깊이정보를 이용한 고속 고정밀 얼굴검출 및 추적 방법

배 윤 진◆, 최 현 준*, 서 영 호**, 김 동 욱**

A Fast and Accurate Face Detection and Tracking Method by using Depth Information

Yun-Jin Bae◆, Hyun-Jun Choi*, Young-Ho Seo**, Dong-Wook Kim**

요 약

본 논문에서는 RGB영상과 깊이영상을 사용하여 얼굴검출 및 추적을 고속으로 수행할 수 있는 방법을 제안한다. 이 방법은 얼굴검출 과정과 얼굴추적 과정으로 구성되며, 얼굴검출 과정은 기본적으로 기존의 Adaboost 방법을 사용하나, 깊이영상을 사용하여 탐색영역을 축소한다. 얼굴추적은 템플릿 매칭방법을 사용하며, 조기종료 기법을 사용하여 수행시간을 줄였다. 이 방법들을 구현하여 실험한 결과, 얼굴검출 방법은 기존의 방법에 비해 약 39%의 수행시간을 보였으며, 얼굴추적 방법은 640×480 해상도의 프레임 당 2.48ms의 추적시간을 보였다. 또한 검출율에 있어서도 제안한 얼굴검출 방법은 기존의 방법에 비해 약간 낮은 검출률을 보였으나, 얼굴로 인식하였지만 실제로는 얼굴이 아닌 경우의 오검출률에 있어서는 기존방법의 약 38% 향상된 성능을 보였다. 또한 얼굴추적 방법은 추적시간과 추적 정확도에 있어서 상보적인 관계를 가지며, 특별한 경우를 제외한 모든 경우에서 약 1%의 낮은 추적오차율을 보였다. 따라서 제안한 얼굴검출 및 추적방법은 각각 또는 결합하여 고속 동작과 높은 정확도를 필요로 하는 응용분야에 사용될 수 있을 것으로 기대된다.

Key Words : Adaboost algorithm, Haar-like Feature, face tracking, face detection, depth information, depth camera, Adaboost 알고리즘, Haar-like 형태, 얼굴 추적, 얼굴 검출, 깊이 정보, 깊이 카메라

ABSTRACT

This paper proposes a fast face detection and tracking method which uses depth images as well as RGB images. It consists of the face detection procedure and the face tracking procedure. The face detection method basically uses an existing method, Adaboost, but it reduces the size of the search area by using the depth image. The proposed face tracking method uses a template matching technique and incorporates an early-termination scheme to reduce the execution time further. The results from implementing and experimenting the proposed methods showed that the proposed face detection method takes only about 39% of the execution time of the existing method. The proposed tracking method takes only 2.48ms per frame with 640×480 resolution. For the exactness, the proposed detection method showed a little lower in detection ratio but in the error ratio, which is for the cases when a detected one as a face is not really a face, the proposed method showed only about 38% of that of the previous method. The proposed face tracking method turned out to have

※ 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행되었음. [KI002058, 대화형 디지털 홀로그램 통합서비스 시스템의 구현을 위한 신호 처리 요소 기술 및 SoC 개발
◆ 주저자 : 광운대학교, zauroum@kw.ac.kr, 준회원
* 목포해양대학교, 정회원
** 광운대학교, dwkim@kw.ac.kr
논문번호 : KICS2012-02-060, 접수일자 : 2012년 2월 14일, 최종논문접수일자 : 2012년 6월 11일

a trade-off relationship between the execution time and the exactness. In all the cases except a special one, the tracking error ratio is as low as about 1%. Therefore, we expect the proposed face detection and tracking methods can be used individually or in combined for many applications that need fast execution and exact detection or tracking.

Ⅰ. 서 론

최근 컴퓨터 기술의 급속한 발전으로 인해 기존의 텍스트 위주의 사용자 환경에서 벗어나 이미지, 그래픽, 오디오 및 비디오 데이터 등을 제공하는 멀티미디어 사용자 환경으로 변화하고 있다.인간생체의 일부를 검출하고 추적하는 방법은 컴퓨터 비전 분야를 비롯한 다양한 분야에서 오래전부터 연구되어 왔으며, 보안시스템, 화상회의, 로봇 비전, HCI(human-computer interface)에 의한 대화형 시스템, 스마트 홈 등에 널리 사용되고 있다[1,2]. 이 중 얼굴에 대한 연구가 가장 활발히 연구되어 왔으며[1~9], 그 목적은 빠르고 정확한 검출과 추적이었다.

기 제안된 얼굴검출 방법은 크게 지식-기반 방법, 특징-기반 방법, 템플릿 매칭(template matching) 방법, 외형-기반 방법(appearance-based methods)으로 분류할 수 있다[3]. 지식-기반 방법은 사람의 얼굴을 구성하는 눈, 코, 입 등의 기하학적인 특성을 파악하여 얼굴을 검출하는 방법[4]이다. 이 방법은 얼굴의 특징요소들에 대한 정보를 정확히 정의하기 어렵다는 단점이 있으며, 정의된 규칙의 엄격성과 검출/추적률 간의 상보적 관계를 보인다. 특징-기반 방법은 얼굴의 특징 성분인 얼굴요소[5], 질감 정보[6], 피부색[7-9], 또는 이들을 복합적으로 사용하여[10] 얼굴을 검출한다. 특히 피부색을 이용한 얼굴검출은 얼굴의 회전, 포즈 등의 변화에 독립적이므로 강인하다는 장점이 있으나, 조명의 영향을 많이 받는다는 단점이 있다. 템플릿 매칭 방법은 수동적으로 미리 대상이 되는 모든 얼굴에 대한 표준 얼굴패턴을 만들고 이를 입력영상과 비교하여 얼굴을 검출하는 방법[11]이다. 이 방법은 적용하기 쉽다는 장점이 있지만 거리에 따른 얼굴의 크기 변화나, 얼굴의 회전, 기울어짐 등에 민감하다는 단점이 있다. 외형-기반 방법은 학습영상 집합을 입력받아 훈련과정을 통해 학습된 모델을 이용하여 얼굴을 검출하는 방법이다. 이러한 방법으로는 주성분 분석에 의해 생성되는 고유얼굴(eigen face)을 이용하는 방법[12,13], 신경망과 서포트 벡터 머신(support vector machine)을 이용하는 방법[14] 등이 있다. 이 방법들은 복잡한 영상에서 얼굴을 검출하기 위해 훈련과정에서 얼굴 영상과 비 얼굴 영상을 입력받아 둘의 차이를 잘 나타낼 수 있는 특징들을 찾은 다음 이를 이용하여 얼굴을 검출한다. 그러나 이 방법들은 훈련 과정과 검출에 많은 계산량을 필요로 하고, 훈련과정에서 도출된 특징들에 대한 의존도가 높아 이 특징들의 정확성에 성능이 크게 좌우된다. 또한 이들 방법들의 특징을 복합적으로 사용하는 방법[15-17]도 발표되었으며, 이를 더욱 확장하는 등[18] 현재까지 이 방법이 가장 널리 사용되고 있다.

얼굴추적은 동영상으로 입력되는 영상 시퀀스에서 움직이는 사람의 얼굴을 검출하고 이동경로를 추적하는 것으로, 실시간 환경에서의 빠른 수행속도에 초점을 맞추어 연구되고 있다. 얼굴을 추적하는 가장 간단한 방법은 얼굴을 하나의 객체로 보고 객체에 해당하는 블록을 매칭시키는 방법[19]이다. 그 외 전처리 또는 수학적, 물리적 현상 등을 사용한 모델링 방법을 이용하여 동적인 배경에서 움직이는 물체를 분리하여 영상 내에서 가장 유사한 객체를 추적하는 방법이 있다[20,22]. 그리고 기존에 2차원 영상을 사용하던 방법과는 달리 3차원적 정보를 사용하기 위하여 3차원적 움직임[23,24]이나 스테레오 매칭에 의한 변이값[25,26]을 사용하는 방법들도 연구되었다. 또한 최근에는 깊이카메라[26] 또는 Microsoft 사의 Kinect[27]을 이용하여 깊이 정보를 실시간으로 획득할 수 있기 때문에 이들로부터 획득된 깊이정보를 얼굴검출 및 추적에 직접 사용하는 연구도 진행되고 있다[28,29]. [28]에서는 깊이정보 기반의 템플릿 정합을 이용하여 얼굴과 손을 검출하였다. 이때 두 종류의 깊이 카메라를 사용하고, 얼굴을 먼저 검출한 후에 이 정보를 바탕으로 손의 위치를 추적한다. [29]에서는 얼굴을 몇 가지 영역으로 분할한 후에 얼굴의 특징을 검출하는 방법으로 얼굴을 찾는 알고리즘을 제안하였다.

본 논문의 목적 또한 빠르고 정확한 얼굴검출 및 추적이다. 본 논문에서는 컬러정보와 깊이정보 모두를 사용하며, 깊이정보는 깊이카메라로부터 획득한다고 가정한다. 본 논문에서 제안하는 시스템은 얼굴 템플릿을 검출하는 검출부와 검출된 템플릿을 추적하는 추적부의 두 부분으로 구성된다. 얼굴 검출부에서는 기

본적으로 Adaboost 방법[15]을 사용하는데, 깊이영상으로 탐색영역을 국한시켜 검출시간을 줄인다. 얼굴추적부에서 검출된 얼굴의 명암성분을 템플릿으로 하여 블록 매칭, 즉 템플릿 매칭 연산을 이용하여 얼굴을 추적한다. 또한 본 시스템은 빠른 추적을 위하여 조기종료(early termination) 기법을 사용한다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문에서 얼굴탐색에 사용하는 Adaboost 방법을 설명하고, 3장에서는 본 논문에서 제안하는 깊이정보를 이용한 고속 얼굴검출 및 추적 알고리즘에 대하여 설명한다. 4장에서는 제안하는 알고리즘의 구현결과 및 성능평가에 대하여 기술하였으며, 마지막으로 5장에서 결론을 맺는다.

Ⅱ. Adaboost 알고리즘

본 장에서는 본 논문이 얼굴검출의 기본적인 방법으로 사용할 Adaboost 알고리즘[15]에 대해 설명한다. 이 방법은 외형적 특징을 기반으로 하는 방법이며, 이 특징들을 미리 선정하고 이들을 블록매칭 방법으로 찾아 얼굴을 인식하는 방법이다. 이 방법의 근간은 Haar 웨이블릿(wavelet) 함수와 같은 단순한 패턴을 객체인식에 사용하는 것이며[16], 이 패턴들은 그 뒤 더욱 확장되어(Haar-유사 특징, Haar-like features) 더욱 정확히 얼굴을 인식하도록 개선되었다[17].

Adaboost 방법은 얼굴의 각 부분을 단순한 패턴에 매칭시켜서 이들을 이용하여 얼굴을 인식하는 방법이다. 이 중 초기에는 에지 성분의 패턴을 사용하였으나, 후에 나머지 패턴들을 추가하여 확장하였다. 기본적으로 패턴 자체의 값은 0 또는 1의 값을 가지며, 그 크기는 탐색대상의 크기에 따라 변화시켜 사용한다.

영상과 패턴을 매칭시키는 과정에서 Adaboost 방법은 누적영상(integral imaging) 방법을 사용하는데 이 방법으로 계산된 화소의 값은 식 (1)과 같다.

$$ii(x,y) = \sum_{x' \le x,\, y' \le y} i(x',y') \qquad (1)$$

여기서 $i(x,y)$는 일반적인 영상의 (x,y)에서의 화소값이며, $ii(x,y)$는 누적영상의 (x,y)에서의 화소값이다. 즉, 누적영상의 화소값은 그 화소의 좌상위의 모든 화소값들을 누적덧셈한 결과값이다.

Adaboost 알고리즘은 선처리 과정인 훈련과정을 통하여 어떤 패턴을 어떻게 사용할 것인지를 미리 결정한다. 이 과정에서 다양한 얼굴영상과 비얼굴영상을 대상으로 각 패턴을 적용하고, 각 패턴을 특정 얼굴부위에 적용할 때 흰색과 검은색의 비중을 얼마로 하면 최적의 패턴매칭이 되는지를 결정한다. 이렇게 결정된 각 패턴에 대한 가중치와 그 패턴을 약분류기(weak classifier)로 정의한다.

실제로 얼굴검출에 사용할 때는 단일 약분류기만 사용하는 것이 아니라 약분류기를 단계적 다중구조(cascade structure)로 하여 강분류기(strong classifier)를 만들어 사용한다. [30]에서 강분류기는 단계적 다중구조로 결합된 약분류기의 수가 증가할수록 에러율이 0에 근접한다는 것을 증명하였다. 강분류기의 구조는 초기에 배경과 얼굴을 가장 잘 구별하는 약분류기를 사용하며, 점차 얼굴의 세세한 부분을 구분하는 약분류기를 추가하여 구성된다. 따라서 강분류기가 많은 수의 약분류기를 포함할수록 더 정확한 검출을 할 수 있지만, 더 많은 연산시간이 소요된다.

Ⅲ. 깊이정보를 이용한 얼굴검출 및 추적 알고리즘

본 장에서는 제안하는 얼굴검출 및 추적 방법에 대해 설명한다. 2차원의 RGB영상과 깊이영상을 사용하며, 깊이영상과 RGB영상의 해상도는 동일할 필요는 없으나 특정 시간에 두 영상에 포함되는 대상은 동일하다고 가정한다.

그림 2에 제안하는 얼굴검출 및 추적 알고리즘을 블록도로 나타내었다. 제안하는 방법은 얼굴검출 과정과 얼굴추적 과정의 두 과정으로 구성되어 있으며, 얼굴검출 과정은 기본적으로 한 장면(scene)의 초기에 한 번만 수행한다. 검출과정에서 검출된 얼굴영역은 얼굴추적 과정에서 템플릿(template)으로 사용되며, 얼굴이 검출된 다음 프레임부터는 추적과정만 수행된다. 그러나 장면이 바뀌거나 추적과정에서 템플릿 매칭이 이루어지지 않은 경우 다시 검출과정을 수행한다.

3.1. 얼굴검출 과정

앞에서 언급한 것과 같이, 본 논문의 얼굴검출 과정은 기본적으로 Adaboost 알고리즘을 사용한다. 그러나 Adaboost 알고리즘을 적용할 대상 영상의 크기를 줄여 연산시간을 줄이도록 하였다.

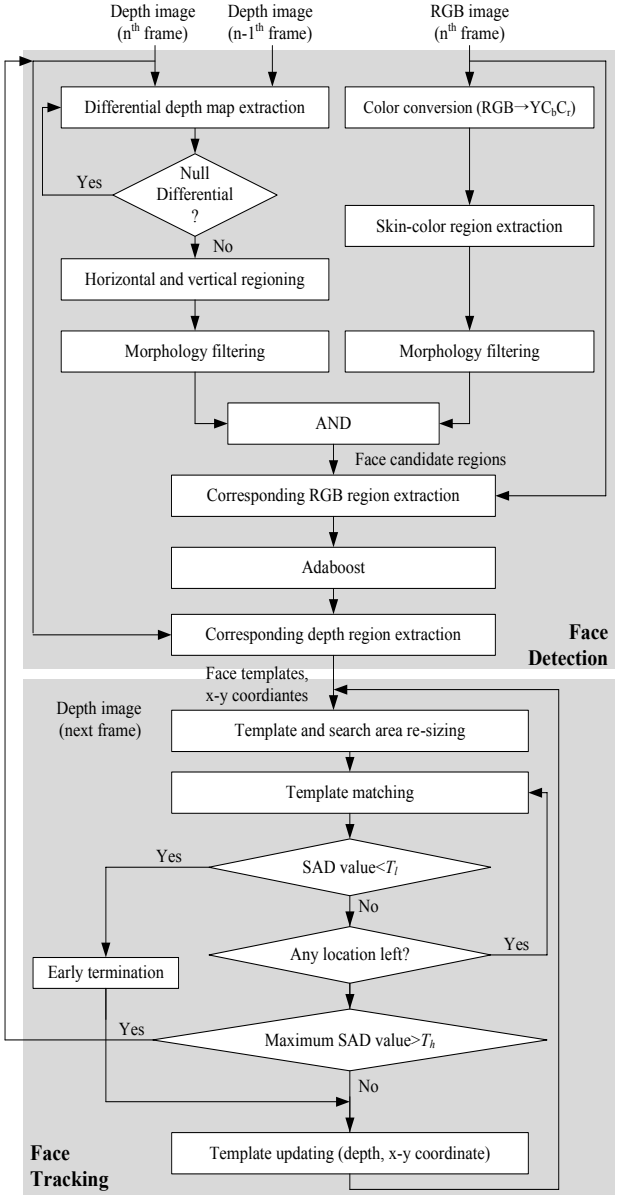


그림 1. 제안하는 얼굴검출 및 추적 알고리즘
Fig. 1. The proposed face detection and tracking algorithm

먼저, 입력된 깊이 동영상으로 움직임 영역을 검출한다. 이것은 이전 프레임과 현재 프레임간의 차이를 구하여 검출하며, 움직임이 전혀 없는 경우는 움직임이 있을 때까지 차영상을 반복하여 구한다. 일단 차영상이 구해지면 그 차영상을 수평방향과 수직방향으로 각각 누적덧셈을 수행한다. 그 결과는 하나의 행과 하나의 열로 나타나며, 각 화소의 값이 0이 아닌 화소들을 행과 열로 각각 확장하여 두 확장 영역의 교집합을 구한다. 그 결과는 모폴로지(morphology) 필터(erosion 필터와 dilation 필터)[33]를 사용하여 잡음과 같은 작은 영역들을 제거하며, 필터링된 영상은 이진화(binarization)한다.

얼굴검출은 얼굴의 색을 검출하는 방법을 사용하였는데, 일반적으로 얼굴색은 조명에 크게 의존하므로, 본 논문에서는 그 의존도를 낮추기 위해 영상포맷을 RGB에서 YCbCr로 바꾸어 사용하였다. 이 중 Y성분은 조명에 가장 민감하므로 Cb와 Cr성분만 사용한다. 본 논문에서 피부색으로 사용한 색의 범위는 [31]의 피부색 참조맵이며, 식 (2)와 같다.

$$B(x,y)=\begin{cases}1 \text{ if}(77\le C_b\le 127)\cap(133\le C_r\le 173)\\ 0\ Otherwise\end{cases} \quad (2)$$

이 식을 만족하는 영역을 추출하고, 그 결과에 모폴로지 필터를 적용하여 잡음과 같은 작은 영역들을 제거하며, 그 결과 역시 이진화한다.

깊이영상으로 획득한 영역과 RGB영상으로 획득한 영상의 교집합(AND)을 구하고, 이를 수평 및 수직방향으로의 누적덧셈하고 그 결과를 다시 전체 영상으로 확장하여 두 확장영역의 교집합을 구한다. 이 결과에 해당하는 RGB영상을 추출하여 그 결과 영역에 Adaboost 알고리즘을 적용함으로써 얼굴영역을 검출한다. 검출된 RGB영상의 Y성분을 얼굴추적 과정에서 템플릿(template)으로 사용하고, 해당 깊이영상을 보조 데이터로 사용한다. 아울러 템플릿의 좌표값도 추적과정으로 보낸다. 그림 3은 얼굴검출 과정의 각 단계별 예를 보이고 있다.

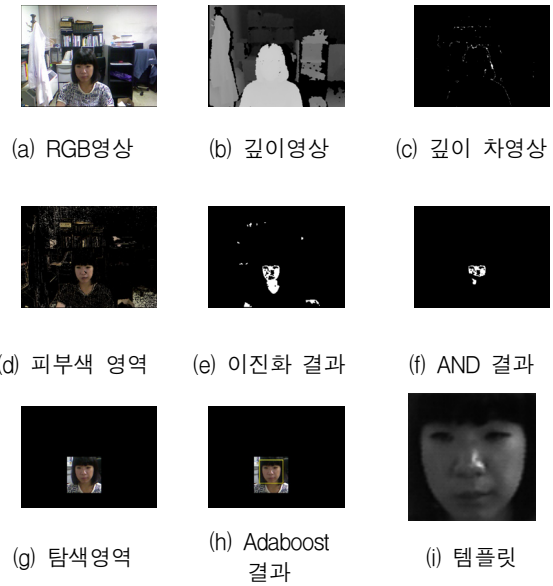


그림 2. 얼굴검출 과정의 단계별 예
Fig. 2. Example for each step in the face detection procedure

3.2. 얼굴추적 과정

얼굴추적 과정은 얼굴검출 과정에서 생성되거나 그 전 프레임에서 갱신(update)된 템플릿이 현재 영상에서 매칭되는 지점을 찾는 과정이다. 그러나 전체 프레임을 대상으로 템플릿 매칭을 수행하면 과다한 시간이 소요된다. 따라서 본 논문에서는 탐색영역을 최소화하는 방법을 제안하며, 얼굴이 상하좌우 뿐만 아니라 앞뒤로 움직이는 경우를 포함하도록 하였다.

3.2.1. 탐색범위의 결정

얼굴추적을 위해 탐색하여야 하는 영역은 얼굴의 움직임 속도와 관련이 있다. 본 논문에서는 추적할 얼굴이 전방의 영상물 등을 시청하고 있다고 가정한다. 따라서 얼굴은 거의 정면을 바라보고 있으며, 영상물을 시청하면서 움직이는 상황을 가정하였다. 얼굴의 움직임에 대해서는 시청하면서 움직일 수 있는 최대의 움직임 속도를 고려하여야 하며, 깊이에 반비례하여 얼굴의 움직임 양이 결정된다는 것을 고려하여야 한다. 거리에 따른 물체의 크기는 동일한 물체가 z_1과 z_2에 있을 때의 크기를 각각 s_1과 s_2라 할 때 식 (3)과 같은 관계를 갖는다.

$$s_2 = \frac{z_2}{z_1} s_1 \tag{3}$$

따라서 특정 깊이에서의 크기를 알면 그 물체가 깊이이동을 하였을 때의 크기를 식 (3)으로 쉽게 구할 수 있다. 물체의 깊이에 따라 움직임 속도, 즉 움직임 양이 달라지므로 깊이에 따른 움직임 양을 일반적으로 표현하기 위해서 본 논문에서는 현재의 템플릿에 대비한 상대적인 크기로 탐색영역을 정의한다.

본 논문에서는 먼저 얼굴의 최대 움직임 속도를 실험을 통하여 전후와 좌우방향에 대해 측정하였으며, 이를 두 프레임 간의 거리로 환산하였다. 이 값을 현재 템플릿에 대한 상대적인 크기로 변환하고, 다양한 거리에 대해 실험적으로 측정한 결과와 식 (3)에 의한 변환결과를 확인하였다. 이와 같이 결정한 탐색영역을 그림 4에 나타내었는데, 좌우 움직임은 각 방향으로 11.7% 범위 내에, 상하의 움직임은 7.2% 내에 각각 있었다.

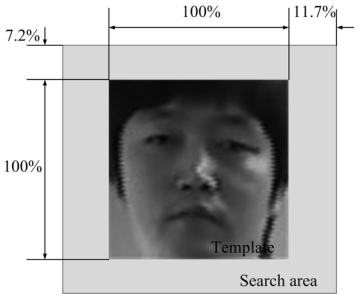


그림 3. 탐색범위
Fig. 3. Search range

3.2.2. 템플릿 및 탐색범위의 재결정

추적과정에서 얼굴이 전후로 움직일 때 깊이가 변화하기 때문에 현재의 템플릿을 사용할 경우 정확한 추적을 할 수 없을 뿐만 아니라 깊이가 변화함에 따라 탐색범위도 변화시켜야 한다. 템플릿의 크기와 탐색범위의 크기는 식 (3)으로 쉽게 변화시킬 수 있다. 그러나 이를 위해서는 전후로 얼굴이 움직인 양을 측정하여야 한다. 본 논문에서는 추적대상인 현재 프레임에서 그 전 프레임에서 만들어진 템플릿과 동일한 위치와 동일한 크기의 부분 깊이영상을 샘플링하여 사용한다. 측정방법은 템플릿과 샘플영상을 각각 $m \times n$ 서브블록으로 나누고, 각 서브블록 (i,j)의 깊이값 평균$\left(a_{i,j}\right)$을 계산하여 그 중 최고치를 각각 템플릿$\left(z_{temp}\right)$과 샘플링된 현재 프레임$\left(z_{cur}\right)$의 깊이값으로 선택한다. 이것을 식 (4)에 나타내었다.

$$z_{temp} = \max(a_{1,1}^{temp}, \ldots, a_{n,m}^{temp}),\ z_{cur} = \max(a_{1,1}^{current}, \ldots, a_{n,m}^{current}) \tag{4}$$

여기서 첨자 $temp$와 cur은 각각 템플릿과 샘플링된 현재 프레임을 표시하며, $\max(\ )$는 괄호내의 값 중 최대치를 선택한다. 식 (4)에 의해 두 깊이값이 측정되면 이를 사용하여 템플릿 및 탐색영역을 식 (3)에 의해 재조정한다. 그림 5은 가까이 다가선 경우에 대한 템플릿 갱신의 예를 보이고 있다.

(a) (b)

그림 4. 전방향 움직임에 대한 템플릿의 갱신: (a) 갱신 전, (b) 갱신 후
Fig. 4. Template update for a forward movement; (a) before, (b) after

3.2.3. 템플릿 매칭과 템플릿 갱신

얼굴추적의 다음 단계는 앞 절에서 재조정된 템플릿을 사용, 재조정된 탐색영역을 조사하여 매칭되는 점을 찾는 것이며, 기본적으로는 탐색영역 전체를 탐색한다. 이 경우 탐색하여야 하는 화소수는 $(s_h^{search} - s_h^{temp}) \times (s_v^{search} - s_v^{temp})$이며, 여기서 s_h^{temp}와 s_v^{temp}는 각각 템플릿의 수평 및 수직 방향 크기이며, 이에 해당하는 현재 프레임의 탐색 범위의 값은 각각 s_h^{search}과 s_v^{search}이다. 본 논문에서는 탐색할 때 비용함수로 SAD(sum-of-absolute differences)[33]를 사용한다. 즉, 탐색범위 내의 모든 위치를 탐색하여 그 중 가장 작은 SAD값을 갖는 위치를 선택한다.

그러나 본 논문의 목적이 고속추적이므로 이를 위한 방안을 제안한다. 일반적으로 위치를 추적하는 목적에 따라 조금의 오차도 없이 추적하는 경우보다는 어느 정도의 근사를 허용하는 경우가 많고, 또 정확한 위치를 찾는 것은 매우 많은 연산을 요한다. 또한 일적으로는 시청하면서 잘 움직이지 않으며, 움직인다고 해도 그 크기가 크기 않다. 또한 상하나 전후의 움직임보다 좌우방향의 움직임이 많다.

이런 점을 착안하여 본 논문에서는 조기종료(early termination) 기법을 사용한다. 이를 위해서 근사적으로 위치를 추적했다고 판단하여야 하는데, 본 논문에서 비용함수로 SAD를 사용하기 때문에 미리 정한 문턱치 SAD값 이하를 갖는 경우 조기종료를 시행한다. 이를 식으로 나타내면 식 (5)와 같다.

$$SAD_{i,j} < T_l \tag{5}$$

여기서 $SAD_{i,j}$는 위치 (i,j)에서의 SAD값이며, T_h는 미리 정한 문턱치이고 이것은 실험적으로 결정된다. 또한 탐색범위 내의 각 위치를 탐색하는 순서도 위에서 언급한 것에 부합하도록 그림 6과 같이 탐색범위의 중심에서부터 나선형으로 탐색하도록 하였다.

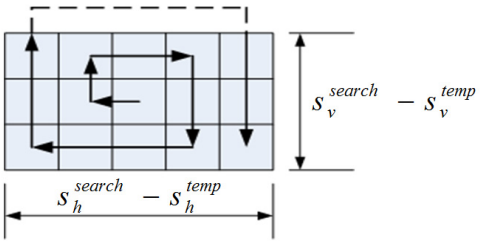


그림 5. 나선형 탐색
Fig. 5. Spiral search

3.2.4. 얼굴검출 과정으로의 귀환

앞에서 언급한 바와 같이 얼굴추적 과정을 수행하다가 템플릿 매칭에 실패하면 얼굴검출 과정으로 귀환하여 얼굴검출 과정부터 다시 수행한다. 이것은 장면이 바뀌거나 사람이 화면에서 사라지는 경우 등에 나타난다. 이와 같은 경우는 추적과정에서 SAD값이 매우 크게 나타나는데, 본 논문에서는 식 (6)과 같이 탐색범위 전체를 탐색한 후 각 위치에서의 SAD값의 최소값이 문턱치 T_h보다 큰 경우로 결정한다.

$$\min(SAD_{i,j} \; for \; all \; pixel \; positions \; within \; the \; seach \; range) > T_h \tag{6}$$

여기서 min( )는 괄호내의 값 중 최소치를 선택한다.

Ⅳ. 구현 및 실험

앞 장에서 설명한 본 논문에서 제안하는 방법을 구현하고 여러 테스트 시퀀스를 대상으로 실험을 수행하였다. 구현은 Microsoft window7 운영 체제에서 Microsoft visual studio 2008과 OpenCV Library 2.1[34]을 이용하였으며, 실험에 사용된 PC의 사양은 2.67GHz의 Intel Core i5 CPU와 4GB RAM이었다. 또한 RGB영상과 해당 깊이영상은 최근 Microsoft사에서 출시한 구조광 방식의 Kinect를 사용하여 640×480 해상도의 깊이정보와 동일 해상도의 컬러정보를 획득하여 사용하였다.

4.1. 실험 방법

3장에서 본 논문에서 제안한 방법을 얼굴검출과 추적과정으로 나누어 설명하였다. 그러나 얼굴추적을 위해서는 얼굴검출 과정을 한 번 거쳐야 하고, 또 본 논문에서 제안하는 얼굴검출 과정 역시 기존의 방법과의 성능비교가 필요하다. 이 검출과정의 비교대상은 Viola와 Jones의 방법, 즉 Adaboost 방법[15]을 택하였으며, 그 이유는 본 논문에서도 기

본적으로 이 방법을 사용하고 있고 또 현재 이 방법을 가장 널리 사용하고 있기 때문이다. 따라서 특정 비디오 시퀀스에 대해 Viola와 Jones 방법, 본 논문에서 제안한 얼굴검출 방법과 얼굴추적 방법을 모두 적용하여 그 성능을 비교하였다. 비교항목은 각 방법의 수행시간과 검출률이었다.

표 1에 실험에 사용한 테스트 시퀀스들을 나열하였다. 표에서 보듯이 자체 제작한 4개의 시퀀스를 사용하였는데, 이 중 '상하 움직임', '좌우 움직임', '전후 움직임' 시퀀스는 각 방향의 움직임에 대해 보다 정확하게 실험하기 위해서 각 방향으로의 움직임만을 포함하고 있으며, '윤진' 시퀀스는 자유로운 움직임을 포함하고 있다. 보다 객관적인 시퀀스를 사용하기 위해 ETRI에서 제작한 Lovebird1'을 테스트 시퀀스에 포함하였다. 이 영상은 두 사람이 멀리서부터 가까이로 다가오는 영상이며, 가까이 다가왔을 때의 얼굴이 상당히 크다. 이 외에도 RGB영상과 깊이영상을 동시에 제공하는 시퀀스들을 MPEG 등에서 제공하고 있으나, 이들 대부분은 사람 얼굴을 추적하는 테스트 시퀀스로 사용하기에는 적합지 않아 포함하지 않았다. 사용한 모든 테스트 시퀀스들은 각각의 비중을 균등하게 하기 위해서 동일한 프레임 수로 조정하여 사용하였다.

4.2. 실험 결과

4.2.1. 파라미터 결정

성능실험에 앞서 식 (5)와 (6)의 문턱치 파라미터 T_l과 T_h를 결정하기 위한 실험을 수행하였다. 이를 위해서는 얼굴추적 결과가 정확한지를 판단하는 기준을 설정하여야 하는데, 그림 7에 그 기준을 나타내었다. 본 논문에서는 제안한 얼굴검출 알고리듬의 결과를 정확한 추적결과로 간주하였으며, 여기에서 오검출이나 검출을 실패한 경우는 수작업으로 검출결과를 정확히 보정하여 사용하였다. 그림 9에서 사각형은 얼굴검출 결과의 얼굴 템플릿을 나타내고, A는 이 얼굴영역의 중심점, B는 추적한 얼굴의 중심점을 각각 나타낸다. 본 논문에서는 추적거리 오차율(tracking distance error ratio)을 식 (7)과 같이 정의하였다.

$$Tracing\ distance\ error\ ratio(\%) = \frac{\sqrt{p^2+q^2}}{\sqrt{X^2+Y^2}} \times 100 \quad (7)$$

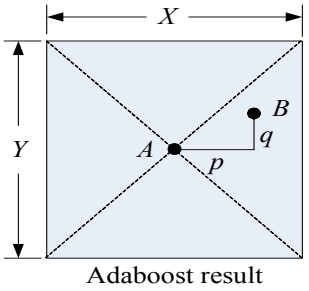


그림 6. 추적 결과의 정확성 판단 기준
Fig. 6. Criterion to find the exactness of the tracking result

문턱치 파라미터를 결정하기 위하여 이 추적거리 오차율을 다양하게 변화시키고, 동시에 조기종료 파라미터 T_l을 변화시키면서 실험하였는데, 그 결과를 그림 8에 나타내었다. 이 실험에서는 표 1의 자체 제작한 4개의 시퀀스만을 사용하였으며, 그림 8의 값들은 이 스퀀스들의 평균을 구한 것이다. 그림에서 보듯이 추적거리 오차율은 5%, 10%, 20%, 30%를 각각 사용하였으며, T_l은 0에서 250까지 측정하였다. 그림에서 좌측 종축은 각 추적거리 오차율보타 큰 오차를 보이는 결과의 비율을 나타내고, 우측 종축은 추적시간을 나타낸다. 그림에서 보듯이, 추적거리 오차율이 30%인 경우를 제외하고는 추적거리 오차율에 대한 측정결과는 비슷한 경향을 보였으며, $T_l = 40$까지는 거의 변화를 보이지 않다가 그 후 급격히 오차율(error

표 1. 실험에 사용된 테스트 시퀀스
Table 1. Test sequences used in the experiments

출처	시퀀스 이름	RGB영상 해상도	깊이영상 해상도	사용 프레임 수
자체 제작	상하 움직임	640×480	640×480	300
	좌우 움직임	640×480	640×480	300
	전후 움직임	640×480	640×480	300
	윤진	640×480	640×480	300
ETRI	Lovebird1	1,024×768	1,024×768	300

ratio)이 증가하였다. 또한 모든 경우 $T_l = 80$ 정도에서 오차율이 포화되는 것을 확인하였다. 따라서 본 논문에서는 추적거리 오차율을 20%로 고정하고, T_l을 20, T_h를 80으로 각각 결정하였다. 이 때 추적 오차율은 약 1.26%, 추적시간은 평균 2.54ms였다.

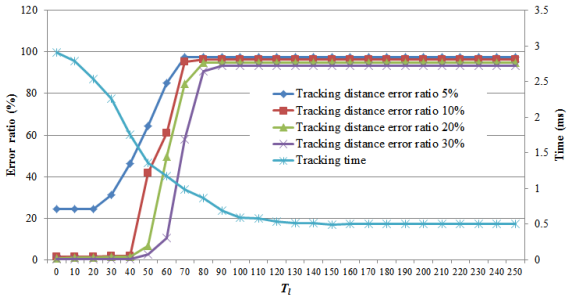


그림 7. 파라미터 결정을 위한 실험 결과
Fig. 7. Experimental results to determine the parameters

4.2.2. 수행시간 실험 결과

그림 9는 표 1의 '전후 움직임', '상하 움직임', 좌우 움직임' 시퀀스를 대상으로 수행시간을 측정한 결과를 그래프로 나타낸 것이다. 이 그림에서 얼굴추적에 대한 식 (5)와 (6)의 문턱치를 $T_l = 20$, $T_h = 80$, 그리고 추적거리 오차율을 20%로 설정한 결과이다. 이 그림에서 'Viola&Jones'는 [15]의 방법을 구현한 프로그램을 OpenCV Library 2.1에서 얻어 사용한 결과이며, 'Proposed face detection method'는 3장에서 제안한 얼굴검출 방법에서 탐색영역을 축소시키는 방법을 적용한 결과에 OpenCV Library 2.1에서 얻은 프로그램을 적용한 결과이다. 'Proposed face tracking method'는 첫 번째 프레임에 대해서 제안한 얼굴검출 방법을 적용하고, 나머지 프레임은 제안한 얼굴추적 방법을 직접 구현하여 적용한 결과이다. 얼굴추적은 그림 9에서 정량적으로 확인할 수 없기 때문에 이 결과만 그림 10에 확대하여 나타내었다. 본 실험에서 Viola&Jones와 제안한 알고리즘이 사용한 Adaboost 알고리즘의 파라미터는 동일하다.

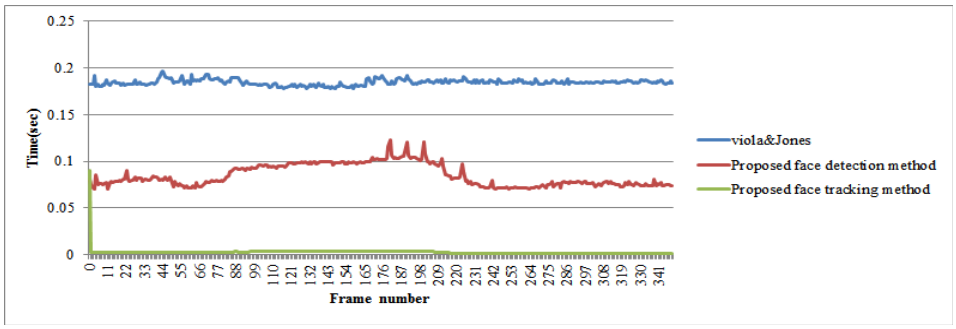
그림에서 보듯이 세 종류의 움직임 모두에서 Viola와 Jones의 방법은 약간의 변화가 있으나 대체로 수행시간이 비슷한 것을 볼 수 있다. 이것은 Viola와 Jones의 방법이 움직임에 상관없이 모든 프레임에서 영상 전체를 탐색하기 때문이다. 반면, 제안한 얼굴검출 방법은 움직임의 종류에 따른 수행시간의 변화가 뚜렷이 나타난다. (a)에서 앞으로 움직인 경우(약 70번째 프레임부터 190번째 프레임까지) 템플릿과 탐색범위가 증가하여 수행시간이 길어지고, 뒤로 움직인 경우(약 190번째 프레임부터 240번째 프레임까지) 그 반대가 되어 수행시간이 줄어드는 것을 볼 수 있다. 그림 10 (b)는 좌우로 연속적으로 움직인 경우인데, 움직임이 많은 부분과 적은 부분이 뚜렷이 나타나고 있다. (c)의 경우는 상하로 움직인 경우로, 서 있는 상태에서 약 100번째 프레임부터 150번째 프레임까지는 앉은 경우이고, 약 150번째 프레임부터 200번째 프레임까지는 다시 일어서는 경우이다. 앉는 경우 수행시간이 줄어드는데, 이것은 신체가 움직이는 영역이 점차 줄어들어 탐색영역이 줄어들기 때문이다.

한편, 제안한 얼굴추적의 경우 초기의 프레임에서는 제안한 얼굴검출 과정을 거치기 때문에 모든 그래프에서 얼굴검출과 같은 수행시간을 보인다. 그러나 일단 추적과정으로 들어가면 모든 움직임에서 그 수행시간이 상당히 줄어드는 것을 볼 수 있다.

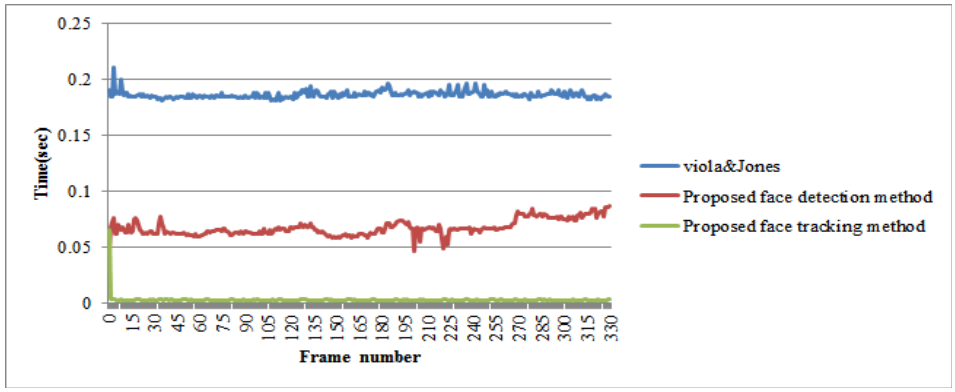
그림 9와 10의 얼굴추적에 대한 수행시간 실험 결과에서는, 전후로 움직이는 경우(a) 가장 많은 시간이 소요되어 최고 프레임 당 4ms의 추적시간을 보였으며, 좌우 움직임의 경우(b)는 2ms가 조금 넘는 정도의 추적시간을 거의 변화없이 소요함을 알 수 있다. 또한 상하의 움직임에 대해서도 다소간의 차이가 있지만 3ms 이내의 추적시간을 보임을 알 수 있다.

전후 움직임의 경우 그림 9의 제안한 얼굴검출 방법에서와 같이 앞으로 움직이는 경우 추적시간이 늘어나는 것을 볼 수 있는데, 이것은 템플릿과 탐색범위를 확대하는 시간과 나선형 탐색시간이 늘어나는 것 때문이다. 반면 뒤로 움직이는 경우는 오히려 추적시간이 줄어드는데, 이것은 템플릿과 탐색범위를 조정하는 시간이 늘어나는 것보다 추적 자체에서 감소하는 시간이 더욱 크기 때문이다. 그림 10(b)에서 보는 바와 같이 좌우의 움직임은 추적시간에 거의 영향을 미치지 못함을 알 수 있으며, 상하의 움직임에 해당하는 (c)의 경우는 움직임에 대하여 오히려 추적시간이 감소하는 것으로 나타났다. 이것은 탐색범위의 설정으로 상하나 좌우의 움직임이 충분히 포함되어 있다는 것을 보여주는 것이다.

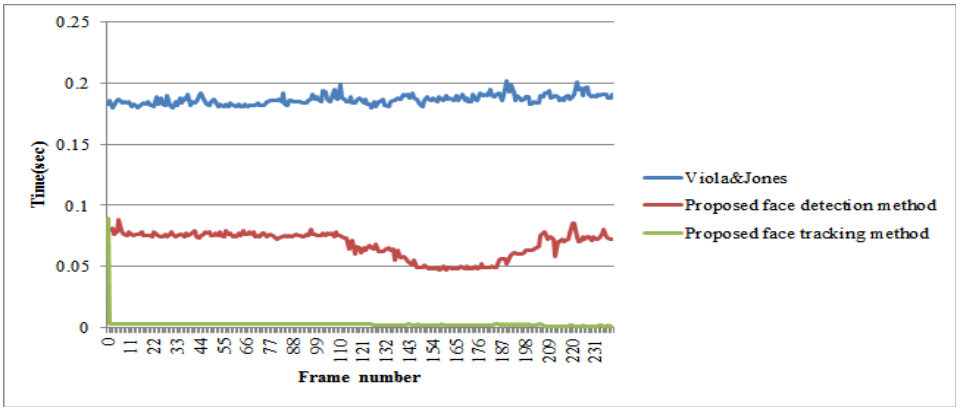
표 2는 표 1의 테스트 시퀀스에 프레임 당 평균 수행시간을 나타내었다. 먼저 Viola와 Jones의 방법은 모든 프레임에서 전체화면을 검색하기 때문에 거의 같은 시간을 보였으며, 표에서 보이는 차이는 컴퓨터 내부적인 요인으로 판단된다. Viola와 Jones



(a)



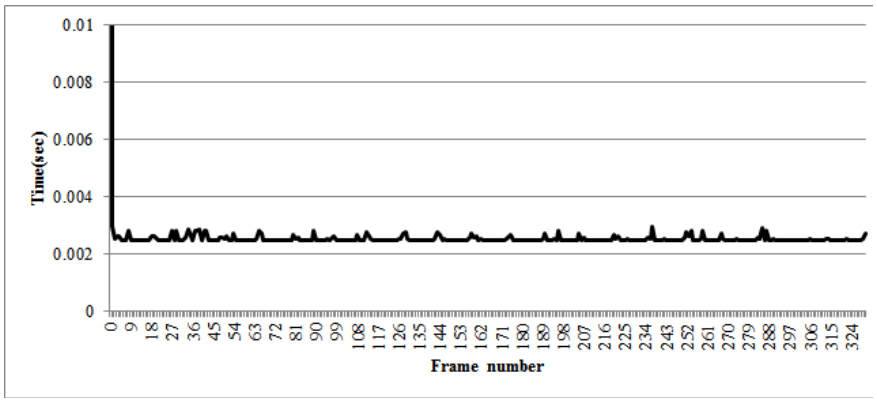
(b)



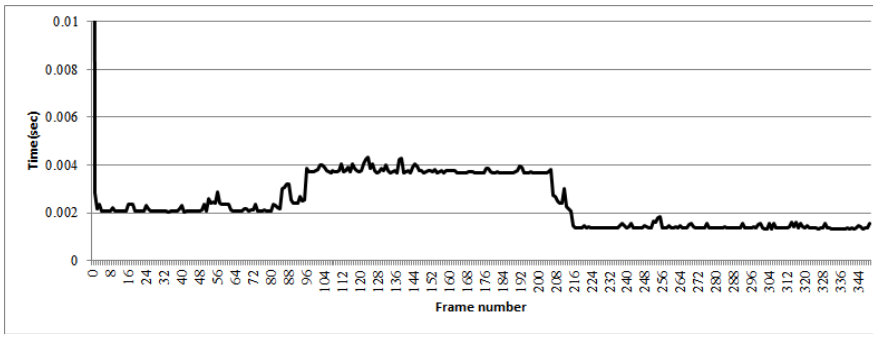
(c)

그림 8. 수행시간에 대한 실험 결과의 예; (a) 전후 움직임, (b) 좌우 움직임, (c) 상하 움직임.
Fig. 8. Examples of the experimental results on the execution time for the movement of; (a) back-and-forth, (b) left-right, (c) up-down.

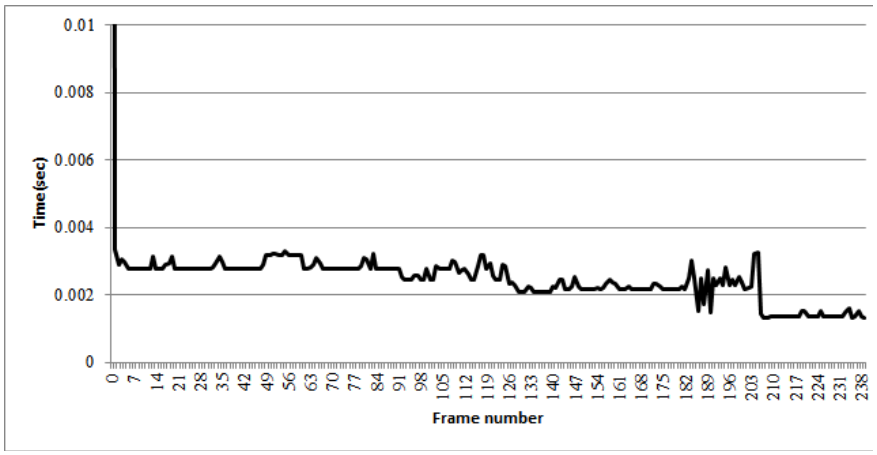
의 방법에 비해 제안한 얼굴검출 방법은 약 38.77%의 수행시간이 소요되어 얼굴검출 그 자체로도 상당한 시간을 절약할 수 있음을 알 수 있다. 앞에서 설명한 것과 같이 전후 움직임의 경우 탐색영역이 증가하였다가 되돌아오기 때문에 검출시간이 다소 증가하였으며, 상하 움직임의 경우는 탐색영역이 감소하였다가 되돌아오기 때문에 오히려 감소하였다. 상하, 좌우, 전후 움직임과 윤진 시퀀스는 대부분 2.5ms 전후의 추적시간을 보여 빠른 추적이 가능하다는 것을 확인할 수 있었다. Lovebird1는 화면의 해상도가 다른 영상에 비해 높지만, 그 해상도를 감안하더라도 상대적으로 높은 추적시간을 보였다. 이것은 사람이 가까이로 다가오면서 얼굴의 크기가 상당히 커지기 때문이다.



(a)



(b)



(c)

그림 9. 제안한 얼굴추적 방법의 수행시간 측정 결과; (a) 전후 움직임, (b) 좌우 움직임, (c) 상하 움직임.
Fig. 9. Examples of the experimental result on the execution time of the proposed face tracking method for the movement of; (a) back-and-forth, (b) left-right, (c) up-down.

표 2. 프레임 당 평균 수행시간 비교
Table 2. Comparison for average execution time per frame

시퀀스	Viola & Jones	제안한 얼굴검출 방법	제안한 얼굴추적 방법
상하 움직임	188.27ms	70.66ms	2.58ms
좌우 움직임	186.43ms	74.84ms	2.25ms
전후 움직임	187.72ms	80.76ms	2.85ms
윤진	187.15ms	75.24ms	2.61ms
Lovebird1	477.52ms	157.31ms	8.42ms

4.2.3. 검출 및 추적 오차율 실험결과

표 3에는 Viola와 Jones의 얼굴추적 방법과 본 논문에서 제안한 얼굴검출 방법의 검출률 통계를 보이고 있는데, 얼굴검출과 얼굴추적은 특성이 매우 다르므로, 여기서는 얼굴검출만 언급하고 얼굴추적은 뒤에서 따로 설명한다. 표에서 '검출성공율'은 검출된 결과의 적합여부와 상관없이 검출에 성공했다고 나타나는 경우이며, '검출실패율'은 검출을 하지 못했다고 나타는 경우이다. 따라서 검출성공율과 검출실패율을 더하면 100%가 된다. 또한 오검출률은 얼굴로 검출되었으나 실제의 경우 얼굴이 아닌 경우(negative true)에 대한 비율이다. 그림 11에 이런 경우의 예를 보이고 있는데, (a)는 Viola와 Jones의 방법, (b)는 같은 프레임을 제안한 얼굴검출 방법을 수행한 결과이다. (a)에서 얼굴 좌하측에 얼굴이 아닌 영역을 얼굴로 인식하고 있는 것을 볼 수 있다.

표 3에서 보듯이 제안한 얼굴검출 방법이 Viola와 Jones의 방법보다 검출률이 약간 낮은 것을 볼 수 있는데, 이것은 본 논문의 방법이 움직임에 의해 탐색영역을 제한하였기 때문이다. 그러나 오검출률의 경우 제안한 검출방법이 약 5.7% 작게 나타나 우수한 결과를 보이고 있음을 확인할 수 있었다.

표 3. 평균 얼굴검출률 비교
Table 3. Comparison for average face detection ratios

방법	검출 성공율(%)	검출 실패율(%)	오검출률 (%)
Viola&Jones	98.66	1.34	15.05
제안한 얼굴검출 방법	98.61	1.39	9.36

제안한 얼굴추적 방법에 대한 오차율을 표 4에 보이고 있다. 이 통계는 추적거리 오차율을 20%로 식 (6)의 T_h를 80으로 고정한 것이며, 비교를 위해 $T_l = 10, 20, 30$의 세 경우를 나타내었다. 그림 10에서 예측한 대로 T_l이 증가할수록 추적시간은 감소하나 추적 오차율이 높아져, 추적시간과 오차율이 상보관계에 있음을 알 수 있다. 또한 전후 움직임의 경우 가장 높은 오차율을 보였는데, 이것은 템플릿 자체를 변경하여야 하기 때문인 것으로 분석되었다. Lovebird1의 경우 얼굴의 크기가 점차 커져서 가까이에 다가왔을 때는 상당히 크고, 중간에 팔로 얼굴을 가리는 부분이 있어서 오차율이 상대적으로 높은 것으로 나타났다. Lovevird1을 제외하면 모든 경우에 있어서 오차율은 1% 전후를 보여 제안한 얼굴추적 방법이 상당히 높은 정확도를 가짐을 알 수 있었다.

그림 10. 오검출의 예; (a) Viola와 Jones의 방법, (b) 제안한 얼굴추적 방법
Fig. 10. Example of negative-true detection; (a) Viola and Jones method, (b) proposed face detection method.

표 4. 제안한 얼굴추적 방법의 추적 오차율
Table 4. Tracking error ratio of the proposed face tracking method

시퀀스 이름	$T_l = 10$	$T_l = 20$	$T_l = 30$
상하 움직임	0.82%	0.85%	0.86%
좌우 움직임	0.91%	1.21%	1.52%
전후 움직임	1.11%	1.26%	1.41%
윤진	0.94%	1.19%	1.48%
Lovebird1	0.50%	2.00%	3.00%
평균	0.86%	1.30%	1.65%

V. 결 론

본 논문에서는 RGB영상과 해당 깊이정보를 사용하여 고속으로 얼굴을 검출하고 추적하는 방법을 제안하였다. 얼굴검출 방법은 기본적으로 기존이

Adaboost 방법을 사용하나, 깊이정보를 사용하여 탐색해야 하는 영역을 축소함으로써 수행시간을 감소시켰다. 얼굴추적 방법은 템플릿 매칭 방법을 사용하며, 깊이정보를 이용하여 상하, 좌우, 전후의 얼굴 움직임을 추적할 수 있도록 탐색범위를 설정하였다. 또한 얼굴의 전후 움직임에 대한 깊이 변화에 템플릿과 탐색범위의 크기를 조정하도록 하였다. 또한 얼굴추적 방법에서 나선형 탐색에 의한 조기종료 기법을 도입하여 수행시간을 줄였다.

직접 제작한 여러 종류의 동영상과 MPEG의 테스트 시퀀스 한 개에 제안한 방법과 기존 방법 중 대표적인 Viola와 Jones의 방법을 적용하여 성능평가를 수행하였다. 제안한 얼굴검출 방법은 Viola와 Jones의 방법에 비해 약 39% 정도의 시간을 소요하였으며, 제안한 얼굴추적 방법은 640×480 해상도의 프레임 당 평균 약 2.5ms의 시간을 소요하였다. 검출율에 있어서도 제안한 얼굴검출 방법이 Viola와 Jones의 방법보다 약간 떨어졌으나, 오검출율에 있어서는 제안한 방법이 상당히 낮은 것을 확인하였다. 또한 제안한 얼굴추적 방법은 조기종료를 위한 문턱치의 변화에 따른 추적 정확도와 추적시간은 예상한 바와 같이 상보적인 관계가 있음을 확인하였으며, 특별한 경우를 제외한 모든 경우 약 1% 정도의 낮은 오차율을 보임을 확인하였다.

References

[1] G, Q, Zhao, et al., "A Simple 3D face Tracking Method based on Depth Information," Int'l Conf. on Machine Learning and Cybernetics, pp. 5022-5027, Aug. 2005.

[2] C. X. Wang and Z. Y. Li, "A New Face Tracking Algorithm Based on Local Binary Pattern and Skin Color Information," ISCSCT, Vol. 2, pp. 20-22, Dec. 2008.

[3] M. H. Yang, et al., "Detecting Faces in Images; A Survey," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, pp. 34-58, Jan. 2002.

[4] G. Z. Yang and T. S. Huang, "Human Face Detection in Complex Background," Pattern Recognition, Vol. 27, No. 1, pp. 53-63, Jan. 1994.

[5] K. C. Yow, R. Cipolla, "Feature-Based Human Face Detection," Image and Vision Computing, Vol. 15, No. 9, pp. 713-735, Sept. 1997.

[6] Y. Dai and Y. Nakano, "Face-texture Model based on SGLD and its Application in Face Detection in a Color Scene," Pattern Recognition, Vol. 29, No. 6, pp. 1007-1017, June 1996.

[7] J. Yang and A. Waibel; "A Real-Time Face Tracker," WACV'96, pp. 142-147, 1996.

[8] S. J. McKenna, S. Gong, and Y. Raja; "Modelling Facial Colour and Identity with Gaussian Mixtures," Pattern Recognition, Vol. 31, No. 12, pp. 1883-1892, 1998.

[9] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A Survey of Skin-color Modeling and Detection Methods," Pattern Recognition, Vol. 40, pp. 1106–1122, Mar. 2007.

[10] R. Kjeldsen and J. Kender, "Finding Skin in Color Images," Proc. Second Int'l Conf. Automatic Face and Gesture Recognition, pp. 312-917, 1996.

[11] L. Craw, D. Tock, and A. Bennett, "Finding Face Features," Proc. Second European Conf. Computer Vision, pp. 92-96, 1992.

[12] M. Turk and A. Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, Vol. 3, pp. 71-86, 1991.

[13] P. N. Belhumeur, et al., " Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp. 711-720, 1997.

[14] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: an Application to Face Detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 130-136, 1997.

[15] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," Computer Vision, Vol. 52, No. 2, pp. 137-154, 2004.

[16] C. P. Papageorgiou, M. Oren, and T. Poggio, "A General Framework for Object Detection," IEEE Int'l Conf. Computer

Vision, pp. 555-562, 1998.

[17] R. Lienhart and J. Maydt, ″An Extended Set of Haar-like Features for Rapid Object Detection,″ Int'l Conf. Image Processing, Vol. 1, pp. 22-25, Sept, 2002.

[18] K. An and M. Chung, "Cognitive face analysis system for future interactive TV," IEEE Trans. Consumer Electronics, Vol. 55, No. 4, pp. 2271-2275, Nov. 2009.

[19] K. Hariharakrishnan and D. Schonfeld, "Fast object tracking using adaptive block matching," IEEE Trans. Multimedia, vol. 7, no. 5, 2005.

[20] M. Lievin and F. Luthon; ″Nonlinear Color Space and Spatiotemporal MRF for Hierarchical Segmentation of Face Features in Video,″ Proc. IEEE Int'l Conf. Image Processing, pp. 63-71, 2004.

[21] Y. Lin et al., "Real-time Tracking and Pose Estimation with Partitioned Sampling and Relevance Vector Machine," IEEE Intl. Conf. Robotics and Automation, pp. 453-458, 2009.

[22] A. An and M. Chung, "Robust Real-time 3D Head Tracking based on Online Illumination Modeling and its Application to Face Recognition," IEEE Intl. Conf. Intelligent Robots and Systems, pp. 14661471, 2009.

[23] R. Okada, Y. Shirai, and J. Miura, ″Tracking a Person with 3-D Motion by Integrating Optical Flow and Depth,″ Proc. Fourth IEEE Int'l Conf. Automatic Face and Gesture Recognition, pp. 336-341, 2000.

[24] G. Zhao, et al., "A Simple 3D Face Tracking Method based on Depth Information," Intl Conf. Machine Learning and Cybernetics, pp. 5022-5027, 2005.

[25] Y. H. Lee et al., ″A Robust Face Tracking using Stereo Camera,″ SICE Annual Conf., pp. 1985-1989, Sept. 2007.

[26] S. Kosov et al., "Rapid Stereo-vision Enhanced Face Recognition," IEEE Intl. Conf. Image Processing, pp. 2437-2440, Sept. 2010.

[27] J. L. Wilson, Microsoft kinect for Xbox 360, PC Mag. Com, Nov. 10, 2010.

[28] S. Xavier, R.-H. Javier, and Josep R. Casas, ″Real-Time Head and Hand Tracking Based on 2.5D Data,″ IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 14, NO. 3, JUNE 2012

[29] Maurício Pamplona Segundo, Luciano Silva, Olga Regina Pereira Bellon, and Chauã C. Queirolo, ″Automatic Face Segmentation and Facial Landmark Detection in Range Images,″ IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS－PART B: CYBERNETICS, VOL. 40, NO. 5, OCTOBER 2010

[30] M. Hacker, et al., ″Geometric Invariants for Facial Feature Tracking with 3D TOF Cameras,″ Int'l Symposium on Signals, Circuits and Systems, Vol. 1, pp. 1-4, 2007.

[31] Douglas Chai, et al., ″Locating Facial Region of a Head-and -Shoulders Color Image,″ Int'l Conf. Automatic Face and Gesture Recognition, pp. 124-129, April 1998.

[32] Y. Freund and R. E. Schapire, ″A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,″ J. Computer and System Sciences, Vol. 55, pp. 119-139, 1997.

[33] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd Ed., Pearson Prentice Hall, Upper Saddle River, NJ, 2008.

[34] http://sourceforge.net/projects/opencvlibrary/files/

배 윤 진 (Yun-Jin Bae)

2010년 2월 광운대학교 전자재료공학과 졸업(공학사)

2012년 2월 광운대학교 일반대학원 졸업(공학석사)

<관심분야> 홀로그램, SVC, Tracking, HCI, Stereo Vision

최 현 준 (Hyun-Jun Choi)

2003월 2월 광운대학교 전자재료공학과 졸업(공학사)
2005년 2월 광운대학교 일반대학원 졸업(공학석사)
2009년 2월 광운대학교 일반대학원 졸업(공학박사)
2009년 3월~2010년 2월 광운대학교 연구교수
2010년 3월~2011년 8월 안양대학교 정보통신공학과 조교수
2011년 9월~현재 목포해양대학교 전자공학과 전임강사
<관심분야> 영상압축, 워터마킹, FPGA/ASIC 설계, 암호학, Design Methodology

서 영 호(Young-Ho Seo)

1999년 2월 광운대학교 전자재료공학과 졸업(공학사)
2001년 2월 광운대학교 일반대학원 졸업(공학석사)
2004년 8월 광운대학교 일반대학원 졸업(공학박사)
2005년 9월~2008년 2월 한성대학교 조교수
2008년 3월~현재 광운대학교 교양학부 부교수
<관심분야> 실감미디어, 2D/3D 영상 신호처리, 디지털 홀로그램, SoC 설계

김 동 욱 (Dong-Wook Kim)

1983년 2월 한양대학교 전자공학과 졸업(공학사)
1985년 2월 한양대학교 공학석사
1991년 9월 Georgia공과대학 전기공학과(공학박사)
1992년 3월~현재 광운대학교 전자재료공학과 정교수
2009년 3월~현재 광운대학교 실감미디어 연구소 연구소장
<관심분야> 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP설계, Wireless Communication