공간기반 객체 외곽선 연결과 배경 저장을 사용한 움직이는 객체 분할
(Moving Object Segmentation using Space-oriented Object Boundary Linking and Background Registration)

이 호 석 †
(Ho Suk Lee)

요 약 동영상에서 움직이는 객체의 외곽선은 객체의 분할을 위하여 매우 중요하다. 그러나 객체의 외곽선에는 끊어진 외곽선(broken boundary)들이 많이 존재한다. 이 논문에서 우리는 새로운 공간 기반 외곽선 연결 알고리즘을 개발하여 끊어진 객체의 외곽선을 연결하였다. 객체 외곽선 연결 알고리즘은 끊어진 외곽선의 말단 픽셀(terminating pixel) 주변에 4분면을 형성한다. 그리고 반지름 범위 내에서 전 방향으로 탐색을 수행하여 가장 가까운 다른 말단 픽셀을 찾아 끊어진 객체의 외곽선을 연결한다. 시스템은 또한 입력된 동영상들로부터 배경을 저장한다. 시스템은 객체의 외곽선 연결 수행 결과로부터 하나의 객체 마스크를 생성하고 저장된 배경으로부터 또 하나의 객체 마스크를 생성하여 이 두 개의 객체 마스크를 함께 사용하여 동영상으로부터 움직이는 객체를 분할한다. 또한 시스템은 Roberts 기울기 연산자를 사용하여 추출된 움직이는 객체로부터 그림자도 제거한다. 제안된 알고리즘의 가장 큰 특징은 더욱 정확한 움직이는 객체의 분할과 내부에 구멍이 존재하는 움직이는 객체의 분할이다. 우리는 개발된 알고리즘을 표준 MPEG-4 테스트 영상과 카메라로 입력된 동영상을 사용하여 실험하였다. 제안된 알고리즘은 매우 좋은 효율을 나타내고 있다. 알고리즘은 2.0GHz Pentium-IV CPU에서 QCIF 영상은 최소한 초당 49 프레임 이상 처리할 수 있으며 CIF 영상은 최소한 초당 19 프레임 이상 처리할 수 있다.

키워드 : 움직임 객체 에지, 움직임 객체 외곽선, 외곽선 연결, 배경 저장, 움직임 객체 분할

Abstract Moving object boundary is very important for moving object segmentation. But the moving object boundary shows broken boundary. We invent a novel space-oriented boundary linking algorithm to link the broken boundary. The boundary linking algorithm forms a quadrant around the terminating pixel in the broken boundary and searches forward other terminating pixel to link within a radius. The boundary linking algorithm guarantees shortest distance linking. We also register the background from image sequence. We construct two object masks, one from the result of boundary linking and the other from the registered background, and use these two complementary object masks together for moving object segmentation. We also suppress the moving cast shadow using Roberts gradient operator. The major advantages of the proposed algorithms are more accurate moving object segmentation and the segmentation of the object which has holes in its region using these two object masks. We experiment the algorithms using the standard MPEG-4 test sequences and real video sequence. The proposed algorithms are very efficient and can process QCIF image more than 48 fps and CIF image more than 19 fps using a 2.0GHz Pentium-4 computer.

Key words : Moving object edge, Moving object boundary, Boundary linking, Background registration, Moving object segmentation

1. 서 론

The moving object segmentation is very important to support object-based functionalities and for the successful application of MPEG-4 video coding standard. The video moving object segmentation

· 이 논문은 2004년도 호서대학교 학술연구조성비에 의하여 연구되었음
† 정 회 원 : 호서대학교 컴퓨터공학부 교수
hslee@office.hoseo.ac.kr
논문접수 : 2004년 8월 23일
심사완료 : 2004년 12월 3일

has been receiving considerable amount of attention from many researchers[1-4].

One of the most important elements for moving object segmentation is to obtain the moving object boundary. The object boundary, however, shows broken boundary caused by the edge detection failure due to the instantaneous halt of the moving object. The object or the part of object often halts temporarily between consecutive frames and this causes the edge detection failure. And the illumination variation and the camera noise can also cause edge detection failure. The broken boundary makes it difficult to obtain the accurate moving object segmentation. The broken boundary should be supplemented with boundaries from other boundary sources and boundary linking and further using object mask. The other boundary sources include previous moving object edge, Canny edge of current frame, and background edge of current frame. 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 forward other terminating pixel to link in a clock-wise concentric circle within a radius. The algorithm does not create a cycle and guarantees the shortest distance linking. The linking algorithm tries to link the broken boundary robustly within a search radius. But the linking algorithm does not try to link the broken boundary which is wider than the search radius, so for some images, there might remain broken boundaries after boundary linking. But for most images, we can obtain a completely linked moving object boundary after boundary linking. We use the object mask to deal with the case of wide broken boundary.

The main characteristics of the proposed algorithm are accurate moving object segmentation by boundary linking, the segmentation of an object which has holes in its region, and real-time performance. We use the standard MPEG-4 test sequences and a real video sequence for experiment. We assume that the camera and background are stationary.

The organization of paper is as follows. Section 2 gives the detailed explanation of the proposed segmentation algorithms. Section 3 shows the experimental results and 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 initial moving object to final moving object segmentation. The initial moving object is actually the background change detection mask. We use the background change detection mask computed from the current frame and the registered background because it can yield more accurate moving object edge. The initial moving object is processed in two complementary ways. The two complementary ways are moving object boundary construction from moving object edge and boundary linking and noise elimination of initial moving object for object segmentation. The constructed object boundary, however, shows broken boundary so we try to link the broken boundary to obtain a completely linked boundary for most images. And finally we combine the results from two complementary object masks for accurate moving object segmentation and the segmentation of an object which has holes in its region. We use the connected component algorithm and morphological operation to eliminate the noise and to enhance the object mask boundary. Fig. 1 shows the overall system architecture.

FD means the pixel-based frame difference between the current frame and previous frame and generates the current frame change detection mask. BD means the frame difference between the current frame and background and generates the background change detection mask. BB means the background buffer which is used to register the background. The count value of FD frame difference is sent to BB to register the background. The moving object edge is constructed using the Canny edge of current frame 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 boundary regions and perform the space-

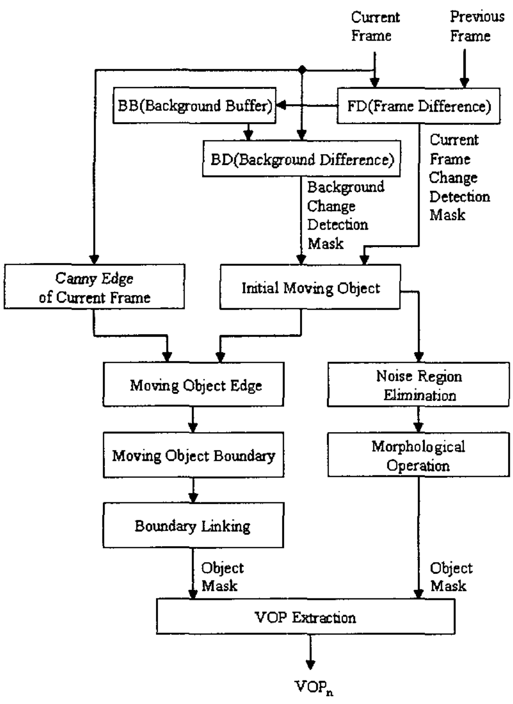


Fig. 1 Architecture of moving object segmentation

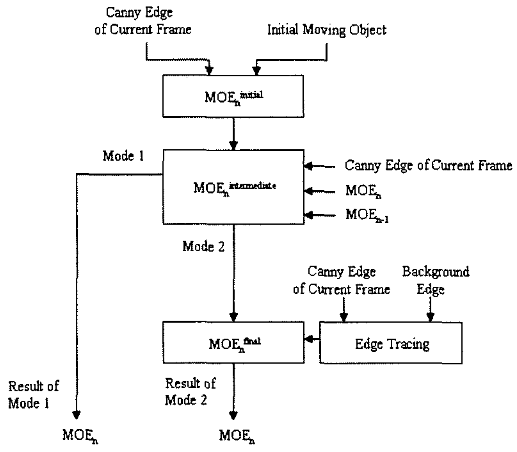


Fig. 2 Moving object edge construction

oriented boundary linking robustly to obtain a completely linked moving object boundary. We construct the object mask of first type using this result. We also eliminate noise from the initial moving object and construct the object mask of second type. Then, we extract the VOP of the current frame using these two object masks.

2.2 Moving object edge construction

We define the moving object edge as the edge of the moving object. We construct the moving object edge as a process from initial moving object edge through intermediate and to final moving object edge. Fig. 2 shows the block diagram of the moving object edge construction.

The initial moving object edge is constructed using Canny edge [5] of current frame and the initial moving object. The intermediate moving object edge is constructed using Canny edge of current frame and the previous moving object edge to link the broken boundary region which is mostly caused by instantaneous halt of the object between the consecutive frames. After this process, 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 video sequences such as "Weather" and "Hall monitor" in mode one and can obtain a sufficient moving object edge for further processing. In mode two, we construct the final moving object edge using edge tracing. The edge tracing is implemented using Canny edge and background edge of current frame. We can process the "Claire" which does not have any background and the "Akiyo" which has a high quality in mode two. User can select the mode using a parameter.

2.2.1 Change detection and background

We use the absolute pixel difference as the test statistics for change detection. Under the 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 [6].

We register the background using stationary background filtering [2,3] and use the background to generate the background change detection mask.

2.2.2 Moving object edge

More specifically, the moving object edge can be defined as the pixels constituting the edges of the moving object. The following is the equation for initial moving object edge construction.

$$MOE_n^{initial} = \{e = e_1 \,\&\, e_2 \mid e_1 \in E_n^c,\ e_2 \in IMO\} \qquad (1)$$

Here, E_n^c is the Canny edge of the current frame and IMO is the initial moving object. e, e_1, and e_2 are the edges. The operation & is the logical AND operation. The intermediate moving object edge is computed as follows.

$$MOE_n^{intermediate} = \{e = e_1 \,\&\, e_2 \mid e_1 \in E_n^c,\ e_2 \in MOE_n^{initial} \,\|\, e_2 \in MOE_{n-1}^{initial}\} \quad (2)$$

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, caused by the instantaneous halt of the object. The final moving object edge is constructed as follows.

$$MOE_n^{final} = \{e = e_1 \,\|\, e_2 \mid e_1 \in MOE_n^{intermediate},\ e_2 \in E_n^c \,\&\, e_2 \notin E_n^b\} \quad (3)$$

Here, E_n^b is the background edge. The logical OR operation means the union of the edges of Canny edge of current frame excluding background edges with the intermediate moving object edges in order to obtain the more linked final moving object edge. We call this edge tracing. We find edge tracing quite effective when the background has no noise or when the image has a high quality such as the "Akiyo" or the "Claire". We give a procedural explanation of edge tracing below.

2.2.3 Edge tracing and copying

Edge tracing and copying is performed in mode two of moving object edge construction. Edge tracing and copying uses Canny edge of current frame E_n^c and the background edge of current frame E_n^b. The followings are edge tracing and copying steps.

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

$$E_n^b(x,y) = E_n^c(x,y) - MOE_n^{intermediate}(x,y) \quad (4)$$

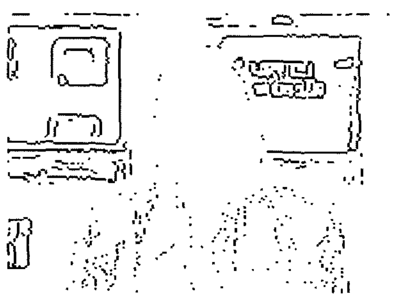
But the background edge contains some still remaining uneliminated object pixels as shown Fig. 3(a). We use the connected component algorithm to eliminate these remaining object pixels in the background edge and obtain a clear background edge as shown in Fig. 3(b).

(step2) Scan $MOE_n^{intermediate}(x,y)$ by horizontal scanline to locate the first pixel encountered.

(step3) Search around the neighboring pixels of the pixel found in **(step2)** in 8-connected way to start the edge tracing in $MOE_n^{intermediate}$.

(step4) Check the presence of a pixel in $E_n^c\ (x,y)$ at the corresponding coordinate position during edge tracing in $MOE_n^{intermediate}$ until the edge tracing comes to the terminating pixel in $MOE_n^{intermediate}(x,y)$. If there exists a pixel at the corresponding coordinate, perform edge tracing along the $E_n^c\ (x,y)$ edge as long as the pixel of $E_n^c\ (x,y)$ under edge tracing does not encounter the background edge $E_n^b(x,y)$ constructed in **(step1)**. And include the traced edge of $E_n^c\ (x,y)$ into $MOE_n^{\ final}$. If it encounters the background edge, edge tracing stops.

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


(a) With remaining object pixels in the background edge

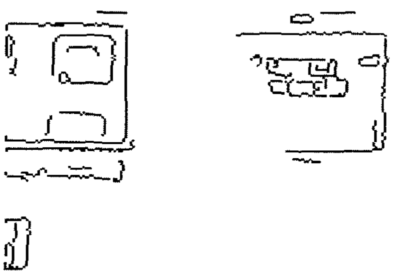

(b) With no object pixels in the background edge

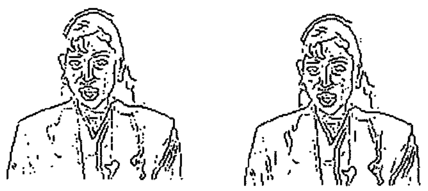
Fig. 3 Background edge (Akiyo 11th)

Fig. 4 shows the moving object edge construction results.

Fig. 4 shows the moving object edge of (a) "Hall monitor", (b) "Akiyo", and (c) "Claire". The left column of (a) and (b) were produced using the current frame change detection mask as in [1] and the right column of (a) and (b) using the background change detection mask. We can see the more accurate moving object edge in the right column. For (a) and (b), we use the intermediate

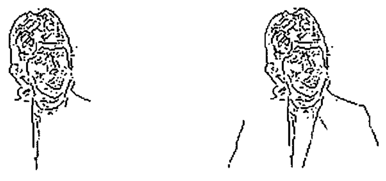
(a1) $MOE^{intermediate}$ (a2) $MOE^{intermediate}$

(a) Hall monitor 45th

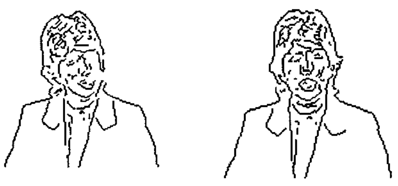


(b1) $MOE^{intermediate}$ (b2) $MOE^{intermediate}$

(b) Akiyo 29th



(c1) $MOE^{initial}$ 90th (c2) $MOE^{intermediate}$ 90th



(c3) MOE^{final} 90th (c4) MOE^{final} 17th

(c) Claire

Fig. 4 Moving object edge

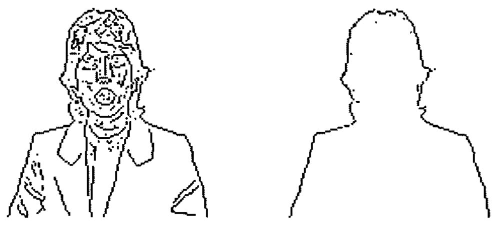
(a) Hall monitor, (b) Akiyo, (c) Claire

moving object edge as the final result. The (c) "Claire"shows the process of moving object edge construction using edge tracing. Fig. 4 (c4) was shown for more demonstration.

2.3 Moving object boundary

2.3.1 Moving object boundary extraction

The moving object boundary is defined as the boundary pixels of moving object edge. First, we eliminate small isolated noise pixels from the moving object edge. Then, we scan the moving object edge by horizontal and vertical scanline and obtain the moving object boundary. The following Fig. 5 shows the moving object boundary of "Claire".



(a) (b)

Fig. 5 Moving object boundary

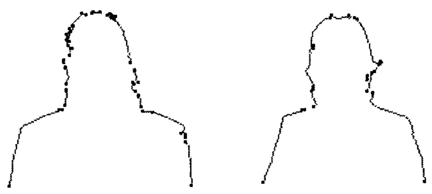
(a) Moving object edge, (b) Moving object boundary

2.3.2 Moving object boundary linking

We examine the moving object boundary image and set terminating pixels at the final end pixels of the broken boundary line. Fig. 6 (a1) and (a2) show the "Hall monitor" with terminating pixels, (b) shows the "Akiyo" with terminating pixels, and (c) shows the "Claire" with terminating pixels. The actual terminating pixel is just a single pixel but we enlarge it for clear vision.

(a1) Hall monitor 54th (a2) Hall monitor 69th

(a) Hall monitor



(b) Akiyo 18th (c) Claire 11th

Fig. 6 Terminating pixel setting

(a) Hall monitor, (b) Akiyo, (c) Claire

Edge linking algorithms using various techniques are explained using examples in [7]. An edge linking algorithm considering three types of break points using a window is explained in [8]. The proposed boundary linking algorithm is a space-oriented algorithm considering boundary pixel direction. The boundary linking algorithm links the broken boundary by inserting pixels from one terminating pixel to the other terminating pixel. The algorithm considers the linking direction first. The algorithm forms a quadrant around the terminating pixel and performs a clockwise concentric forward search within a search radius from the terminating pixel. The algorithm always guarantees shortest distance linking in object boundary and does not create a cycle. We see that the boundary linking algorithm forms the object boundary accurately in the broken boundary region because the terminating pixels used for boundary linking 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 virtual pixel is to predetermine the general direction toward which the broken boundaryis going to be linked from the terminating pixel. The following Fig. 7 (a) shows the moving object boundary with terminating pixels and (b) shows the enlargement of block area in (a) to show the positions of the virtual pixels for object boundary linking.

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 varies significantly according to the characteristics of the input image. Thus, we form a quadrant around the terminating pixel $T_0=(0,0)$ for a more general search range 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 as the angle between T_0 and V_0 we compute

$$\theta_V = \cos^{-1}\frac{V_x}{\sqrt{V_x^2+V_y^2}}*\pi/180 \quad (5)$$

and we search another terminating pixel T_1 to link around the terminating pixel T_0 within the quadrant in clockwise concentric circle within a given 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.

$$\theta_S = \cos^{-1}\frac{T_x}{\sqrt{T_x^2+T_y^2}}*\pi/180 \quad (6)$$

The condition and operation of boundary linking can be summarized as follows.

```
procedure boundary_linking( )
{
  if $(2\pi+\theta_v-\frac{\pi}{4}\le\theta_s\le2\pi+\theta_v+\frac{\pi}{4})$ then {
     link $T_0$ = (0,0) to $T_1$ = $(T_x, T_y)$
  } else expand the quadrant to half plane
 and perform boundary linking from $T_0$ = (0,0)
}
```

If the angle θ_s of the terminating pixel T_1 is within the range as computed above, the boun-

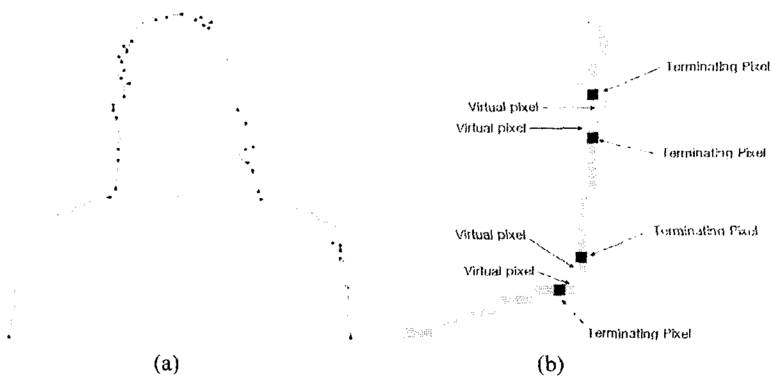


Fig. 7 Virtual pixel setting
(a) Moving object boundary with terminating pixels
(b) Imaginary virtual pixel setting

darylinking 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 half plane and perform the boundary linking iteratively from T_0 = (0,0).

We find other terminating pixel to link within the radius around the starting terminating pixel within 5 iterations most of the time in the experiment. In other cases, the boundary linking algorithm searches the pixel in the expanded half plane. We continue boundary linking until there remain no more terminating pixels to link. Fig. 8 shows the constructed moving object boundary of Fig. 6 after boundary linking.

The linking algorithm tries to link the broken boundary robustly within a search radius which is determined experimentally. The linking algorithm does not try to link the terminating pixel which is located farther than the search radius, because it can link the wrong terminating pixel in trying to do so. For example, the search radius for "Hall monitor" is set at 8 and "Akiyo" at 12.

The algorithm sets 20 terminating pixels in Fig. 6 (a1) and performed 11 boundary linking in Fig. 8 (a1) "Hall monitor". In Fig. 6 (b) "Akiyo" 18th, the algorithm sets 39 terminating pixels and performed 18 boundary linking. We can see several broken boundaries in Fig. 6 (a1), (a2), (b), and (c) and can see their corresponding completely linked object boundary in Fig. 8 (a1), (a2), (b), and (c) respectively. Currently, we can see two object boundaries

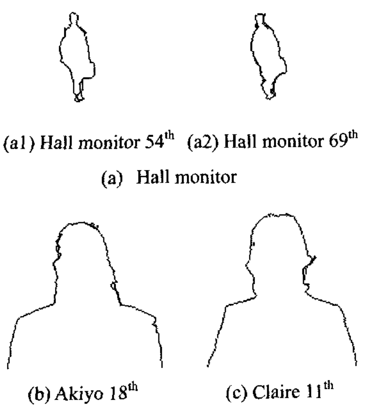


Fig. 8 Moving object boundary linking
(a) Hall monitor, (b) Akiyo, (c) Claire

in some boundary, one by moving object edge itself, and the other by boundary linking, but these seemingly double boundaries do not cause a problem. We map the object texture to the outermost boundary.

2.3.3 Moving object linking at the bottom

We construct the moving object boundary as shown in Fig. 8. But the moving object shows boundary disconnection at the bottom. Fig. 9(a) and (b) show the connected moving object boundary of Fig. 8 (b) and (c).

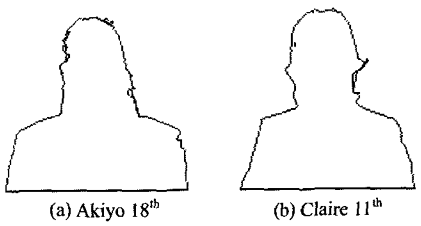


Fig. 9 Object boundary connection at the bottom
(a) Akiyo, (b) Claire

2.4 Video object plane extraction

We perform moving object extraction with two object masks. We use the object mask made from boundary linking and the object mask from initial moving object. We call the former the object mask of first type and the latter the object mask of second type. The advantage of the first type is the accurate moving object extraction. But it cannot handle the case of an unlinked broken boundary which can still exist in the wide broken boundary and the segmentation of an object which has holes in its region. The second type can handle these situations but the boundary it shows is somewhat coarse. So, we employ these two complementary object masks.

2.4.1 Using the object mask of first type

Fig. 10 shows moving object extraction using the object mask of first type. Fig. 10(a) is the moving object mask and (b) is the segmentation result. Fig. 10(b) shows an accurate object boundary

2.4.2 Using the object mask of second type

Fig. 11 shows the block diagram of the second type object mask construction. We apply the connected component algorithm to remove the noise

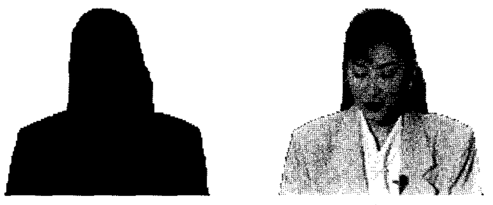


Fig. 10 Object extraction with first mask (Akiyo)
(a) Moving object mask, (b) Segmentation result

and then also apply the morphological opening and closing operation to enhance the quality of the initial moving object in order to use it as the moving object mask, which we call the object mask of second type.

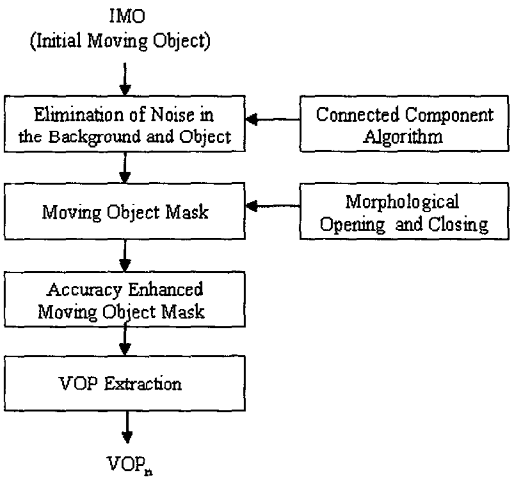

IMO
(Initial Moving Object)
Elimination of Noise in the Background and Object
Connected Component Algorithm
Moving Object Mask
Morphological Opening and Closing
Accuracy Enhanced Moving Object Mask
VOP Extraction
VOP_n

Fig. 11 Construction of object mask of second type

We show in Fig. 12the moving object extraction using the object mask of second type. Fig. 12(a) shows the initial moving object with noise, (b) shows the object mask of second type with clear background and object images, and (c) shows the segmentation result.

Fig. 13 shows moving object extraction result of "Akiyo" using the object mask of second type. The "Akiyo" object shows somewhat coarse object boundary.

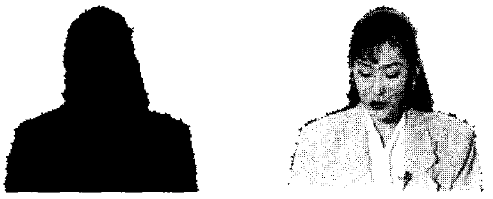


Fig. 13 Object extraction with second mask (Akiyo)
(a) Moving object mask, (b) Segmentation result

2.4.3 Using both object masks

Following is an expression for moving object extraction using the two complementary object masks.

$$Moving\_object_n(x,y) = \begin{cases} Current\_frame_n(x,y) & \text{if } mask\_first_n(x,y)=1 \\ & \quad \& \ mask\_second_n(x,y)=1 \\ 0 & otherwise \end{cases} \tag{7}$$

Moving_object$_n$(x,y) means the extracted moving object. The *mask_first$_n$(x,y)*means the object mask of first type and *mask_second$_n$(x,y)* 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 wide broken

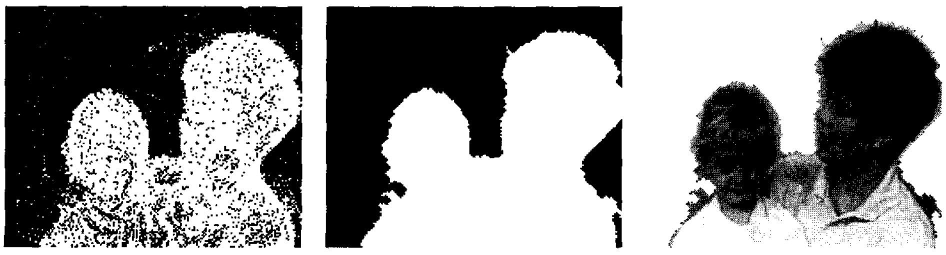


Fig. 12 Moving object extraction 45th
(a) Initial moving object (b) Object mask of second type (c) Segmentation result

boundaries and can segment the object which has holes using the object mask of second type. Fig. 14 (a) and (b) show the moving object extraction of "Mother&Daughter" with a wide broken boundary using expression (7). Fig. 14 (c) and (d) show the segmentation of an object which has holes in its region using expression (7), too.

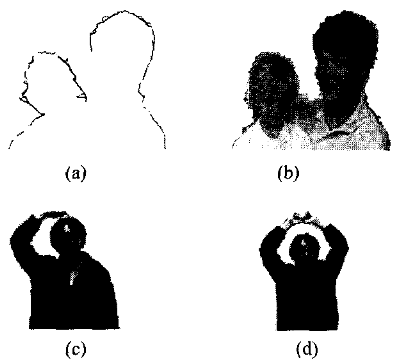


Fig. 14 Moving object extraction with both masks
(a) Mother&Daughter with wide broken boundary,
(b) Mother&Daughter segmentation
(c) Junki real video A, (d) Junki real video B

2.4.4 Moving cast shadow suppression

The moving cast shadow consists of umbra and penumbra [9]. We apply the Roberts gradient operator to suppress the moving cast shadow in the moving object image by taking advantage of the weak response property of the Roberts gradient operator [7].

3. Experiment

We experiment the moving object segmentation algorithm using the standard MPEG-4 test image sequences such as "Akiyo", "Hall monitor", "Claire", "Silence", "Weather", "Grandma" and "Junki" real video.

3.1 Objective evaluation

We use the error percentage measure [3] for the objective evaluation. This measures the cumulative error between the original and the segmented images.

$$Error\_Percentage = \frac{\sum_{W} \sum_{H} (a(x,y) \oplus OM(x,y))}{W \times H} \times 100\% \tag{8}$$

The $a(x, y)$ is the alpha map of the original image used for reference. $\oplus$ is the Exclusive-OR operation. The $OM(x, y)$ is the alpha map of the segmented image. W is the width and H is the height. Fig. 15 shows the error percentage graph for "Claire", "Akiyo", and "Weather".

The size of image sequence is CIF(352x288) and

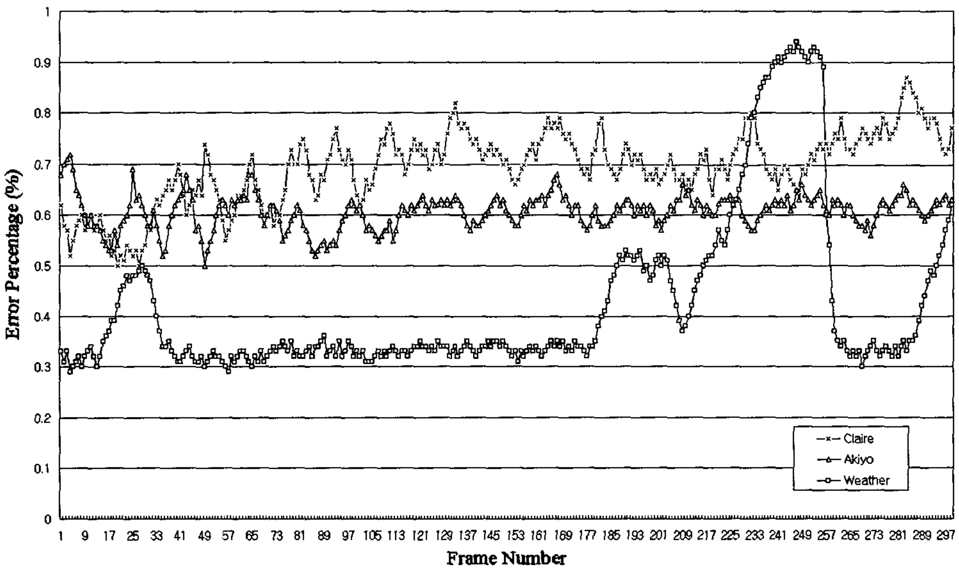


Fig. 15 Error percentage graph

it is input 25 frames/sec. 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. Also, we obtained the *OM(x, y)* by applying the proposed object segmentation algorithm.

If there is no difference, the error percentage is 0%. If the error percentage is above 1.5%, then the segmented moving object could show some visually recognizable difference from the input moving object.

The error percentage values we obtained from our experiments are less than 0.8% for these image sequences. This value indicates that the segmented moving object by our algorithm is visually very similar to the manually segmented moving object frames from the input image sequence and validates that the proposed segmentation algorithm can segment the objects very accurately. In Fig. 15, frames 25 and 26 of "Akiyo" and "Weather" show some high value of error percentage and frame 190, 193, and 200 of "Weather" show high value of error percentage because the object moves somewhat faster in these frames.

We can compare Fig. 15 with Fig. 7 of [3] and can see that the error percentage results of "Akiyo" and "Weather" of our approach are comparable with those results.

Table 1 shows the performance statistics of our algorithm and other approaches for comparison. We use the 2.0GHz P-IV CPU as the experiment CPU. We have optimized the implementation of computation intensive algorithms to satisfy the real-time requirement of various multimedia applications.

Table 1 shows the performance statistics. The table shows the test sequences, total frames, total processing time, and frames per second. The test sequences are CIF(352x288) and QCIF(176x144) standard MPEG-4 test sequences such as "Hall monitor", "Akiyo", "Claire", "Mother&Daughter", and "Weather". For example, the proposed system can process 72.25 frames per second for "Hall monitor" QCIF sequence, in other words, the system spends 0.0138 second per frame for "Hall monitor" QCIF sequence. We have included the performance results of [1][3][4] in the table for comparison. The approach [1] can process 2.5 fps using P-II 450MHz. The approach [3] can process QCIF at 25 fps and CIF at 10 fps using P-III 450MHz. The approach [4] can process QCIF at 9.4 fps using P-III 800MHz. The test CPUs are different but we can conclude that our proposed algorithms are more efficient than these other algorithms.

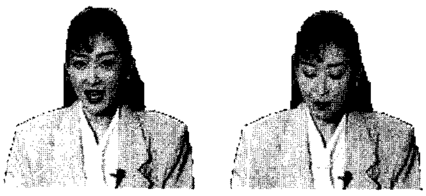
3.2 Subjective evaluation

We show the moving object segmentation results of "Akiyo", "Hall monitor", "Claire", "Silence", "Weather(360x243)", "Grandma" in Fig. 16. We also use "Junki"real video sequence to demonstrate the segmentation of an object which has holes in its region more clearly.

We see that our algorithm can produce the accurate moving object segmentation and can segment the multiple moving objects and the object which has holes in its region using these two complementary object masks. We can see a single tiny hole in Fig. 16 (e1) between the hair and

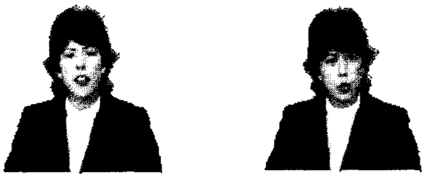
Table 1 Performance statistics

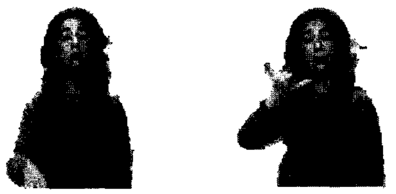
Algorithm	Test CPU	Sequence	Frames	Total processing time(msec)	Frames/second
Proposed	P IV 2.0G	Hall Monitor(CIF)	298	15691	19.02
		Hall Monitor(QCIF)	328	44541	72.25
		Akiyo(CIF)	298	15397	19.38
		Akiyo(QCIF)	298	5030	59.25
		Claire(QCIF)	492	7002	70027
		Mother&Daughter(QCIF)	298	6107	48.83
		Weather(360×243)	298	12844	23.22
Changick Kim[1]	P II 450M			(about 0.4 sec/frame)	
Shao-Yi Chien[3]	P-III 450M	CIF type			10
		QCIF type			25
Shao Yi Chien[4]	P III 450M	QCIF type		(106.64 msec/frame)	9.4

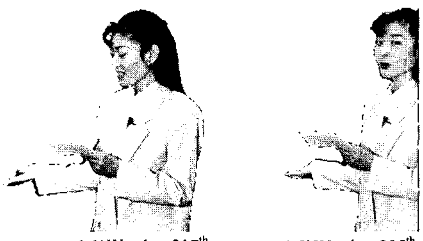

(a1) Akiyo 29th (a2) Akiyo 50th
(a) Akiyo

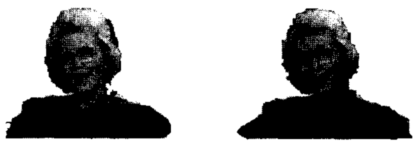
(b1) Hall monitor 45th and 56th

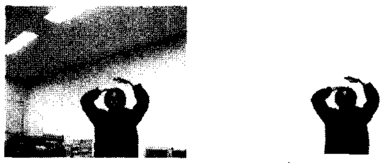
(b2) Hall monitor 152th and 219th
(b) Hall monitor


(c1) Claire 16th (c2) Claire 44th
(c) Claire


(d1) Silence 10th (d2) Silence 149th
(d) Silence


(e1)Weather 217th (e2)Weather 285th
(e) Weather


(f1) Grandma 31th (f2) Grandma 647th
(f) Grandma


(g1) Junki real video 149th

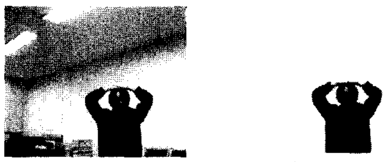

(g2) Junki real video 158th

Fig. 16 Moving object segmentation
(a) Akiyo, (b) Hall monitor, (c) Claire, (d) Silence, (e) Weather, (f) Grandma, (g) Junki real video

forehead. We can also see two holes clearly in (g2).

We can compare these segmentation results with the results in [1][2][3][4]. Our segmentation results are better than those results in terms of the accuracy of the segmentation result because we use the object boundary linking for accuracy. The proposed segmentation algorithm can segment the object which has holes in its region as shown in (e) and (g) in Fig. 16. But none of the references demonstrate this. Particularly, we can see in (d) and (e) of Fig. 9 of [3] that the approach does not segment the object which has a hole in its region. We can also compare our "Mother&Daughter" segmentation results of Fig. 14 (b) with those of Fig. 16 and Fig. 17 of [4]and we can see that our results are better. Our results show the face of the child accurately without any dip in the face.

4. Conclusion

We propose a novel moving object edge construction algorithm, a space-oriented geometric boundary linking algorithm, and a segmentation algorithm usingtwo object masks. We can achieve more accurate moving object segmentation, multiple moving objects segmentation, and the segmentation of the object which has holes in its region using these algorithms.

We have shown the performance of our proposed algorithms using standard MPEG-4 test image

sequence and a real video from camera. The proposed algorithms are very efficient and can process QCIF image more than 48 fps and CIF image over 19 fps in a 2.0GHz Pentium-4 personal computer for real-time content-based applications.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] T. Aach, A. Kaup, R. Mester, "Statistical model-based change detection in moving video," Signal Processing, Vol. 31, pp.165-180, March 1993.

[7] Rafael Gonzalez, Richard Woods, Digital Image Processing(2nd Edition), Addison-Wesley Pub Co., 2002.

[8] 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, 1999.

[9] 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

1979년 3월~1983년 2월, 서울대학교 전자계산기공학과 공학사 졸업. 1983년 3월~1985년 2월, 서울대학교 컴퓨터공학과 대학원 공학석사 졸업. 1989년 3월~1993년 8월, 서울대학교 컴퓨터공학부 대학원 공학박사 졸업. 1994년 3월~현재, 호서대학교 컴퓨터공학부 교수. 관심분야는 영상처리, 영상 분할, MPEG-4, MPEG-7, JPEG-2000