깊이정보를 이용한 템플릿 매칭 기반의 효율적인 얼굴 추적 알고리즘

*김우열 **서영호 ***김동욱

광운대학교

*wykim@kw.ac.kr

Template Matching-based Efficient Face Tracking Algorithm using Depth Information

*Kim, Woo-Youl **Seo, Young-Ho ***Kim, Dong-Wook

Kwangwoon University

요약

본 논문에서는 키넥트 센서의 RGB영상과 깊이영상을 사용하여 얼굴을 검출하고, 검출 된 템플릿을 이용하여 얼굴을 추적하는 방법을 제안한다. 얼굴검출은 기본적으로 기존의 Adaboost 방법을 사용하나, 깊이정보와 피부색을 사용하여 탐색영역을 최대한 축소하여 수행시간 및 오검출율을 줄였다. 그리고 얼굴추적은 깊이정보를 이용하여 템플릿의 크기, 탐색영역을 조정하였다. 또한, RGB영상보다 조명변화에 강한 깊이영상을 이용하여 효율적인 템플릿 매칭을 하였다.

1. 서론

인간생체의 일부를 검출하고 추적하는 방법은 컴퓨터 비전분야를 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, 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. 특징-기반 방법은 얼굴의 특징 성분인 얼굴요소, 질감 정보, 피부색, 또는 이들을 복합적으로 사용하여 얼굴을 검출한다. 템플릿 매칭 방법은 수동적으로 미리 대상이 되는 모든 얼굴에 대한 표준 얼굴패턴을 만들고 이를 입력영상과 비교하여 얼굴을 검출하는 방법[5]이다. 외형-기반 방법은 학습영상 집합을 입력받아 훈련과정을 통해 학습된 모델을 이용하여 얼굴을 검출하는 방법이다.

얼굴 추적은 동영상으로 입력되는 영상 시퀀스에서 움직이는 사람의 얼굴을 검출하여 이동 경로를 추적하는 것으로 실시간 환경에서의 빠른 수행속도에 초점을 맞추어 연구가 진행되고 있다. 얼굴을 추적하는 방법에는 기존에 2차원 영상을 사용하던 방법과는 달리 3차원적 정보의 깊이 값을 이용하는 방법들도 연구가 진행되고 있다. 또한 최근에는 깊이카메라 또는 Microsoft사의 KINECT를 이용하여 깊이정보를 실시간으로 획득하여 얼굴 검출 및 추적에 직접 사용하는 연구도 진행되고 있다.

본 논문에서는 KINECT 센서를 이용하여 얼굴을 검출하고 추적하는 알고리즘을 제안한다. 본 논문은 다음과 같이 구성된다. 2장에서는 제안하는 알고리즘에 대하여 설명하고, 3장에서는 실험결과를 기술하고, 4장에서 결론을 맺는다.

2. 제안하는 알고리즘

본 장에서는 본 논문에서 제안하는 얼굴을 검출하여 효율적으로 얼굴을 추적하는 방법에 대하여 설명한다.

2.1. 얼굴 검출 방법

얼굴검출에서는 RGB영상과 깊이영상을 사용한다. 그림 1은 제안하는 얼굴검출 알고리즘을 블록도로 나타내었다.

본 논문에서 제안하는 얼굴 검출 방법은 움직임 검출과 피부색 영역에 대한 두 가지 정보를 가지고, 가능한 네 가지 경우로 분류하였다. 얼굴검출은 초기에 한 번만 수행하며, 얼굴이 검출된 다음 프레임부터는 추적과정만 수행된다. 그러나 장면이 바뀌거나 추적과정에서 템플릿 매칭이 이루어지지 않은 경우 다시 검출과정을 수행한다. 본 논문의 얼굴검출 방법은 기본적으로 Adaboost 알고리즘을 사용한다. 그러나 본 논문에서는 Adaboost 알고리즘에 적용할 영상의 크기를 움직임 검출과 피부색 영역을 이용하여 상당히 많은 부분을 줄여 연산시간을 감소시켰다.

2.1.1. 움직임 영역 검출

먼저, 입력된 깊이영상으로 움직임 영역을 검출한다. 이것은 이전 프레임과 현재 프레임간의 차이를 구하여 검출한다. 일단 차영상이 구해지면 그 차영상을 수평방향과 수직방향으로 각각 누적덧셈을 수행한다. 그 결과는 하나의 행과 하나의 열로 나타나며, 각 화소의 값이

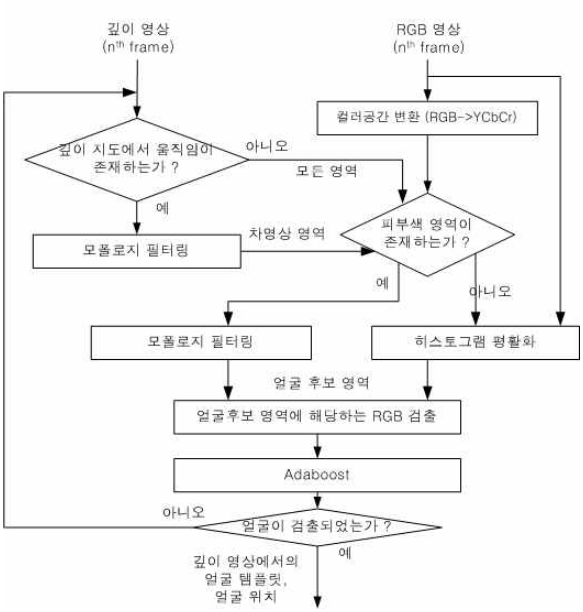

그림 1. 제안하는 얼굴검출 알고리즘

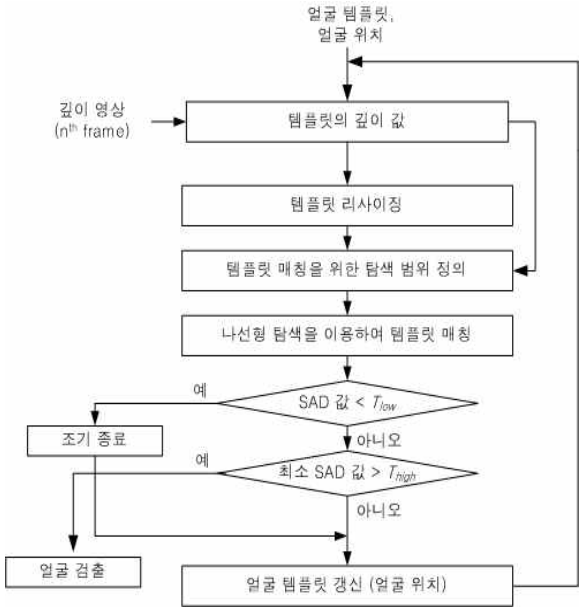

그림 2. 제안하는 얼굴추적 알고리즘

0이 아닌 화소들을 행과 열로 각각 확장하여 두 확장 영역의 교집합을 구한다.

2.1.2. 피부색 영역 검출

피부색 영역에 대한 검출은 앞에서 정의한 움직임 영역에 대해서만 검출하였다. 만약 움직임 영역이 검출되지 않았을 때는 모든 영역에 대해서 피부색을 검출하였다.

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 [6]의 피부색 참조맵이며, 식 (1)와 같다.

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

만약 조명변화로 인해 영상이 어둡거나 할 때, 피부색을 검출하지 못하는 경우가 있다. 이때는 히스토그램 평활화(Histogram Equalize)를 이용하여 피부색을 검출하였다.

2.2. 얼굴추적 방법

얼굴추적에서는 깊이영상 만을 사용한다. 그림 2은 제안하는 얼굴추적 알고리즘을 블록도로 나타내었다.

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

2.2.1. 템플릿 크기 조절

추적과정에서 얼굴이 전후로 움직일 때 깊이가 변화하기 때문에 현재의 템플릿을 사용할 경우 정확한 추적을 할 수 없다. 따라서 깊이가 변화함에 따라 템플릿의 크기도 변화시켜야 한다. 그림 3은 깊이에 따른 템플릿의 크기를 측정한 결과이다. 물체가 깊이 z에 있을 때의 템플릿의 크기를 s라 하면 그림 3의 결과를 fitting하여 식 (2)과 같은 관계를 얻을 수 있다.

$$s = 0.00004z^3 - 0.0165z^2 + 2.3613z - 84.032 \quad (2)$$

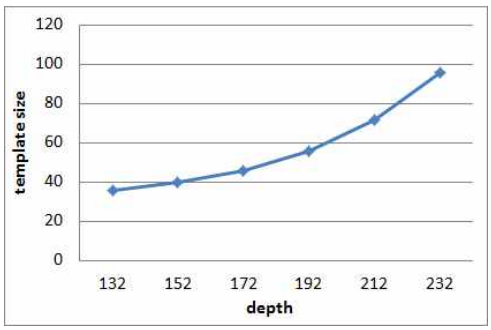

그림 3. 깊이에 따른 템플릿의 크기

그러나 이를 위해서는 전후로 얼굴의 깊이 값을 측정하여야 한다. 본 논문에서는 그 전 프레임에서 만들어진 템플릿 사용한다. 측정방법은 템플릿을 각각 $m \times n$ 서브블록으로 나누고, 각 서브블록 (i,j)의 깊이 값 평균$(a_{i,j})$을 계산하여 그 중 최고치를 템플릿(z_{temp})의 깊이 값으로 선택한다. 이것을 식 (3)에 나타내었다.

$$z_{temp} = \max(a_{1,1}^{temp},\ldots,a_{n,m}^{temp}) \quad (3)$$

여기서 첨자 $temp$은 템플릿임을 표시하며, max( )는 괄호내의

값 중 최대치를 선택한다. 식 (3)에 의해 깊이 값이 측정되면 이를 사용하여 템플릿을 식 (2)에 의해 조정한다.

2.2.2. 탐색범위의 결정

얼굴추적을 위해 탐색하여야 하는 영역은 얼굴의 움직임 속도와 관련이 있다. 본 논문에서는 추적할 얼굴이 전방의 영상을 시청하고 있다고 가정한다. 따라서 얼굴은 거의 정면을 바라보고 있으며, 영상을 시청하면서 움직이는 상황을 가정하였다. 얼굴의 움직임에 대해서는 시청하면서 움직일 수 있는 최대의 움직임 속도를 고려하여야 한다.

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 전후, 좌우, 상하방향에 대해 측정하였으며, 이를 두 프레임 간의 거리를 환산하여 탐색영역을 정의하였다. 표 1는 전후, 좌우, 상하방향에 대하여 실험을 통해 측정 한 깊이에 따른 움직임 양을 보여준다. 표 1을 이용하여 깊이에 따라 탐색영역을 정의하였다. 이와 같이 결정한 탐색영역을 그림 4에 나타내었는데, 좌우상하 움직임은 각 방향으로 28% 범위 내에 각각 있었다.

표 1. 깊이에 따른 움직임에 대한 변화량.

좌우 움직임	
깊이 값	최대 움직임 양 (픽셀 변화량)
132	8
148	8
167	10
188	12
212	16
228	20

상하 움직임	
깊이 값	최대 움직임 양 (픽셀 변화량)
132	8
148	10
172	10
196	14
220	18
234	24

전진 움직임		
깊이 값		최대 움직임 양 (픽셀 변화량)
t-1	t	
149	155	4
160	164	6
180	188	6
194	196	8
214	219	10
236	243	14

후진 움직임		
깊이 값		최대 움직임 양 (픽셀 변화량)
t-1	t	
151	148	8
169	164	8
188	180	8
201	196	10
220	215	10
236	230	14

2.2.3. 템플릿 매칭

얼굴추적의 다음 단계는 앞 절에서 재조정된 템플릿을 사용, 재조정된 탐색영역을 조사하여 매칭되는 점을 찾는 것이다. 하지만 RGB영상을 가지고 템플릿 매칭을 했을 경우, 조명변화로 인해 다음 프레임에서 얼굴을 찾지 못하고 다른 부분을 찾는 경우가 발생한다. 그래서 본 논문에서는 RGB영상보다 조명변화에 강한 깊이정보 만을 이용하여 템플릿을 매칭을 하였다.

템플릿 매칭을 할 경우에 기본적으로는 탐색영역 전체를 탐색한다. 이 경우 탐색하여야 하는 위치 수는 $(s_h^{search} - s_h^{temp}) \times (s_v^{search} - s_v^{temp})$이며, 여기서 s_h^{temp}와 s_v^{temp}는 각각 템플릿의 수평 및 수직방향 크기이며, 이에 해당하는 현재 프레임의 탐색범위의 값은 각각 s_h^{search}과 s_v^{search}이다. 본 논문에서는 탐색할 때 비용함수로 SAD(sum-of-absolute differences)를 사용한다. 즉, 탐색범위 내의 모든 위치를 탐색하여 그 중 가장 작은 SAD값을 갖는 위치를 선택한다.

그러나 전체영역을 탐색하면 연산시간이 오래 걸리기 때문에 본 논문에서는 이를 위한 방안으로 조기종료(early termination) 기법을 제시한다. 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 (4)와 같다.

$$SAD_{i,j} < T_l \tag{4}$$

여기서 $SAD_{i,j}$는 위치 (i,j)에서의 SAD값이며, T_l는 미리 정한 문턱치 값이고, 이것은 실험적으로 결정된다.

2.2.4. 얼굴검출 과정으로 귀환

앞에서 언급한 바와 같이 얼굴추적 과정을 수행하다가 템플릿 매칭에 실패하면 얼굴검출 과정으로 귀환하여 얼굴검출을 다시 수행한다. 이것은 장면이 바뀌거나 사람이 화면에서 사라지는 경우 등에 나타난다. 이와 같은 경우는 추적과정에서 SAD값이 매우 크게 나타나는데, 본 논문에서는 식 (5)과 같이 탐색범위 전체를 탐색한 후 각 위치에서의 SAD값의 최소값이 문턱치 값 T_h보다 큰 경우로 결정한다.

$$\min() > T_h \tag{5}$$

여기서 min( )는 탐색범위 내에서 SAD 값 중 최소치를 말한다.

3. 실험결과

본 논문에서 제안하는 방법을 구현하고 여러 테스트 시퀀스를 대상으로 실험을 수행하였다. 구현은 Microsoft window7 운영 체제에서 Microsoft visual studio 2010과 OpenCV Library 2.2를 이용하였으며, 실험에 사용된 PC의 사양은 2.67GHz의 Intel Core i5 CPU와 6GB RAM이었다.

알고리즘의 테스트를 위해 직접 제작한 전후, 좌우, 상하 움직임에 대한 테스트 시퀀스를 이용하였다. 각 테스트 시퀀스는 200프레임으로 제작되었다.

3.1. 검출 및 추적 오차율

표 2는 평균적인 얼굴 검출률에 대하여 Viola&Jones 방법과 본 논문에서 제안한 방법을 비교한 결과이다. 표 2에서 보듯이 검출 성공률은 유사하나, 오검출률이 14.84%가 작게 나타나 우수한 결과를 보이고 있음을 확인할 수 있다.

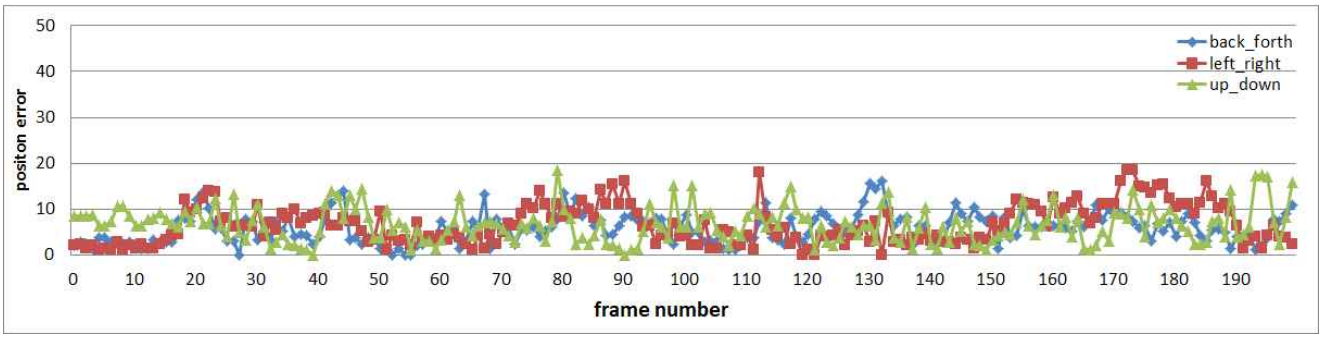

그림 4. 각 테스트 시퀀스에 해당하는 위치에러

표 2. 평균 얼굴검출률 비교

방법	검출 성공율(%)	검출 실패율(%)	오검출률(%)
Viola&Jones	98.66	1.34	15.05
제안한 검출방법	98.67	1.33	0.66

추적 위치 에러는 식 (6)와 같이 ground-truth 위치와 추적한 위치 사이를 유클리드 거리로 계산하였다. 식 (6)의 x, y는 ground-truth의 위치이다. 그리고 x', y'는 추적한 위치에 해당한다.

$$E_d = \sqrt[2]{(x'-x)^2+(y'-y)^2} \quad (6)$$

그림 4는 각각의 시퀀스를 가지고 추적한 위치 에러를 측정한 결과를 보여준다. 그림 4에서 보여주듯이 전후, 좌우, 상하방향에 추적위치에러 평균은 각각 5.74, 6.55, 6.60으로 ground-truth의 위치와 거의 차이가 없는 것을 알 수 있다.

3.2. 검출 및 추적 수행시간

표 3는 제안한 얼굴 검출과 추적에 대한 평균 수행시간을 계산한 것이다. 표 3에서 볼 수 있듯이, 제안하는 검출 방법의 속도는 Viola&Jones 방법과 비교하여 평균 26.65ms로 비교적 빠른 속도를 보였다. 그리고 제안하는 얼굴 추적 방법 또한 16.4ms로 비교적 빠른 속도를 보였다.

표 3. 프레임 당 평균 수행시간 비교.

시퀀스	Viola& Jones	제안한 검출 방법	제안한 추적 방법
좌우 움직임	186.43ms	28.55ms	10.79ms
전후 움직임	187.72ms	26.78ms	18.26ms
상하 움직임	188.27ms	24.63ms	20.15ms
평균	188.14ms	26.65ms	16.4ms

4. 결론

본 논문에서는 KINECT 센서를 이용하여 얼굴을 검출하고 효율적으로 얼굴을 추적하는 알고리즘을 제안하였다. 얼굴 검출 방법에서는 기존의 방법인 Adaboost를 이용하지만 깊이영상과 RGB영상을 이용하여 Adaboost의 입력영상을 제한하여 얼굴을 정확하고 빠르게 검출하였다. 그리고 얼굴 추적 방법에서는 영상의 깊이에 따른 적절한 템플릿의 크기조절과 탐색영역을 설정하여 좀 더 정확하게 얼굴을 추적할 수 있도록 하였다. 또한 깊이 영상을 이용하여 템플릿 매칭을 하였기 때문에 얼굴 추적 시 조명변화에도 강하였다. 빠른 추적을 위해서는 조기종료 기법을 사용하여 수행시간을 줄였다.

따라서 제안한 방법은 초당 30 프레임 이상의 실시간 얼굴추적 시스템에 사용하기 적합하며, 추적시간과 추적정확도의 상보적 관계를 활용하면 다양한 분야에서 사용할 수 있을 것으로 사료된다.

감사의 글

이 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2012-0004985)

참고문헌

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

[6] 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.