특집논문 (Special Paper)
방송공학회논문지 제21권 제6호, 2016년 11월 (JBE Vol. 21, No. 6, November 2016)
http://dx.doi.org/10.5909/JBE.2016.21.6.861
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

향상된 캠쉬프트를 사용한 실시간 얼굴추적 방법

이 준 환[a)], 유 지 상[a)‡]

Real-time Face Tracking Method using Improved CamShift

Jun-Hwan Lee[a)], and Jisang Yoo[a)‡]

요 약

본 논문에서는 실시간 얼굴 추적을 위하여 기존의 CamShift 알고리즘의 단점을 보완한 새로운 CamShift 알고리즘을 제안한다. 배경 내 추적 객체와 색상이 유사한 객체가 존재할 경우 기존 CamShift 알고리즘은 불안정한 추적을 보여준다. 이러한 문제점을 화소 단위로 거리정보를 획득할 수 있는 Kinect의 깊이 정보와 HSV 색공간 기반의 피부색 후보영역을 추출하는 Skin Detection 알고리즘을 이용하여 색상분포만 이용하는 기존의 CamShift의 단점을 보완한다. 또한 추적하던 객체가 사라지거나 가려짐이 발생할 경우에도 다시 추적할 수 있는 특징점 기반의 매칭 알고리즘을 통하여 차폐영역에 강인한 특성을 가지게 한다. 이러한 향상된 CamShift 알고리즘을 사람의 얼굴 추적에 적용함으로써 다양한 분야에 활용 가능한 강인한 얼굴추적 알고리즘을 제안하고자 한다. 실험결과 제안하는 알고리즘은 기존의 추적 알고리즘인 TLD보다 월등히 빠른 처리속도와 더 우수한 추적성능을 보여주었고, CamShift 보다 조금 느리지만 기존의 CamShift가 가지고 있는 문제점들을 해결하였다.

Abstract

This paper first discusses the disadvantages of the existing CamShift Algorithm for real time face tracking, and then proposes a new Camshift Algorithm that performs better than the existing algorithm. The existing CamShift Algorithm shows unstable tracking when tracing similar colors in the background of objects. This drawback of the existing CamShift is resolved by using Kinect's pixel-by-pixel depth information and the Skin Detection algorithm to extract candidate skin regions based on HSV color space. Additionally, even when the tracking object is not found, or when occlusion occurs, the feature point-based matching algorithm makes it robust to occlusion. By applying the improved CamShift algorithm to face tracking, the proposed real-time face tracking algorithm can be applied to various fields. The results from the experiment prove that the proposed algorithm is superior in tracking performance to that of existing TLD tracking algorithm, and offers faster processing speed. Also, while the proposed algorithm has a slower processing speed than CamShift, it overcomes all the existing shortfalls of the existing CamShift.

Keyword : Face tracking, Face-TLD, Haar-Feature, FAST, BRIEF, CamShift, Kinect

a) 광운대학교 전자공학과(Department of Electrical Engineering, KwangWoon University)
‡ Corresponding Author : 유지상(Jisang Yoo)
E-mail: jsyoo@kw.ac.kr
Tel: +82-2-940-5112
ORCID: http://orcid.org/0000-0002-3766-9854
※ 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. R0126-16-1034, 채널/객체 융합형 하이브리드 오디오 콘텐츠 제작 및 재생기술 개발)
※ 이 논문의 연구결과 중 일부는 "2016년 한국방송·미디어공학회 하계학술대회"에서 발표한 바 있음.
· Manuscript received September 7, 2016; Revised October 31, 2016; Accepted Novemver 7, 2016.

Ⅰ. 서 론

실시간 영상에서 사람의 얼굴을 검출하는 연구는 꾸준히 연구되고 있는 주제이다. 단조로운 배경이 아닌 동적 환경에서 얼굴을 추적하기 위한 연구가 활발히 진행 되어왔다[1-5]. 하지만, 복잡한 배경과 동적인 환경에서 실시간으로 높은 정확도의 얼굴 추적은 여전히 연구과제로 남아있다. 실시간 얼굴 추적은 피부색, 움직임 정보, 다양한 특징 변화 등 고려 사항이 많기 때문에 쉬운 문제가 아니다. 정확하게 얼굴 영역을 찾았다고 해도 추적을 하려면 시간이 오래 걸리거나 또는 추적에 실패하는 경우도 있다. 이러한 경우는 실시간 얼굴추적이 필요한 HCI(Human Computer Interaction) 시스템이나 감시 시스템에 적합하지 않다. 실시간으로 얼굴 추적이 가능하기 위해서는 알고리즘의 계산량이 굉장히 중요하다. 아무리 정확한 얼굴 추적기라도 계산량이 많아 속도가 느리다면 활용할 수 있는 분야에 제약이 있기 마련이다. 뛰어난 추적성능을 자랑하는 Face-TLD[6,7] 같은 경우에도 계산량이 많아 초당 5프레임 이하의 처리속도를 보여준다.

얼굴 추적을 위해 필요한 선행 작업은 얼굴영역을 검출하고 크기와 위치를 구하는 것이다. 실시간으로 얼굴을 검출하는 작업은 빠른 연산이 요구되기 때문에 기존의 연구[8-10]에서는 주로 특징 기반 방법으로 색상정보와 얼굴형태 정보를 이용한 방법들이 제시 되어왔다. 색상정보를 이용한 방법은 구현이 쉽고 계산량이 적다는 장점이 있으나 입력된 영상이 유사한 컬러 성분으로 존재 할 경우 검출의 정확도가 낮아지는 문제점이 있다. 또한 얼굴형태 정보를 이용하는 방법[8-10]은 조명변화에는 강건하지만 유사한 형태를 갖는 객체에 대해서 쉽게 오류를 범하는 경우가 있다. 본 논문에서는 획득된 동영상에서 실시간으로 얼굴영역을 검출하고 추적하는 새로운 알고리즘을 제안하고자 한다. 얼굴 영역의 검출은 색상정보를 이용하여 얼굴의 후보영역을 생성한 뒤 Haar feature[11]를 이용하여 특징 기반으로 얼굴을 검출한다. 얼굴 검출 시 Haar feature 기반의 Adaboost 강분류기[11,12]가 줄무늬나 글자와 같은 영역을 얼굴로 오검출하는 경우가 가끔 발생하는데 피부색의 색상정보[13,14]를 이용하여 얼굴의 후보영역을 생성한 뒤 얼굴을 검출하게 되면 오검출의 빈도수가 확연하게 줄어든다.

검출된 얼굴을 추적하기 위해서는 색상정보를 이용한 추적 방법인 CamShift 알고리즘[15]을 사용한다. CamShift 알고리즘은 목표 모델과 후보군 사이의 색상 분포인 히스토그램의 유사도를 비교하는 탐색 알고리즘으로서, 계산의 단순성, 안정성 면에서 좋은 성능을 가지고 있다. 그러나 목표모델과 유사한 색상분포를 갖는 객체나 배경에 취약하다는 단점이 있다. CamShift 알고리즘은 가변적인 탐색 윈도우를 사용하는데, 이는 추적하던 객체의 크기가 변해도 추적할 수 있는 장점이기도 하지만, 유사한 색상분포를 갖는 배경과 추적하던 객체가 근접하면 배경 전체를 추적하던 객체로 인식해 잘못된 추적을 하게 되는 문제점도 발생시킨다. 따라서 얼굴영역을 정확히 검출하여도 추적하는 과정에서 실패하는 경우가 발생할 수 있다[16,17].

얼굴을 추적하는 과정에서 색상정보 이외에 깊이 정보 등의 부가적인 정보를 사용하여 성능을 향상시킬 수 있다. 기본적으로 깊이 정보를 얻는 방법은 다수의 카메라에서 영상을 획득한 후 각 화소의 시차 값을 계산하는 방식으로 이루어진다[18-20]. 하지만 여러 대의 카메라를 캘리브레이션 하는 방법은 카메라를 설치한 뒤 캘리브레이션을 하는데 시간이 소요되며, 카메라의 위치가 변경될 때마다 다시 계산을 해줘야 하는 번거로움이 있다. 또한 여러 대의 카메라에서 획득한 이미지를 통해 디스패리티 맵을 추출하는데[21] 상당한 계산 시간을 요구하기 때문에 얼굴추적과 같은 실시간 추적 알고리즘에서는 적합하지 못하다. Microsoft사의 Kinect는 일반 카메라의 RGB 컬러카메라 기능과 적외선 센서를 이용하여 3차원의 깊이 정보를 실시간으로 획득하는 기능이 동시에 가능하다[22,23]. 본 논문에서 제안하는 알고리즘은 Kinect를 통해 화소 단위의 거리정보를 획득한 후 이를 이용하여 기존 CamShift의 단점을 개선한 향상된 추적 알고리즘이다. 또한 추적하던 얼굴이 고속으로 이동하거나 차폐영역 즉, 가려짐이 발생하여 추적이 실패하였을 경우에도 이전 얼굴 추적 시 저장해두었던 얼굴의 템플릿과 현재 프레임간의 특징점 기반 매칭[24,25]을 수행함으로써 재 추적이 가능하다. 본 논문에서 제안하는 알고리즘은 일반적인 실내 환경에서 단일객체, 즉 한사람의 얼굴추적을 목표로 개발되었다.

본 논문의 구성은 다음과 같다. 2장에서는 일반적으로 얼굴 영역 검출을 위해 필요한 알고리즘에 대해 소개하고, 3장에서는 깊이 정보를 사용하여 CamShift 알고리즘 기반의 실시간 얼굴추적 방법을 새로이 제안한다. 4장에서는 실험을 통해 제안하는 알고리즘과 기존의 알고리즘의 성능 비교를 제시하고 5장에서 결론을 맺는다.

Ⅱ. 얼굴 영역 추적을 위한 기본 알고리즘

1. Skin Detection

본 논문에서는 추적할 객체가 사람의 얼굴이라는 사실을 이용하여 카메라의 컬러영상에서 피부색의 후보영역을 추출하게 된다[13,14]. 피부색을 정확하게 검출하는 것은 쉽지 않다. 사람에 따라서 피부색이 어둡거나 밝기도 하고, 인종에 따라서도 피부색이 매우 다양하기 때문에 모든 피부에 최적인 알고리즘을 찾기 어렵다. 하지만 인간은 공통적으로 붉은 색의 피가 흐르기 때문에 피부색과 상관없이 붉은색 계열의 색을 포함하고 있다. 이 붉은색을 이용하여 주어진 영상에서 피부색의 후보영역을 추출하게 된다. 이때 일반적인 색공간인 RGB 색공간 보다 밝기 성분과 색상성분이 분리하기 쉬운 HSV 색공간을 사용한다. HSV 색공간 중에서도 색상성분인 Hue값의 특정범위를 사용하여 피부색의 후보영역을 추출하게 된다. 본 논문에서는 [13, 14]를 통해 얻은 정보를 바탕으로 실험해본 결과 $(0 < H < 20, 10 < S < 150, 60 < V < 255)$이 적정 값이라는 결론을 얻었고 사용하였다. 이는 다양한 인종의 피부색이 고려된 값이며 일반적인 실내조명이라고 가정하였을 때 피부색 후보영역을 잘 검출하였다. 본 논문에서 제안하는 알고리즘은 일반적인 실내 환경에서의 얼굴 추적이 목적이기 때문에 충분한 성능임을 확인하였다. 추출된 피부색 후보영역은 얼굴을 검출하는 Haar Detection[11], 검출된 얼굴을 추적하는 CamShift[15], 추적하던 객체가 사라졌을 때 저장된 템플릿과의 특징점 매칭[24,25]에 모두 사용된다. Skin Detection 알고리즘으로 피부색의 얼굴 후보영역이 추출되면 이 영역을 제외한 나머지 영역은 모두 색상정보를 버린다. 그림 1은 Skin Detection 알고리즘을 적용하여 검출된 얼굴 후보영역을 나타낸다.

Haar Detection 알고리즘을 얼굴 후보영역에 적용하여 얼굴 영역을 검출하는 경우 후보영역에서만 탐색을 하기 때문에 연산속도가 빨라진다는 장점이 있다. 일반적으로 주어진 컬러영상에 Haar Detection 알고리즘을 적용하면 Adaboost 강분류기[12]가 줄무늬나 글자와 같은 영역을 얼굴로 오검출하는 경우가 가끔 발생한다. 그러나 검출된 피부색 후보영역에 알고리즘을 적용하게 되면 오검출률을 줄일 수 있다.

2. Haar feature를 이용한 Face Detection

얼굴을 추적하기 전에 영상 내에 얼굴이 존재하는지 판단하고 그 위치를 먼저 파악하여야 한다. 본 논문에서는 Haar feature를 사용한 face detection 알고리즘을 적용한다. Viola와 Jones에 의해 제안된 Haar feature와 AdaBoost에 기반을 둔 객체검출 방법은 높은 정확도와 빠른 처리속도를 가진다. Haar feature는 적분영상을 사용하여 계산 속도를 높이며, AdaBoost 알고리즘은 분별력이 가장 높은 fea-

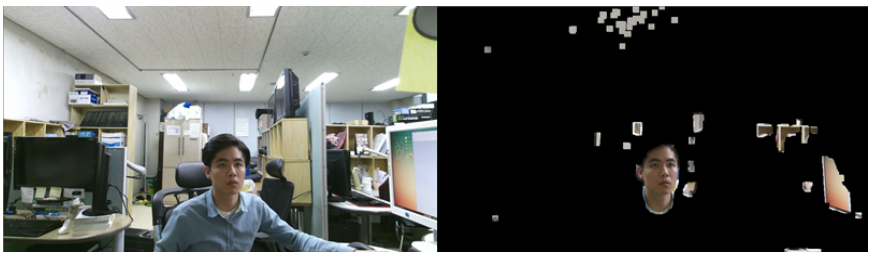


그림 1. (a) 주어진 컬러영상 (b) Skin Detection을 적용하여 검출한 피부색 후보영역
Fig. 1. (a) Example image, and (b) candidate areas remaining after applying Skin Detection

ture를 선택하여 분류기를 학습시키는데 사용한다[1]. 여러 분류기를 간단한 것부터 복잡한 순서의 cascade 구조로 배치하여 정확도를 유지하면서도 처리속도가 높은 효율적인 검출기이다. 이러한 Haar feature를 사람의 얼굴에 대해 학습시킨 후 AdaBoost에 기반을 둔 cascade 구조를 사용하여 얼굴영역을 검출하게 된다[11,12].

Haar feature를 이용하는 경우 영상에서 인식할 대상의 특징 값을 추출하기 위해서 그림 2와 같이 다양한 종류의 Haar feature들이 사용된다. Haar feature는 2개 이상의 인접한 사각형 영역들로 구성되며 feature에 대한 특징값은 흰색으로 표현된 영역에 해당하는 픽셀들의 밝기 합에서 검은색으로 표현된 영역에 해당하는 픽셀들의 밝기 합을 뺀 값으로 정의된다. 그림 2 에서 나타낸 feature를 흑백 반전, X축 Y축 스케일 확대 및 축소 등을 고려하면 거의 무한대에 가까운 feature 조합이 가능하다. 이들 중 의미 있는 feature들을 선별하여 사용하는 것이 중요하다. 여기서 의미 있는 feature란 인식하고자 하는 대상들에서는 일관성 있게 비슷한 값을 나타내면서 대상이 아닌 경우에는 고정되지 않은 값을 내는 feature들로 볼 수 있다. 이러한 의미 있는 feature 선별은 사용자의 수작업이 아닌 Boosting 알고리즘과 같이 자동화된 학습 알고리즘을 통해 이루어진다[11,12]. 그림 3은 다양한 종류와 위치, 크기의 Haar feature를 대조하여 얼굴을 검출하는 과정을 나타낸 그림이다.

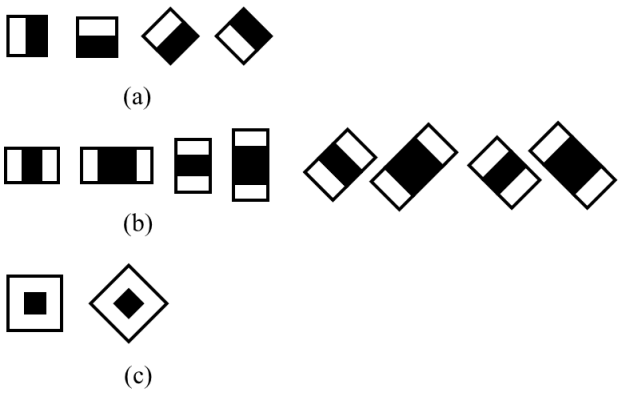


그림. 2. 다양한 크기와 모양의 Haar feature (a) Edge features (b) Line features (c) Center-surround features
Fig. 2. Various sizes and shapes of Haar features: (a) Edge features, (b) Line features, and (c) Center-surround features

그림. 3. 다양한 Haar feature를 대조하여 얼굴을 검출하는 과정[2]
Fig. 3. Process of differentiating face detection through using various Haar features [26]

3. CamShift(Continuously Adaptive Mean SHIFT) 알고리즘

CamShift(Continuously Adaptive Mean SHIFT)는 MeanShift 알고리즘[27]을 개선한 것으로 탐색 윈도우의 크기를 스스로 조절한다[15]. MeanShift는 nonparametric 한 Kernel 밀도 추정 알고리즘으로, 반복적으로 수행하여 확률 분포의 지역 최댓값을 찾는다. CamShift는 MeanShift를 기반으로 색상정보 분포(Histogram)를 통해 객체를 고속으로 추적하는 알고리즘으로 초기의 검색 영역의 크기와 위치를 사용자가 입력하면 반복적인 histogram의 비교와 대조를 통해 객체를 추적하게 된다. MeanShift와의 가장 큰 차이점은 가변적인 search window를 적용하여 크기가 변하는 객체도 손쉽게 추적할 수 있다는 점이 있다. CamShift는 다음과 같은 과정으로 동작한다[15].

a. 탐색 윈도우의 초기 위치와 크기를 설정한다.
b. 색상정보의 확률 분포를 계산한 뒤 탐색 윈도우의 중심을 찾기 위해 MeanShift 알고리즘을 수행한다.

c. 영상의 모멘트를 계산하여 색상 분포의 중심 위치와 크기를 구하고 탐색 윈도우를 재설정한다.

d. 재설정된 탐색 윈도우를 사용하여 MeanShift 알고리즘을 반복적으로 수행하며 탐색 윈도우가 수렴되거나 정해진 횟수만큼 b-d 과정을 반복 수행한다. 이때 앞에서 탐색 윈도우의 위치와 크기는 탐색 윈도우 내의 색상분포에 대한 0차, 1차, 2차 모멘트의 계산을 통해 구해진다. 0차, 1차, 2차 모멘트에 대한 수식은 각각 식 (1), (2), (3)과 같다.

$$M_{00} = \sum_x \sum_y I(x,y) \tag{1}$$

$$M_{10} = \sum_x \sum_y xI(x,y), M_{01} = \sum_x \sum_y yI(x,y) \tag{2}$$

$$M_{20} = \sum_x \sum_y x^2 I(x,y), M_{02} = \sum_x \sum_y y^2 I(x,y) \tag{3}$$

여기서 $I(x,y)$는 탐색 윈도우 내 (x,y) 좌표에 해당하는 화소의 값을 나타낸다. 0차, 1차, 2차 모멘트를 이용하여 탐색 윈도우의 중심 위치 (x_c, y_c)를 식 (4)와 같이 구할 수 있다[15].

$$x_c = \frac{M_{10}}{M_{00}}, y_c = \frac{M_{01}}{M_{00}} \tag{4}$$

Ⅲ. 가려짐에 강인한 CamShift 기반 실시간 얼굴추적 방법

본 논문에서 제안하는 알고리즘은 그림 4와 같이 크게 세 과정으로 구성되어 있다. 본 논문에서 제안하는 알고리즘은 그림 4와 같이 크게 세 과정으로 구성되어 있다. 먼저 얼굴을 찾는 Detection 부분에서는 키넥트로부터 들어온 입력영상에 2.1절에서 설명한 Skin detection 알고리즘을 적용하여 얼굴 후보군을 생성한 후 2.2절에서 설명한 Haar feature를 이용하여 얼굴을 검출한다. 검출된 얼굴의 위치와 크기를 CamShift의 초기 값으로 설정한다. 검출된 얼굴을 추적하는 Tracking 부분에서는 키넥트로부터 얻은 컬러영상과 Depth map의 Calibration 과정을 거쳐 앞서 검출된 얼굴의 위치의 깊이 값을 얻을 수 있다. 3.1절에서 설명할 깊이정보를 사용한 CamShift를 통해 얼굴을 추적한다. 추적을 수행함과 동시에 3.2절에서 설명할 Bhattacharyya distance[28]를 계산하여 추후 추적에 실패하였을 때에 대비한 얼굴 템플릿을 저장하거나 추적의 실패를 판단한다. Re-Tracking 부분에서는 추적이 실패했다고 판단한 경우에 저장된 얼굴 템플릿과 현재 프레임간의 특징점 매칭을 수행한다. 올바른 매칭이 4쌍 이상일 경우 호모그래피 계산을 통해 유효성을 검사한 뒤[29] 유효하다면 CamShift의 초기 값으로 설정하고 추적을 시작하게 된다. 유효성을 검사하는 조건은 3.4절에서 상세히 다룬다. 이러한 과정들을 통해 기존의 CamShift의 불안정한 추적과 재추적이 불가능하다는 큰 문제점을 보완함으로서 보다 안정적이며 가려짐에 강인한 추적기를 제안하고자 한다.

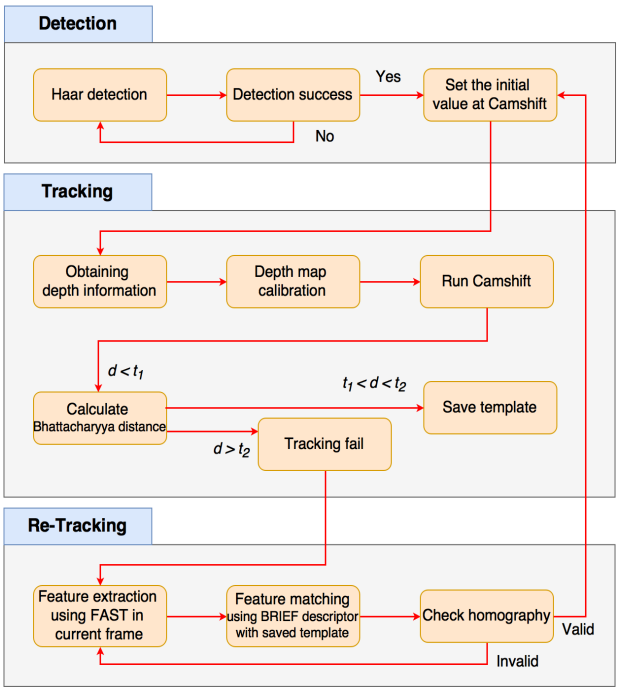


그림 4. 제안하는 알고리즘의 흐름도
Fig. 4. A flowchart of the proposed algorithm

1. 깊이 정보를 사용한 개선된 CamShift

기존의 CamShift 알고리즘[15]은 객체 추적 등의 다양한 영상처리에 활용되고 있으나, 아직까지 개선의 여지가 있다. CamShift 알고리즘은 초기 탐색 윈도우 설정 문제, 가변적인 탐색윈도우의 잘못된 추적 문제, 탐색 윈도우를 벗어난 대상 객체의 추적 문제 등을 가지고 있다. 본 논문에서 제안하는 알고리즘은 이 세 가지 문제를 개선한 향상된 추적 알고리즘이다. CamShift 알고리즘을 수행하기 위해서는 사용자에 의해 초기 탐색 윈도우의 위치 및 크기가 설정되어야 한다. 추적할 대상의 영역 설정은 불가피하지만 알고리즘을 반복적으로 수행하거나 실제 시스템에 적용할 경우 탐색 윈도우를 매번 설정해야 하는 번거로움이 있다. 추적을 시작할 때마다 초기 탐색 윈도우를 설정해야 하는 문제는 앞서 설명한 Haar feature를 이용한 face detection으로 해결한다. 영상에서 얼굴영역을 찾고 그 위치를 자동으로 초기 탐색 윈도우로 설정함으로써 사용자가 매번 탐색 윈도우를 설정해야 하는 번거로움을 덜 수 있다.

CamShift 알고리즘은 MeanShift의 고정된 탐색 윈도우와 다르게 가변적인 탐색 윈도우를 사용하는데 이는 추적하는 객체의 크기가 변해도 추적이 가능하다는 장점이지만, 유사한 색상분포를 갖는 배경과 추적하던 객체가 근접하면 배경 전체를 추적하는 객체로 인식하는 문제가 발생한다. 따라서 정확한 얼굴영역을 검출하여도 추적하는 과정에서 실패하는 경우가 빈번하게 발생한다. 그림 5 은 유사한 색상에 의한 불안정한 추적 영상의 예시이다.

그림 5. 유사한 색상에 의한 불안정한 추적영상의 예시
Fig. 5. Unstable tracking in an image due to similar colors

제안하는 알고리즘에서는 이 문제를 해결하기 위해 색상정보 이외의 부가 정보인 깊이 정보를 사용한다.

그림 6(a)는 CamShift를 이용한 추적 과정 중 초기 탐색 윈도우로 지정된 객체의 색상정보 분포를 계산하여 추적 대상과의 색상 유사확률을 0부터 255까지의 밝기 값으로 시각화한 Back-projection 영상이다[15]. 그림 6(b)는 이전프레임에서 추적된 얼굴영역의 중심점 깊이 값에 ±5%를 가지는 화소만을 255의 밝기 값으로 시각화한 그림이다. 여기서 5%는 얼굴의 입체적인 구조를 고려하여 실험적으로 선택된 값이다. 실험결과 얼굴의 가장 오목한 부분과 볼록한 부분까지 포함할 수 있는 값이며, 얼굴이 카메라를 비스듬히 바라보고 있을 때와 카메라의 앞, 뒤 방향으로 움직일 때 까지 잘 추적하는 것을 볼 수 있다. 그림 6(c)는 그림 6(a) 와 그림 6(b)의 교집합 즉, AND 연산의 결과이다. 그림 6(d)는 입력영상인 RGB 컬러영상에서 추적의 결과를 빨간색 타원으로 표시한 영상이다. 기존의 CamShift 알고리즘에서는 그림 6(a)와 같은 Back-projection 영상을 사용하여 색상기반의 추적을 한다. 반면에 본 논문에서 제안하는 개선된 CamShift 알고리즘에서는 그림 6(c)에 주어진 깊이 정보를 사용하여 색상기반의 추적을 함으로서 유사색상을 가진 배경에 강인한 특성을 가지게 된다.

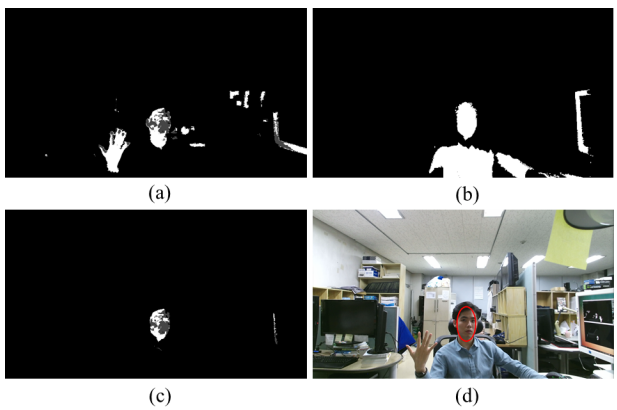


그림 6. (a) CamShift의 Backprojection (b) 이전 프레임 추적결과 중심점의 깊이와 같은 깊이를 가지는 화소 (c) (a)와 (b)의 AND 연산 결과 (d) 얼굴 추적의 결과
Fig. 6. (a) Backprojection of Camshift, (b) pixels with same depth center point as previously tracked frame results, (c) result of (a) and (b) AND calculation, and (d) face tracking results

그림 7에서 동일한 프레임에서 제안하는 알고리즘이 기존의 CamShift 알고리즘과는 다르게 깊이 정보를 사용하여 유사색상을 가진 배경이나 객체와 인접했을 때 올바른 추적을 하는 것을 알 수 있다.

(a) (b)

그림 7. (a) 깊이 정보를 사용한 개선 CamShift 알고리즘 결과 (b) 기존의 CamShift 알고리즘 결과
Fig. 7. (a) Result of the improved CamShift algorithm using depth information, and (b) result of the existing CamShift

기존의 CamShift 알고리즘은 탐색 윈도우로 불리는 특정 커널을 기반으로 객체의 추적이 이루어지며 커널 내의 색상 분포를 통해 반복적으로 탐색이 이루어진다. 그렇기 때문에 탐색 윈도우 내 대상 객체가 존재하지 않을 경우에는 탐색이 불가능하다. 특히 추적하던 객체가 고속이동을 하는 경우 영상처리에 소요되는 연산시간이 길어지거나 카메라의 특성에 따라 갑자기 객체의 위치에 변화가 발생할 수 있으며, 이럴 경우 탐색 윈도우 내 객체가 존재하지 않기 때문에 더 이상의 탐색이 불가능하다. 또한 추적하던 객체가 카메라의 화각 바깥으로 이동하여 영상 내에 존재하지 않게 되어도 추적이 불가능하다. 제안하는 알고리즘에서는 추적하는 얼굴 객체가 고속으로 이동하거나 다른 물체에 의해 가려지거나 또는 화면 밖으로 사라져 추적에 실패했을 경우에도 특징점 기반의 매칭을 통해 다시 추적할 수 있다는 장점이 있다. 재추적하는 방법은 3.3절에서 자세하게 다룬다.

2. 얼굴 움직임의 판단

음성의 단어인식과 문자, 영상 인식 등의 패턴인식에서 거리의 개념은 패턴들이 특징 공간상에서 떨어져 있는 정도를 나타내며 패턴 간의 유사도 측정을 위한 기준으로 사용한다. 즉, 특징 공간상에서 매우 근접한 거리에 위치한 두 패턴은 거의 동일한 특징을 가지므로 큰 유사도를 갖는다. 거리를 측정하는 방법으로는 유클리디언 거리(Euclidean Distance)[30], DTW(Dynamic Time Warping) 알고리즘[31], 바타차야 거리(Bhattacharyya Distance)[28] 측정 방법 등 다양한 알고리즘들이 사용된다.

본 논문에서는 추적하던 얼굴의 변화량을 확인하고 추적 실패를 확인하는 기준으로 바타차야 거리를 사용한다. 바타차야 거리 d는 이전프레임과 현재프레임의 히스토그램을 비교하여 식 (5)와 같이 구하게 된다.[28]

$$d(H_1,H_2)=\sqrt{1-\sum_I \frac{\sqrt{H_1(I)\cdot H_2(I)}}{\sqrt{\sum_I H_1(I)\cdot \sum_I H_2(I)}}} \tag{5}$$

여기서 H_1은 비교할 첫 번째 영상의 히스토그램이고 H_2는 두 번째 영상의 히스토그램이다. I는 히스토그램 빈(bin)의 인덱스를 의미하고 $H_1(I)$와 $H_2(I)$는 히스토그램의 I 번째 빈을 의미한다. 비교하는 두 영상의 히스토그램은 먼저 정규화 작업을 거친 뒤 식 (5)을 사용하여 비교하게 된다. 바타차야 거리 d는 비교하는 두 영상이 완벽하게 일치할 때 0의 값을 갖고 유사도가 떨어질수록 1에 가까운 값을 가지게 된다. CamShift의 결과를 매 프레임 비교하여 유사성이 일정 임계값 보다 큰 프레임을 템플릿에 저장한다. 일정 임계값 보다 바타차야 거리가 크다는 것은 객체가

그림 8. 다양한 크기와 각도로 저장된 얼굴 템플릿
Fig. 8. Various size and angles of saved face templates

완전히 정지해 있는 것이 아니라 회전하거나 이동하여 영상에서 밝기 분포의 변화가 생겼다는 의미이다. 그림 8은 다양한 크기와 각도로 저장된 얼굴 템플릿의 예시이다.

제안하는 알고리즘에서는 t_1, t_2 $(0 < t_1 < t_2 < 1)$ 두 개의 임계값을 설정한다. t_1과 t_2는 실험적으로 값을 정하는데 하드웨어의 성능과 카메라의 성능, 조명에 따라 달라질 수 있다. 식 (5)을 통해 계산된 바타차야 거리 d와 t_1, t_2를 비교하여 세 가지 경우를 고려한다. 첫 째, 계산한 바타차야 거리 d의 값이 t_1보다 작은 경우에는 추적하던 얼굴의 이전 프레임과 현재 프레임간의 히스토그램 차이가 적은 것으로 얼굴 객체의 움직임이 거의 없고 정지해있는 경우이다. 이때는 아무 처리도 하지 않는다. 이 경우 큰 변화 없이 정지해 있는 객체의 추적 결과를 템플릿에 중복 저장하는 문제도 방지할 수 있다. 따라서 효과적인 메모리 사용이 가능하며 추후에 저장된 템플릿과 특징점 매칭을 통한 재추적의 시간을 단축시키는 효과를 얻을 수 있다.

두 번째, 계산한 바타차야 거리 d의 값이 t_1보다는 크고 t_2보다는 작은 경우에는 얼굴의 움직임으로 이전 프레임과 현재 프레임간의 히스토그램에 변화가 있는 경우이다. 이 경우 추적하던 얼굴을 템플릿에 저장한다. 제안하는 알고리즘에서는 성공적으로 추적한 경우가 많아질수록 저장된 템플릿이 많아지기 때문에 재추적에 강인한 특성을 보인다. 다시 말하면 알고리즘이 실행될수록 객체의 학습이 이루어지기 때문에 성능이 더 좋아지게 된다. 하지만 특징점 기반으로 템플릿을 매칭하기 때문에 저장된 템플릿이 많아질수록 재추적 시 연산시간이 증가하는 단점이 있다. 따라서 저장된 템플릿의 수를 적절하게 조절하는 것이 재추적의 시간과 성능을 결정하는 중요한 요소가 된다. 마지막으로 바타차야 거리 d의 값이 t_2보다 큰 경우에는 추적하던 얼굴 히스토그램의 변화가 아주 크다는 것을 의미한다. 이때는 얼굴이 고속으로 이동하거나 다른 물체에 의해 가려져서 추적에 실패한 경우이다. 이 경우에는 얼굴 객체의 빠른 변화로 미리 추적이 실패하였다고 판단하고 재추적 단계로 돌입하게 된다.

3. 특징점 매칭을 이용한 재추적

객체의 고속 이동이나 가려짐으로 인해 추적에 실패한 경우 선별되어 저장된 템플릿 이미지들과 현재 프레임간의 특징점 매칭을 수행하여 현재 프레임에서 추적할 객체를 다시 찾는다. 하지만 영상 초기에 가려짐이 발생했을 때 재추적을 할 경우 저장된 템플릿이 부족하기 때문에 재추적에 어려움이 있다. 따라서 저장된 템플릿이 50개 미만일 경우 다시 얼굴을 검출하는 단계로 돌아가 Haar Detection을 통한 얼굴검출을 하게 된다. 특징점을 추출하는 방법과 추출된 특징점을 기술하는 방법은 다양하다. 본 논문에서 제안하는 알고리즘은 실시간으로 적용되어야 하기 때문에 연산 속도에 우선순위를 두고 특징점 추출 알고리즘과 기술자를 선택한다. 특징점을 추출하는 알고리즘은 속도가 빠르고 반복성(Repeatability)이 뛰어난 FAST 알고리즘[24]을 적용한다. 특징점의 기술자에서 SIFT 알고리즘[32]의 경우는 128차원으로 가장 자세하게 기술되어 높은 정확도를 기대할 수 있지만, 기술자의 차수가 높은 만큼 계산량이 많아지기 때문에 제안하는 알고리즘처럼 실시간 기반의 시스템에서는 적합하지 않다. 따라서 제안하는 알고리즘에서는 속도에 큰 강점을 가지고 있는 BRIEF 기술자[25]를 사용하여 특징점 매칭을 수행한다.

3.1 FAST(Features from Accelerated Segment Test)를 이용한 고속 특징점 추출

FAST(Features from Accelerated Segment Test) 알고리즘[24]은 실시간 특징점 추출에 적합한 알고리즘으로 기존의 특징점 추출 알고리즘인 Harris Corner Detector[33], SIFT[32], SURF[34] 보다 빠른 속도로 특징점 추출이 가능하다. 아래 그림 9는 특징점을 결정하기 위한 중심 화소와 주변 화소들과의 관계를 보여준다. FAST 알고리즘은 주어진 영상에서 기준 화소 P로부터 거리가 3인 원을 정의한다. 그림 9에서 파란색으로 나타낸 정의된 원에 걸쳐있는 16개의 화소를 $p \rightarrow x_k$라고 정의하면 밝기 값$(I_{p \rightarrow x_k})$과 기준 화소 P의 밝기 값(I_p)에서 임계값을 더하거나 뺀 값을 비교하여 기준 화소 P의 특징점 여부를 판별한다[24]. 만약 16개 화소

중 그 밝기 값이 기준 화소 P의 밝기 값과 임계값을 더한 값보다 더 크거나 기준 화소 P의 밝기 값에서 임계값을 뺀 값보다 더 작은 경우가 N번 이상 연속으로 있게 되면 기준 화소 P를 특징점으로 판단한다. N의 값은 주로 9, 10, 11, 12 등 다양한데 N이 9일 경우 특징점의 반복성(Repeatability)이 가장 높다[24]. N값과 마찬가지로 임계값은 사용자가 임의로 설정할 수 있으며 낮은 임계값을 설정할 경우 특징점이 많이 선택되고 임계값을 크게 설정할 경우 특징점은 적게 선택된다.

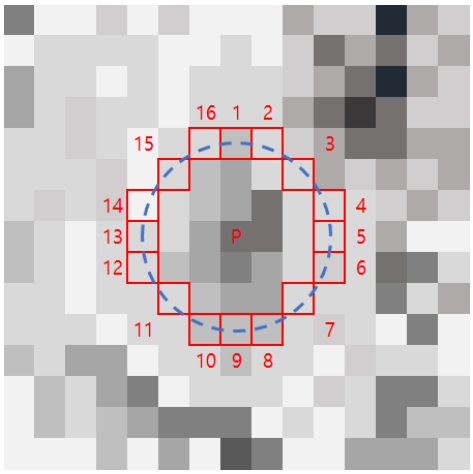


그림. 9. 중심 화소 P로부터 거리가 3인 원에 해당하는 16개의 화소
Fig. 9. Pixels on a circle centered at P with distance of 3

FAST 알고리즘에서는 기준 화소 P가 특징점인지 판단하기 위해 P의 밝기 값에 임계값을 더하거나 뺀 값과 16개 화소의 밝기 값을 비교하지만 더 빠른 연산속도를 위해 Decision Tree를 사용한다. 기준 화소 P로부터 정의된 16개 화소의 밝기 값을 P보다 훨씬 큰 경우, P보다 훨씬 작은 경우, P와 유사한 경우 이렇게 3가지로 분류하고 이를 이용하여 화소들의 밝기 분포를 16차원의 Ternary 벡터로 표현한다. 그리고 이 16차원의 벡터를 Decision Tree에 입력하여 특징점 여부를 보다 빠르게 판단할 수 있게 된다.

위와 같은 방법으로 추출된 특징점은 주변 화소와의 밝기 값의 차이만을 이용하여 추출되었기 때문에 영상에서 특징점들이 모여 있는 경우가 자주 발생한다. 군집되어 있는 유사한 특징점 중에서 대표 점을 뽑는 NMS(Non Maximum Suppression) 방법을 사용하여 유사한 특징점들을 하나의 대표되는 특징점으로 대체한다. FAST 알고리즘은 같은 영상이어도 영상의 크기가 크면 많은 특징점을 추출하고 영상의 크기가 작으면 적은 개수의 특징점을 추출하기 때문에 영상의 크기에 따라 민감한 특성을 지니고 있다. NMS을 통하여 크기 변화에 강인한 특성을 가지게 되는데 큰 영상에서 추출된 다수의 특징점을 대표 값으로 대체함으로써 특징점의 개수를 줄여주는 효과가 있기 때문이다.

3.2 이진 기술자(Binary Descriptor) BRIEF: Binary Robust Independent Elementary Features를 이용한 고속 정합

BRIEF[25]는 이진화된 기술자를 사용하기 때문에 메모리 사용 시 큰 장점을 가진다. 일반적인 기술자들보다 상대적으로 적은 비트를 이용하여 간단한 방법으로 비교하지만 좋은 성능을 보여주는 기술자이다. 특징점끼리 매칭하는 과정에서 BRIEF 기술자는 Euclidean distance[30]를 비교하는 대신 Hamming distance[35]를 비교하여 빠르고 효율적인 비교를 한다. Hamming distance는 같은 비트 수를 갖는 이진화된 기술자들 사이에 대응되는 비트 값이 일치하지 않는 것의 개수를 의미한다. 0과 1로만 이루어진 이진화된 기술자를 사용하면 64차원의 float형 기술자를 사용하는 SURF[34]에 비해서 매칭과 인식의 정확도가 떨어진다고 생각하기 쉽다. 하지만 BRIEF는 SURF와 비교하여 훨씬 빠른 속도로 비슷하거나 더 나은 인식 성능을 얻을 수 있다[25].

4. 호모그래피(Homography)의 유효성 판단

호모그래피는 대응되는 두 영상간의 투영관계를 나타내는 3x3 행렬이다. 제안하는 알고리즘에서는 저장된 얼굴 템플릿과 현재 프레임간의 특징점 기반의 매칭을 통해 얼굴을 재추적할 때 호모그래피를 계산하여 정상적인 투영관계를 가지는지 판별하는 용도로 사용한다[29]. 호모그래피 행렬을 구하기 위해서는 최소한 정합된 네 쌍의 특징점이 필요하다. 저장되어있는 템플릿들과 현재 프레임 간의 특징점 기반 매칭을 통하여 네 점 이상의 올바른 매칭쌍이 존재하면 매칭된 특징점들 사이의 투영관계인 호모그래피 계산

이 가능하다. 이때 오류와 잡음이 섞여있는 데이터들로부터 올바른 모델을 예측하여 오차를 줄이는 알고리즘인 RANSAC(RANdom SAmple Consensus)[36]을 이용하여 이상점(Outlier)들을 제거한다. 이때 구해진 변환관계는 이상점(outlier)을 제거하는 RANSAC이 실패한 경우도 포함되어 있기 때문에 항상 올바른 결과가 아니다. 따라서 계산된 호모그래피가 올바른지 확인하는 과정이 필요하다[29].

3행 3열의 원소가 1로 정규화 된 호모그래피 행렬 H를 식 (6)과 같이 정의한다. 3행 3열의 원소를 제외한 나머지 원소를 1행 1열부터 차례대로 h_1~h_8으로 나타낸다. 행렬 H를 위에서 정의한 몇 가지 Factor들로 판별한다. 식 (7)의 D는 행렬 H의 2x2 부분행렬의 determinant값으로 D가 0보다 작다면 회전 순서가 지켜지지 않았음을 알 수 있다. 뒤틀림이나 뒤집힘이 발생한 경우 잘못된 호모그래피를 판별할 수 있는 가장 중요한 Factor이다. 식 (8)의 X_s는 X축 방향의 길이가 얼마나 확대 또는 축소되었는지를 나타내는 Scale Factor이다. 식 (9)의 Y_s는 X축 방향이 아닌 Y축 방향의 Scale Factor이다. 식 (10)의 P는 Perspective Factor로 사각형의 사다리꼴 정도를 나타낸다. P 값이 0이라면 완전한 직사각형이고 P값이 커질수록 직사각형과 거리가 먼 뒤틀린 사다리꼴을 의미한다.

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \quad (6)$$

$$D = h_1 h_5 - h_2 h_4 \quad (7)$$

$$X_s = \sqrt{h_1^2 + h_4^2} \quad (8)$$

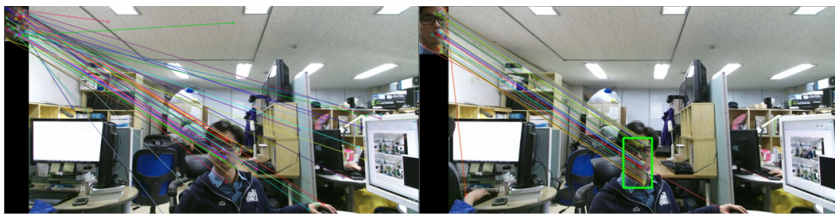
$$Y_s = \sqrt{h_2^2 + h_5^2} \quad (9)$$

$$P = \sqrt{h_7^2 + h_8^2} \quad (10)$$

위와 같은 여러 가지 factor들을 가지고 잘못 계산된 호모그래피의 여부를 확인하게 되는데 식(11)의 6가지 판별식이 이용된다. 이중에 하나라도 해당된다면 비정상적인 변환이라고 판단하게 된다[29].

$$D \le 0 \,\square\, X_s < 0.1 \,\square\, X_s > 3 \,\square\, Y_s < 0.1 \,\square\, Y_s > 3 \,\square\, P > 0.002 \quad (11)$$

정상적으로 호모그래피를 계산했다면 호모그래피를 통한 현재 프레임의 투영된 위치를 초기 위치로 설정하고 다시 추적을 시작하게 된다. 그림 10은 차폐영역에 의해 객체가 가려져 추적이 중단된 후 객체를 재추적 하는 경우이다. 그림 10에서 왼쪽 위의 작은 영역은 저장되어있는 템플릿 중의 하나이고 오른쪽 영역은 현재 프레임이다. 그림 10(a)는 저장되어있는 여러 템플릿들과 특징점 매칭을 수행하고 있는 영상이고 그림 10(b)는 특징점 매칭 후 호모그래피 계산을 통해 성공적으로 추적하던 객체를 다시 찾은 영상이다.



(a) (b)

그림 10. (a) 여러 템플릿들과의 특징점 매칭 결과 (b) 호모그래피 변환으로 다시 찾은 추적 객체
Fig. 10. (a) Result from feature matching various templates, and (b) success in re-tracking the object through homography conversion

Ⅳ. 실험결과

제안하는 알고리즘은 Microsoft사의 Kinect-v2를 통해 3채널 1920x1080 해상도의 RGB 영상과 1채널 512x424 해상도의 깊이 영상을 입력 받아 RGB 영상의 해상도를 960x540로 리사이즈 한 후 그래픽카드(GPU)의 고속화 없이 Intel i5-4690 3.50GHz CPU, 16GB RAM, Visual Studio 2013 환경에서 실험하였다. 제안하는 알고리즘은 키넥트로부터 획득한 깊이 영상을 사용하기 때문에 키넥트 깊이획득 가능 범위인 0.5m~4.5m 안에서 객체가 움직인다고 가정하고 실험하였다. 제안하는 알고리즘은 Face-TLD: Tracking-Learning-Detection[6][7], CamShift[15]와 비교하였

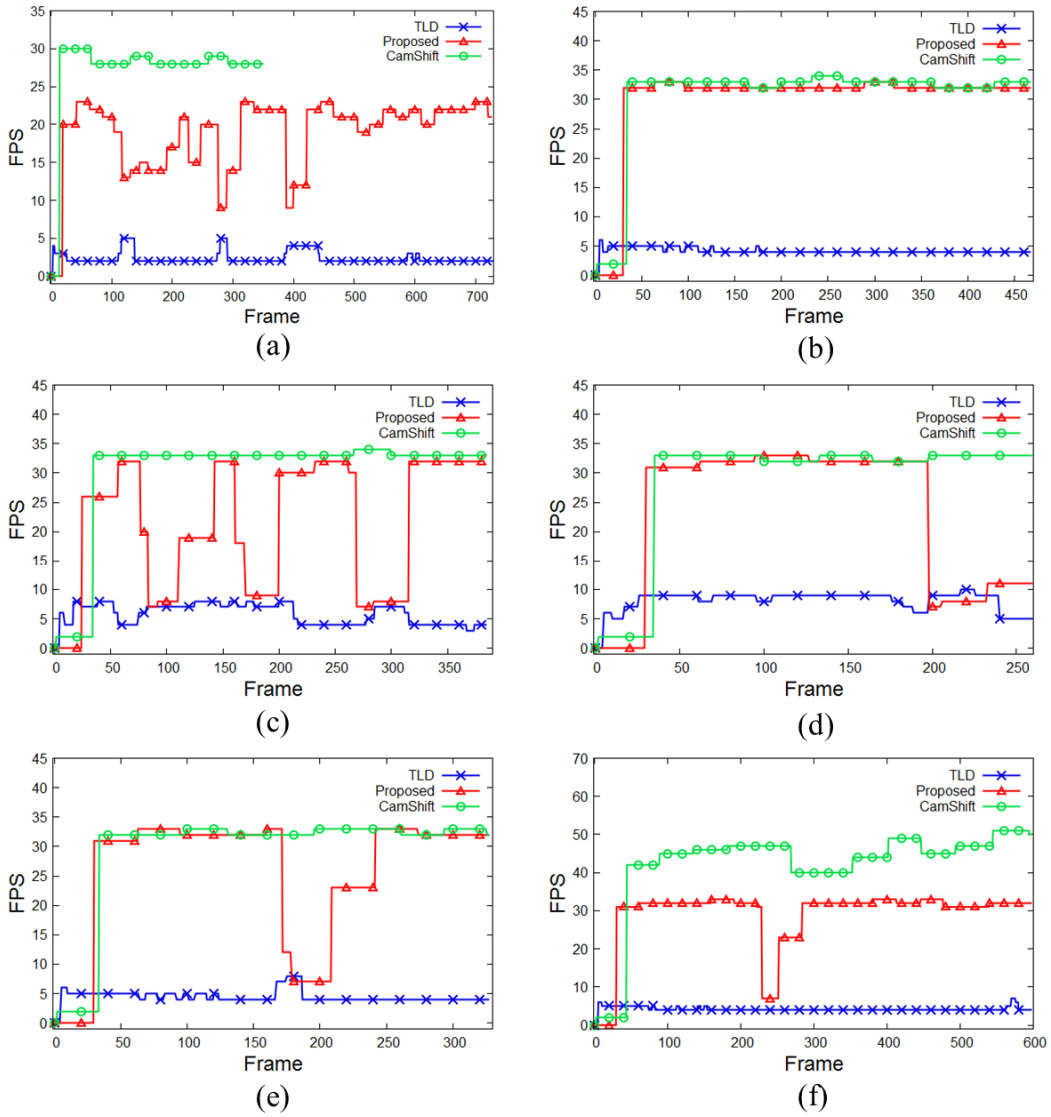


그림 11. 세 알고리즘의 처리속도 비교 (a) 자체 촬영 시퀀스 (b) face_move1 시퀀스 (c) face_occ2 시퀀스 (d) face_occ3 시퀀스 (e) face_occ5 시퀀스 (f) face_turn2 시퀀스

Fig. 11. Comparison of the processing speed of the three algorithms. (a) self shooting sequence (b) face_move1 sequence (c) face_occ2 sequence (d) face_occ3 sequence (e) face_occ5 sequence (f) face_turn2 sequence

으며 제안하는 알고리즘의 특성상 컬러영상과 깊이정보를 가진 깊이영상 동시에 필요하다. 따라서 모든 공인된 데이터에 대해 실험할 수 없어 컬러영상과 깊이정보를 같이 제공하는 시퀀스[37]중 사람의 얼굴이 나오는 5개의 시퀀스와 총 726프레임으로 구성되어 있는 자체 촬영한 시퀀스를 이용하여 실험하였다. 제공되는 시퀀스는 Kinect-v1 으로 촬영된 영상으로서 컬러영상과 깊이영상 모두 640x480의 해상도로 매프레임 PNG파일 포맷으로 제공된다[37]. 자체 촬영한 시퀀스는 키넥트에서 컬러영상과 깊이영상을 실시간으로 받아옴과 동시에 영상처리 알고리즘 처리하기 때문에 저장된 영상을 불러들여 영상처리를 하는 다른 5개의 시퀀스의 결과보다 조금 더 느린 속도를 보여주고 있다. 그림 11은 비교하는 세 알고리즘의 처리속도를 비교한 그림이다. 비교하는 세 알고리즘 모두 처리속도가 0FPS 부터 시작하는데 추적할 객체의 초기 값을 설정하기 전이므로 최초 0FPS 속도를 갖는 몇 개의 프레임은 최소 FPS와 평균 FPS 계산 시 제외하였다. TLD 알고리즘은 얼굴이 가려져 있어 얼굴영역을 추출할 수 없는 상황에서 처리속도가 빨라지는 특징을 보였다. 반면에 제안하는 알고리즘은 얼굴이 가려져 영역을 찾을 수 없는 상황에서 처리속도가 저하되는 특징이 있다. 이는 일반적인 상황에서는 깊이 정보를 사용한 CamShift 알고리즘으로 처리속도가 상대적으로 빠르지만 객체를 찾을 수 없는 상황에서는 저장된 얼굴 템플릿과 현재 프레임간의 특징점에 기반을 둔 매칭 작업이 많은 연산을 요구하기 때문이다. CamShift 알고리즘은 최소 28FPS의 속도로 빠르고 안정적인 속도를 보여주지만 추적에 실패한 뒤 재추적 하지 못하는 문제점으로 인해 자체 촬영 시퀀스에서 349프레임 이후에는 그림 11(a)에서와 같이 처리속도가 존재하지 않는다. 자체 촬영 시퀀스를 제외한 나머지 5개의 시퀀스에 대해서는 재추적 실패를 하는 경우는 없지만 실제 결과 영상을 보면 모두 잘못된 추적을 하는 것을 알 수 있다.

표 1은 TLD, CamShift, 그리고 제안하는 알고리즘에 대한 성능 비교 결과를 정리한 것이다. 속도 면에서는 CamShift 알고리즘, 제안하는 알고리즘, TLD 알고리즘 순으로 좋은 성능을 보여주었고 정확도 면에서는 제안하는 알고리즘, TLD 알고리즘, CamShift 알고리즘 순으로 좋은 성능을 보여주었다. 세 알고리즘 중 CamShift 알고리즘은 자체 촬영한 시퀀스에서 보여준 재추적이 불가능한 문제점으로 인해 추적 알고리즘으로는 부적합하다는 판단이다.

표 1. 알고리즘 성능 비교
Table 1. Comparison of algorithm performance

Sequence	Algorithm	Max FPS (FPS)	Min FPS (FPS)	Average (FPS)	Object Miss Frame (Frame)	Total Frame (Frame)	Tracking Success rate (%)
self shooting	TLD	5	2	2.362	61	726	91.597
	Proposed	23	9	19.336	43	726	94.077
	CamShift	30	28	28.500	377	726	48.071
face_move1	TLD	6	4	4.223	0	469	100.000
	Proposed	33	32	32.147	0	469	100.000
	CamShift	34	32	32.850	0	469	100.000
face_occ2	TLD	8	3	5.829	8	387	97.933
	Proposed	33	7	22.608	59	387	84.755
	CamShift	34	33	33.094	0	387	100.000
face_occ3	TLD	10	5	8.231	40	262	84.733
	Proposed	33	7	25.848	39	262	85.115
	CamShift	33	32	32.707	0	262	100.000
face_occ5	TLD	8	4	4.529	0	330	100.000
	Proposed	33	7	28.161	22	330	93.333
	CamShift	33	32	32.439	0	330	100.000
face_turn2	TLD	7	4	4.207	0	600	100.000
	Proposed	33	7	30.468	10	600	98.333
	CamShift	51	40	45.386	0	600	100.000

나머지 5개의 시퀀스에 대해서는 단순한 색상정보만을 이용하기 때문에 모두 잘못된 추적을 보여준다. 추적하려는 객체가 가려졌음에도 영상 전체를 객체라고 판단해 추적 성공률은 가장 높다. TLD 알고리즘과 제안하는 알고리즘은 높은 추적 성공률을 보여주었다. TLD 알고리즘은 실제 객체가 아닌 영역을 추적하는 문제점이 6개의 시퀀스중 4개의 시퀀스에서 발견되었고 제안하는 알고리즘은 6개의 시퀀스중 1개의 시퀀스에서 발견되었다. TLD 알고리즘은 처리속도가 평균 4.897 FPS인 반면 제안하는 알고리즘은 평균 26.428 FPS로 GPU을 사용하여 고속화 한다면 실시간 구현에 문제가 없어 보인다.

그림 12는 자체 촬영한 시퀀스 112번째 프레임으로 얼굴이 반쯤 가려진 상황에서도 세 알고리즘 모두 얼굴 영역을 잘 추적하는 모습이다. 그림 12(b)의 TLD알고리즘과 달리 그림 12(c)의 제안하는 알고리즘과 그림 12(d)의 CamShift 알고리즘은 형태나 탐색윈도우의 크기가 가변적인 성질에 의해 가려지지 않은 얼굴영역만을 추적하는 특징이 있다.

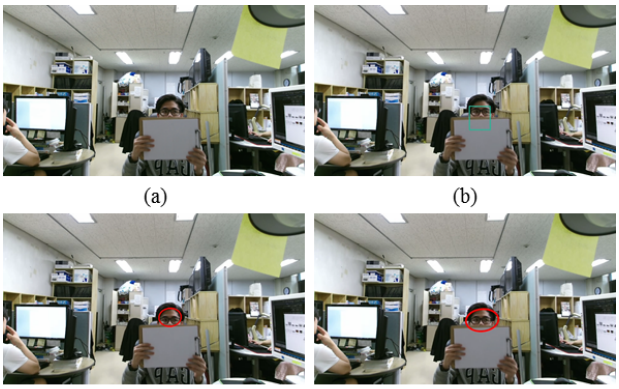


그림 12. 자체촬영 시퀀스 프레임 #112의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘 (d) CamShift 알고리즘
Fig. 12. The result of face tracking for self shooting sequence frame #112: (a) reference image, (b) TLD algorithm, (c) Proposed algorithm, and (d) CamShift algorithm

그림 13은 자체 촬영한 시퀀스의 133번째 프레임으로 얼굴이 완전히 가려졌을 때의 상황이다. 그림 13(b)와 (d)를 보면 TLD알고리즘과 CamShift 알고리즘은 추적할 객체가 사라졌음에도 불구하고 잘못된 추적을 하고 있는 모습이다. 얼굴이 갑작스럽게 사라진 것이 아닌 연속적인 프레임으로 조금씩 가려진 경우이다. 그림 13(c)의 제안하는 알고리즘은 바타차야 거리 계산을 통해 추적하는 객체가 가려진 것으로 판단하고 재추적 단계로 전환하게 된다.

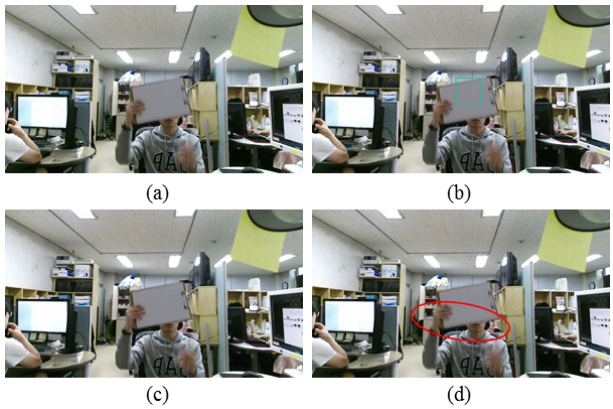


그림 13. 자체촬영 시퀀스 프레임 #133의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘 (d) CamShift 알고리즘
Fig. 13. The result of face tracking for self shooting sequence frame #133: (a) reference image, (b) TLD algorithm, (c) the proposed algorithm, and (d) CamShift algorithm

그림 14는 자체 촬영한 시퀀스의 360번째 프레임으로 얼굴이 가려진 후 다시 나타났을 때의 상황이다. 그림 14(b)와

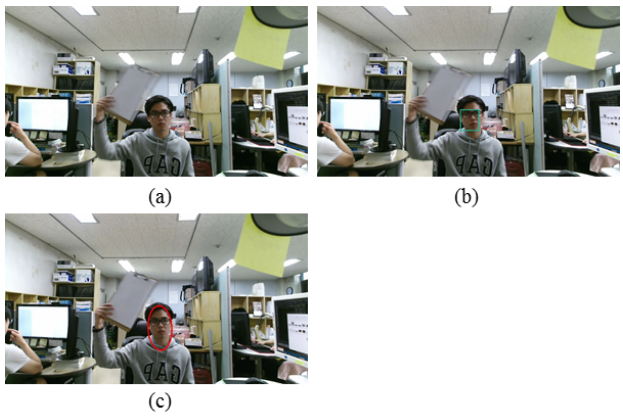


그림 14. 자체촬영 시퀀스 프레임 #360의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘
Fig. 14. The result of face tracking for self shooting sequence frame #360: (a) reference image, (b) TLD algorithm, and (c) the proposed algorithm

(c)의 TLD알고리즘과 제안하는 알고리즘은 각각 재추적에 성공한 모습이고, CamShift 알고리즘은 가려짐에 의해 추적에 실패해 349번째 프레임 이후는 얼굴 추적 결과가 존재하지 않는다.

그림 15는 자체 촬영한 시퀀스의 403번째 프레임으로 얼굴이 안 보이는 상황이다. 그림 15(b)의 TLD알고리즘은 잘못된 추적을 보여준다.

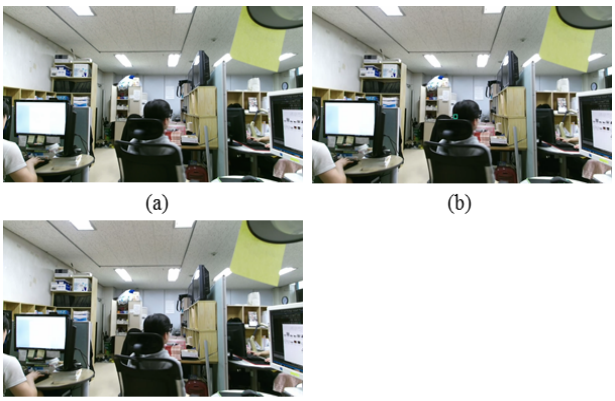


그림 15. 자체촬영 시퀀스 프레임 #403의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘
Fig. 15. The result of face tracking for self shooting sequence frame #403: (a) reference image, (b) TLD algorithm, and (c) the proposed algorithm

자체 촬영한 시퀀스에서 TLD 알고리즘은 총 726프레임 중 얼굴이 가려져 객체를 찾을 수 없는 프레임이 61프레임이고 계산된 추적 성공률은 91.597%이다. 하지만 시퀀스에 대한 알고리즘 테스트 결과 그림 13(b), 그림 15(b)와 같이 알고리즘은 성공적인 추적이라고 인식하지만 실제로는 잘못된 객체를 추적 하고 있는 프레임이 간혹 발견되었다. 따라서 해당 시퀀스에 대한 TLD 알고리즘의 올바른 추적 성공률은 계산한 91.597% 보다 조금 더 낮을 것으로 예상한다. 제안하는 알고리즘에서는 TLD 알고리즘과 같이 잘못된 객체를 추적하고있는 프레임은 발견되지 않았으며 94.077%의 높은 추적 성공률을 보인다. CamShift 알고리즘은 빠른 속도를 보여주는 반면 그림 13의 (d)와 같이 잘못된 추적을 하는 문제점과 얼굴 객체가 가려짐에 의해 영상 내에 존재하지 않을 때 추적에 실패하여 더 이상 추적하지 못하는 문제점을 보여주었다.

그림 16은 face_occ2 시퀀스의 24번째 프레임으로 TLD 알고리즘과 CamShift알고리즘은 잘못된 추적을 하는 것을 보여준다. TLD알고리즘 같은 경우는 추적하던 객체가 가려지기 직전에 잘못된 추적을 하는 경우가 6개의 시퀀스중 4개의 시퀀스에서 발견되었다. 자체 촬영 시퀀스를 제외한 나머지 5개의 시퀀스에 대해서는 CamShift 알고리즘의 특징인 색상정보와 가변적인 윈도우를 사용하는 점이 큰문제점으로 작용하여 추적하려는 얼굴 이외의 배경을 모두 객체라고 인식하는 잘못된 추적을 보여준다.

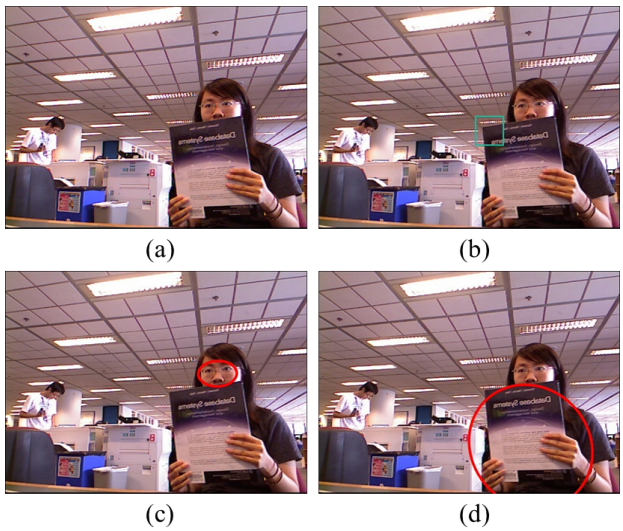


그림 16. face_occ2 시퀀스 프레임 #24의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘
Fig. 16. The result of face tracking for face_occ2 sequence frame #24: (a) reference image, (b) TLD algorithm, and (c) the proposed algorithm (d) CamShift algorithm

그림 17은 face_occ2 시퀀스의 175번째 프레임으로 TLD 알고리즘은 그림 17(b)와 같이 손으로 얼굴을 가렸지만 이전프레임에서 얼굴의 위치를 계속 추적하고있는 모습을 보여주며, 그림 17(c)의 제안하는 알고리즘은 추적하던 얼굴이 가려짐에 의해 사라졌을 때 잘못된 추적을 하는 경우를 보인다. 제안하는 알고리즘은 6개의 시퀀스중 한 번에 잘못된 추적을 보인다.

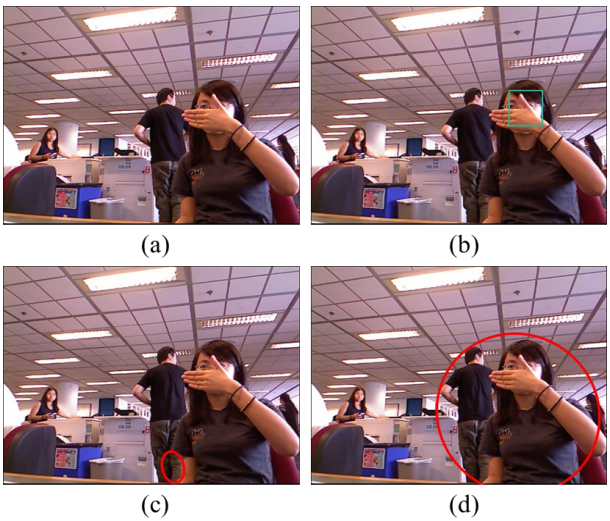


그림 17. face_occ2 시퀀스 프레임 #175의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘
Fig. 17. The result of face tracking for face_occ2 sequence frame #175: (a) reference image, (b) TLD algorithm, and (c) the proposed algorithm (d) CamShift algorithm

그림 18은 face_occ3 시퀀스의 104번째 프레임이고 그림 19는 face_occ5 시퀀스의 178번째 프레임이다. 이는 위에서 언급한 것과 마찬가지로 TLD알고리즘은 가려짐 직전과 직후에 추적하던 객체를 오추적하는 모습을 보여주고, 제안하는 알고리즘은 연속된 프레임 간의 바타차야거리 계산을 통해 가려짐을 잘 구분해내는 모습을 보인다. CamShift는 모든 프레임에 대한 잘못된 추적하는 모습을 보인다.

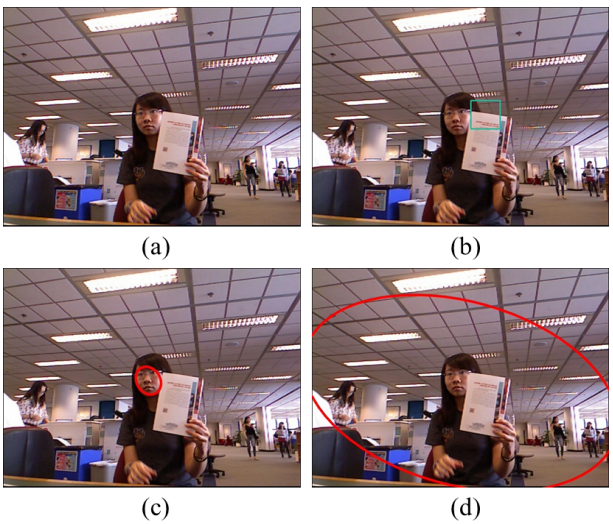


그림 18. face_occ3 시퀀스 프레임 #104의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘
Fig. 18. The result of face tracking for face_occ3 sequence frame #104: (a) reference image, (b) TLD algorithm, and (c) the proposed algorithm (d) CamShift algorithm

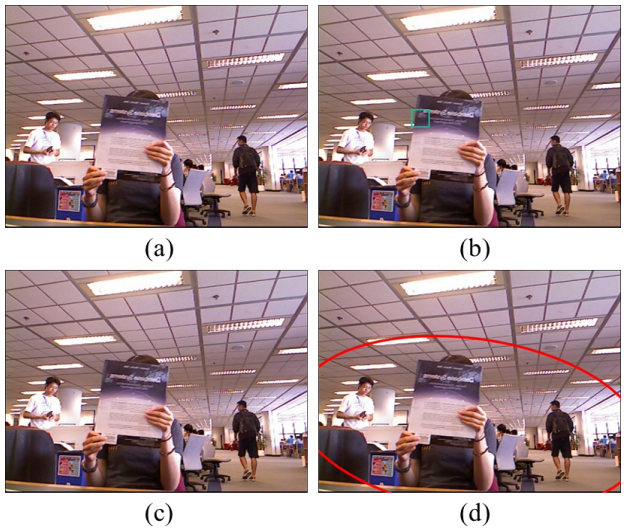


그림 19. face_occ5 시퀀스 프레임 #178의 얼굴 추적결과 (a) 레퍼런스 영상 (b) TLD 알고리즘 (c) 제안하는 알고리즘
Fig. 19. The result of face tracking for face_occ5 sequence frame #178: (a) reference image, (b) TLD algorithm, and (c) the proposed algorithm (d) CamShift algorithm

실험결과에 대한 영상은 https://www.youtube.com/channel/UCrNCHOY3CUhWpndRbP6O7Yg 에서 확인할 수 있다.

V. 결 론

본 논문에서는 Microsoft사의 Kinect로부터 획득된 깊이 정보를 사용하여 기존 CamShift의 단점을 개선한 실내 환경의 단일객체의 얼굴 추적 알고리즘을 제안하였다. 제안한 알고리즘에서는 CamShift가 가지고 있는 유사한 색상의 객체가 인접했을 때의 잘못된 추적, 추적 실패 시 재추적하지 못하는 문제, 추적할 대상을 사용자가 직접 입력해야

하는 문제 등을 해결하였다. 기존에 얼굴 추적에 대한 연구는 많이 이루어져있지만 대체로 연산량이 많고 연산 시간이 오래 걸려 실시간 구현이 어려운 문제가 있다. 제안하는 알고리즘은 그래픽카드(GPU)의 고속화 없이도 Intel i5-4690 3.50GHz CPU, 16GB RAM, Visual Studio 2013 환경에서 평균 26.428 FPS 정도의 높은 처리 속도를 보여준다. 또한 TLD알고리즘과 비슷하거나 실험한 시퀀스에 대해서는 보다 높은 얼굴 추적 성공률을 보여주며 깊이정보가 부가적으로 필요하다는 제약조건이 있지만 TLD알고리즘에 비해 월등한 추적속도를 보여준다.

참 고 문 헌 (References)

[1] Viola, Paul, and Michael J. Jones. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154.

[2] Rowley, Henry A., Shumeet Baluja, and Takeo Kanade. "Neural network-based face detection." IEEE Transactions on pattern analysis and machine intelligence 20.1 (1998): 23-38.

[3] Osuna, Edgar, Robert Freund, and Federico Girosit. "Training support vector machines: an application to face detection." Computer vision and pattern recognition, 1997. Proceedings., 1997 IEEE computer society conference on. IEEE, 1997.

[4] Hsu, Rein-Lien, Mohamed Abdel-Mottaleb, and Anil K. Jain. "Face detection in color images." IEEE transactions on pattern analysis and machine intelligence 24.5 (2002): 696-706.

[5] Hjelmås, Erik, and Boon Kee Low. "Face detection: A survey." Computer vision and image understanding 83.3 (2001): 236-274.

[6] Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Face-tld: Tracking-learning-detection applied to faces." Image Processing (ICIP), 2010 17th IEEE International Conference on. IEEE, 2010.

[7] Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on vol. 34, no. 7, pp. 1409-1422, 2012.

[8] Young-Gon Kim, Rae-Hong Park, and Seong-Su Mun “Face Detection Using Adaboost and Template Matching of Depth Map based Block Rank Patterns”,JBE, 437-446, Vol. 17, No. 3, May 2012.

[9] Hoo Hyun Kim, Dong-Chan Cho, Jong Yeop Bae, Whoi-Yul Kim. "Rotation Invariant Face Detection with Boosted Random Ferns." The Korean Institute of Broadcast and Media Engineers Summer Conference, (2013.6): 52-55.

[10] Kyong-Ho Lee. "Face Tracking Using Face Feature and Color Information." Journal of the Korea Society of Computer and Information , 18.11 (2013.11): 167-174.

[11] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. 511, 2001

[12] Viola, Paul, and Michael Jones. "Fast and robust classification using asymmetric adaboost and a detector cascade." Advances in Neural Information Processing System 14 (2001).

[13] Jones, Michael J., and James M. Rehg. "Statistical color models with application to skin detection." International Journal of Computer Vision 46.1 (2002): 81-96.

[14] Vezhnevets, Vladimir, Vassili Sazonov, and Alla Andreeva. "A survey on pixel-based skin color detection techniques." Proc. Graphicon. vol. 3, pp.85-92, September, 2003.

[15] Bradski, Gary R. "Computer vision face tracking for use in a perceptual user interface." (1998).

[16] Allen, John G., Richard YD Xu, and Jesse S. Jin. "Object tracking using camshift algorithm and multiple quantized feature spaces." Proceedings of the Pan-Sydney area workshop on Visual information processing. Australian Computer Society, Inc., 2004.

[17] Wang, Zhaowen, et al. "CamShift guided particle filter for visual tracking." Pattern Recognition Letters 30.4 (2009): 407-413.

[18] Zhang, Zhengyou. "A flexible new technique for camera calibration." IEEE Transactions on pattern analysis and machine intelligence 22.11 (2000): 1330-1334.

[19] Tsai, Roger. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses." IEEE Journal on Robotics and Automation 3.4 (1987): 323-344.

[20] Weng, Juyang, Paul Cohen, and Marc Herniou. "Camera calibration with distortion models and accuracy evaluation." IEEE Transactions on pattern analysis and machine intelligence 14.10 (1992): 965-980.

[21] Mühlmann, Karsten, et al. "Calculating dense disparity maps from color stereo images, an efficient implementation." International Journal of Computer Vision 47.1-3 (2002): 79-88.

[22] Zhang, Zhengyou. "Microsoft kinect sensor and its effect." IEEE multimedia 19.2 (2012): 4-10.

[23] Pagliari, Diana, and Livio Pinto. "Calibration of kinect for xbox one and comparison between the two generations of Microsoft sensors." Sensors 15.11 (2015): 27569-27589.

[24] Rosten, Edward, and Tom Drummond. "Machine learning for high-speed corner detection." Computer Vision–ECCV 2006. Springer Berlin Heidelberg. pp. 430-443, 2006.

[25] Calonder, Michael, et al. "Brief: Binary robust independent elementary features." Computer Vision–ECCV 2010 pp. 778-792, 2010.

[26] https://www.flickr.com/photos/unavoidablegrain/6884354772/in/photostream/ (Image by Greg Borenstein)

[27] Comaniciu, Dorin, and Peter Meer. "Mean shift: A robust approach toward feature space analysis." IEEE Transactions on pattern analysis and machine intelligence 24.5 (2002): 603-619.

[28] Bhattacharyya, Anil. "On a measure of divergence between two multinomial populations." Sankhyā: the indian journal of statistics (1946): 401-406.

[29] Trzcinski, Tomasz, and Vincent Lepetit. "Efficient discriminative projections for compact binary descriptors." Computer Vision–ECCV

2012. Springer Berlin Heidelberg. pp. 228-242, 2012.
[30] Danielsson, Per-Erik. "Euclidean distance mapping." Computer Graphics and image processing 14.3 (1980): 227-248.
[31] Müller, Meinard. "Dynamic time warping." Information retrieval for music and motion (2007): 69-84.
[32] Lowe, David G. "Distinctive image features from scale-invariant keypoints."International journal of computer vision vol. 60, no. 2, pp. 91-110, 2004.
[33] Harris, Chris, and Mike Stephens. "A combined corner and edge detector."Alvey vision conference. vol. 15. 1988.
[34] Bay, Herbert, et al. "Speeded-up robust features (SURF)." Computer vision and image understanding vol. 110, no. 3, pp. 346-359, 2008.
[35] Hamming, Richard W. "Error detecting and error correcting codes." Bell System technical journal 29.2 (1950): 147-160.
[36] Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Communications of the ACM 24.6 (1981): 381-395.
[37] http://tracking.cs.princeton.edu/index.html
[38] Song, Shuran, and Jianxiong Xiao. "Tracking revisited using rgbd camera: Unified benchmark and baselines." Proceedings of the IEEE international conference on computer vision. 2013.
[39] http://darkpgmr.tistory.com/80

저 자 소 개

이 준 환

- 2016년 2월 : 광운대학교 전자공학과 학사
- 2016년 3월 ~ 현재 : 광운대학교 전자공학과 석사
- ORCID : http://orcid.org/0000-0001-8967-0761
- 주관심분야 : 영상처리, 컴퓨터비전, 딥러닝

유 지 상

- 1985년 2월 : 서울대학교 전자공학과 학사
- 1987년 2월 : 서울대학교 전자공학과 석사
- 1993년 5월 : Purdue Univ. EE, ph.D
- 1997년 9월 ~ 현재 : 광운대학교 전자공학과 교수
- ORCID : http://orcid.org/0000-0002-3766-9854
- 주관심분야 : 웨이블릿 기반 영상처리, 영상압축, 영상인식, 비선형 신호처리