Journal of the Korea Industrial Information Systems Research Vol. 22 No.3, June. 2017 :27-38
http://dx.doi.org./10.9723/jksiis.2017.22.3.027 ISSN:1229-3741

다중 사용자 촬영 영상의 영상 스티칭

(Stitching Method of Videos Recorded by Multiple Handheld Cameras)

미어 사데크 빌라흐[1], 안 희 준[2]*
(Meer Sadeq Billah and Ahn Heejune)

요 약 본 연구는 다수의 개인 사용자들이 휴대폰 카메라를 통하여 공연장 등에서 녹화한 다수의 영상을 스티칭하는 방법을 제시한다. 기존 고정형 리그(Rig)를 사용하는 360 카메라 솔루션과 대비하여, 시간 동기화, 반복적 변환행렬계산 및 카메라 센서 불일치 보정과 같은 새로운 문제들을 해결해야한다. 이 논문에서는 오디오를 사용한 시각동기화 방법, 색상 전달 방식에 따른 센서 불일치 제거, 전역 동작 안정화 알고리즘을 사용하여 변환 행렬의 업데이트를 함으로써 이러한 문제를 해결하였다. 또한, 카메라의 움직임이 크지 않은 경우에서, 제안 된 알고리즘은 화면 별 스티칭을 하는 경우 보다, 계산 속도와 화질 면에서도 우수한 성능을 보임을 실험을 통하여 확인하였다.

핵심주제어 : 스티칭, 360도 VR, 스마트폰 영상, 영상처리

Abstract This Paper Presents a Method for Stitching a Large Number of Images Recorded by a Large Number of Individual Users Through a Cellular Phone Camera at a Venue. In Contrast to 360 Camera Solutions that Use Existing Fixed Rigs, these Conditions must Address New Challenges Such as Time Synchronization, Repeated Transformation Matrix Calculations, and Camera Sensor Mismatch Correction. In this Paper, we Solve this Problem by Updating the Transformation Matrix Using Time Synchronization Method Using Audio, Sensor Mismatch Removal by Color Transfer Method, and Global Operation Stabilization Algorithm. Experimental Results Show that the Proposed Algorithm Shows better Performance in Terms of Computation Speed and Subjective Image Quality than that of Screen Stitching.

Key Words : Stitching, Smartphone Video, Image Processing, 360 Degree Video

1. 서 론

최근 가상현실이 다시 관심을 모으고 있다. 이 중에서도 파노라마 영상을 사용한 시각 정보 기술은 VR기술 중 주목받는 기술이다[1, 2]. 파노라마 사진은 특수 장비 또는 소프트웨어를 사용하여 가로로 긴 시야로 이미지를 캡처하는 사진 기술이다. 이에 사용되는 이미지 스티칭 기술은 이미지들의 투영기하행렬을 계산하고, 이를 기하 변환을 통하여 합치는 작업을 수행한다. 이 분야의 대표적인 연구로 SIFT 특징점 알고리즘으로 유명한 Lowe 교수팀의 '자동 스티치 (Auto-

* Corresponding Author : heejune@seoultech.ac.kr
† 이 논문은 본 연구는 서울과학기술대학교 이 연구는 서울과학기술대학교 교내연구비의 지원으로 수행되었습니다 (2016-0833)
Manuscript receivrd May 29, 2017 / revised Jun 21, 2017 / accepted Jun 27, 2017
1) 서울과학기술대학교 전기정보공학과, 제1저자
2) 서울과학기술대학교 전기정보공학과, 교신저자

stitch)'[3] 방법이 있으며, OpenCV Image Stitching 모듈로 제공되고 있다.

동영상 파노라마는 최근 '360VR'이라는 표현으로 불리며, 2015년경부터 Youtube 등에서 콘텐트 서비스를 제공하고 있다. 파노라마 동영상을 제작하기 위해서는 FoV (Field of View)가 수평 360도+수직180도인 특수한 카메라가 필요한데, 어안렌즈를 사용하여 2개의 카메라 (각 180도+180도)를 사용하는 방식과, 다수개의 일반 FoV (60-90도 내외) 카메라로 촬영한 영상을 스티칭하는 방식이 주로 사용된다[4]. 예를 들어 PTGui [5], Kolor [6] 및 Vahana Video-Stitch [7]는 현재 이미지나 동영상 스티칭에 사용되는 대표적 무료 또는 유료 소프트웨어이다. 이처럼, 기존 전용 360도 카메라를 사용하여 촬영된 영상은, 카메라의 특성이 거의 동일하고, 고정된 상대위치를 갖고 있으므로, 한 번의 기하변환 추정을 하면 합성에 필요한 변환행렬을 얻을 수 있다.

본 연구에서 다루려는 응용은 이와 같이 특수 제작된 360도 카메라와는 달리, 다수개의 스마트폰으로 촬영된 영상을 스티칭하여 파노라마영상을 만드는 것이다. 예를 들어 Fig. 1과 같이 이벤트나 공연장에서 여러 명의 친구들이 동시 촬영한 휴대폰 영상을 결합하는 것이다. 최근 EU 프로젝트인 ICoSOLE (icosole.eu) [8]와 국내 Giga Korea 연구과제 프로젝트, 미래창조부 창조씨앗과제 등에서 이러한 기술이 연구되고 있다. 특히, 본 논문의 결과는 창조씨앗과제를 위한 기초결과에 해당한다. 기술적으로 기존의 360도 카메라와 달리, 시간 동기화, 반복 등록 및 카메라 센서 불일치 보정과 같은 새로운 문제들을 해결하여야 한다. 이러한 문제 중 손 떨림이나 움직임 등에 때문에 발생하는 기하변환 재추정의 문제가 발생한다. 기하변환 재추정은 많은 계산량이 요구되기 때문에 매 화면마다 사용하는 것은, 실용적인 방법이 되지 못한다. 또한, 시간적으로 화면간의 기하적인 높은 상관성이 존재하므로 이를 이용할 필요가 있다.

본 논문에서 이 3가지 문제들에 대한 해결 방법을 제시한다. 제2절에서 동영상 스티칭 문제를 정의하고, 제안된 처리 절차에 대하여 총괄적으로 설명한다. 제3절에서는 각 단계별 세부 알고리즘을 (시간동기화 및 컬러보정) 상세히 설명한다. 특히 전역 움직임에 기반한 안정화 방식을 통하여 화면마다 기하변환을 재추정하는 대신 2차원 이동만을 보상하도록 한다. 제 4장에서는 이를 통해서 얻은 결과를 제시하고 성능을 평가한다. 마지막으로 5장에서 본 연구의 한계점과 보완점을 기술한다.

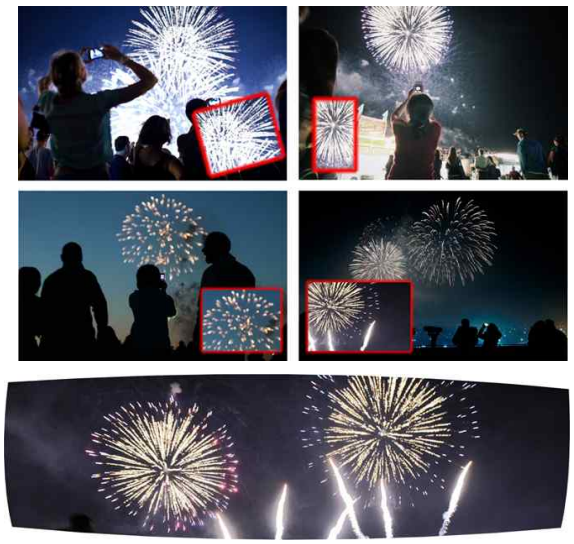


Fig. 1 UGC based Panorama Application Scenario

2. 스티칭 시스템

2.1 이미지 & 동영상 스티칭

현재 대부분의 영상 스티칭 알고리즘은 이미지 스티칭을 목적으로 개발된 알고리즘에 기반하고 있다. Fig. 2은 'Auto Stitching' 알고리즘[3]에서 제안되었고 OpenCV의 이미지 스티칭 모듈에서 사용하여, 연구자들에게 실제적인 표준이 되고 있는 구조이다.

이미지 스티칭과정은 변환행렬추정(영어로는 'registration'이라고 표현됨)과 합성(composition)의 두 단계를 거치게 된다. 변환행렬의 추정단계는 SIFT나 SURF와 같은 특징점 추출과 특징점의 기술자를 계산하는 작업으로 시작한다. 각 화면에서 추출된 특징점 정보는 다른 화면에서의 특징

점 정보와 비교를 통하여 대응점(corresponding points)을 찾아낸다. 이 대응점을 바탕으로 각 카메라간의 카메라 이동행렬을 계산해낸다. 여러 장의 영상의 카메라 이동을 동시에 추정하는 번들 조정(Bundle Adjustment)과 웨이브 교정 (Wave Correction) 등을 하여 프로젝션 행렬을 추정한다. 합성단계에서 기하변환을 수행하고, 겹치는 영역과, 독립적인 영역을 나누고, 겹치는 영역에 대하여 합성(blending)을 수행하게 된다. 이때 영상의 조도 등에 차이가 있는 경우 밝기보정작업을 수행한다.

본 연구자들이 OpenCV의 이미지 스티칭 모듈을 사용하여 프로파일을 해본 결과, 두 단계에서 모두 계산양이 많았다. 그러나 합성단계는 현재 다중밴드 (multiband: 주파수를 이미) 합성 알고리즘을 생략한다면, 상당히 계산양을 줄일 수 있고, 본 연구에서 제한하는 바와 같이 생상보정을 하는 경우에는 다중밴드 합성을 생략하여도 합성 영상의 화질에 큰 열화가 발생하지 않을 것으로 보고 연구하고 있다. 본 논문에서는 이들 중 계산양에 가장 크게 영향을 주는 단계인 특징점 추출과 대응점 계산, 그리고 이를 사용한 변환행렬계산이다.

기존 고정형 360 카메라 소프트웨어도 이러함 점을 고려하여, 처음 화면에만 변환행렬추정을 수행하고, 이후 영상에는 함성 절차만을 수행하고 있습니다. 그러나 본 연구에서 사용하는 입력 영상은 사용자들이 휴대폰을 통하여 취득한 영상으로 일정 수준의 움직임을 피할 수 없다[9]. 또한 현재 매화면 정합 변수 추정을 하는 경우, 기준 시각이 일정하지 않아서, 한 장 한 장은 잘 합성이 되더라도 동영상으로 합쳤을 때 영상은 흔들림이 발생하여 바람직하지 못한 결과를 보인다.

2.2 제안된 동영상 스티칭

앞서 기술한 화면단위 변환행렬 추정의 문제를 해결 또는 완화하기 위하여 Fig. 3과 같은 스티칭 파이프라인을 구성하였다. 알고리즘의 핵심 생각은 다음과 같습니다. 카메라 움직임이 크지 않은 상황에서 촬영된 영상들은 화면별 계산된 변환행렬이 크게 변화지 않는다. 또한 동일 동영상 내에서 움직이는 전경 물체의 비디오에서 커다란 고정 된 배경 물체를 찾을 수 있다면 비디오의 영상내 움직임의 추적하는 것이 가능하며 그 움직임이 크지 않을 것이다. 따라서 첫 번째 프레임에서 구한 변환 행렬과 화면내 카메라 이동을 고려하면, 매 화면 계산양이 많은 추정을 생략할 수 있다.

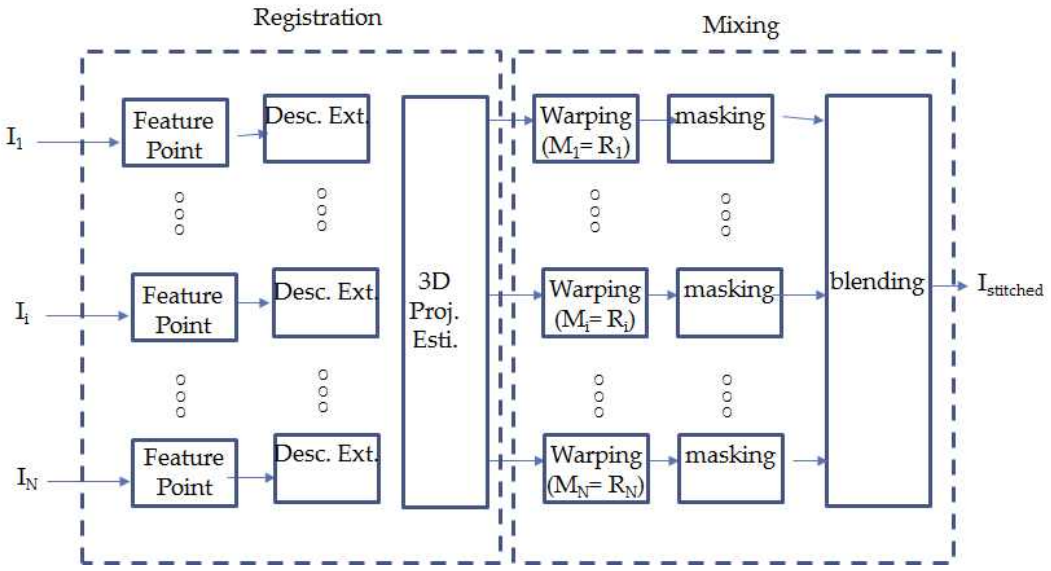


Fig. 2 Image Stitching pipeline in [3] and OpenCV Image Stitching module

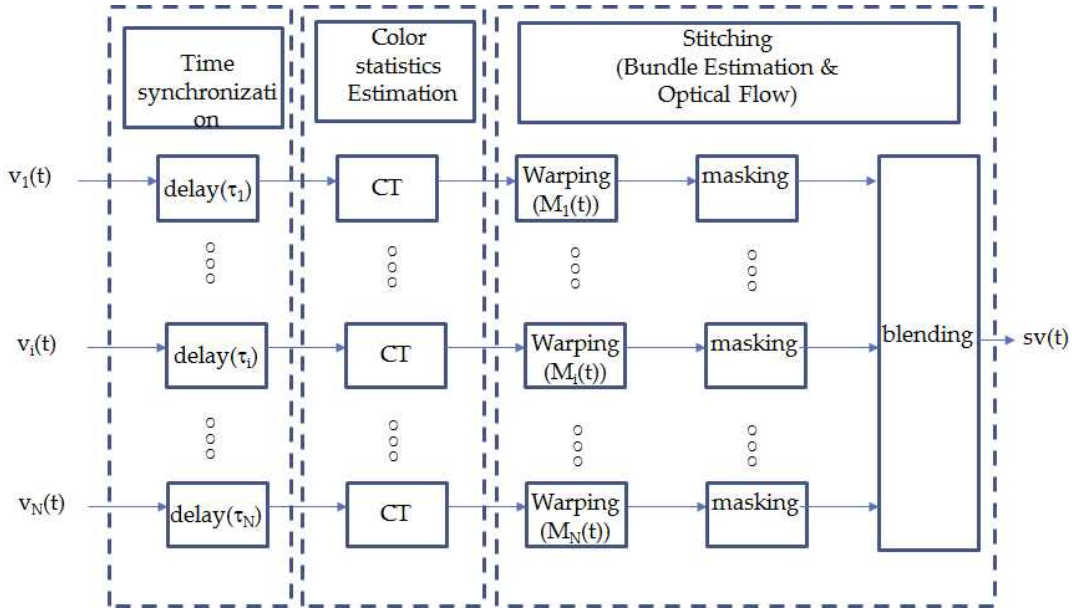


Fig. 3 The Proposed Fast Video Stitching Pipeline

이를 바탕으로 기존의 이미지 스티칭 시스템을 다음과 같이 확장하여 설계하였다.

- 제 1단계: 각 카메라에서 촬영된 영상을 시각적으로 동기화한다. 동기화에 사용하는 방식은 영상과 같이 녹화된 오디오정보를 사용합니다. 영상 정보대신 오디오 정보를 사용하는 이유는 음성정보의 시간 해상도가 영상보다는 훨씬 높고, 일차원 정보라 계산 양이 작기 때문이다.
- 제 2단계: 시간 동기화 정보를 사용하여 각 개별 비디오에서 프레임을 시작한다. 이러한 특정 프레임은 시간 동기화 방식으로 초기 프레임으로 다음 단계에 공급된다.
- 제 3단계: 비디오는 4 단계에서 6 단계까지 반복하여 프레임 단위로 처리된다.
- 제 4단계: 효율적인 컬러 전송 방법은 기타 화면에서 기준화면으로 방식으로 비디오 프레임의 순서에 따라 적용된다.
- 제 5 단계 : 변환행렬은 첫 화면에 대해서는 이미지 스티칭과 같은 방식으로 계산하면, 이후 개별 동영상의 연속 프레임 간의 변환은 첫 번째 행렬에 전역이동행렬을 곱하여 추정된다.
- 제 6 단계 : 개별 프레임을 변환하고 변환된 화면을 합성하여 사용하여 스티칭한다.
- 제 7 단계 : 모든 프레임이 처리되면 출력 비디오가 저장된다.

3. 각 단계별 세부 알고리즘

3.1 시간동기화 처리

앞서 언급한 바와 같이, 사용자들에 의하여 녹화된 영상은 실험 입력을 비디오는 각자 휴대폰을 사용하여 촬영되므로 전역 시각 동기화정보가 포함되어 있지 않습니다. 시각동기화를 위하여 특별한 시각정보를 저장하는 시스템을 개발할 수도 있으나, 본 논문에서는 일반적인 경우를 가정한다. 우리는 여기서 1-2분 정도 오차이내의 촬영시간정보를 알고 있다고 가정한다.

본 연구에서는 Fig. 4에서 사용된 오디오기반 동기화 알고리즘 중용 주파수 집합 비교 알고리즘을 사용하여 프레임레벨의 동기화를 수행한다. 즉. 다음과 같이 기준 오디오신호$a_0(t)$ 를 바탕으로 각 오디오 신호 $a_i(t)$의 시간지연 τ_i를 추정한다.

$$\begin{cases} a_0(t) \\ a_i(t) = c_i * a_0(t-\tau_i) + n_i(t)\ i = 1, \cdots, N-1 \end{cases} \quad (1)$$

영상 정보대신 오디오 정보를 사용하는 것은 오디오정보의 시간해상도가 40KHz정도로 25~30Hz인 영상에 비하여 월등히 높아 보다 정밀한 동기화가 가능하고, 영상데이터에 비하여 상대적으로 계산양이 작고, 관련 알고리즘이 연구되어 있기 때문이다. 본 연구에서 적용한 알고리즘은 주파수 매칭방식을 사용합니다. 개별 비디오 파일에서 오디오를 추출한 후, 오디오 신호가 고정 된 시간 간격 (bin이라고 칭함)으로 분할한다. 각 단위 시간 영역을 푸리에 변환(Fourier Transformation)을 수행하고, 기존 연구[17]에 간단하면서 좋은 성능을 보인 절대치가 큰 m개의 주파수를 선별합니다. 이후 각 연속 비디오에 대해 시간 오프셋 τ_i을 변화시키면서 가장 일치도가 높은 시간 오프셋 τ_i을 결정한다.

$$\tau_i^* = \underset{\tau_i}{\arg max} \sum (F_m(a_i(t-\tau_i)) \cap F_m(a_0(t))) \quad (2)$$

여기서 $F_m(\bullet)$은 푸리에 변환계수 중 절대치가 큰 m개의 주파수의 집합을 의미한다. 이 계산은 검색하여야하는 시간 법위가 클 때 선형적으로 계산양이 증가한다. 하지만, 본 연구에서는 1-2분 이내의 동기는 사용가가 알고 있다는 것을 가정하므로, 계산양의 문제는 크게 발생하지 않는다. Fig. 5는 3 개의 입력 비디오에 동반된 오디오 신호와 동기화 포인트의 시점을 출출해 낸 결과이다. 실험에 사용한 영상에 대하여 1과 2 번째 비디오는 0.104 밀리 초의 시간차가 있었고, 3 번째 비디오와 2 번째 비디오 사이에는 0.070 밀리 초의 간격이 있었다. 실제로 화면 주기보다 훨씬 높은 해상도로 동기화가 가능하지만, 이 경우 해당 시간의 화면을 생성하여야하기 때문에 시스템 복잡도가 높아지므로, 영상의 해상도 정밀도까지만 동기화를 수행하였다. 비디오 헤더에서 렌 값을 고려하여 지연 프레임만큼 건너 띠면 된다.

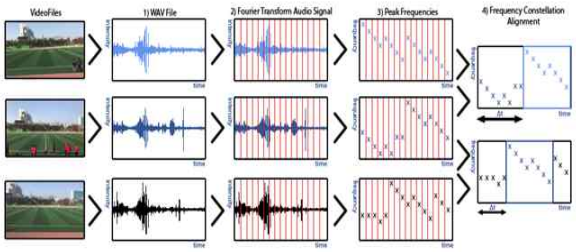


Fig. 4 Time synchronization Algorithm based key frequencies constellation matching between Three Recorded Contents

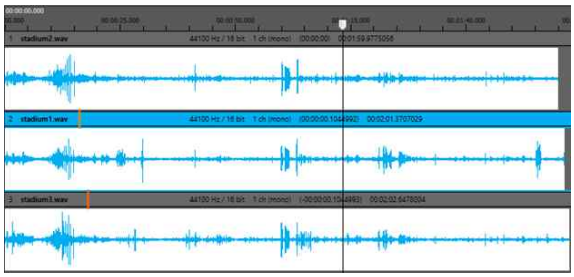


Fig. 5 Audio based Time Synchronization Example

3.2 색상 매칭

사용자들이 사용한 카메라 센서의 특성이 다르거나 설정이 다른 경우, 즉 노출, 색상 프로필 및 해상도가 다른 경우, 동시에 동일한 환경에서 촬영을 하였어도 영상의 컬러는 많이 다르게 되며 따라서 스티칭하면 이음새 등에 눈에 보이는 차이가 발생한다. 이런 경우 이미지의 컬러 특성을 일치시키는 방법을 색상전송 (color transfer) 또는 색상매칭(color matching)[10, 11]이라고 한다.

본 논문에서 사용하는 알고리즘은, 영상 입력 중에서 임으로, 또는 사용자가 지시에 의하여 기준 색상 입력을 정하고, 참조 프레임의 색상은 대상 프레임으로 전달된다. 기존 연구된 방식[10, 11] 중, 본 연구에 사용한 방식은 비교적 단순한 방법으로 영상간의 생상별 평균과 표준편차만을 동일하게 일치시키는 것이다.

1. 우선 RGB영상을 CIE lαβ 색상공간[12]으로 변환한다. lαβ 색상공간은 l축이 무채색 채널을 나타내고 α 및 β 채널이 반음계 옐로우 블루 및 레드 그린 상대 채널 인 사람의 시각적 인식 시

스템을 기반으로 한다.

2. 기준과 목표화면의 평균과 표준편차 $(\overline{l_t}(\sigma_t^l), \overline{\alpha_t}(\sigma_t^\alpha), \overline{\beta_t}(\sigma_t^\beta), \overline{l_s}(\sigma_s^l), \overline{\alpha_s}(\sigma_s^\alpha), \overline{\beta_s}(\sigma_s^\beta)$ 를 계산하고, 이를 이용하여 색상변환을 수행한다.

$$l'(x,y) = \frac{\sigma_t^l}{\sigma_s^l}(l_s(x,y) - \overline{l_s}) + \overline{l_t}, \quad (3)$$
$$\alpha'(x,y) = \frac{\sigma_t^\alpha}{\sigma_s^\alpha}(\alpha_s(x,y) - \overline{\alpha_s}) + \overline{\alpha_t},$$
$$\beta'(x,y) = \frac{\sigma_t^\beta}{\sigma_s^\beta}(\beta_s(x,y) - \overline{\beta_s}) + \beta_t$$

3. 결과를 다시 RGB 영상으로 변환한다.

Fig. 6은 두 개의 샘플 비디오 프레임에 대한 컬러 전송의 출력을 보여준다. 실험영상은 왼쪽 영상에 비하여 우측 영상의 생상대비가 약한 것을 볼 수 있습니다. 색상전달 없이 합성한 경우 (가운데) 연결 부분의 색차가 보이는 반면, 색상 전달을 한 경우 이 부분이 깔끔하고 양쪽의 차이가 없다.

3.3 전역 움직임 예측을 통한 변환행렬 갱신

화면별 변환행렬 계산에서 오는 문제를 해결하기 위하여, 본 연구에서는 비디오를 안정화하기 위해 프레임 사이에서 옵티컬-플로우 (Optical Flow)를 사용하는 알고리즘 [13]을 사용한다. 각 영상별로, 카메라 움직임은 옵티컬-플로우 계산에서 얻은 2차원 회전 및 이동 행렬을 계산하고, 초기 화면에서 얻은 변환행렬과 곱하여 업데이트 한다. 첫 번째 프레임은 'Auto Stitching'방식에 따라 스티칭을 수행한다. 이 방식에 따르면 3차원 기하회전행렬 $M = R(\theta,\psi,\phi)$만 구해진다. 두 번째 프레임부터는 사용될 때, 우리는 전역움직임 행렬 (2차원 강체변환)을 계산하고, 이를 첫 번째 행렬과 곱하여 변환행렬을 구하게 된다. 이렇게 함으로써 얻는 장점은 기존 방식에서 지원할 수 없었던 카메라간의 변위를 고려할 수 있다는 점이다. 그러나 첫 번째 프레임의 카메라 시차의 반영에 대해서는 아직 완벽한 해가 발표된 적이 없

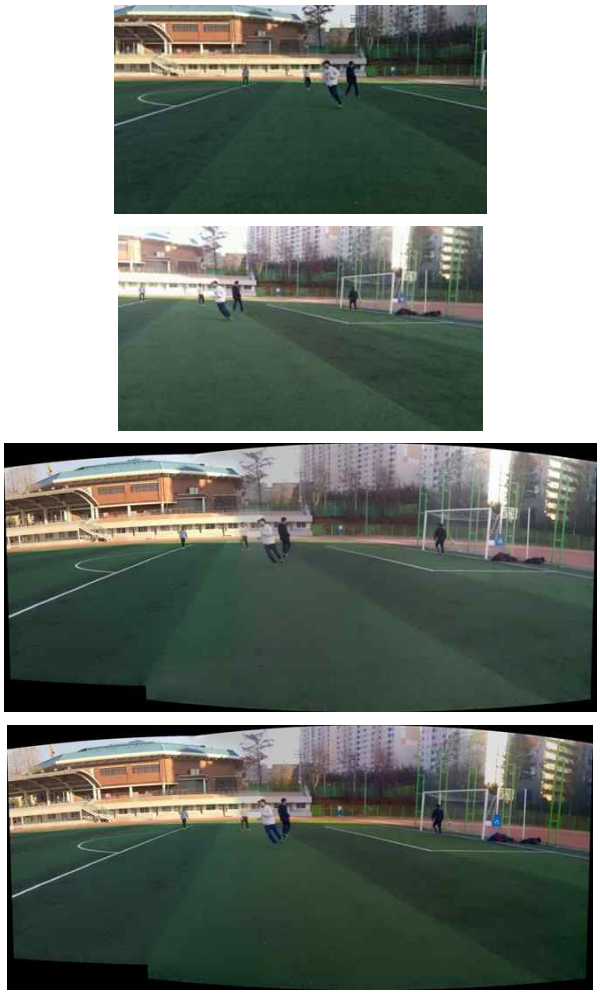


Fig. 6 Color Transfer based Color matching (top: Left Video Frame (iPhone 6 plus), Right Video Frame (HTC One M8), middle: Stitched Video Frame without Color Transfer. bottom: Stitched Video Frame applying Color Transfer)

으며, 본 연주자들도 현재 연구 중에 있다.

1. 옵티컬-플로우에 사용할 화면 F_n 및 F_{n+1}의 특징점[14]을 구한다.

2. 특징점의 프레임 간 움직임을 찾기 위해 Lucas-Kanade 옵티컬 플로우 [16]의 희소 반복 버전 [15]을 사용한다.

$$u = \frac{-\sum f_{y_i}^2 \sum f_{x_i} f_{t_i} + \sum f_{x_i} f_{y_i} \sum f_{y_i} f_{t_i}}{\sum f_{x_i}^2 f_{y_i}^2 - (\sum f_{x_i} f_{y_i})^2},$$
$$v = \frac{-\sum f_{x_i}^2 \sum f_{y_i} f_{t_i} + \sum f_{x_i} f_{y_i} \sum f_{x_i} f_{t_i}}{\sum f_{x_i}^2 f_{y_i}^2 - (\sum f_{x_i} f_{y_i})^2} \quad (4)$$

이를 통하여 이전 프레임과 현재 프레임간의 매칭점 쌍의 집합을 얻는다.

$$Mathing\,Points = \{(P_n(i), P_{n-1}(i))\} \quad (5)$$

3. 2 세트의 매칭점을 사용하여 그들 사이의 최적의 어파인 (affine)변환을 추정합니다. 우리는 회전 행렬 R (2 × 2)과 이동벡터 $\mathbf{t} = (t_x, t_y)$벡터를 다음 식을 통하여 같이 얻는다. 각각의 현재화면의 매칭점 $P_n(i)$과 이전화면의 매칭점 $P_{n-1}(i)$ 들에 대해 최소가 되는 행렬을 찾아낸다. (OpenCV의 estimateRigidTransform를 사용)

$$[R^*|t^*] = \arg min_{R,t} \sum_i |P_n(i) - P_{n-1}(i) - \vec{t}|^2 \quad (6)$$

계산된 어파인 행렬의 2차원 데카르트 좌표와 3차원 Homogeneous 좌표는 다음과 같다.

$$\begin{bmatrix} \cos\theta & -\sin\theta & -t_x \\ \sin\theta & \cos\theta & -t_y \end{bmatrix}$$

또는

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & -t_x \\ \sin\theta & \cos\theta & 0 & -t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad (7)$$

4. F_{n+1}에 처음 화면에서 계산한 M을 곱한 변환행렬을 이용하여 화면의 기하변환에 사용한다.

4. 실험 결과

4.1 평가 방법

본 연구의 성능지표는 제안된 방법을 통하여 사용자가 만족할 만한 화질의 스티칭 결과 영상을 얻을 수 있는 가하는 것과, 계산양 감소에 어느 정도 효과가 있는 가이다. 스티칭 영상의 화질의 평가는 기존의 연구에서는 기준 영상을 정의하지 못해서 정량적으로 수행되지 못했다. 그 이유는 영상 화질평가에 주로 사용하는 PSNR은 기하외곡이 없는 경우에 화소값의 외곡을 평가하기 위한 방법으로, 본 연구와 같이 기하외곡을 평가하는 경우에는 적합하지 않기 때문이다. 본 연구에서는 기하외곡 등 주관적인 시각정보를 보다 잘 반영하는 MSSIM (Mean Structural Similarity Index Measure)를 사용한다. 평가를 위하여 기준 영상이 필요한데, 실험에 사용된 3개의 영상 중 좌우 영상만을 사용하여 스티칭한 후 중앙의 영상을 기준영상으로 사용하여 겹치는 영역을 MSSIM으로 평가하는 새로운 방식을 제안하고 사용한다.

$$MSSIM = MMSIM_{oveerlapped\,region}(I_{center\,warped}(x,y)), I_{warped\,stitched}(x,y)) \quad (8)$$

간 단계별 실행 시간을 측정하고 이를 평균하여 비교하였다. 비교를 위하여 화면단위 스티칭 방법의 결과를 함께 제시하며, 실험에 사용한 환경은 OpenCV 3.2.0의 Image stitching 모듈을 사용하고, 컴퓨팅 환경은 Windows 10, Visual studio 2013, 메인 프로세서 Intel Xeon으로 구성하였습니다. 참고로, 현재 배포된 OpenCV의 이미지 스티칭모듈의 GPU버전은 제대로 최적화가 되어 있지 않아서 CPU 버전에 비하여 속도가 빠르지 않으며, 다중 쓰레딩을 사용하지 않았기 때문에 CPU core와 상관없이 하나의 core만을 사용한다. 따라서 이를 고려하여 결과해석을 해야 한다.

실험에는 두 종류의 영상을 사용하였다. Soccer 영상은 Left-Center-Right 모두 2K 해상도 영상이고, Concert는 Center 영상의 2K Left와 Right 영상은 1K 영상으로 구성하였다. 각 각 길이는 30초와 60초로 구성하였고, 녹화 시작 시간은 3초 이내의 차이를 주었다.

4.2 평가 결과

(a) Frames of the input Videos V_1, V_2, V_3. (Left to Right)

(b) Frames of the input videos V_1, V_2, V_3 in the Panorama Grid.

(c) Optical Flow between each individual current and previous Frames

(d) Calculated Rigid Transform, M for each current Frame and Warp Affine with M.

(e) Warped current Frames of V_1, V_2, V_3 aligned to the position of the previous Frame

Fig. 7 Transformation of the Video Frames according to the proposed algorithm.

Table 1은 Soccer 영상의 50 개의 연속적인 비디오 프레임에 걸친 스티칭 파이프라인단계에 필요한 평균 시간을 측정한 결과이다. 프레임수가 증가하면 영향이 미미하기 때문에 계산에서 초기 프레임은 제외했다. 제안 된 알고리즘에서는 두 번째 프레임에서 전역 움직임을 계산하기 전에 이미지의 특징점을 검색하는 과정의 시간이 포함됩니다. 변환 행렬을 얻는 과정에서 실행시간이 무려 5%정도로 감소한다. 그러나 총 계산시간을 비교 한 경우 약 25%정도의 감소를 얻었다. (Concert 영상도 거의 유사한 결과를 얻음) 그 이유는 멀티밴드 블렌딩의 시간이 전체의 비중에서 50%이상을 차지하기 때문입니다. 따라서 멀티밴드 블렌딩을 대치하는 고속 기법의 연구가 필요하다.

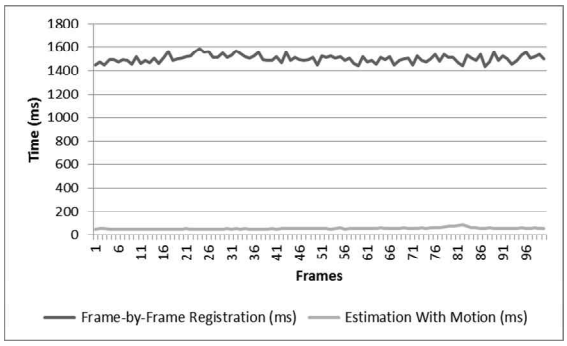
Fig. 8은 2 종의 실험영상의 결과 파노라마 프레임의 출력을 보여준다. 비디오의 처음 100 프레임에 대한 상세한 계산시간 비교는 다음과 같다. 첫 번째 데이터 세트 (그림 8. (a))가 고려 될 때,

Table 1 Average Time Consumption of Frame-by-Frame and Proposed Algorithm (Based on the next fifty frames after initial registration)

	Frame-by-Frame [ms]	Proposed [ms]
Feature Finder	2747.30	0
Feature Matcher	1495.06	0
3-D Projection Estimation	4564.54	0
Optical Flow	0	336.16
Registration Sub-Total	**8806.9**	**336.16**
Color Transfer	5054.89	5033.70
Warp	4383.34	4464.65
Exposure Compensation	1629.58	1420.78
Seam Selection	4049.43	5055.83
Blending	5711.88	5891.95
Compositing Sub-Total	**20829.12**	**21866.91**
Total	**29636.02**	**22203.07**

조밀 한 질감의 배경, 무작위 플래시 조명 효과, 움직이는 전경층, 연기자 및 백그라운드의 디지털 화면이 프레임에서 감지된다. 우리는 프레임 사이의 매칭 포인트의 진폭이 낮으면서 각 프레임에서 더 높은 특징 포인트의 빈도를 관찰했다. BestOf2NearestMatcher 알고리즘은 동위 원소 결정에 상당한 영향을 미치는 기본 RANSAC 방법에서 많은 인라이어(inliers) 및 아웃라이어(outliers)를 발견했다. 두 실험 영상을 비교 했을 때, 콘서트 영상은 1400-1600 ms도 50-70ms 로 약 26배, 축구장 영상 (750 프레임) 700-800 ms 대 50-70ms로 평균 13배 정도의 성능 향상을 보였다. 이 차이는 화면별 변환 행렬 계산 방식이 영상의 특징 (아웃라이어 정도)에 따라 계산양에 차이가 있기 때문이다.

Fig. 9에서 Soccer 영상의 스티칭 후의 영상품질성능을 MSSIM로 보인다. 화면간 비교의 경우 제안된 실험과 거의 비슷한 결과를 얻었고, 두 방법 간의 편차는 그래프 상으로는 거의 구분할 수 없을 정도의 차를 보여서 포함하지 않았다. 최종 합성된 데모 영상은 인터넷 https://www.youtube.com/watch?v=v8uwV8lP0Xw 에서 확인할 수 있다.



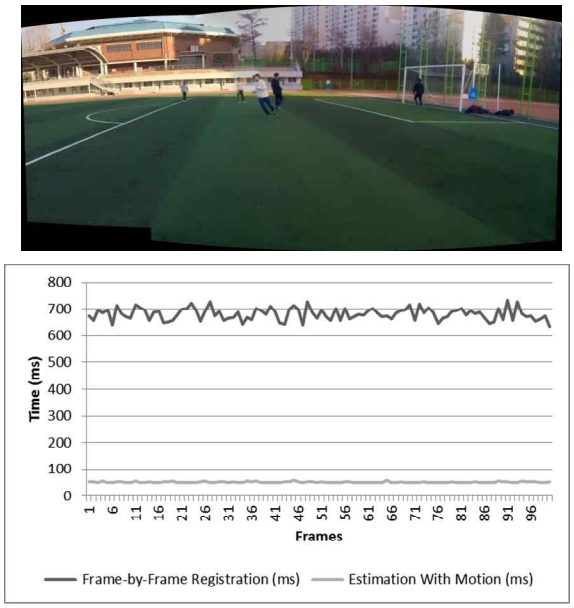


Fig. 8 Comparison of Performance of the two methods for a (First) Concert, (Second) Soccer Game use cases.

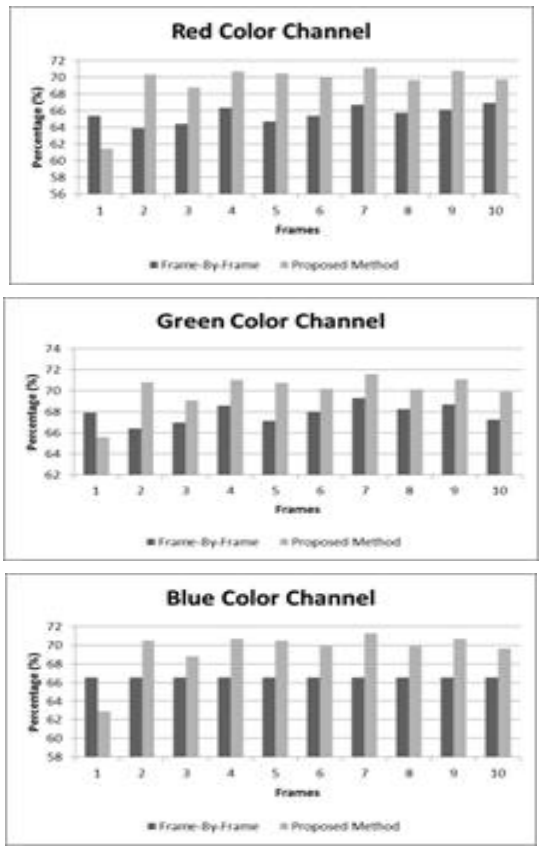


Fig. 9 Stitching Accuracy of different methodologies (MSSIM).

5. 결 론

특수 제작된 카메라가 아닌 휴대폰으로 촬영된 다수의 영상을 합쳐서 파노라마영상을 만드는 작업은 응용측면에서 필요하나, 기존 개발된 방식으로는 어려운 문제이다. 본 연구에서는 이러한 응용을 수행하기 위하여 필요한 기술인 시간동기화, 색상 일치, 화면 일치방법을 개발하고 실험하였다. 특히, 계산양이 많이 요구되는 화면변환 변수를 계산함에 있어서, 매화면마다 새롭게 변환을 계산하는 대신, 처음 계산된 변환을 바탕으로 각 영상의 전역움직임을 추정하여 갱신하는 방식으로 계산양을 줄임과 동시에 영상 화질 면에서도 동일한 결과를 얻을 수 있음을 확인하였다.

그러나, 본 연구에서는 카메라를 들고 있는 사람들이 큰 움직임이 없다는 가정 하에 알고리즘을 개발하고 실험하였으며, 걸어 다니거나 하는 경우는 고려하지 않았다. 또한 다수의 영상을 촬영하여 다양한 실험을 통하여 검증하지 못하였으며, 화면의 화질을 계산할 때도 MSSIM외의 화면 스티칭에 적합한 기준을 개발하여 평가하지 못한 점 등 제약사항이 존재한다. 이외에도 기존 이미지 스티칭에서 해결하지 못한 문제들도 여전한 해결하야 할 과제들이다. 다만 이러한 문제들은 본 연구의 한계이기도 하지만, 현재 이 분야의 연구에서 지속적으로 연구해야하는 문제들입니다. 이러한 문제 중에서 본 연구자들은 GPU를 사용한 실시간 스티칭 시스템의 개발과 움직임이 크게 발생한 경우에 대한 해결책 등에 대하여 추가 개발을 진행 중이다.

References

[1] Ko H.-S., “Cyberspace and Protection of Personal Data,” The Journal of Internet Electronic Commerce Research, Vol. 3, No. 2, pp. 37-63, 2003.

[2] Joo J.-H., “The Development of a Cyber World Culture Expo and Electronic Tourism Market System,” The Journal of Information Systems, Vol. 9, No. 1, pp. 87 ~ 108, 2000.

[3] Brown, M. and Lowe, D.G., “Automatic Panoramic Image Stitching Using Invariant Features. International Journal of Computer Vision,” Vol. 74, No. 1, pp.59-73, 2007.
[4] Collection of 360° Video Rigs, https://thefulldomeblog.com/2015/11/17/collection-of-360-video-rigs/ (last access, 2017. May, 26)
[5] PTGui, https://www.ptgui.com/(last access, 2017. May, 26)
[6] Kolor, http://www.kolor.com/panotour/ (last access, 2017. May, 26)
[7] Vahana Video-Stitch, http://www.video-stitch.com/(last access, 2017. May, 26)
[8] ICoSOLE EU project, http://icosole.eu (last access, 2017. May, 26)
[9] El-Saban, M.A., Refaat, M., Kaheel, A. and Abdul-Hamid, A., “Stitching Videos Streamed by Mobile Phones in Real-Time,” The 17th ACM International Conference on Multimedia, pp. 1009-1010. 2007
[10] Reinhard E., Ashikhmin M., Gooch B., and Shirley P., “Color Transfer between Images. In Applied Perception,” IEEE Computer Graphics and Applications, September/October 2001.
[11] Learning OpenCV: Color Transfer Between Images algorithm. available online http://www.programering.com/a/MzM4gDMwATU.html
[12] Ruderman D.L., Cronin T.W., and Chiao C.C., “Statistics of Cone Responses to Natural Images: Implications for Visual Coding,” J. Optical Soc. of America, Vol. 15, No. 8, pp. 2036-2045, 1988
[13] Nghia Ho, Simple Video Stabilization Using OpenCV. At http://nghiaho.com/?p=2093
[14] Shi J. and Tomasi. C. “Good Features to Track,” IEEE Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.
[15] Jean-Yves Bouguet. “Pyramidal Implementation of the Lucas Kanade Feature Tracker,” Intel Corporation, Vol. 5, No. 4, 2001
[16] Lucas, B., and Kanade, T. “An Iterative Image Registration Technique with an Application to Stereo Vision,” the 7th International Joint Conference on Artificial Intelligence, pp. 674-679, 1981.
[17] Bano, S., and Cavallaro, A., “ViComp: Composition of User-Generated Videos.” Multimedia Tools and Applications Vol, 75, No. 12, pp.7187-7210, 2016.
[18] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. "Image Quality Assessment: from Error Visibility to Structural Similarity." IEEE Transactions on Image Processing Vol.13. No. 4, pp. 600-612, 2004.

미어 사데크 빌라흐
(Meer Sadeq Billah)

- 정회원
- 2007년 7월: BGC Trust University Bangladesh 컴퓨터공학 학사
- 2008년 2월- 2015년 6월: 방글라데시 Sansons ㈜ 소프트웨어 엔지니어
- 2012년 7월- 2015년 8월: 삼성전자 발글라데시, 소프트웨어 엔지니어
- 현재: 서울과학기술대학교 전기정보공학과 석사과정
- 관심분야: 컴퓨터 비젼, 3D 영상

안 희 준 (Ahn Heejune)

- 정회원
- KAIST 전기정보공학과 박사
- 서울과학기술대학교 전기정보공학과 정교수
- 관심분야 : 인터넷 프로토콜, 영상처리, 데이터마이닝