

Object Tracking Initialization Using Automatic Moving Object Detection

Ka Ki Ng and Edward J. Delp
Video and Image Processing Laboratories (VIPER)
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana USA

ABSTRACT

In this paper we present new methods for object tracking initialization using automated moving object detection based on background subtraction. The new methods are integrated into the real-time object tracking system we previously proposed. Our proposed new background model updating method and adaptive thresholding are used to produce a foreground object mask for object tracking initialization.

Traditional background subtraction method detects moving objects by subtracting the background model from the current image. Compare to other common moving object detection algorithms, background subtraction segments foreground objects more accurately and detects foreground objects even if they are motionless. However, one drawback of traditional background subtraction is that it is susceptible to environmental changes, for example, gradual or sudden illumination changes. The reason of this drawback is that it assumes a static background, and hence a background model update is required for dynamic backgrounds. The major challenges then are how to update the background model, and how to determine the threshold for classification of foreground and background pixels. We proposed a method to determine the threshold automatically and dynamically depending on the intensities of the pixels in the current frame and a method to update the background model with learning rate depending on the differences of the pixels in the background model and the previous frame.

Keywords: moving object detection, background subtraction, adaptive threshold, background model update, object tracking, particle filtering, color features, edge features, histograms

1. INTRODUCTION

Visual surveillance has been a very active research topic in the last few years due to the growing importance for security in public places. A typical automated visual surveillance system consists of moving object detection, object classification, object tracking, activity understanding, and semantic description.

Moving object detection is not only useful for object tracking initialization for a visual surveillance system, it is always the first step of many different computer vision applications, for example, automated visual surveillance, video indexing, video compression, and human machine interaction. Since subsequent processes are greatly dependent on the performance of this stage, it is important that the classified foreground pixels accurately correspond to the moving objects of interests.

Moving object detection aims at extracting moving objects that are of interest in video sequences with background which can be static or dynamic. Some examples of interesting objects include walking pedestrians and running vehicles, but usually not waving tree leaves. Most algorithms use either temporal or spatial information in the image sequence to perform moving object detection, and the most commonly used feature is pixel intensity. Many algorithms for moving object detection have been proposed in the literature, most of them can be categorized into one of the three most popular approaches. Some of them can perform moving object detection in real-time, they include background subtraction [1-3] and frame differencing [4]. The third approach, optical flow [5] is a more complex method and it does not work in real-time. More details of these methods will be discussed in Section 2.

This work was supported by the US Department of Homeland Security Regional Visualization and Analytics Center (RVAC) Center of Excellence. Address all correspondents to E. J. Delp at ace@ecn.purdue.edu.

There are a lot of common difficulties and problems encountered when performing moving object detection. Some examples of these are illumination change (at different times of the day in the case of outdoor) and repetitive motion from clutter such as waving tree leaves. Due to these problems with dynamic environmental conditions, moving object detection from the background becomes very challenging.

The major challenges for background subtraction are how to update the background model, and how to determine the threshold for classification of foreground and background pixels. We proposed an algorithm to determine the threshold automatically and dynamically depending on the pixel intensities of the current frame, and a method to update the background model with learning rate depending on the pixel differences between the background model and the previous frame.

The output of moving object detection module is a noisy motion mask which indicates the results of the background-foreground pixel classification. It will then be further processed using morphological operations to obtain an accurate motion mask. This mask is used to initialize the object tracking system we previously proposed in [6].

2. TRACKING INITIALIZATION USING MOVING OBJECT DETECTION

We use moving object detection to generate a motion mask which provides information such as position and size of the target object that is required to initialize the object tracking module.

Moving object detection is always the first step of a typical surveillance system. Moving object detection aims at extracting moving objects that are interesting out of a background which can be static or dynamic. Since subsequent processes are greatly dependent on the performance of this stage, it is important that the classified foreground pixels accurately correspond to the moving objects of interests. The three most popular approaches to moving object detection are background subtraction, temporal frame differencing, and optical flow.

Background subtraction is widely used for moving object detection especially for cases where the background is relatively static because of its low computational cost. The name “background subtraction” comes from the simple technique of subtracting the background model from the current frame to obtain the difference image, and by thresholding the difference frame, a mask of the moving object in the current frame is obtained. The entire process is done in a pixel-by-pixel fashion. The algorithm makes the assumption that every frame in a sequence is made up of a fixed background with moving objects on it, and the pixels of the foreground object have different grayscale intensities (or color channels for color images) from those of the background pixels. There are many variations of background subtraction, but the major difference among most of them is how the background model is constructed and updated. Some background models are adaptive and are updated dynamically, and some of them use statistical model for each pixel of the background.

The traditional background subtraction method subtracts the background model from the current image. It segments foreground objects accurately. It also detects foreground objects even if they are motionless. However, traditional background subtraction is susceptible to environmental changes, for example, in the cases of gradual or sudden illumination change. These changes alter the background model. The result of background subtraction is always contaminated by a large number of erroneous foreground pixels. The major drawback of background subtraction is that it only works for static background, and hence background model update is required for dynamic background scenes.

Temporal frame differencing is also an algorithm using a pixel-wise difference between frames. However, unlike background subtraction, temporal differencing does not have any background model. The current frame is simply subtracted from the previous frame, and if the difference in pixel values for a given pixel is greater than a threshold, the pixel is considered part of the foreground. Frame differencing has very low computational cost so it can be done in real-time. It is also very adaptive to dynamic background. However it has two major drawbacks, objects with uniformly distributed intensity values (homogeneous region such as the side of a car), the interior pixels are interpreted as part of the background. Another problem is that objects must be continuously moving in every frame. If an object stays still for more than a frame period it becomes part of the background. A challenging task for frame differencing is to determine the value of the threshold. Different sequence requires a different threshold to classify the pixel as a foreground or background pixel.

The third approach, optical flow [5] is a more complex method and it does not work in real-time. Optical flow methods make use of the flow vectors of moving objects over time to detect moving regions in an image. It describes coherent motion of points or features between image frames. They can detect motion in video sequences even from a moving camera, however, most of the optical flow methods are computationally complex, very sensitive to noise, and cannot run in real-time.

The major challenges for background subtraction are how to update the background model, and how to determine the threshold for classification of foreground and background pixels. We proposed an algorithm to determine the threshold automatically and dynamically depending on the pixel intensities of the current frame, and a method to update the background model depending on the pixel differences between the background model and the previous frame.

2.1. Background Model Initialization

To detect moving objects using background subtraction, the first step is to obtain the background model at the beginning of a sequence. We assume that the sequence starts with a background in the absence of moving objects. Then we use the selective averaging method [7] to extract the background model and it is defined as:

$$BM_N(x, y) = \frac{\sum_{m=1}^N I_m(x, y)}{N} \quad (1)$$

where $BM_N(x, y)$ is the intensity of pixel (x,y) of the background model, $I_m(x, y)$ is the intensity of pixel (x,y) of the m^{th} current frame, and N is the number of frame used to construct the background model. It is shown that 100 is a good choice for N [7].

2.2. Extracting Moving Object Using Adaptive Thresholding

After obtaining the background model, we need to compute the difference between the current frame and the background model.

$$D_t(x, y) = |I_t(x, y) - BM_t(x, y)| \quad (2)$$

where $I_t(x, y)$ is the intensity of pixel (x,y) of the current frame at time t .

A motion mask representing regions of significant motion in the scene is produced by a threshold value, and this threshold, $Th_{Ad,t}$ is computed adaptively for each frame using the current difference image, $D_t(x, y)$ (details will be discussed below). If the difference between two pixel values is greater than the threshold, the movement of that pixel is considered as significant. Another fixed threshold, Th_{fix} , is introduced here. The fixed threshold which is set to be a very small value is used to identify pixels that are very likely to be part of the background. The results will be used to determine the learning rate of background update and details will be given in Section 2.3.

$$D_{Th,t}(x, y) = \begin{cases} 255 & \text{if } D_t(x, y) \geq Th_{Ad,t} \\ 0 & \text{if } D_t(x, y) < Th_{fix} \\ 127 & \text{if } Th_{fix} \leq D_t(x, y) < Th_{Ad,t} \end{cases} \quad (3)$$

where $D_{Th,t}(x, y)$ is the thresholded difference image, and $Th_{fix} < Th_{Ad,t}$.

To obtain the adaptive threshold, an iterative method proposed by Rilder and Calvar [8] is used. The histogram of the difference image, $D_t(x, y)$ is segmented into two parts. The initial threshold is set to be the middle value of the range of intensities which is 127 for our experiments, and then the image is thresholded with it. For each iteration, the sample means of the intensities associated with the foreground pixels and the sample means of the intensities associated with the background pixels are computed. Then a new threshold value is computed as the average of the two means, the iteration stops when the threshold stops changing.

After thresholding the difference image, each pixel is classified as a foreground or background image pixel. For the case where $D_{Th,t}(x, y) = 255$, that pixel is classified as a foreground pixel. For $D_{Th,t}(x, y) < 255$, the pixel

is classified as a background pixel. This motion mask is usually contaminated by scattered noise. Morphological operations are used to remove the scattered noise.

$$(x, y) \in \begin{cases} foreground & \text{if } D_{Th,t}(x, y) = 255 \\ background & \text{if } D_{Th,t}(x, y) < 255 \end{cases} \quad (4)$$

A brief introduction of morphology will be given in Section 2.4.

2.3. Adaptive Background Model Update

Background model should be adaptive to the background dynamics such as illumination changes. To update the background model, each pixel is updated according to the difference of the background model and the previous frame. If the pixel (x,y) is classified as a foreground pixel (i.e. $D_{Th,t}(x, y) = 255$), the background model at that location will stay the same, the background model at (x,y) will not be updated. If (x,y) is classified as a background pixel, then the background model updates according to how certain it is to be a background pixel. A smaller fixed threshold, Th_{fix} is used to define a pixel with high likelihood of being a background pixel. (The fixed threshold is assigned as 10 in our experiments.) That is the case when $D_{Th,t}(x, y) = 0$. In this case we update the background model at that location to the corresponding pixel in the current frame (i.e. setting the learning rate, α as 1). If the difference is less than the adaptive threshold, and is greater than the fixed threshold, i.e. $D_{Th,t}(x, y) = 127$, then we update the background model with a much lower learning rate. In this case, the learning rate is assigned as 0.1 in our experiments.

$$BM_t(x, y) = \begin{cases} BM_{t-1}(x, y) & \text{if } D_{Th,t}(x, y) = 255 \\ I_t(x, y) & \text{if } D_{Th,t}(x, y) = 0 \\ \alpha I_t(x, y) + (1 - \alpha)BM_{t-1}(x, y) & \text{if } D_{Th,t}(x, y) = 127 \end{cases} \quad (5)$$

2.4. Grayscale Morphology

The motion mask is usually contaminated by a large number of erroneous foreground pixels due to image noise, background motion, and camera jitter. Morphology filters are used to remove the noises and holes (removing holes is the same as filling small gaps) of the motion mask and segment the objects. The fundamental morphological operations are erosion and dilation.

Erosion To erode an image, each object pixel that is touching a background pixel is changed into a background pixel, erosion removes isolated background pixel. The definition of erosion is

$$f(x, y) = \min\{ I(x, y) \text{ and its neighboring pixels } \} \quad (6)$$

where $f(x, y)$ is the eroded image at pixel (x,y), and $I(x, y)$ is the pixel intensity. 8-point neighborhood is used to define neighboring pixels. The structuring element is selected empirically to be a 3x3 grid with its origin at the center.

Dilation Dilation is the dual of erosion. Each background pixel that is touching an object pixel is changed into an object pixel when it is dilated. Dilation adds pixels to the boundary of the object and closes isolated background pixel (fills up the holes of the object). The definition of dilation is

$$g(x, y) = \max\{ I(x, y) \text{ and its neighboring pixels } \} \quad (7)$$

where $g(x, y)$ is the dilated image at pixel (x,y).

Opening Opening is an erosion followed by a dilation operation. Opening removes small islands and thin filaments of object pixels. It is used to remove fine object, separate a target from fine points, and smooth a boundary without changing the area or shape. The definition of opening is

$$Opening(M) = Dilation(Erosion(M)) \tag{8}$$

Closing Closing is a dilation operation followed by an erosion operation. Closing removes islands and thin filaments of background pixels. It fills small holes within objects, and join the neighboring objects without changing the area or shape.

$$Closing(M) = Erosion(Dilation(M)) \tag{9}$$

These techniques are useful for processing noisy images where some pixels have the wrong classification results. Successive operations of opening and closing are used to improve these defects. We perform closing once followed by opening once for our experiments.

After applying morphological operations, the accurate motion mask is obtained, and the position and shape of the moving objects can be extracted from it. The position is computed as the centroid of the moving segment in the motion mask, and the shape is modeled by the smallest ellipse that encloses the moving segment in the motion mask. This information will be used to initialize object tracking. A diagram of the moving object

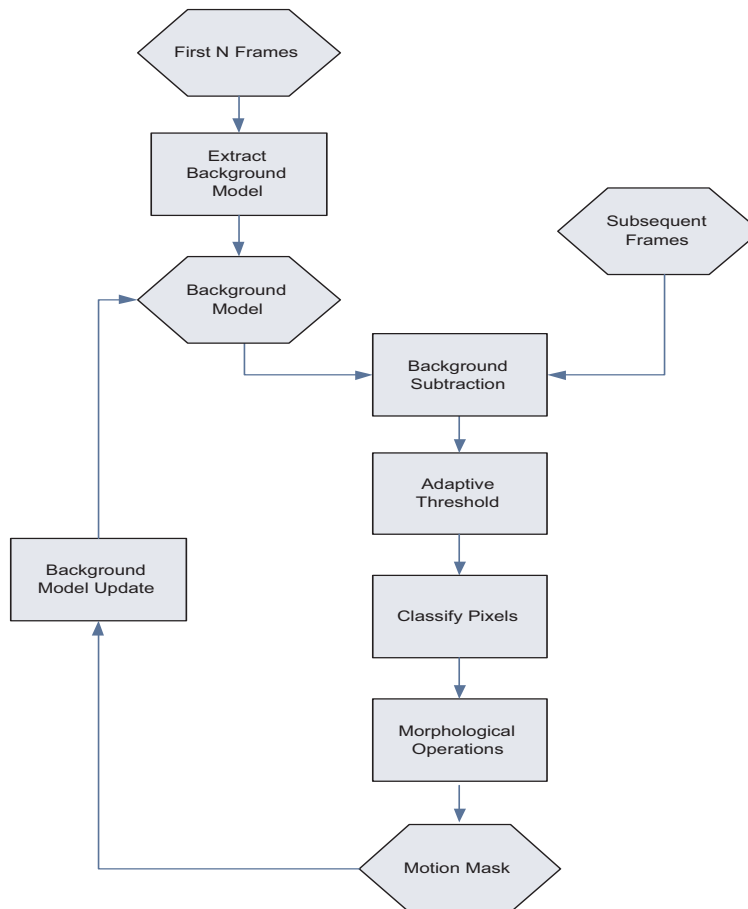


Figure 1. Tracking Initialization System Diagram.

detection algorithm is shown in Figure 1.

3. PARTICLE FILTERING

Object tracking problem can be formulated as a state estimation problem given available observation. Particle filtering can be used as a mathematical tool in object tracking to estimate the target state. Particle filtering is popularly used for object tracking because it has been shown to be very successful for non-linear and non-Gaussian dynamic state estimation problems and is still very reliable in cases like clutter and occlusions.

3.1. Filtering

The “filtering” problem is the process of estimating a system’s current state which is hidden, based on past and current observations [9]. This is represented by the probability density function $p(x_t|x_{t-1}, z_{0:t})$. Depending on the application, the state at time t , x_t and observation at time t , z_t can be different entities. For the case of visual tracking, the state can be position, velocity, orientation, or scale of an object, and the observation can be the color histogram, edge orientation histogram, or contours of the image data.

In general, the object tracking problem can be modeled by the state-space representation

1. State Equation:

$$x_t = f_t(x_{t-1}, v_t) \quad (10)$$

2. Measurement Equation:

$$z_t = h_t(x_t, n_t) \quad (11)$$

where f_t is the state equation (transition equation) at time t , and h_t is the measurement equation (observation equation) at time t . These are non-linear and time varying in general. The system noise and the measurement noise are denoted by v_t and n_t respectively.

3.2. Bayesian Recursive Filtering

The optimal estimator \hat{x}_t (optimal in the minimum variance sense) of the state x_t is provided by the conditional expectation $E[x_t|z_{1:t}]$. To obtain $E[x_t|z_{1:t}]$, we need to estimate the posterior probability density function $p(x_t|z_{1:t})$ at each time instant t , and this estimation can be done by using Bayesian filtering.

The goal of Bayesian filtering is to construct the posterior probability density function $p(x_t|z_{1:t})$, of the state x_t at time t given all available observations $z_{1:t}$ up to time t . It is assumed that the initial probability density function of the state (prior probability density function) $p(x_0)$ is known. Then the posterior probability density function $p(x_t|z_{1:t})$ can be obtained recursively with a two-stage process: prediction and update. Recursive filtering means that the received or observed data can be processed sequentially rather than as a batch:

- Prediction: Assume that the posterior probability density function $p(x_{t-1}|z_{1:t-1})$ is available, the prediction step yields $p(x_t|z_{1:t-1})$ using the Chapman-Kolmogorov integral equation:

$$p(x_t|z_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1} \quad (12)$$

where x_t is the state at time t and $z_{1:t-1}$ is the observation sequence from time 1 to time $t-1$. The probability density function $p(x_t|x_{t-1})$ expresses the state model (transition model).

- Update: At time t , the observation z_t is available, the prior probability density function $p(x_t|z_{1:t-1})$ is updated via Bayes’ rule:

$$p(x_t|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{\int p(z_t|x_t)p(x_t|z_{1:t-1})dx_t} \quad (13)$$

where x_t is the state at time t and $z_{1:t}$ is the observation sequence from time 1 to time t . The probability density function $p(z_t|x_t)$ expresses the likelihood function that describes the measurement model.

In general Equations (12) and (13) cannot be evaluated in closed form. In some special cases, such as linear and Gaussian state-space models which can be analyzed using a Kalman filter, and hidden finite-state space Markov chains closed form solutions are available. To solve this problem, particle filtering is used to approximate Bayesian filtering when a closed-form solution cannot be obtained (i.e. when the transition and observation models are non-linear and non-Gaussian).

The key idea of particle filtering is to represent the posterior probability density function by a set of discrete samples known as particles. A sample is also referred to as a particle because of its discrete nature and its discrete representation by the probability density function. Each particle represents a hypothesis of the state and it is randomly drawn from the prior density. After a particle is drawn, it is then propagated according to the transition model. Each propagated particle is verified by a weight assignment using the likelihood model. The weight characterizes the quality of a specific particle. A large weight will be assigned to a good particle, and a small weight will be assigned to a bad particle.

The posterior probability density function is constructed recursively by the set of weighted random samples $\{x_t^{(i)}, w_t^{(i)}; i = 1, \dots, N\}$ where N is the total number of particles. At each time t , the particle filtering algorithm repeats a two-stage procedure: prediction and update:

- Prediction: Each particle $x_t^{(i)}$ evolves independently according to the state model, including the addition of random noise in order to simulate the unknown disturbance. This step yields an approximation of the prior probability density function:

$$p(x_t) \approx \frac{1}{N} \sum_{i=1}^N \delta(x_t - x_t^{(i)}) \quad (14)$$

- Update: Each particle's weight is evaluated based on the latest measurement according to the measurement model (likelihood model). The posterior probability density function at time t in the form of a discrete approximation can be written as

$$p(x_t | z_{1:t}) \approx \sum_{i=1}^N w_t^{(i)} \delta(x_t - x_t^{(i)}) \quad (15)$$

where

$$w_t^{(i)} = \frac{\mathcal{L}(z_t | x_t^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)})}{q(x_t^{(i)} | x_{t-1}, z_t)} \quad (16)$$

where $q(\cdot)$ is the importance density, $\mathcal{L}(\cdot)$ is the observation likelihood model, and the set of weights sums to 1.

4. OBJECT TRACKING USING PARTICLE FILTERING

This section provides a brief overview of the object tracking algorithm we developed previously [6]. A particle filtering framework in conjunction with color histogram and edge orientation histogram is used for object tracking. Particle filtering is used for object tracking because it is very robust for non-linear and non-Gaussian dynamic state estimation problems and performs well even when clutter and occlusions are present. The target object can be represented by its features in a feature space. Spatially weighted normalized color histogram and edge orientation histogram are used to estimate the probability density functions of the object's color and edge features.

4.1. Target Representation

Color histograms are used for object tracking because they are robust to partial occlusion and are rotation and scale invariant. They are also flexible in the types of object that they can be used to track, including rigid (e.g. vehicles) and non-rigid objects (e.g. people). However, problems arise when the target object and background have similar color distribution. It may be difficult to distinguish the object from background. To solve this problem, multiple-feature tracking is used because it provides more information about the object. Edge features are introduced as a complementary feature to our observation model because it is useful for modeling the structure of the object to be tracked. The edge features are used with a histogram based on the estimated strength and orientation of the edges. Since we assume color and edge features are independent, the similarity measures introduced below for each of them are computed separately.

The classical way of estimating the probability density function is to use every pixel in a region R as a sample for the histogram (color or edge histogram for our experiments) followed by applying a kernel on that sample. In our experiments, the efficient histogram computation [6] is used to estimate the probability density function with higher efficiency. The observed information will be unstable at peripheral points in the region and at its center when the object is undergoing partial occlusions and background changes. To avoid this, assigning larger weights for the central points of the region is achieved by using the Epanechnikov kernel $K(x)$ [10–12]. The probability of the quantized color vector or edge orientation u in the target object model is given by

$$q_u(x) = C \sum_{i=1}^n K(\|d(x_i)\|^2) \delta[b(x_i) - u] \quad (17)$$

where $C = \frac{1}{\sum_{i=1}^n K(\|d(x_i)\|^2)}$ is the normalization factor, and it ensures $\sum q_u(x)=1$. The kernel K we use is the Epanechnikov kernel and it is a convex and monotonically decreasing function. $b(x_i)$ is the bin index where the color or edge component at x_i falls into, and δ is the Kronecker delta function.

While the histogram of the target object model, $q_u(x)$ uses the region R defined by the position and size obtained by the motion mask, the histogram of the candidate s_t , $p_u(s_t)$ is defined similarly except that the region R is centered at the particle’s position.

4.2. Similarity Measure

To evaluate the similarity between the model histogram, q and the particle’s histogram, $p(s_t^{(i)})$, where $s_t^{(i)}$ is the i^{th} particle at time t , we employ the Bhattacharyya coefficient ρ .

$$\rho[p(s_t), q] = \sum_{u=1}^m \sqrt{p_u(s_t)q_u} \quad (18)$$

where u is the histogram bin index and m is the number of bins. The larger ρ is, the more similar the two distributions are. For two identical normalized histograms we obtain $\rho = 1$, indicating a perfect match. To quantify the distance between two distributions, the distance d is defined as

$$d = \sqrt{1 - \rho[p, q]} \quad (19)$$

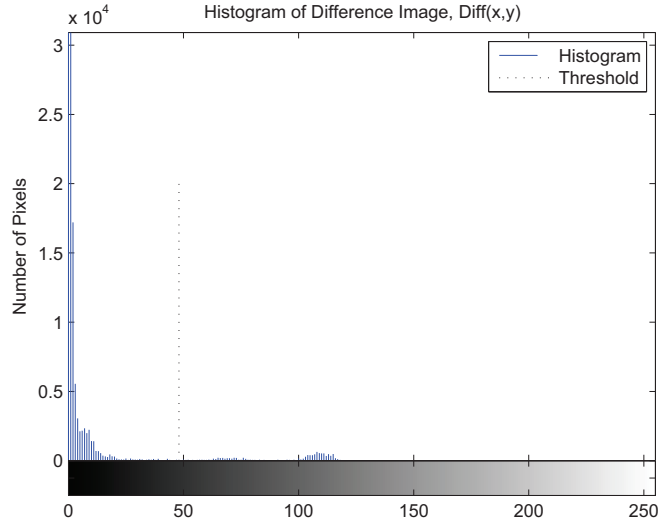
which is known as the Bhattacharyya distance. The observation likelihood function uses this distance to assign a weight to each particle. A sample with small Bhattacharyya distance corresponds to a larger weight; similarly, a sample with large Bhattacharyya distance corresponds to a smaller weight.

For each particle, the observation likelihood function, Equation (20), uses d from Equation (19) to assign a weight defined as

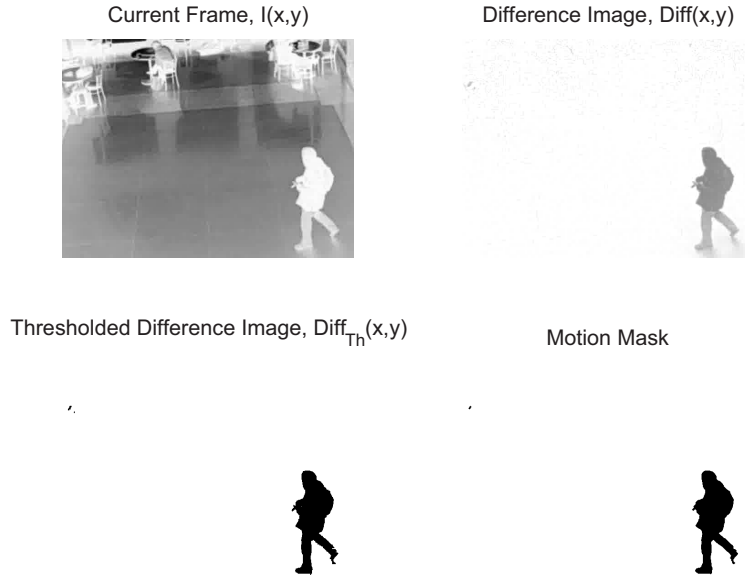
$$w = e^{-\frac{d^2}{2\sigma^2}} \quad (20)$$

The value of the parameter σ of Equation (20) is set dynamically for each frame using the algorithm we proposed previously [6].

5. EXPERIMENTAL RESULTS



(a)



(b)

Figure 2. Moving Object Detection Using Automatic Thresholding.

Figure 2a. is the histogram of the difference image and the adaptive threshold computed automatically associated with it. It takes 3 - 5 iterations on average to compute the threshold for each frame using Rilder and Calvar's method. Most of the pixels' differences are zero because they are the differences between the background pixels of the current frame and the pixels of the background model. This is the case when $D_{Th,t}(x, y) = 0$ in Equation (3). There is another peak at around 110 which comes from the differences between the foreground

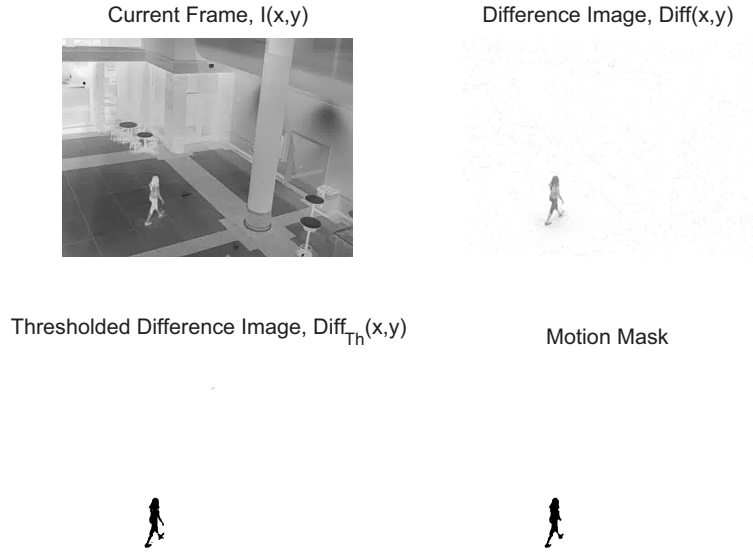


Figure 3. Moving Object Detection.

pixels of the current frame and the background model. These pixels have $D_{Th,t}(x, y) = 255$ in Equations (3). The pixels between 10 and 48 are those with $D_{Th,t}(x, y) = 127$. They are classified as background pixels, however, they are more likely to be foreground pixels compare to those with $D_{Th,t}(x, y) = 0$. Hence background model of those pixels are updated with a lower learning rate.

It is very unlikely to be able to find a threshold that can result in perfect classification, so there will be a lot of isolated noise and holes in the resulting image. Fortunately, this problem can be improved significantly by morphological operations, in our application. Some results of motion masks are shown in Figure 2b and Figure 3.

The result of moving object detection is shown in Figure 2b. The difference image has a lot of non-zero values in the background. They are removed after the difference image is thresholded. The motion mask is obtained after applying morphological operations to the thresholded difference image. The result of another sequence with lower illumination is shown in Figure 3.

Figure 5 shows the $D_{Th,t}(x, y)$ of a sequence with dynamic background (gradual illumination change).

Figure 4 shows the result of the sequence in Figure 2a using traditional background subtraction (without adaptive threshold) for background subtraction. A lot of shadow pixels are classified as foreground pixels.

Figure 6 shows some results of object tracking with initialization of a person walking. The size of the input frames is 352×240 . The blue dot represents the estimated position of the object, and the red ellipse represents the size of the object.

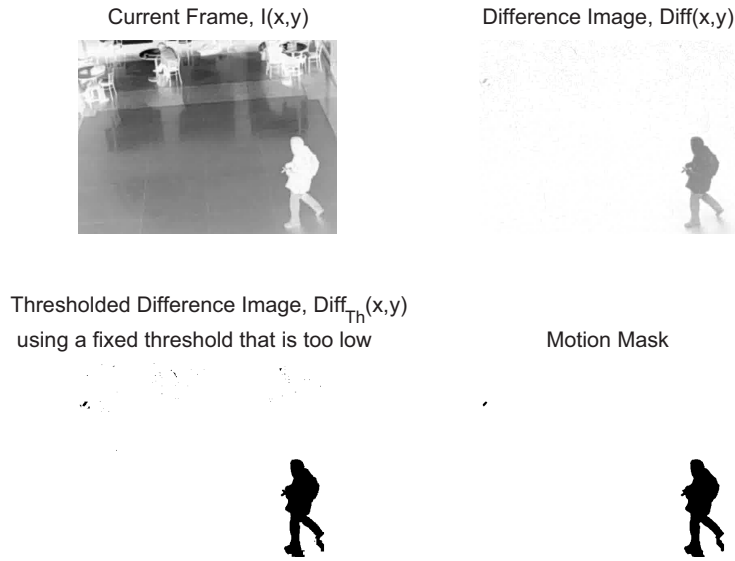


Figure 4. A Threshold That Is Too Low.

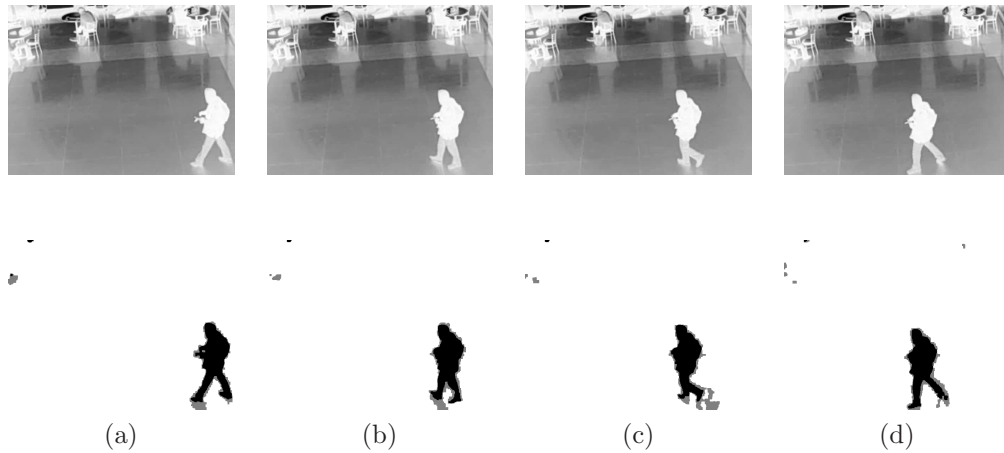


Figure 5. $D_{Th,t}(x,y)$ Of A Sequence With Gradual Illumination Change.

6. CONCLUSIONS

In this paper we introduced an algorithm for moving object detection for object tracking initialization. A technique for background subtraction with adaptive background model update with learning rate depending on pixel differences between the background model and the previous frame is described, and an automatic setting of threshold for foreground-background pixel classification is presented. The new approaches are integrated into our real-time object tracking system to produce a more accurate motion segmentation result which is used to initialize object tracking.

We thank our colleague Nitin Khanna of the Video and Image Processing Laboratory at Purdue University for his suggestions for improvements in this manuscript.

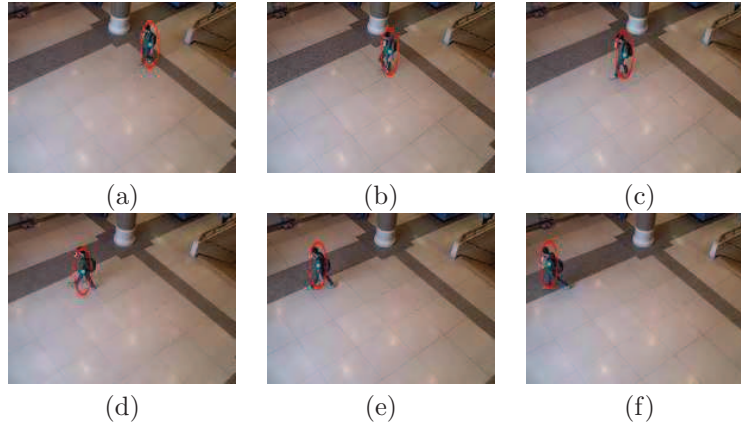


Figure 6. Moving Object Detection And Tracking Results Of A Sequence With A Person Walking, Frame Number (a)20 (b)40 (c)60 (d)80 (e)100 (f)120.

REFERENCES

1. I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8. Washington, DC, USA: IEEE Computer Society, August 2000, pp. 809–830.
2. C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7. Washington, DC, USA: IEEE Computer Society, 1997, pp. 780–785.
3. C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8. Washington, DC, USA: IEEE Computer Society, 2000, pp. 747–757.
4. A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision*. Washington, DC, USA: IEEE Computer Society, 1998, p. 8.
5. D. Meyer and J. Denzler, "Model based extraction of articulated objects in image sequences for gait analysis," *Proceedings of International Conference on Image Processing*, vol. 3. Washington, DC, USA: IEEE Computer Society, 1997, p. 78.
6. K. K. Ng and E. J. Delp, "New models for real-time tracking using particle filtering," *Proceedings of SPIE Visual Communications and Image Processing*, vol. 7257, August 2009.
7. S.-K. Wang, B. Qin, Z.-H. Fang, and Z.-S. Ma, "Fast shadow detection according to the moving region," *Machine Learning and Cybernetics, 2007 International Conference on*, vol. 3, Aug. 2007, pp. 1590–1595.
8. T. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 8, no. 8, pp. 630–632, Aug. 1978.
9. Y. Rui and Y. Chen, "Better proposal distributions: object tracking using unscented particle filter," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2001, pp. 786–793.
10. D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003.
11. K. Deguchi, O. Kawanaka, and T. Okatani, "Object tracking by the mean-shift of regional color distribution combined with the particle-filter algorithm," *Proceedings of the Pattern Recognition, 17th International Conference*, vol. 3. Washington, DC, USA: IEEE Computer Society, 2004, pp. 506–509.
12. K. Nummiaro, E. Koller-Meier, and L. V. Gool, "Color features for tracking non-rigid objects," *Special Issue on Visual Surveillance Chinese Journal of Automation*, 2003, pp. 345–355.