새로운 객체 외곽선 연결 방법을 사용한 비디오 객체 분할

이 호 석†

요 약

비디오에서 움직이는 객체의 외곽선은 객체를 정확하게 분할하기 위하여 매우 중요하다. 그러나 움직이는 객체의 외곽선에는 단락된 외곽선들이 존재하게 된다. 우리는 단락된 외곽선을 연결할 수 있는 새로운 외곽선 연결 알고리즘을 개발하였다. 외곽선 연결 알고리즘은 단락된 외곽선의 말단 픽셀에 사분면을 형성하고 동심원을 구성하면서 반지름 내에서 다른 말단 픽셀을 찾는 탐색을 전진하면서 수행한다. 외곽선 연결 알고리즘은 객체의 외곽선에서 가장 짧게 외곽선을 연결한다. 그리고 시스템은 비디오로부터 배경을 구하여 저장한다. 시스템은 외곽선 연결로부터 객체 마스크를 생성하고, 배경된 저장으로부터 또 하나의 객체 마스크를 생성하여 이 두 개의 객체 마스크를 보완적으로 사용하여 움직이는 객체를 분할한다. 논문의 주요 장점은 정확한 객체 분할을 위한 새로운 객체 외곽선 연결 알고리즘의 개발이다. 제안된 알고리즘은 개발된 새로운 객체 외곽선 연결 알고리즘과 배경 저장을 이용하여 정확한 객체 분할, 다중 객체 분할, 내부에 구멍이 존재하는 객체의 분할, 가느다란 객체의 분할, 그리고 복잡한 배경을 가진 객체를 자동으로 분할하여 보여주었다. 우리는 알고리즘들을 표준 MPEG-4 실험 영상과 카메라로 입력된 실제 영상을 가지고 실험하였다. 제안된 알고리즘들은 매우 효율이 좋으며 펜티엄-IV 3.4GHz CPU에서 평균적으로 QCIF 영상을 1초당 70.20 프레임 그리고 CIF 영상을 1초당 19.7 프레임을 실시간 객체 응용을 위하여 처리할 수 있다.

키워드 : 움직이는 객체의 에지, 움직이는 객체의 외곽선, 외곽선 연결, 두개의 객체 마스크 사용, 자동 객체 분할

Video object segmentation using a novel object boundary linking

Lee Ho Suk†

ABSTRACT

Moving object boundary is very important for the accurate segmentation of moving object. We extract the moving object boundary from the moving object edge. But the object boundary shows broken boundaries so we develop a novel boundary linking algorithm to link the broken boundaries. The boundary linking algorithm forms a quadrant around the terminating pixel in the broken boundaries and searches for other terminating pixels to link in concentric circles clockwise within a search radius in the forward direction. The boundary linking algorithm guarantees the shortest distance linking. We register the background from the image sequence using the stationary background filtering. We construct two object masks, one object mask from the boundary linking and the other object mask from the initial moving object, and use these two complementary object masks to segment the moving objects. The main contribution of the proposed algorithms is the development of the novel object boundary linking algorithm for the accurate segmentation. We achieve the accurate segmentation of moving object, the segmentation of multiple moving objects, the segmentation of the object which has a hole within the object, the segmentation of thin objects, and the segmentation of moving objects in the complex background using the novel object boundary linking and the background automatically. We experiment the algorithms using standard MPEG-4 test video sequences and real video sequences of indoor and outdoor environments. The proposed algorithms are efficient and can process 70.20 QCIF frames per second and 19.7 CIF frames per second on the average on a Pentium-IV 3.4GHz personal computer for real-time object-based processing.

Key Words : Moving Object Edge, Moving Object Boundary, Object Boundary Linking, Using Two Object Masks, Automatic Segmentation

1. 서 론

The MPEG-4 video coding standard introduced the concept of Video Object Plane(VOP) to enable the object-based processing for various applications. Since the introduction of Video Object Plane, the moving object segmentation has become an important research subject and has been receiving considerable amount of attention from many researchers.

※ 본 연구는 2005년도 호서대학교 교내학술특별연구비 지원에 의하여 이루어졌음.

† 정 회 원 : 호서대학교 공과대학 컴퓨터공학 뉴미디어학과 교수

논문접수 : 2005년 9월 14일, 심사완료 : 2006년 5월 17일

We use the edge detection approach for the moving object segmentation because we consider that the constructions of moving object edge and the moving object boundary are the most important things for the video moving object segmentation and because we consider that the Canny edge detector used for the construction of moving object edge has the advantages of accurate edge detection and fast execution for real-time application.

The algorithm [1, 2] shows an approach of obtaining the moving edge map using the absolute frame difference and the Canny edge detection. The algorithm uses the Gaussian convolution filtering to remove the noise. The algorithm obtained the final moving edge by the set union of changed moving edge and the temporarily still moving edge. The algorithm used the logical AND of horizontal and vertical filling-in for the moving object extraction. But the moving edge map and the result of the logical AND of spatial filling-in show incomplete moving object boundaries. The morphological operation has been used to handle this problem. But the extracted moving object still shows inaccurate object boundaries. The algorithm[3] assumed a dominant global motion of the background and used motion detection by the morphological motion filtering or change detection mask, Hausdorff distance, object model initialization and update, horizontal and vertical region filling, and the shortest path algorithm for the moving object edge construction. The moving object edge generally shows accurate object boundary. This algorithm uses the connected component labeling algorithm to remove the noise. But the shortest spanning of pixels in the object boundary of binary object model for the object boundary replacement does not work if any one of the pixels in the object boundary is missing. Furthermore, the edge pixel additions by the Hausdorff distance matching may not be sufficient in the new binary object model when the broken edge region is relatively wide in the initial object model. The approach [4] uses the background registration technique for the moving object segmentation. The basic idea of this algorithm is the change detection but the algorithm focuses on the registered background rather than the foreground, because the background is observed to be more reliable than the changing foreground in producing the change detection mask. But the object boundary is not accurate. This approach uses connected component labeling algorithm to remove the noise. The approach[5] uses the same background registration technique but uses the predictive watershed algorithm to generate the moving object edge. But the predictive watershed algorithm has to handle error propagation and can suffer from over-segmentation. And the overall algorithm execution is slow. A contour tracing algorithm which can trace a hole inside an object is explained in [6] but the algorithm assumes a completely linked object boundary beforehand. The method in [7] explains an interesting shape error concealment technique using the Bézier curve that works for a binary alpha plane of a single smooth object boundary. The approach in [8] explains a shape information concealment method using the MAP estimator for missing shape blocks in the binary alpha plane.

Object-oriented coding has been introduced in [9]. In [10][11], polygon/B-spline approximation approaches have been used for the shape coding for rate-distortion optimization. These approaches focus on the object-oriented coding once the object mask is given. The paper [12] explains the Hausdorff distance matching between the objects using the distance transform, ranking, and partial matching. The paper [3] explains the use of Hausdorff distance matching in the moving object edge matching for model update. But the Hausdorff distance computation for object matching is time-consuming. The papers [1][2] also use a kind of distance computation for the moving object edge construction. But we find in the experiment that it is impossible to use the edge pixels of the current frame by the Canny edge detector for moving object edge construction when the matched region of the moving object edge is a wide broken edge region, because we find that the distance between edge pixels in a wide broken edge region cannot be computed using the distance computation method.

The broken boundaries of moving object is caused by the edge detection failure due to the reasons such as the instantaneous halt of the moving object, a color similarity coincidence between the background and the moving object, illumination variation, and a camera noise. The object or the part of the object sometimes halts temporarily between the consecutive frames and this causes the edge detection failure. The color similarity coincidence means the color similarity between the background and the foreground moving object. The illumination variation means the non-uniform or varying intensity levels of illumination. These factors can cause the edge detection failure. The noise is the inherent problem in the moving object segmentation and the noise must be eliminated. We use the connected component labeling algorithm to eliminate the noise from the image.

The broken boundaries make it difficult to obtain the accurate moving object segmentation. We propose that

the broken boundaries should be supplemented with the boundaries from other boundary sources and linked by the boundary linking. The other boundary sources include the previous moving object edge, the edge of current frame by the Canny edge detector, and the background edge of current frame. We use an edge set union operation and the edge tracing and copying operation for the construction of moving object edge. The boundary linking is defined as the linking of two terminating pixels in the broken boundary. The boundary linking algorithm forms a quadrant around the terminating pixel and searches for other terminating pixels to link in concentric circles clockwise within a search radius. The algorithm does not create a cycle and guarantees the shortest distance linking. The linking algorithm tries to link the broken boundaries robustly within a search radius. But the linking algorithm does not try to link the broken boundaries which is wider than the search radius, thus for some images with a relatively wide broken boundary there might remain broken boundaries after the boundary linking. But for the most images, we can obtain a completely linked moving object boundary after the boundary linking. Later, we use the object mask to deal with the relatively wide broken boundary.

The main advantage of the proposed algorithm is the accurate segmentation of moving object using the object boundary linking. And we achieved the segmentation of multiple moving objects, the segmentation of thin objects, the segmentation of the object which has a hole within the object, and the segmentation of moving objects in the complex background using the object boundary linking and the background automatically. We also suppress the moving cast shadow using the Roberts gradient operator. We use standard MPEG-4 test image sequences such as *Akiyo*, *Hall monitor*, *Claire*, *Mother&Daughter*, *Silence*, *Weather*, and *Grandma* to experiment the system. Particularly, we use *Akiyo* 18^{th} frame, *Hall monitor* 54^{th} frame, *Claire* 10^{th} frame, and *Mother&Daughter* 45^{th} frame to test the algorithm. The other frames of these standard sequences are used for more demonstrations. We also use real video sequences to experiment the system. The real video sequences include a sequence with a hole called Junki, two thin object sequences, and walking student image sequences in the complex background. The Junki sequence will be shown in section 2.4.3 and the other sequences will be shown in section 3.2.

We assume that the camera and the background are stationary in the algorithm development. The background global motion and camera motion can be represented by an affine transformation and thus can be compensated by a motion estimation and compensation[14].

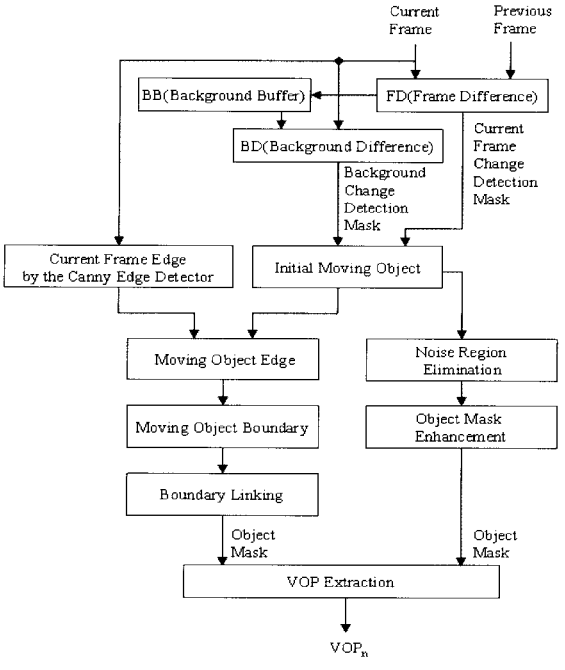
The organization of paper is as follows. Section 2 gives the detailed explanation of the proposed moving object segmentation algorithm from the moving object edge construction to the video object plane extraction. Section 3 shows the experimental results of the algorithm. The section 4 draws the conclusion.

2. Moving object segmentation

2.1 Overall architecture

We consider the moving object segmentation as a process of development from the initial moving object to the final moving object segmentation. The initial moving object is actually the background change detection mask and it is used as the initialization of moving object for the segmentation. We use the background change detection mask computed from the current frame and the registered background because it shows a more accurate moving object edge than the foreground change detection mask computed from the two consecutive image frames. The initial moving object is processed in two separate ways but the two object masks produced as shown in (Fig. 1) are used together in a complementary way to produce the final segmentation of moving object. To explain it in more detail, one of the two complementary object masks is produced from the moving object boundary after the moving object boundary linking and the other object mask is produced from the initial moving object after the noise elimination and the object mask image enhancement. The constructed moving object boundary, however, shows broken boundaries so we try to link the broken boundaries to obtain a completely linked object boundary. We use the connected component labeling algorithm to eliminate the noise and we use the binary morphological opening and closing operation to enhance the object mask. And finally we use the two complementary object masks for the accurate segmentation of moving object. (Fig. 1) shows the overall system architecture for the automatic segmentation of moving object. VOP means the video object plane.

In (Fig. 1), FD means the pixel-based frame difference between the current frame and the previous frame and generates the current frame change detection mask. BD means the frame difference between the current frame and the registered background and generates the background change detection mask. BB indicates the backgr ound buffer which is used to register the background. The current pixel value determined by the registration



(Fig. 1) Overall architecture of the automatic moving object segmentation

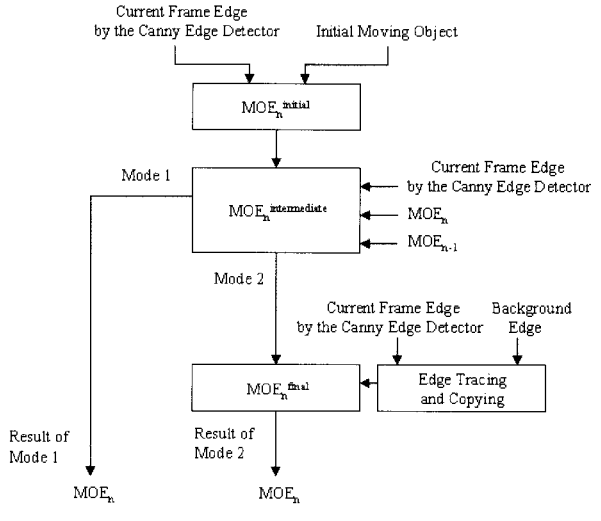
count is sent to BB for the background registration. We use the current frame change detection mask as the initial moving object in the beginning period and then we use the background change detection mask as the initial moving object after we register the background. The moving object edge is constructed using the edge of current frame by the Canny edge detector and the initial moving object. We scan the constructed moving object edge horizontally and vertically and extract the moving object boundary. Then, we examine the object boundary and set the terminating pixels in the broken boundaries and perform the space-oriented boundary linking to obtain a completely linked moving object boundary. We construct the object mask of first type using this result. We also eliminate the noise and enhance the initial moving object and construct the object mask of second type. We, then, extract the moving object of the current frame using these two complementary object masks.

2.2 Automatic segmentation

The system can segment the moving object automatically from the video sequence. The system stores the default threshold values for the change detection and uses the means and the standard deviations for the selection of the appropriate threshold value for the automatic change detection. The system computes the means and the standard deviations of the frames and also of the frame differences. The system considers the deviations of the frame difference and the means of the frames of the video sequence in a binary way. If the deviation of the frame difference and the mean of the frame are low, the system selects the lowest threshold value. For example, the system can obtain 0.8~4.5 as the range of deviation and 93 as the value of mean for *Akiyo* and selects 0.1 as the threshold value for the automatic change detection.

2.3 Moving object edge construction

We construct the moving object edge progressively from the initial moving object edge through the intermediate and to the final moving object edge. (Fig. 2) shows the block diagram of the progressive construction of the moving object edge. MOE_n and MOE_{n-1} mean the current and the previous moving object edge respectively.



(Fig. 2) Moving object edge construction

The initial moving object edge is constructed using the edge of current frame by the Canny edge detector [13] and the initial moving object. We use the threshold value 5 for the Canny edge detector. The threshold value has been determined by experimentation. The intermediate moving object edge is constructed using the edge of current frame by the Canny edge detector and the previous moving object edge to link the broken boundaries which are mostly caused by the instantaneous halt of the moving object between the consecutive frames. After this stage, the moving object edge construction is divided into two modes. In mode one, we regard the intermediate moving object edge as the final result. We can process the image sequences such as *Weather* and *Hall monitor* in mode one and can obtain a sufficient moving object edge for the object boundary extraction. In mode two, we

<Table 1> Threshold value

Sequence Type	Sequence	Threshold of Gramian Matrix Approach	Threshold of Significance Test Approach
MPEG-4 Test Image Sequence	Akiyo(QCIF)	0.1x10-14	0.1
	Claire(QCIF)	0.6x10-7	2.0
	Hall monitor(QCIF)	0.5x10-4	200.0
	Mother & Daughter(QCIF)	0.5x10-7	3.0
Real Video	Junki(QCIF)	0.1x10-5	30.0
	A thin tree branch(QCIF)	0.1x10-2	30.0
	Walking students(352x240)	0.1x10-2	400.0

construct the final moving object edge using the edge tracing and copying. The edge tracing and copying uses the edge of current frame by the Canny edge detector and can use the background edge of current frame. We can process *Akiyo* which has a high quality and *Claire* in which the background edge does not touch the object edge in mode two.

2.3.1 Change detection

We use the absolute pixel difference as a statistical test for change detection. Under the H_0 null hypothesis that there is no change in the current pixel, the pixel difference can be modeled using a zero-mean Gaussian distribution N(0, σ) with variance σ^2. The following equation (1) shows the Gaussian distribution function of absolute pixel difference[14, 15].

$$p(PD \mid H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{PD^2}{2\sigma^2}\right) \quad (1)$$

where PD means the absolute pixel difference and H_0 means the null hypothesis. The "changed" and "unchanged" decision can be determined by the significance test of equation (2).

$$\alpha = prob(|PD| > T \mid H_0) \quad (2)$$

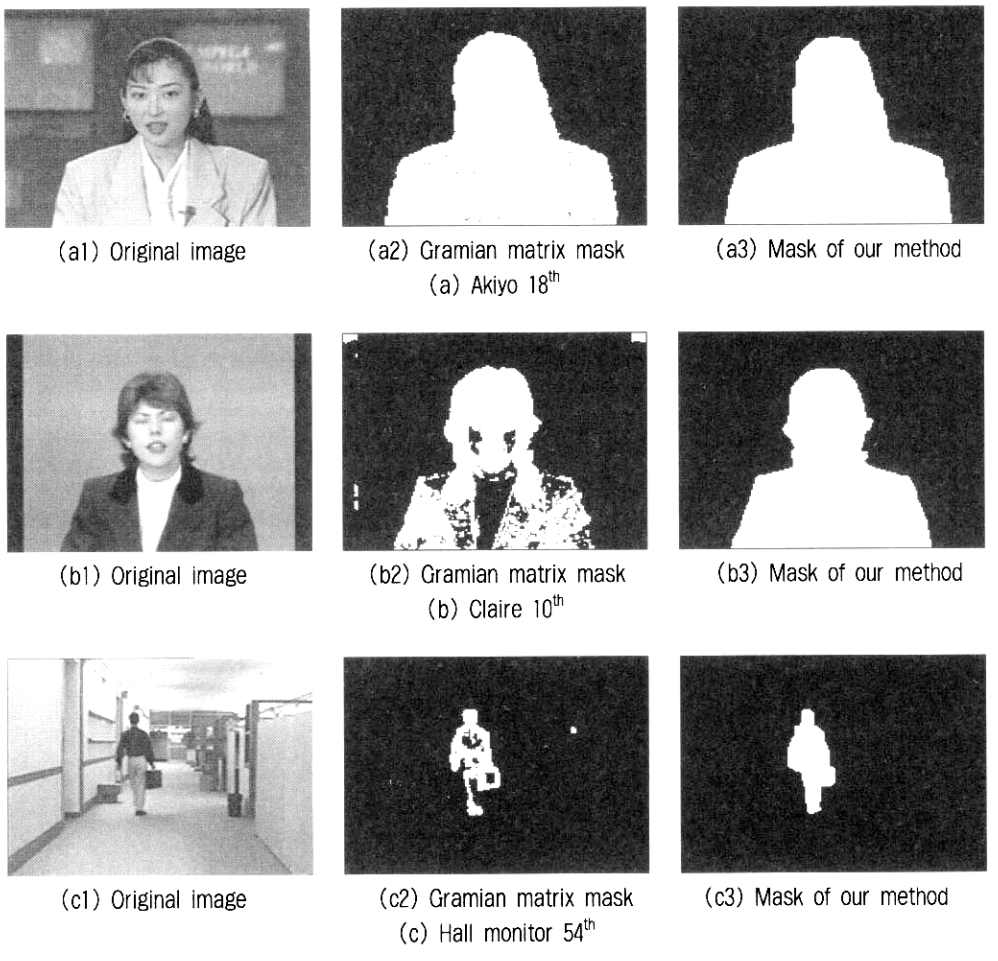
where α is the significance level and T is the threshold value. The significance level α is determined by experimentation[15]. The threshold value for the change detection is global for all the images in the image sequence. For example, we can find the threshold value 200.0 for all images in *Hall monitor* sequence by the significance test.

In [16], the Gramian matrix approach for change detection is explained. The approach mentions that the linear dependence decision[16, 17] by the computation of Gramian matrix determinant can be used for the detection of moving objects. We implemented the Gramian matrix approach using one reference vector u and two vectors v and w using equation (4) and (5) of [16]. We use 3x3 pixels as the region of support for the vector and replace the center pixel of a region of support by the vectors. We tested this approach using (a) *Akiyo*, (b) *Claire*, and (c) *Hall monitor*. We find the global threshold value for the decision of "changed" and "unchanged" state of the pixel for each image sequence in the Gramian matrix approach.

<Table 1> below shows the threshold values for the Gramian matrix approach and the significance test approach for each image sequence. The threshold value for *Akiyo* sequence is the lowest in each approach because the *Akiyo* sequence has the lowest noise. The threshold value for *walking students* real video is the highest in each approach because the *walking students* real video has the highest noise.

In (Fig. 3) below, we show the original image, the mask generated by the Gramian matrix approach named Gramian matrix mask, and the mask of our approach named mask of our method for the object segmentation. We can see that the mask of our approach is better than the mask generated by the Gramian matrix approach.

In the Gramian matrix approach, we find in our experiment that when we lower the threshold to eliminate the noise, the noise in the background and in the change detection mask generally decreases but the noise within the change detection mask is not completely eliminated and the shape of the change detection mask shrinks. We use double precision floating-point operations to compute the determinant of the Gramian matrix. The determinant computation consumed much of system resources and the computation was slow. Thus, we see that the change detection mask generated by the Gramian matrix approach also needs post-processing to eliminate the noise in the object mask and in the background and to link the broken object boundaries. And this approach may also need optimization for fast execution.



(Fig. 3) Comparison of Gramian matrix mask and our mask : (a) Akiyo 18th, (b) Claire 10th, (c) Hall monitor 54th

2.3.2 Background registration

We register the background using the stationary background filtering technique[3, 4]. We use the background to generate the background change detection mask and it is used to obtain the initial moving object. The followings are the background registration expressions.

$$Static\_buffer_n(x,y) = \begin{cases} Static\_buffer_{n-1}(x,y)+1 & if \quad PD_n(x,y) \quad is \quad unchanged \\ 0 & otherwise \end{cases}$$

$$Background\_buffer_n(x,y) = \begin{cases} Current\_frame_n(x,y) & if \quad Static\_buffer_n(x,y) >= \\ & \quad \text{Re}gistration\_count \\ No \quad operation & otherwise \end{cases} \quad (3)$$

$Static\_buffer_n(x,y)$ increments the count value, which indicates the number of frames the pixel at the coordinate position (x,y) remains unchanged in the frame differences. $PD_n(x,y)$ means the pixel difference between the current frame and the previous frame. $Current\_frame_n(x,y)$ means the current frame and when the value of $Static\_buffer_n(x,y)$ stores the value of $Current\_frame_n(x,y)$ and registers the background. We find the global registration count value 4 for the background registration by extensive experimentation.

2.3.3 Moving object edge

We define the edge as the pixels constituting the edge in the image. The initial moving object edge is constructed as follows. MOE_n means the current moving object edge.

$$MOE_n^{\ initial} = \{e = e_1 \,\&\, e_2 \mid e_1 \in E_n^{\ c}, \ \ e_2 \in IMO\} \quad (4)$$

(4) Here, $E_n^{\ c}$ is the edge of the current frame by the Canny edge detector and IMO is the initial moving object. e, e_1, and e_2 mean the edges. The operation & is the logical AND operation. The intermediate moving object edge is constructed as follows.

$$MOE_n^{\ intermediate} = \{e = e_1 \,\&\, e_2 \mid e_1 \in E_n^{\ c}, \ \ e_2 \in MOE_n \| \ e_2 \in MOE_{n-1}\} \quad (5)$$

Here, MOE_n is the current moving object edge and MOE_{n-1} is the previous moving object edge. The || operation means the logical OR operation. It means the union of the edges of MOE_n with the edges of MOE_{n-1} to link the broken boundaries in MOE_n, which are mostly caused by the instantaneous halt of the moving object. The final moving object edge is constructed as follows.

$$MOE_n^{final} = \{e = e_1 \,||\, e_2 \mid \; e_1 \in MOE_n^{intermediate}, \; e_2 \in E_n^c \;\&\; e_2 \notin E_n^b\} \tag{6}$$

Here, E_n^b is the background edge. The logical OR operation takes the union of the edges of the current frame by the Canny edge detector excluding the background edges with the intermediate moving object edges in order to obtain the final moving object edge which is linked more. We call this edge tracing and copying. We find the edge tracing and copying quite effective when the image has a high quality such as *Akiyo* or when the background edge is separated from the object edge such as *Claire*. We can see the results of *Claire* in (Fig. 5) (c3) and (c4). We can compare (Fig. 5) (c2) with (c3) and can see the effectiveness of the edge tracing and copying.

2.3.4 Edge tracing and copying

The edge tracing and copying is performed in mode two of the moving object edge construction. For *Akiyo*, the edge tracing and copying uses the edge of current frame E_n^c by the Canny edge detector and the background edge of current frame E_n^b. The followings are the steps of the algorithm.

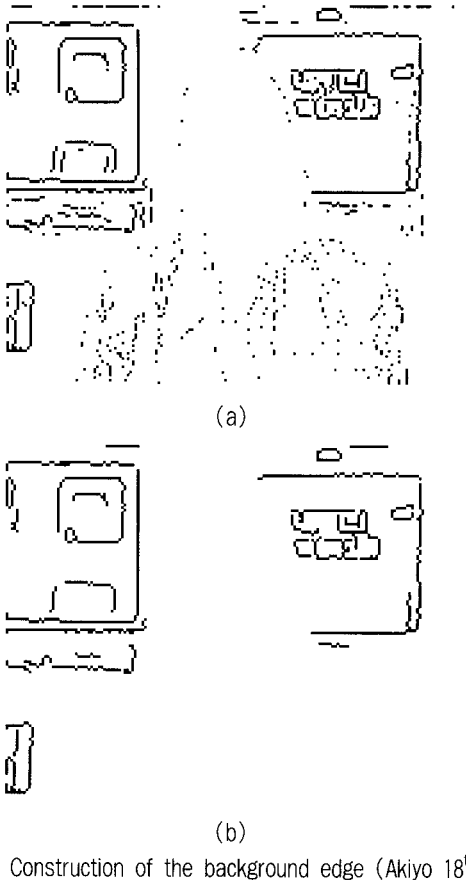
(step1) Construct the background edge $E_n^b(x,y)$ automatically using the following equation.

$$E_n^b(x,y) = E_n^c(x,y) - MOE_n^{intermediate}(x,\, y) \tag{7}$$

But the background edge image still contains some remaining object pixels as shown in (Fig. 4) (a). We use the connected component labeling algorithm[3, 18, 19] to eliminate these remaining object pixels in the background edge image and obtain a clear background edge image as shown in (Fig. 4) (b).

(step2) Scan $MOE_n^{intermediate}(x,y)$ horizontally and locate the first pixel encountered .

(step3) Search the neighboring pixels of the pixel found in **(step2)** in 8-neighborhood[18], which surrounds the pixel found immediately, to locate the terminating pixel, which indicates the location of the broken edge, in $MOE_n^{intermediate}(x,y)$ to start the edge tracing.



(Fig. 4) Construction of the background edge (Akiyo 18th) : (a) Background edge image with remaining object pixels, (b) Background edge image without remaining object pixels

(step4) Check the existence of a pixel in $E_n^c(x,y)$ at the corresponding coordinate position of the terminating pixel found in **(step3)**. If there exists a pixel at the corresponding coordinate position, then perform the edge tracing along the edge line of $E_n^c(x,y)$ as long as the pixel of $E_n^c(x,y)$ under the edge tracing does not encounter the background edge in $E_n^b(x,y)$ constructed in (step1). And copy the traced edge of $E_n^c(x,y)$ into $MOE_n^{intermediate}(x,y)$ to construct $MOE_n^{final}(x,y)$. If the edge tracing encounters the background edge, then the edge tracing stops.

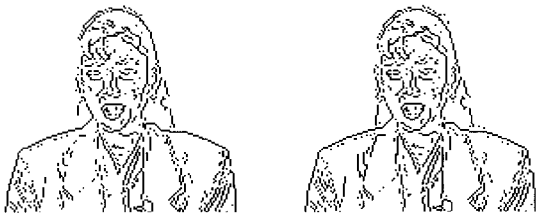
(step5) Return to **(step2)** and continue.

The following (Fig. 4) shows the moving object edge construction results.

(Fig. 5) shows the moving object edge of (a) *Hall monitor* 54th, (b) *Akiyo* 18th, and (c) *Claire* 90th, 10th. The left column of (a) and (b) were produced using the current frame change detection mask as used in [1][3] and the right column of (a) and (b) were produced using the background change detection mask. We can see the more accurate moving object edge using the background change detection mask. For (a) and (b), we use the intermediate moving object edge as the final result. (Fig. 5)

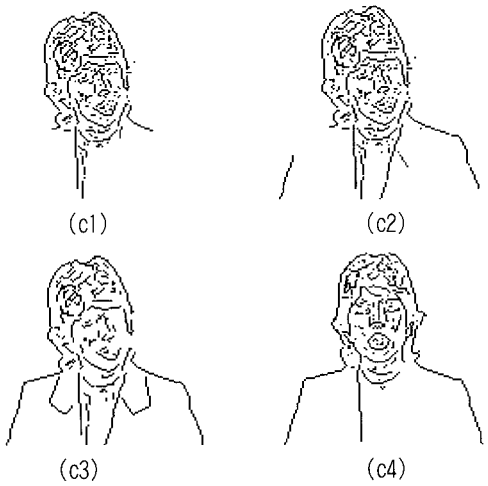
(a1) (a2)

(a) Hall monitor 54^{th} : (a1) $MOE^{intermediate}$ using the current frame difference, (a2) $MOE^{intermediate}$ using the background frame difference



(b1) (b2)

(b) Akiyo 18^{th} : (b1) $MOE^{intermediate}$ using the current frame difference, (b2) $MOE^{intermediate}$ using the background frame difference



(c1) (c2)

(c3) (c4)

(c) Claire 90^{th}, 10^{th} : (c1) $MOE^{initial}$ 90^{th}, (c2) $MOE^{intermediate}$ 90^{th}, (c3) MOE^{final} 90^{th}, (c4) MOE^{final} 10^{th}

(Fig. 5) Moving object edge : (a) Hall monitor 54^{th}, (b) Akiyo 18^{th}, (c) Claire 90^{th}, 10^{th}

(c) *Claire* shows the process of moving object edge construction using the edge tracing and copying. But for *Claire*, the construction of the background edge is not necessary because the background edge and the object edge are clearly separated. (Fig. 5) (c4) is shown for another demonstration.

2.4 Moving object boundary construction

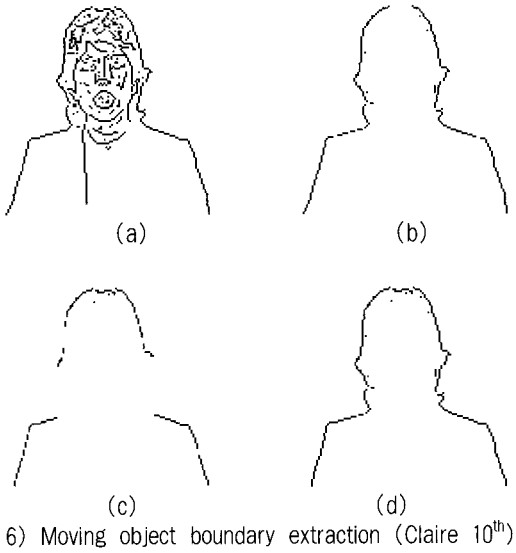
The completely linked moving object boundary is very important for the accurate segmentation of moving object. However, the extracted moving object boundary shows broken boundaries caused by the edge detection failures. We try to link these broken boundaries to construct the accurate moving object boundary using a novel boundary linking algorithm.

2.4.1 Moving object boundary extraction

First, we eliminate the tiny aggregate of isolated noise pixels from the moving object edge image. Then, we scan the moving object edge horizontally and obtain the object boundary as shown in (Fig. 6) (b). And we scan the object vertically and obtain the object boundary as shown in (Fig. 6) (c). We combine all the boundary pixels and extract the moving object boundary as shown in (Fig. 6) (d). The following is the expression for the moving object boundary extraction.

$$Moving\_object\_boundary_n(x,y) = \begin{cases} 1 & if \quad horizontal\_boundary\_pixel_n(x,y)=1 \\ & \quad || \quad vertical\_boundary\_pixel_n(x,y)=1 \\ 0 & otherwise \end{cases} \tag{8}$$

$Moving\_object\_boundary_n(x,y)$ is the result of the moving object boundary extraction. $Horizontal\_boundary\_pixel_n(x,y)$ and $vertical\_boundary\_pixel_n(x,y)$ mean the object boundary pixels extracted by the horizontal and vertical scanning respectively. The || is the logical OR operation. We set the first pixel and the last pixel to value 1 in the horizontal and vertical scanning to indicate that this is the boundary pixel. The following (Fig. 6) shows the moving object boundary extraction of *Claire*. The extracted moving object boundary, however, can have broken boundaries.



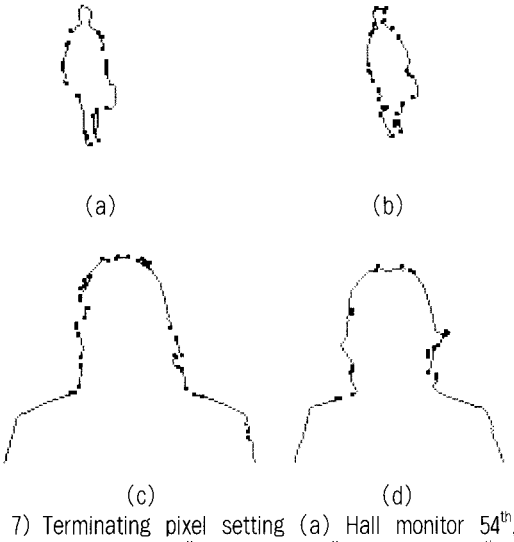
(a) (b)

(c) (d)

(Fig. 6) Moving object boundary extraction (Claire 10^{th}) : (a) Moving object edge, (b) Object boundary by horizontal scanning, (c) Object boundary by vertical scanning, (d) Moving object boundary

2.4.2 Moving object boundary linking

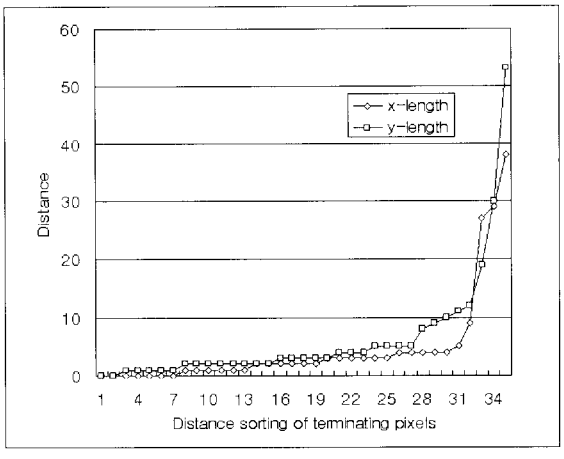
We examine the moving object boundary image horizontally and locate the terminating pixels at the two final end pixels of the broken boundary line, one by a downward tracing and the other by an upward tracing. (Fig. 7) (a) and (b) show the *Hall monitor* 54th and 69th with terminating pixels, (c) shows the *Akiyo* 18th with terminating pixels, and (d) shows the *Claire* 10th with terminating pixels. The actual terminating pixel is an ordinary single pixel but we enlarge it to distinguish it from other boundary pixels.



(Fig. 7) Terminating pixel setting (a) Hall monitor 54th, (b) Hall monitor 69th, (c) Akiyo 18th, (d) Claire 10th

Edge linking algorithms using various techniques are explained using examples in [19]. An edge linking algorithm considering three types of break points within a window is explained in [20]. The algorithm performs edge linking within a window over the image.

We propose a boundary linking algorithm which is a space-oriented geometric algorithm considering the boundary pixel linking direction only. The boundary linking algorithm links the broken boundaries by inserting pixels from one terminating pixel to the other terminating pixel. The purpose of this algorithm is to separate the moving object clearly from the background and to obtain a more accurate object boundary than the approach using the morphological operation to link the broken boundary [1, 4]. This algorithm is a pixel search algorithm looking for pixels to link along the object boundaries. The algorithm first considers the pixel linking direction. The algorithm forms a quadrant around the terminating pixel and searches for other terminating pixels to link in concentric circles clockwise within a search radius from the terminating pixel in the forward direction. The specific search radius can be determined automatically by the computation of differences of the sorted distances between the terminating pixels. The following (Fig. 8) shows the graph of sorting of distances between the terminating pixels of *Akiyo* 18th QCIF image.



(Fig. 8) Distance sorting of terminating pixels(Akiyo 18th QCIF)

We try to find the greatest difference which occurs first by computing the differences of the sorted distances between the terminating pixels. The greatest difference can be used for the selection of the search radius because the distance beyond the greatest difference can mean a long distance between the terminating pixels and a long distance can imply a wrong boundary linking in the object boundary. For example, we see that we can select the distance 9 as the search radius for *Akiyo* 18th QCIF image automatically from (Fig. 8).

The algorithm always guarantees the shortest distance linking in the object boundary because the algorithm always tries to link the terminating pixel which is the shortest in distance from the terminating pixel during the search within the search radius. And the algorithm does not create a cycle during the boundary linking because the algorithm always searches for other terminating pixels to link in the forward direction from the terminating pixel. We see that the boundary linking algorithm forms the object boundary accurately in the broken boundaries because the terminating pixels used for the boundary linking in the broken boundaries exist on the object boundary.

We give a more detailed explanation. We assume a virtual pixel around the terminating pixel. The virtual pixel is assumed in the forward direction of the terminating pixel by considering the slope of the terminating pixel. The purpose of a virtual pixel is to predetermine the general direction toward which the broken boundary

will be linked from the terminating pixel.

However, it is usually seldom the case that every terminating pixel ought to be linked only in the direction of the virtual pixel, because the pixel distribution can vary significantly according to the characteristics of the input image. Thus, we form a quadrant for a more general search range around the terminating pixel $T_0 = (0,0)$ in the direction of the virtual pixel $V_0 = (V_x, V_y)$. The algorithm begins by positioning the terminating pixel at the coordinate origin. The boundary linking operation is now carried out within the quadrant from the terminating pixel. First, we compute the angle between the terminating pixel and the virtual pixel. If we let θ_v be the angle between T_0 and, V_0 then we compute

$$\theta_v = \cos^{-1} \frac{V_x}{\sqrt{V_x^2 + V_y^2}} * \pi / 180 \tag{9}$$

and we search for another terminating pixel T_1 to link around the terminating pixel T_0 within the quadrant in concentric circles clockwise within the search radius. Now, if we find another terminating pixel $T_1 = (T_x, T_y)$ around the current terminating pixel T_0. We compute the angle θ_s between T_0 and T_1.

$$\theta_s = \cos^{-1} \frac{T_x}{\sqrt{T_x^2 + T_y^2}} * \pi / 180 \tag{10}$$

The condition and operation of object boundary linking can be summarized in a procedure as follows.

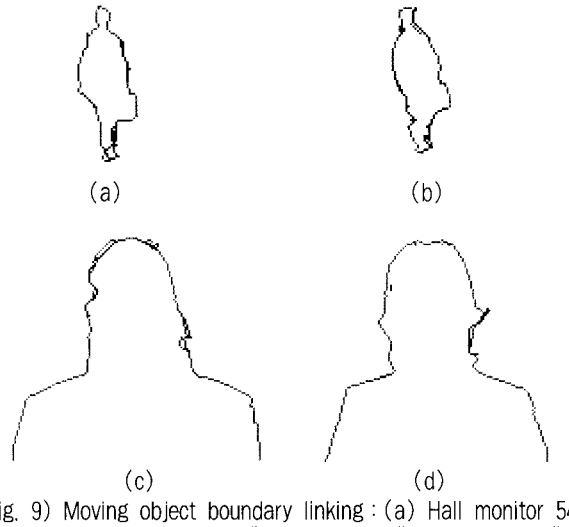
```
procedure object_boundary_linking( )
{
  if (2π + θv − π/4 ≤ θs ≤ 2π + θv + π/4) then {
      link T0 = (0,0) to T1 = (Tx, Ty)
  } else expand the quadrant to half plane and
      perform object boundary linking from T0 = (0,0)
}
```

If the angle θ_s of the terminating pixel T_1 is within the range as computed above, the boundary linking is carried out from the terminating pixel T_0 to the terminating pixel T_1. If the angle θ_s is not within the range, we expand the quadrant to the half plane and perform the boundary linking iteratively from the terminating pixel $T_0 = (0,0)$.

We find other terminating pixel to link within the search radius around the starting terminating pixel within 5 iterations most of the time in the experiment. In other cases, the boundary linking algorithm searches for other terminating pixel to link in the expanded half plane. We continue the boundary linking operation until there remain no more terminating pixels to link. (Fig. 9) shows the constructed moving object boundary of (Fig. 7) after the object boundary linking.

The boundary linking algorithm tries to link the broken boundary robustly within the search radius. But the boundary linking algorithm does not link the terminating pixel which is located farther than the search radius, because it can link the wrong terminating pixel in trying to link the terminating pixel located farther than the search radius. For example, the search radius 9 can be selected for *Hall monitor* 54th QCIF image and the search radius 14 can be selected for the thin tree branch 67th QCIF image.

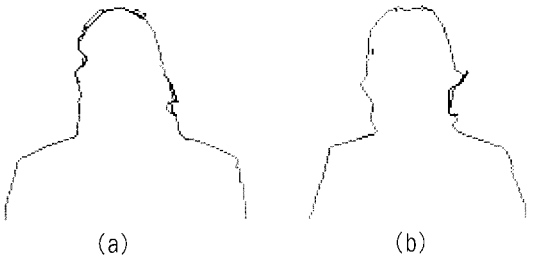
The algorithm locates 20 terminating pixels as shown previously in (Fig. 7) (a) in this section and performed 12 boundary linkings as shown in (Fig. 9) (a) *Hall monitor* 54th. In (Fig. 6) (c) *Akiyo* 18th moving object edge, the algorithm locates 35 terminating pixels and performed 23 boundary linkings. We can see several broken boundaries in (Fig. 7) (a) and we can see that they are completely linked in (Fig. 9) (a). We can also see broken boundaries in (Fig. 7) (b), (c), and (d) and we can also see the corresponding completely linked object boundary in (Fig. 9) (b), (c), and (d) respectively. However, we can see two object boundaries on the same boundary where the boundary linking is performed, one by the moving object edge itself, and the other by the boundary linking, but these seemingly double object boundaries do not cause a problem. We map the object texture to the outermost boundary.



(Fig. 9) Moving object boundary linking : (a) Hall monitor 54th, (b) Hall monitor 69th, (c) Akiyo 18th, (d) Claire 10th

2.4.3 Moving object boundary connection at the bottom

We construct the moving object boundary as shown in (Fig. 9). But the moving object shows boundary disconnection at the bottom where the moving object boundary touches the image rectangular frame. (Fig. 9) (c) and (d) show the disconnected regions of moving object boundary at the bottom. This disconnection naturally arises because there is inherently no object boundary in this region. (Fig. 10) (a) and (b) show the connected moving object boundary of (Fig. 9) (c) and (d).



(a) (b)

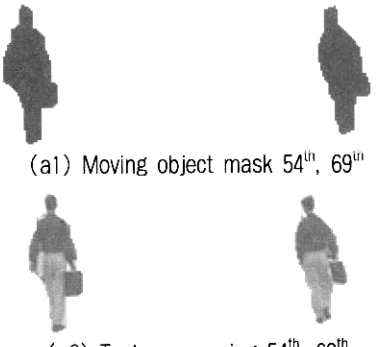
(Fig. 10) Moving object boundary connection at the bottom : (a) Akiyo 18th, (b) Claire 10th

2.5 Video object plane extraction

We perform the moving object extraction using two object masks. One object mask is constructed by the boundary linking and is called the object mask of first type and the other object mask is made from the initial moving object itself and is called the object mask of second type. We can use the object mask of first type for the image sequence in which only a single object is present showing only the head-and-shoulder type motion such as *Akiyo* and *Claire*. The advantage of the object mask of first type is the accurate moving object boundary. But it cannot handle the case of an unlinked broken boundary which could exist in the relatively wide broken boundary and the segmentation of the object which has holes within the object. The object mask of second type can handle these situations but it produces the coarse object boundary. Therefore, we use these two object masks together in a complementary way to obtain the accurate moving object segmentation and to segment the object which has holes within the object.

2.5.1 Object extraction using the object mask of first type

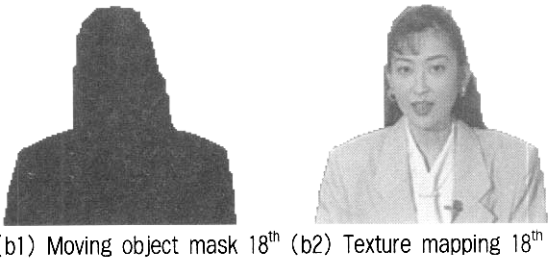
(Fig. 11) shows the moving object extraction using the object mask of first type. (Fig. 11) (a1) is the moving object mask of first type for *Hall monitor* 54th, 69th and (a2) is the texture mapping result. (Fig. 11) (b1) is also



(a1) Moving object mask 54th, 69th

(a2) Texture mapping 54th, 69th

(a) Hall monitor 54th, 69th



(b1) Moving object mask 18th (b2) Texture mapping 18th

(b) Akiyo 18th

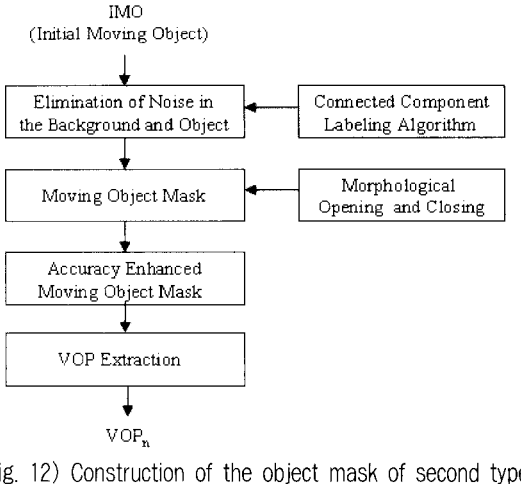
(Fig. 11) Moving object extraction(using the object mask of first type) : (a) Hall monitor 54th, 69th, (b) Akiyo 18th

the moving object mask of first type for *Akiyo* 18th and (b2) is the texture mapping result.

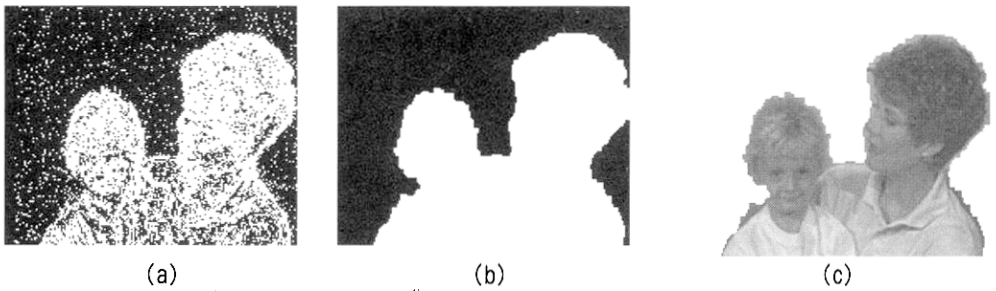
The figures in (Fig. 11) (a2) show the accurate moving object boundaries in the upper body region and (b2) shows a quite accurate object boundary

2.5.2 Object extraction using the object mask of second type

(Fig. 12) shows the block diagram of the construction of the object mask of second type and the moving object extraction using the object mask of second type. The ini-



(Fig. 12) Construction of the object mask of second type



(a) (b) (c)

(Fig. 13) Moving object extraction (Mother&Daughter 45th) : (a) Initial moving object, (b) Object mask of second type, (c) Texture mapping

tial moving object is actually the background change detection mask. The connected component labeling algorithm eliminates noise in the background and the moving object mask. The morphological operation is applied to the object mask and produces a boundary enhanced moving object mask. VOP means the video object plane.

(Fig. 13) shows the moving object extraction with an example using the object mask of second type. (Fig. 13) (a) shows the initial moving object with the salt-and-pepper noise, and (b) shows the object mask of second type with a clear background and an object image, in which the noise is eliminated using the connected component labeling algorithm and the boundary enhanced by the binary morphological opening and closing operation with a 3x3 structuring element, and (c) shows the texture mapping result. The morphological opening operation generally smoothes the contour of an object and eliminates thin protrusions and morphological closing operation also smoothes the contours by filling gaps in the contour and eliminates tiny holes [18].

(Fig. 14) shows the moving object extraction results of *Hall monitor* 54th, 69th and *Akiyo* 18th using the object mask of second type. The extracted moving objects show the coarse object boundaries.

(a1) Moving object mask 54th, 69th

(a2) Texture mapping 54th, 69th
(a) Hall monitor 54th, 69th

(b1) Moving object mask 18th (b2) Texture mapping 18th
(b) Akiyo 18th

(Fig. 14) Moving object extraction (using the object mask of second type) : (a) Hall monitor 54th, 69th, (b) Akiyo 18th

2.5.3 Object extraction using both object masks

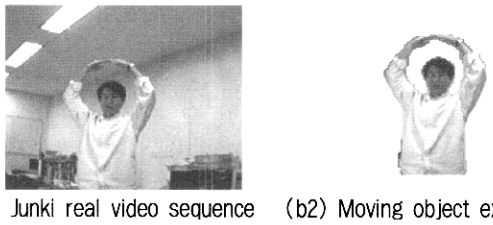
We use the two object masks together to obtain the accurate moving object boundary and to segment the object which has holes within the object. The following is the expression for the moving object extraction using both object masks.

$$Moving\_object_n(x,y) = \begin{cases} Current\_frame_n(x,y) & if \quad object\_mask\_first_n(x,y)=1 \\ & \quad \& \quad object\_mask\_second_n(x,y)=1 \\ 0 & otherwise \end{cases} \tag{11}$$

$Moving\_object_n(x,y)$ means the extracted moving object. The $object\_mask\_first_n(x,y)$ means the object mask of first type and $object\_mask\_second_n(x,y)$ means the object mask of second type. The & is the logical AND operation. The moving object is extracted from the current frame if and only if both object masks are equal to 1. Thus we can handle the wide broken boundary region which might still exist in the object mask of first type using the object mask of second type. We can also segment the object which has holes within the object using the object mask of second type. (Fig. 15) (a1), (a2), and (a3) show the moving object extraction of *Mother& Daughter* 45th with a wide broken boundary using expression (11). Particularly, (Fig. 15) (a1) shows *Mother& Daughter* with 42 terminating pixels, (a2) shows *Mother&*

(a1) Terminating pixel setting. (a2) Boundary linking. (a3) Moving object extraction

(a) Mother&Daughter 45th



(b1) Junki real video sequence (b2) Moving object extraction

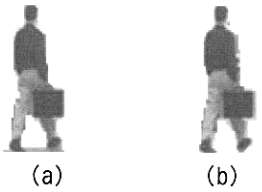
(b) Junki real video 84th

(Fig. 15) Moving object extraction (using both object masks) : (a) Mother&Daughter 45th, (b) Junki real video 84th

Daughter with 31 boundary linkings but still with a wide broken boundary, and (a3) shows the final the moving object extraction. The search radius is set at 14 for *Mother&Daughter* 45th QCIF frame. (Fig. 15) (b1) and (b2) show the extraction of Junki real video 84th which has a hole within the object using expression (11), too.

2.5.4 Moving cast shadow suppression

(Fig. 16) (a) *Hall monitor* 45th with a shadow image below shows the moving cast shadow in the image. The moving cast shadow is usually composed of a dark region called umbra and a soft transition region called penumbra[21]. In reference[4], a morphological gradient operation using the dilation and erosion is used to reduce the shadow effect. We apply the Roberts gradient operator[18, 19] to *Hall monitor* sequence to suppress the moving cast shadow because the Roberts gradient operator produces virtually no responses to the weak pixel differences of the moving cast shadow in the image and is efficient and is also easy to implement. (Fig. 16) (b) illustrates the suppression result of a moving cast shadow by the application of the Roberts gradient operator. We can see no shadow in (Fig. 16) (b). The shadow has been suppressed.



(a) (b)

(Fig. 16) Moving cast shadow suppression : (a) Hall monitor 45th with a shadow, (b) Hall monitor 45th without a shadow

3. Experiment

We experiment the automatic moving object segmentation algorithms using standard MPEG-4 test video sequences such as *Akiyo*, *Claire*, *Grandma*, and *Mother& Daughter* showing head-and-shoulder type motion and *Weather*, *Hall monitor*, and *Silence* showing active motion in the complex background. We also use the thin objects such as a thin tree branch and a thin string to show the segmentation of thin objects. We also use the walking students sequence in the complex background to show the segmentation of moving object in the complex background.

3.1 Objective evaluation

We use the error percentage measure[4] as the objective evaluation. This is a simple but effective relative evaluation using the ground truth. The error percentage measures the objective cumulative pixel differences between the alpha map of the original image and that of the segmented image. The following is the error percentage equation. Alpha map is the object mask.

$$Error\_Percentage = \frac{\sum_{W}\sum_{H}(a(x,y) \oplus s(x,y))}{W \times H} \times 100\% \quad (12)$$

The $a(x, y)$ is the alpha map of the original image used for reference. $\oplus$ is the Exclusive-OR operation. The $s(x, y)$ is the alpha map of the segmented image. W is the width and H is the height. (Fig. 17) below shows the error percentage graphs of *Claire*, *Akiyo*, *Weather*, *Hall monitor*, a thin string, and walking students.

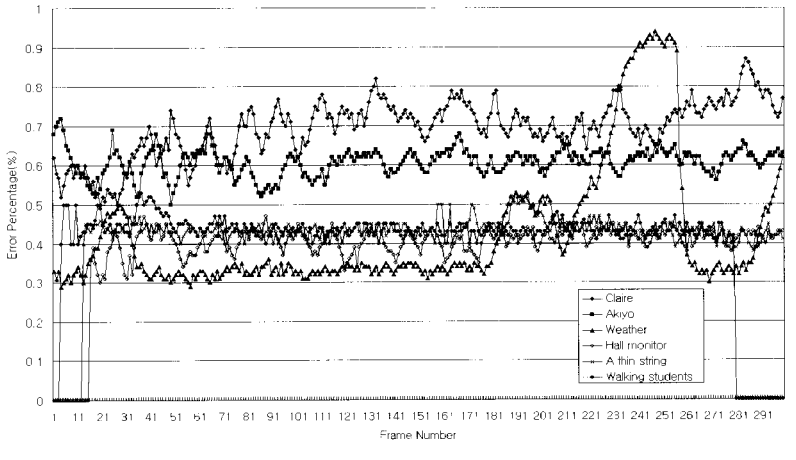
We obtained the $a(x, y)$ reference image sequence by segmenting it manually using the PHOTOSHOP tool in order to obtain the accurate reference image. We obtained the $s(x, y)$ by applying the developed moving object segmentation algorithm. The size of image sequence is CIF(352x288) and we input the image sequence by 25 frames/sec.

The error percentage values of *Claire*, *Akiyo*, and *Weather* are less than 0.8% except for some frames. This means that the segmented moving objects for these sequences are similar to the objectively segmented moving objects from 1 to 298 total images of *Claire*, *Akiyo*, and *Weather*. This validates that the proposed moving object segmentation algorithm can segment the moving objects accurately in the objective measure when the quality of the sequence is high. For example in (Fig. 16), the frames from 17 to 33 of *Weather* show 0.43% error percentage and thus 372 pixel differences on the average because the object moves much during these frames. The frames from 177 to 209 show 0.46% error percentage and 375 pixel differences on the average and the frames from 210 to 260 show 0.71% error percentage and 619 pixel differences on the average because the object also moves much during these frames. And for *Claire*, the frames from 130 to 135 show 0.78% error percentage and 679 pixel differences on the average and the frames from 280 to 283 show 0.81% error percentage and 705 pixel differences because the object moves much during these frames.

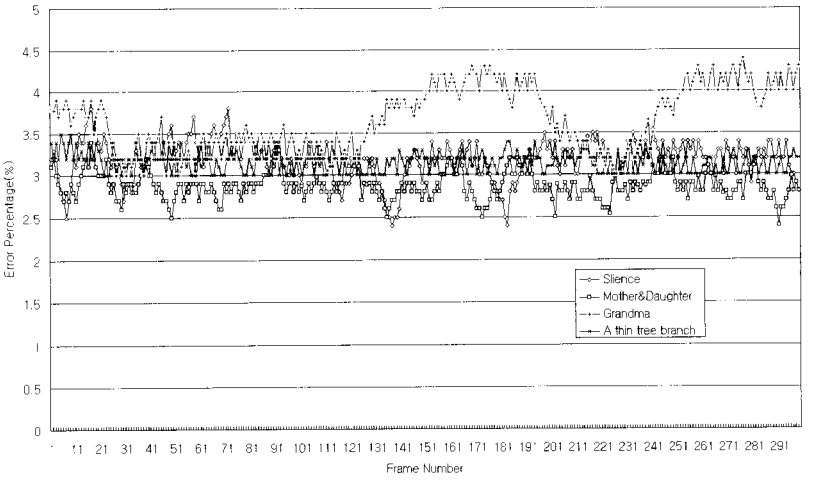
Comparing the result of *Weather* in (Fig. 17) with the result in (Fig. 7) of [4], we conclude that the error percentage values of *Weather* of our approach are better than the error percentage values of *Weather* of [4]. The error percentage values of *Weather* of our approach range from 0.3% to 0.95% but the error percentage values of *Weather* of [4] range from 0.5% to 1.5%.

The error percentage values of *Hall monitor* and the thin string range from 0.3% to 0.5% and the error percentage values of walking students range from 0.5% to 0.9%.

We show in (Fig. 18) error percentage graphs for *Silence*, *Mother&Daughter*, *Grandma*, and a thin tree branch sequences. The error percentage values of these



(Fig. 17) Error percentage graph I



(Fig. 18) Error percentage graph II

〈Table 2〉 Performance statistics

Test CPU	Sequence type	Sequence	Total Frames	Total Processing Time(msec)	Frames/Second
Pentium-IV 3.4GHz	Standard MPEG-4 Test Image Sequence	Hall Monitor(CIF)	298	14643	20.36
		Hall Monitor(QCIF)	328	4063	80.74
		Akiyo(CIF)	298	17529	17.01
		Akiyo(QCIF)	298	4457	66.87
		Claire(QCIF)	492	6571	74.88
		Monitor&Daughter(QCIF)	298	5025	59.31
		Weather(360×243)	298	17253	17.28
	Real Video	Junki real video(QCIF)	595	7711	74.60
		A thin tree branch(QCIF)	622	9603	64.78
		Walking students(352× 240)	1898	78983	24.04

sequences are relatively high and these are due to the low qualities of these sequences. For example, the error percentage values of *Grandma* range from 3% to 4.5% and those of *Mother&Daughter* range from 2.4% to 3.5%. The error percentage values of *Silence* range from 2.5% to 3.8% and those of a thin tree branch range from 3% to 3.5%. We can, however, see in (Fig. 19), (Fig. 20), and (Fig. 21) in section 3.2 that the segmentation results of the sequences with relatively high error percentage values show visual quality good enough for easy recognition.

<Table 2> shows the performance statistics of our algorithm using standard MPEG-4 test video sequences and real video sequences. We use a Pentium-IV 3.4GHz CPU as the test CPU. We have optimized the implementation of computation intensive algorithms for the real-time requirement of object-based applications.

<Table 2> shows the test CPU, sequence type, sequences, total frames, total processing time, and frames per second. The test image sequences are CIF and QCIF (176x144) standard MPEG-4 test image sequences such as *Hall monitor*, *Akiyo*, *Claire*, *Mother&Daughter*, and *Weather* and real video sequences such as Junki, a thin tree branch, and walking students. For example, the proposed system can process 20.36 CIF frames per second for *Hall monitor* CIF sequence and 80.74 QCIF frames per second for *Hall monitor* QCIF sequence. This system can process 70.20 QCIF frames and 19.7 CIF frames including *Weather*(360x243) and walking students(352x240) on the average on a Pentium-IV 3.4GHz CPU.

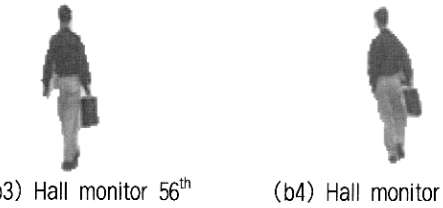
3.2 Subjective evaluation

We tested the moving object segmentation algorithm first using standard MPEG-4 test video sequences such as *Akiyo, Hall monitor, Claire, Silence, Weather, Grandma,* and *Mother&Daughter*. The following (Fig. 19) shows the segmentation results.

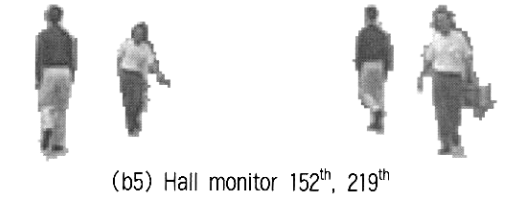
(a1) Akiyo 18th (a2) Akiyo 50th

(a)Akiyo

(b1) Hall monitor 45th (b2) Hall monitor 54th



(b3) Hall monitor 56th (b4) Hall monitor 69th

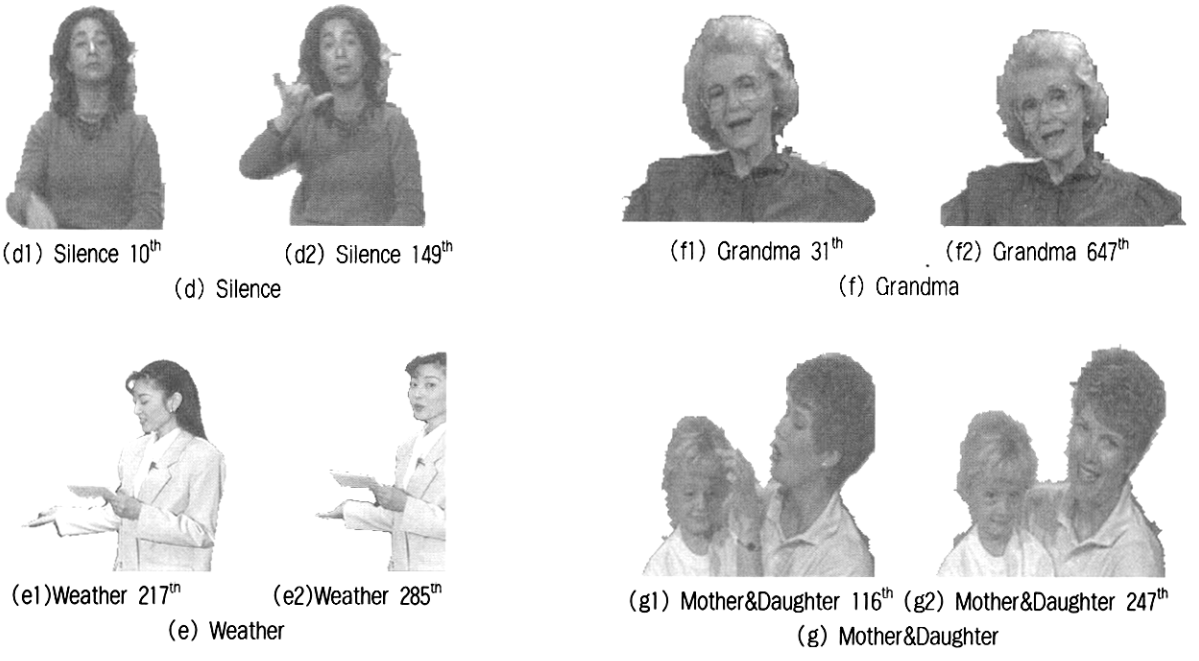


(b5) Hall monitor 152th, 219th

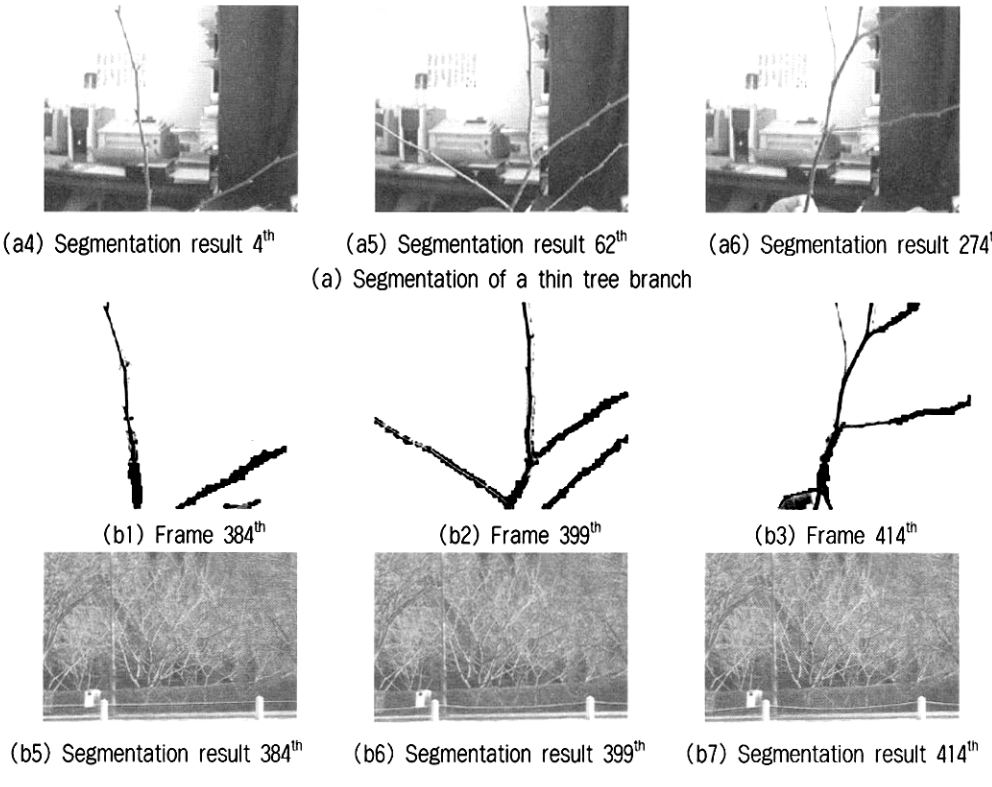
(b) Hall monitor

(c1) Claire 10th (c2) Claire 44th (c3) Claire 90th

(c) Claire



(Fig. 19) Segmentation of standard MPEG-4 test video sequences : (a) Akiyo, (b) Hall monitor, (c) Claire, (d) Silence, (e) Weather, (f) Grandma, (g) Mother&Daughter

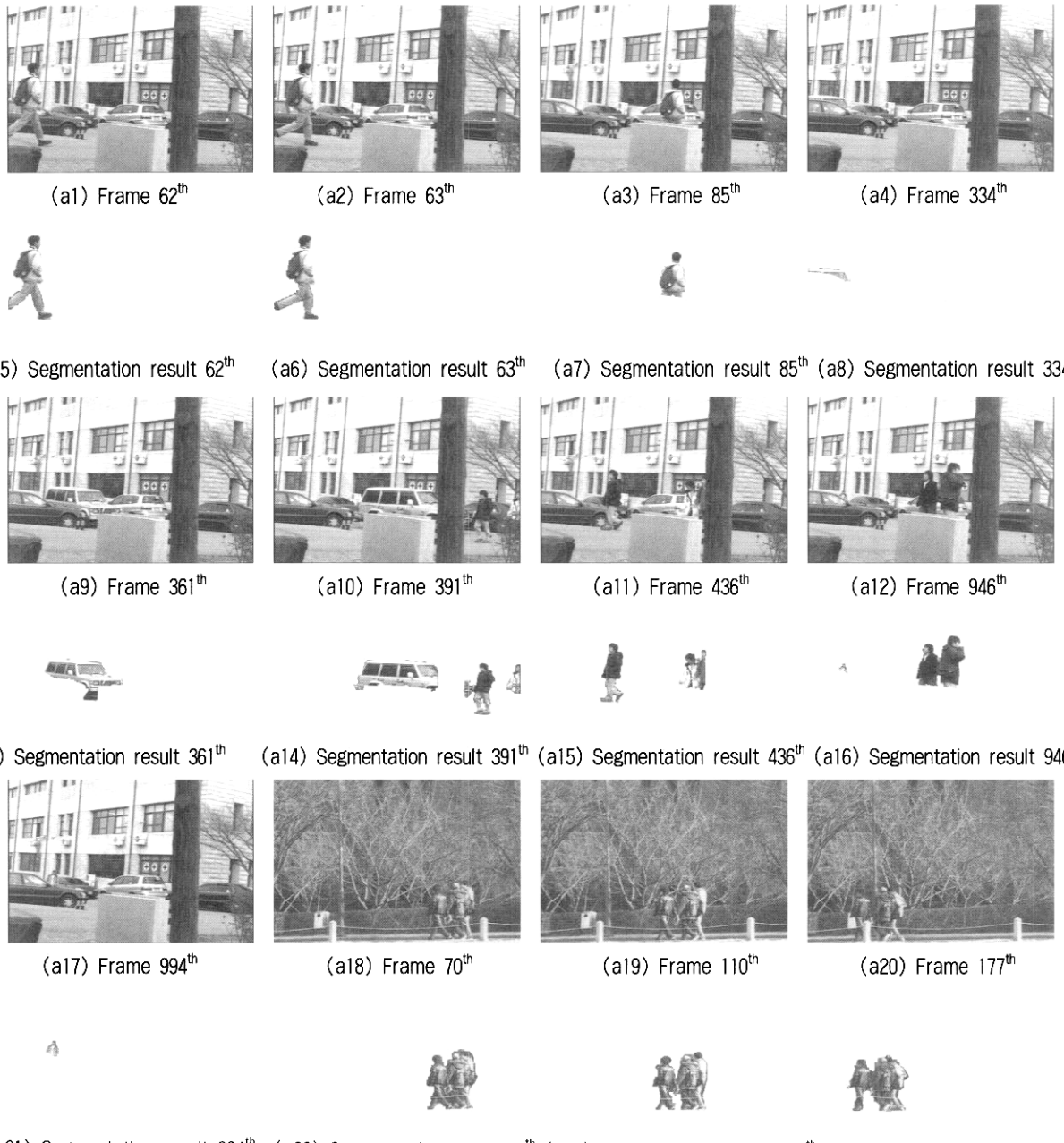


(Fig. 20) Segmentation of thin objects : (a) Segmentation of a thin tree branch, (b) Segmentation of a thin string

We can see that our algorithm can segment the moving object accurately and can also segment the multiple moving objects accurately in the MPEG-4 test image sequences and can also detect and segment an object which has a hole within the object. We can see a hole formed by the arms in (Fig. 15) (b2) in section 2.4.3. We can also see a single small hole in the upper part of (Fig. 19) (e1) *Weather* 217th between the hair and the forehead. We can also notice in (Fig. 19) (b1) *Hall monitor* 45th that the moving cast shadow has been suppressed by the application of the Roberts gradient operator.

We also tested our algorithm with a thin tree branch and a thin string hung between the standing poles in the complex background composed of grass and trees to show the detection and segmentation of thin objects. The thin tree branch image is a QCIF image and the thin string is a 352x240 size image. The following (Fig. 20) shows the segmentation results of the thin objects.

We can clearly see the segmentation results of a thin tree branch in (Fig. 20) (a4), (a5), and (a6). We can no-



(a1) Frame 62th (a2) Frame 63th (a3) Frame 85th (a4) Frame 334th

(a5) Segmentation result 62th (a6) Segmentation result 63th (a7) Segmentation result 85th (a8) Segmentation result 334th

(a9) Frame 361th (a10) Frame 391th (a11) Frame 436th (a12) Frame 946th

a13) Segmentation result 361th (a14) Segmentation result 391th (a15) Segmentation result 436th (a16) Segmentation result 946th

(a17) Frame 994th (a18) Frame 70th (a19) Frame 110th (a20) Frame 177th

(a21) Segmentation result 994th (a22) Segmentation result 70th (a23) Segmentation result 110th (a24) Segmentation result 177th

(Fig. 21) Segmentation of moving objects in the complex background : (a) Segmentation of walking students and a moving car in the complex cluttered background

tice in (Fig. 20) (b5), (b6), and (b7) that a thin string swaying gently is detected and segmented accurately in the complex background, too. The broken regions in the thin string segmentation are caused by the standing poles which prop the string and the seemingly identical color of the string with that of the standing can box.

We test our algorithm using the moving objects in the complex background in the outdoor environment. The objects include the walking students and a moving car on the campus road in the complex cluttered background and another group of walking students in the complex cluttered background. The following (Fig. 21) shows the segmentation results.

We can observe the accurate segmentation of a walking student and also the accurate segmentation of a moving car in the outdoor complex cluttered background from (Fig. 21) (a5), (a6), and (a7) and from (a8), (a13), and (a14). We can see the walking legs of a walking student accurately in (Fig. 21) (a15) and we see a person in a small shape walking up the road in (Fig. 21) (a16) and (a21). We show more segmentation results of a group of walking students from (Fig. 21) (a22), (a23), and (a24) in the complex cluttered background. The segmentation results of a group of walking students are accurate in the complex cluttered background. We can even see the white trace of the string in the legs of the walking students.

We can compare these segmentation results with the results in [1][2][3][4][5]. We notice that our segmentation results are generally better than those results in terms of the accuracy of the moving object boundary because we use the object mask of first type constructed by the object boundary linking.

Summarizing the main contribution of our segmentation approach, we have developed a novel object boundary linking algorithm for the accurate segmentation of moving object. And as the results, we show the detection and segmentation of the thin objects accurately in (Fig. 20) (a4), (a5), and (a6) of a thin tree branch and in (Fig. 20) (b5), (b6), and (b7) of a thin string. And we show the segmentation result of the walking students in the complex cluttered campus building background and also the segmentation result of a group of walking students in the complex cluttered background in (Fig. 21) (a). We show the detection and segmentation of a small object in (Fig. 21) (a16) and (a21), too.

And we show the segmentation of the object which has a hole within the object in (Fig. 15) (b2) and in (Fig. 19) (e1) using the object boundary linking and the background. But none of the references [1][2][3][4][5] show this type of segmentation of an object which has a hole within the object.

3.3 Discussion of results

The major advantage of the developed techniques is the accurate segmentation of moving object using the novel object boundary linking, the segmentation of thin objects, and the segmentation of the object which has a hole within the object using two object masks. We show the segmentation results in (Fig. 19), (Fig. 20), and (Fig. 21) in section 3.2.

The system can segment the moving object automatically by the selection of the appropriate threshold value for the automatic change detection and accurately using the edge tracing and copying technique and the object boundary linking.

4. Conclusion

We consider the moving object segmentation as a process of development from the initial moving object to the final moving object segmentation. We have developed a novel moving object edge construction algorithm, a novel space-oriented geometric boundary linking algorithm, and the segmentation of the moving object using two complementary object masks. We have achieved the automatic and accurate segmentation of moving object, the segmentation of multiple moving objects, the detection and segmentation of thin objects, the detection and segmentation of a small object, and the segmentation of an object which has a hole within the object using the developed novel object boundary linking algorithm and the background.

We show the segmentation of moving objects in the complex cluttered background such as the campus building background and the background made of grass and trees.

We construct the moving object edge progressively from the initial moving object edge through the intermediate and to the final moving object edge using the edge union operation and the edge tracing and copying technique to obtain a more accurate moving object edge. The boundary linking algorithm forms a quadrant around the terminating pixel and searches for other terminating pixels to link in concentric circles clockwise within the search radius in the forward direction. The search radius is determined automatically by the sorting of distances between the terminating pixels. The boundary linking algorithm does not create a cycle and guarantees the shortest distance linking. The boundary linking algorithm

can solve the problem of the broken boundaries which are caused by the instantaneous halt of the moving object, color similarity between the background and the object, illumination variation, and camera noise. The boundary linking algorithm mainly contributes to the accurate moving object segmentation in our algorithms. We also suppress the moving cast shadow using the Roberts gradient operator.

We have shown the automatic and accurate segmentation of the proposed algorithms using standard MPEG-4 test video sequences and real video sequences from a stationary camera. We have optimized the implementation of the computation intensive algorithms for the requirement of real-time object-based applications. The proposed algorithms are effective and efficient and can process 70.20 QCIF frames per second and 19.7 CIF frames per second on the average on a Pentium-IV 3.4GHz personal computer for real-time object-based processing.

We have assumed in the introduction that the camera and the background are stationary. For future research, we will work for a moving background and a moving camera.

REFERENCES

[1] Changick Kim, Jenq-Neng Hwang, "Fast and Automatic Video Object Segmentation and Tracking for Content-Based Applications," IEEE Trans. on Circuits and Systems for Video Technology, Vol.12, No.2, pp.122-129, Feb., 2002.

[2] Changick Kim, Jenq-Neng Hwang, "Object-Based Video Abstraction for Video Surveillance Systems," IEEE Trans. on Circuits and Systems for Video Technology, Vol.12, No.12, pp.1128-1138, Dec., 2002.

[3] Thomas Meier, King N. Ngan, "Video Segmentation for Content-Based Coding," IEEE Trans. on Circuits and Systems for Video Technology, Vol.9, No.8, pp.1190-1203, Dec., 1999.

[4] Shao-Yi Chien, Shyh-Yih Ma, Liang-Gee Chen, "Efficient Moving Object Segmentation Algorithm Using Background Registration Technique," IEEE Trans. on Circuits and Systems for Video Technology, Vol.12, No.7, pp.577-586, July, 2002.

[5] Shao-Yi Chien, Yu-Wen Huang, Liang-Gee Chen, "Predictive Watershed : A Fast Watershed Algorithm for Video Segmentation," IEEE Trans. on Circuits and Systems for Video Technology, Vol.13, No.5, pp.453-461, May, 2003.

[6] Fu Chang, Chun-Jen Chen, Chi-Jen Lu, "A linear-time component-labeling algorithm using contour tracing technique," Computer Vision and Image Understanding 93, pp.206-220, 2004.

[7] Luis Ducla Soares, Fernando Pereira, "Spatial Shape Error Concealment for Object-Based Image and Video Coding," IEEE Trans. on Image Processing, Vo.13, No.4, pp.586-599, April, 2004.

[8] Shahram Shirani, Berna Erol, Faouzi Kossentini, "A Concealment Method for Shape Information in MPEG-4 Coded Video Sequences," IEEE Trans. on Multimedia, Vol.2, No.3, pp.185-190, Sep., 2000.

[9] Hans Georg Mussmann, Michael Hötter, Jörn Ostermann, "Object-oriented analysis-synthesis coding of moving images," Signal Processing : Image Communications, Vol.1, No.2, pp. 117-138, Oct., 1989.

[10] Peter Gerken, Michael Wollborn, Stefan Schultz, "Polygon/spline approximation of arbitrary region shapes as proposal for MPEG-4 tool evaluation-Technical description," ISO/IEC JTC1/SC29/WG11 MPEG 95/360, Nov., 1995.

[11] Fabian W. Meier, Guido M. Schuster, Aggelos K. Katsaggelos, "A Mathematical Model for Shape Coding with B-Splines," Signal Processing : Image Communication, Vol.1, pp.685-701, 2000.

[12] Daniel P. Huttenlocher, Gregory A. Klanderman, William J. Rucklidge, "Comparing Images Using the Hausdorff Distance," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.15, No.9, pp.850-863, Sept., 1993.

[13] John Canny, "A computational approach to edge detection," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.8, No.6, pp.679-698, Nov., 1986.

[14] Roland Mech, Michael Wollborn, "A noise robust method for 2D shape estimation of moving objects in video sequence considering a moving camera," Signal Processing, Vol.66, pp.203-217, Sept., 1997.

[15] Til Aach, André Kaup, Rudolf Mester, "Statistical model-based change detection in moving video," Signal Processing, Vol.31, pp.165-180, March, 1993.

[16] Emrullah Durucan, Touradj Ebrahimi, "Moving Object Detection Between Multiple and Color Images," IEEE Conf. on Advanced Video and Signal Based Surveillance (AVSS'03), pp.243-251, July 21-23, Miami Florida, 2003.

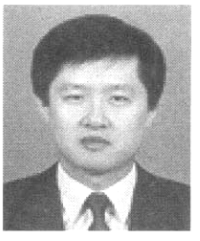
[17] Emrullah Durucan, Touradj Ebrahimi, "Change Detection and Background Extraction by Linear Algebra," Proc. of the IEEE, Vol.89, No.10, pp.1368-1381, October, 2001.

[18] Linda Shapiro, George Stockman, Computer Vision, Prentice-Hall, Inc., 2001.

[19] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, Prentice-Hall, Inc., 2002.

[20] Amjad Hajjar, Tom Chen, "A VLSI Architecture for Real-Time Edge Linking," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.21, No.1, pp.89-94, Jan., 1999.

[21] Jürgen Stauder, Roland Mech, Jörn Ostermann, "Detection of Moving Cast Shadows," IEEE Trans. on Multimedia, Vol. 1, No.1, pp.65-76, March, 1999.



이 호 석

e-mail : hslee@office.hoseo.ac.kr

1983년 서울대학교 전자계산기공학과 (공학사)

1985년 서울대학교 대학원 컴퓨터공학과 (공학석사)

1993년 서울대학교 대학원 컴퓨터공학과 (공학박사)

1994년~현재 호서대학교 공과대학 컴퓨터공학 뉴미디어학과 교수

관심분야 : 컴퓨터공학, 웹공학, 멀티미디어