_s$ on torso ($R_s$ locates in the position just a little lower than the face, its width is the same as the face and its height is one half the height of the face, which guarantees that $R_s$ is within the real torso region) and analyze its histogram to locate the whole torso. The maximum peak of the histogram represents the color of the torso region. So the torso can be detected. The histogram (smoothed with a Gaussian mask) of the small rectangle region on the torso in the moving region of Fig. 1(i) is shown in Fig. 3(b). Fig. 3(c) draws the detected face and torso.

## 4. Legs detection

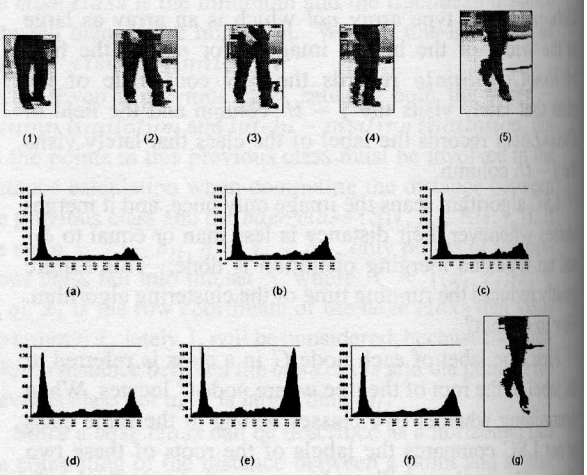


**Figure 4. Legs detection. (1)~(5) Sequence of moving leg regions; (a)~(e) Sequence of histograms of moving leg regions; (f) Accumulated histogram; (g) Detected legs.**

This section will be focused on the detection of legs which is very useful in the recognition of human activities. Once the face, torso and the bounding box of a moving person are detected, the bounding box of the leg region of the moving person is easily determined by a simple subtraction. Hence we can obtain a sequence of bounding boxes of leg regions in the image sequence. For example, Fig. 4(1)~(5) show a sequence of moving leg regions. The histogram for each frame is calculated and shown in Fig. 4(a)~(e). In most of frames of leg regions, legs are the main parts as shown in Fig. 4 (1)~(4). Form the histograms of Fig. 4 (a)~(d), there are prominent peaks representing the leg region, which are higher than the other peaks caused by background. This feature is deduced from the view of the whole sequence of moving legs. However in some images, for instance Fig. 4(5) and corresponding histogram Fig. 4(e), leg region is not the dominant and its corresponding peak is not as significant. Thus we cannot use only one image to find out the leg region. When adding all the histograms of the sequence an accumulated histogram is obtained as shown in Fig. 4(f). In the accumulated histogram, the peak representing the leg region is much higher than the other peaks caused by the background. This is because of two reasons:

1) Within the sequence, the color of legs is rigid and is relatively constant and its occurrence rate accumulated stably while the background in the bounding box keeps changing in every sequential image and the corresponding accumulation is therefore relatively weak;

2) The accumulated histogram reflects the feature of the whole sequence of images.

The prominent peak in the accumulated histogram represents the color of the leg region. So legs can be detected. Fig. 4(g) draws the detected legs.

## 5. Experimental results

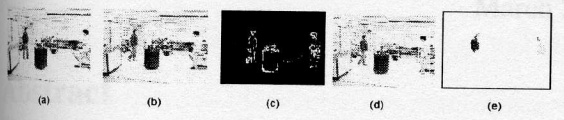


**Figure 5. Detecting and labeling of human body example I. (a),(b) Two continuous frames; (c) Binary edge-difference image; (d) Moving regions detected; (e) Faces and torsos of moving persons detected.**

Our current application background is the indoor moving person surveillance. We have tested the proposed method on a great number of indoor color image sequences. The resolution of the color image is 320×240. The method is implemented in visual C++ and the running time is about 30 ms on a PentiumIII(500MHZ) PC. Here we give two typical examples. Fig. 5 shows an example of detecting moving persons from two continuous frames Fig. 5(a) and (b), where there are two moving persons and one moving robot which is controlled by the moving man. Fig. 5(c) is the bianry edge-difference image. Performing LSC algorithm on Fig. 5(c) three moving objects are detected and the bounding boxes of them are shown in Fig. 5(d). After face detection only two persons are detected and the detected faces and torsos are shown in Fig. 5(e). Fig. 6 shows another example of detecting two walking persons in a sequence, where Fig. 6(a) and (b) are the first two frames of the sequence, Fig. 6(c) shows the detected persons from the two continuous frames but their legs cannot be detected now. Fig. 6(d) ~ Fig. 6(f) give the detecting result in the fifth, eighth and tenth frames. In the fifth frame (Fig. 6(d)) the legs of the left person are detected, while the legs of the right person are detected until the eighth frame (Fig. 6(e)). In the tenth frame (Fig. 6(f)) the two persons meet, however they can be detected because there are two faces detected in the moving region.

The experimental results are very inspiring. Some false detecting results are caused by the skin-colored object existing in the background which is included in the bounding box of moving regions. We hope to develop more powerful face location technique to reduce the rate of false detection.

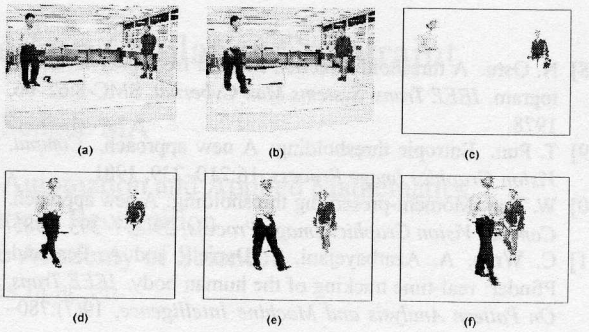


**Figure 6. Detecting and labeling of human body example II. (a),(b) First tw oframes in the sequence; (c) Moving persons detected in the second frame; (d) Moving persons detected in the fifth frame; (e) Moving persons detected in the eighth frame; (f) Moving persons detected in the tenth frame.**

## 6. Conclusions and future work

We have proposed a real time method for detecting and labeling of multiple moving persons. The experimental results are very satisfying since the face, torso and leg region are accurately detected. The future work will be centered on tracking multiple persons and recognizing their activities.

## References

[1] M. Dubuisson and A. Jain. Contour extraction of moving objects in complex outdoor scenes. *International journal of computer vision*, 14(1):83–105, 1995.

[2] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Who? when? where? what? a real time system for detecting and tracking people. *Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 222–227, 1998.

[3] B. Heisele, U. Kressel, and W. Ritter. Tracking non-rigid, moving objects based on color cluster flow. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 257–260, 1997.

[4] B. Horn and B. Schunck. Determining optic flow. *Artificial Intelligence*, 17:185–203, 1981.

[5] M. Kass, A. witkin, and D. Terzopoulos. Snakes: active contour models. *Proceedings of the First International Conf. on Computer Vision*, pages 259–268, 1987.

[6] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5):449–459, 1994.

[7] T. Olson and F. Brill. Moving object detection and event recognition algorithms for smart cameras. *Proc. DARPA Image Understanding Workshop*, pages 159–176, 1997.

[8] N. Ostu. A threshold selection method from gray-level histogram. *IEEE Trans. Systems Man Cybernet*, SMC-8:62–66, 1978.

[9] T. Pun. Entropic thresholding: A new approach. *Comput. Vision Graphics Image Process*, 16:210–239, 1981.

[10] W. Tsai. Moment-preserving thresholding: A new approach. *Comput. Vision Graphics Image Process*, 29:377–393, 1985.

[11] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.