이미지 Edge Line Segmentation 알고리즘을 통한 고속 이미지 스티칭 기법

채호균, 박혜림, 김윤정, 임지헌, 김규헌
경희대학교
hkchae8@naver.com, hl.park@khu.ac.kr, yunjung6629@gmail.com
dlawlgjs1780@khu.ac.kr, kyuheonkim@khu.ac.kr

Fast Image Stitching Based on Image Edge Line Segmentation Algorithm

Chae, Hogyun Park, Healim Kim, Yunjung Im, Jiheon Kim, Kyuheon
Kyung Hee University

요 약

지금까지 영상 콘텐츠 제작 기술의 발전은 SD(Standard Definition)에서 시작하여 HD(High Definition)와 FHD(Full High Definition)를 거쳐, UHD(Ultra High Definition)에 이르기까지 화질을 중심으로 이루어져 왔다. UHD 에 이르며 육안으로는 그 이상의 해상도로 제작된 콘텐츠와 구분하는 것이 힘들어졌으며, 이에 영상 콘텐츠 제작은 화질이 아닌 제한된 촬영 장비들로부터 촬영 방법, 영상 화각의 개선 작업 등으로 그 방향을 전환하고 있다. 이의 연장선 상에서 360 도 영상에 대한 기술개발이 활발히 이루어 지고 있다. 방송 분야에서는 360 도 영상의 실시간 스트리밍 적용 가능성이 모색되고 있는데, 이것이 가능 하려면 대량의 동영상 데이터를 실시간으로 스티칭하여 전달하는 기술이 필요하다. 따라서 고속 이미지 스티칭이 가능해질 경우 실시간 동영상 스티칭을 통해 방송 통신 분야에서의 서비스 향상에 기여할 것으로 보인다. 본 논문은 이미지의 edge 정보를 방향성을 가진 데이터로 분할하여 특징점을 추출하고, 이후 가중치를 통한 특징점 매칭으로 기존의 이미지 스티칭 방법 보다 빠른 속도의 알고리즘을 제안한다.

1. 서론

고성능 디지털 카메라의 발전으로 영상은 스마트폰, 드론 등을 이용해 쉽게 획득되고 방송, 영화 산업, 철도 기술 분야 등의 많은 곳에서 활용되고 있다. 그 중에서 이미지를 정합하여 사용하는 이미지 스티칭 방법은 좀더 넓은 화각의 이미지로서, 이미지 품질 향상을 위해 많은 연구가 진행되고 있다. 하지만 기존에 활용되는 SIFT, SURF 등의 알고리즘을 통한 이미지 스티칭에서는 회전 불변성, 조명, 이미지의 크기 등에 Robust 하기 위해 특징점 추출 과정에서 적분 연산, wavelet 등을 사용하고 이를 통한 정보 추출 과정이 계산량이 많아 오랜 수행 시간이 소요되며 실시간 이미지 스티칭을 하기에는 부족함이 존재한다. 따라서 우리는 기존 영상 처리 알고리즘의 방대한 이미지 스티칭의 실시간 처리를 가능하게 하도록 특징점 추출 및 매칭 과정에서 계산량을 줄일 필요성을 느꼈다. 본 논문에서는 이미지 edge 정보를 기반으로 기존의 특징점 추출 및 매칭 방법을 수정해 연산 속도를 줄이는 방식을 제안 하고자 한다.

제안하는 스티칭 알고리즘에서 이미지 edge 정보를 추출하고 해당 정보를 분할하여 그에 따른 특징점 추출 방식을 설명하고, 이후 분할된 edge 정보를 이용하여 이미지 스티칭 대상 각각의 특징점들을 비교하는 알고리즘을 소개한다. 매칭이 완료된 특징점들은 Outlier 제거 방법을 통해 대상 이미지 들을 최종 정합한다. 실험 결과로 알고리즘 수행 시간의 단축으로 고속화 된 스티칭 알고리즘을 보인다. 마지막으로 이미지의 edge 정보들 만을 활용한 스티칭 기법의 한계와 그것을 극복하기 위한 방법을 제시한다.

본 논문에서 다루는 전체적인 스티칭 과정은 그림 1 과 같은 구조도를 따른다.

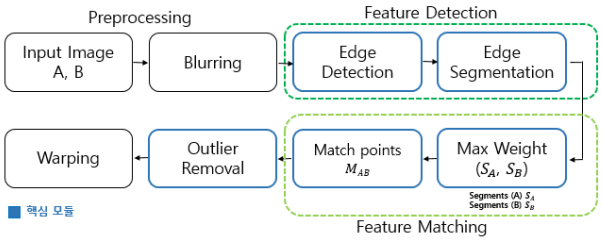

그림 1. 구조도

본 논문의 구성은 다음과 같다. 2 절과 3 절에서 제안하는 방법의 핵심인 이미지의 edge 정보 분할 및 특징점 추출 방법과 이에 따른 특징점 매칭 방법, 4 절에서 제안하는 방법과 기존의 스티칭 알고리즘과의 차이를 보이고, 5 절에서 본 논문에 대한 결론 및 향후 연구 과제로 마무리 한다.

2. Image edge segmentation

이미지 스티칭 알고리즘에서 가장 중요한 것은 각 이미지들의 특징점을 추출한 후 해당 특징점들에 대응하는

좌표들을 연결하는 매칭 작업이다. 하지만 이 과정에서 많은 계산이 필요하기 때문에 시간을 많이 소요하게 된다. 제안하는 방법에서는 기존의 특징점 추출 방법 대신 이 절에서 설명하는 방법을 사용하여 계산량을 줄인다. Image edge segmentation 과정은 특징점을 추출하기 위한 작업으로, edge detection 과정을 통해 얻은 edge 정보를 주위 픽셀들의 방향 정보로 분할하는 과정을 거쳐 여러 개의 segment 들로 나눈다.

edge 정보를 추출하기 위해 전처리 작업으로 이미지 내의 잡음을 제거하는 Blurring 과정이 필요하다. 이때 Bilateral filter 를 사용하여 사소한 잡음을 제거하고 제안하는 방법에서 필요로 하는 edge 부분만을 강조하였다.

edge detection 은 대상 픽셀 주위의 3*3 픽셀에 대하여 $^{R}/_{3}+^{G}/_{3}+^{B}/_{3} > T$ (T: threshold) 일때 데이터를 표시한다. 이후 추출한 edge 정보는 주위픽셀들의 방향 정보로 분할하여 segmentation 작업을 하게된다.

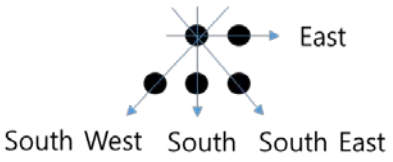

그림 2. Edge 의 방향 정보

변환된 edge 이미지의 최상단 제일 왼쪽 픽셀부터 오른쪽 방향으로 탐색하여 처음으로 값이 존재하는 픽셀의 좌표값(X,Y)을 root 로 저장하게 되고, 그림 3 에서와 같이 root 픽셀 주위의 8 개의 방향 중 4 개의 방향인 동쪽, 남동쪽, 남쪽, 남서쪽의 방향 픽셀들에 대해서 값이 존재하는지 찾는다. 값이 존재하는 방향을 찾았다면 해당 방향을 root 좌표값 이후에 저장하게 된다.

픽셀 주위 방향들을 'Type' 이라는 형태로 데이터에 저장하고 방향이 저장된 픽셀 위치 주위의 8 개 픽셀들에 대하여 데이터가 없을 때까지 위 과정을 반복적으로 진행한다.

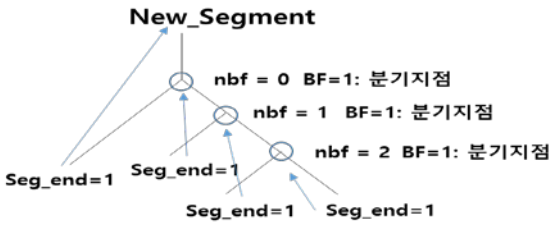

그림 3. Edge Segmentation

만약 픽셀 주위에서 2 개 이상의 방향들에 대해 데이터가 존재한다면 해당 픽셀을 분기점(BF)으로 표시한 뒤, 발견한 방향으로 'Type' 정보를 저장하며 주위의 8 개 픽셀들에 대하여 데이터가 없을 때까지 진행한다. 데이터가 없다면 분기점(BF)으로 되돌아와 마찬가지 방법으로 주위의 8 개 픽셀들에 대해 데이터가 없을 때까지 반복한다.(그림 3)

<1 segment data>

size	edgecount	(X, Y, Type)(Edge Type)*

그림 4. Segment Data

위 작업을 거친 edge 정보는 그림 4 에서와 같은 여러 개의 segment 들로 분할된다. 이렇게 나뉜 segment 들의 root 좌표와 분기점(BF) 좌표들을 이미지 스티칭 과정에서 특징점 후보로 사용한다.

3. Feature Matching

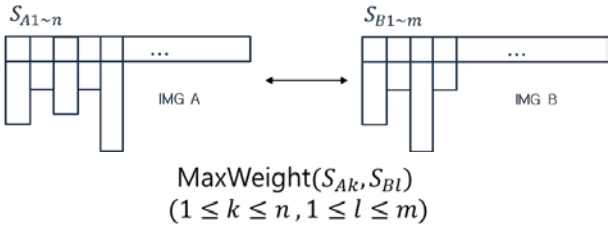

그림 5. Segment 비교

edge segmentation 을 통해 얻은 이미지 A, B 각각에 대한 edge 정보들은 방향 정보들을 포함한 여러 개의 segment 들 ($S_{A1\sim An}$, $S_{B1\sim Bm}$)로 나뉜다. Feature Matching 단계에서는 segment 에 저장되어 있는 'Type' 정보들의 배열에 대하여 유사도에 따라 방향이 동일 하다면 +20, 방향이 다르다면 −50 의 가중치를 부여한다. 가중치 값이 가장 큰 연결점이 가장 유사한 segment 들로 정해져 특징점 매칭이 이루어진다. 이렇게 연결된 segment 들은 root 좌표들의 값을 통해 전체 이미지 연결점에서 크게 벗어난 각각의 연결점에 대해 두 종류의 소거 작업이 진행된다.

매칭을 위한 소거작업으로 먼저 대상 이미지와 참조 이미지 간의 위치에 따라 특징점 X, Y 값 좌표 차이에 대한 부호를 고려하면 이미지 정합 위치에 따라 다음과 같이 표현된다.

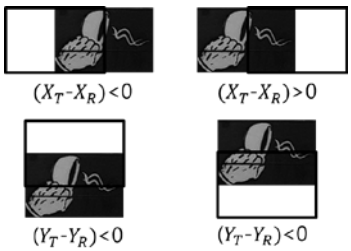

그림 6. 정합 위치 선정

따라서 segment 간의 root 좌표 차이에서 오는 부호에 따라 이미지 정합 위치를 선정하고, 그와 반대되는 부호의 매칭 좌표들을 소거한다.

그 다음은 X, Y 좌표 차이들의 평균에 크게 벗어난 후보 특징점 좌표들을 소거하는 작업을 수행한다.

$$|(X_A-X_B)| < \frac{5}{4} * Avg(X_A-X_B),\ |(Y_A-Y_B)| < \frac{5}{4} * Avg\,(Y_A-Y_B)$$

위의 식은 스티칭 대상 이미지들(A, B)로부터 추출한 특징점의 X 와 Y 좌표 차로부터 소거 작업을 하는 과정을 나타낸다. 여러 개의 데이터 세트로 실험해 본 후, 귀납적 방법으로 위와 같은 소거 과정의 식을 도출하였다. 좌표 차이들의 평균을 구한 후, 각 segment 의 좌표 차이들 중 평균값의 일정 배수보다 작은 값만 보존하고 이 범위에서 벗어난 값은 소거한다. 이 과정은 실험적인 방법이므로 추가적인 연구를 통해 수정 될 수 있다. segment 정보들을 통해 feature

matching 까지 이르는 과정은 다음과 같다.

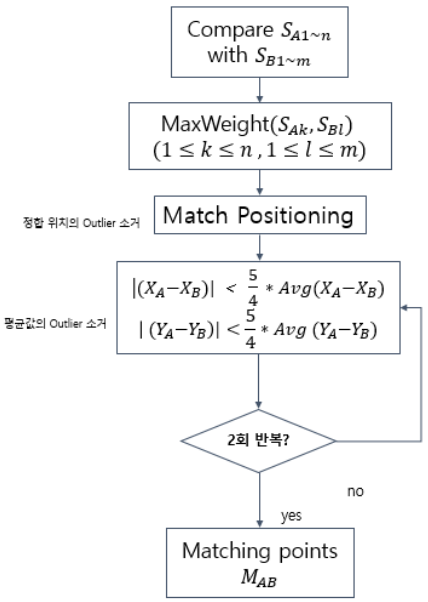

그림 7. Feature Matching 흐름도

매칭 포인트를 추출하고 RANSAC 알고리즘을 이용해 outlier 를 제거하였다.

4. 실험 결과

알고리즘을 검증하기 위하여 본 논문에서는 HD 규격의 두 개의 이미지 세트를 사용한다. 첫 번째 실험 이미지는 모바일 휴대폰 내의 소형 카메라로 근거리에서 촬영한 이미지이며, 두 번째 실험 이미지는 풍경의 일부를 담은 원거리 이미지이다. 실험 환경은 Microsoft Visual Studio 2010 과 OpenCV 2.7.9 라이브러리를 이용하여 구현하고, 인텔 i5 프로세서를 사용하였다.

그림 8. 실험 이미지 세트 (Frame)

위 이미지는 촬영 거리 5m 이내의 근거리 이미지이며, edge 정보를 추출하기 위해서는 전처리 작업으로 배경 내의 노이즈 제거가 필요하다. bilateral filter 로 blurring 작업을 수행하고 제안하는 알고리즘으로 스티칭을 한 후 정합한 결과는 다음과 같다.

그림 9. Edge Line Segmentation 알고리즘 스티칭 후 정합된 이미지(좌), SURF 알고리즘 스티칭 후 정합된 이미지(우)

정합 전까지의 수행 시간을 측정한 결과, 특징점을 추출하여 매칭하기까지 걸린 시간은 954ms 이다. SURF 알고리즘의 측정 시간은 2755ms 로, SURF 대비 65% 시간이 단축되었다.

그림 10. 실험 이미지 세트 (Plaza)

위 원거리 이미지를 입력으로 하여 스티칭을 한 후 정합한 결과는 다음과 같다.

그림 11. Edge Line Segmentation 알고리즘 스티칭 후 정합된 이미지(좌), SURF 알고리즘 스티칭 후 정합된 이미지(우)

정합 전까지 소요된 수행 시간은 181ms 이다. SURF 알고리즘의 수행시간은 1087ms 로, SURF 대비 83% 시간이 단축되었다.

추가적으로 HD 규격에는 미치지 못하지만 qHD 규격의 두 개의 실험 이미지 세트에 대해서 실험해본 결과는 다음과 같다.

그림 12. 실험 이미지 세트 (Rooftop)

그림 13. Edge Line Segmentation 알고리즘 스티칭 후 정합된 이미지(좌), SURF 알고리즘 스티칭 후 정합된 이미지(우)

정합 전까지 소요된 수행 시간은 257ms 이다. SURF 알고리즘의 수행시간은 1182ms 로, SURF 대비 78% 시간이 단축되었다.

그림 14. 실험 이미지 세트 (Poster)

그림 15. Edge Line Segmentation 알고리즘 스티칭 후 정합된 이미지(좌), SURF 알고리즘 스티칭 후 정합된 이미지(우)

정합 전까지 소요된 수행 시간은 115ms 이다. SURF 알고리즘의 수행시간은 587ms 로, SURF 대비 80% 시간이 단축되었다.

	영상크기	Edge Segmentation (sec)	SURF(sec)	감소율(%)
Frame	1440*1080	0.954	2.755	65.372051
Plaza	1328*747	0.181	1.087	83.348666
Rooftop	960*720	0.257	1.182	78.257191
Poster	720*960	0.115	0.587	80.408859
평균		0.37675	1.40275	73.14204

표 1. 이미지 세트 별 실험 결과 속도 비교

표 1 에서 보는 것과 같이 기존 스티칭 방법보다 수행시간이 평균 70% 이상 단축되었으며 스티칭 된 최종이미지도 서브젝티브 테스트 결과 만족스러운 결과를 제공하고 있다.

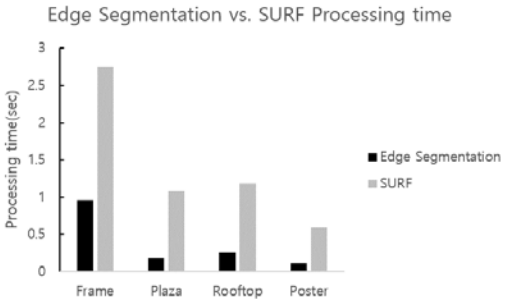

그림 16. 실험 결과 속도 비교

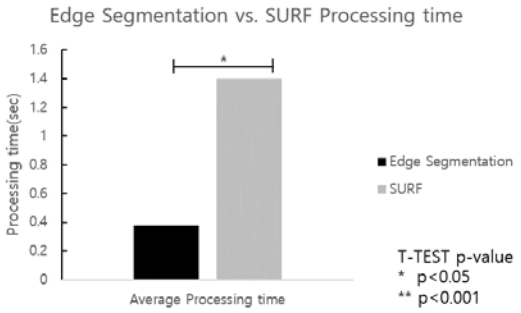

그림 17. T-test 결과

본 논문에 실은 4 가지 실험 영상 세트를 입력으로 하였을 때의 processing time 은 SURF 대비 평균 73.1% 감소했다. T-test 결과, processing time 감소는 통계적으로 유의하다.

5. 결론 및 향후 연구 과제

본 연구에서 진행한 이미지 edge line segmentation 알고리즘을 통한 고속 이미지 스티칭 기법에서 이미지의 edge 정보만을 통해서 기존 스티칭 방법에서보다 3 배 이상 향상된 스티칭 속도를 확인하였다. 이 방법을 사용하면 왜곡이 적은 영상을 빠르게 정합하는 작업에 매우 효과적이다. 하지만 전처리 작업으로 bilateral filter 를 이용하였기 때문에 정합이 끝난 이미지 값이 부분적으로 blur 처리가 된다는 점이 존재한다. blur 처리된 영상의 문제점을 해결하기 위해서는 bilateral filter 된 영상을 이용해 매칭 좌표를 추출하고 추출된 매칭 좌표를 원래의 이미지에 적용하여 결과값을 도출하는 방식을 통해 문제를 해결할 수 있다.

제안하는 방법은 이미지 영상에서 위에서 아래로의 edge 데이터 방향성을 저장하는 방식을 사용하기 때문에 회전 각도에 제한적(rotationally asymmetric)인 스티칭 방법이다.

향후 제안하는 방법에서 회전 각도를 고려하여 회전된 이미지에서도 Robust 한 스티칭에 대한 연구가 필요하다.

* 본 논문은 미래창조과학부 및 정보통신기술진흥센터의 정보통신 · 방송 연구개발사업의 일환으로 수행하였음. [2017-0-00224, UHD 방송콘텐츠 기반 지능형 Dynamic Media 생성, 분배 및 소비 기술 개발]

참고문헌

[1] Hyo Chang Ahn, Sang Burm Rhe, “Fast Image Stitching Based on Improved SURF Algorithm Using Meaningful Features”, Korea information processing society, journal B, Vol.19, No.2, 2012.

[2] H Bay et al,. “Surf: Speeded up robust features”, EECV 2006, pp 404-417, 2006

[3] Son Jong-In et al., “View invariant image matching using SURF”, 한국방송공학회 학술발표대회 논문집, Vol.2011 No.7, 2011

[4] Bongjoe Kim et al., “Illumination Robust Feature Descriptor Based on Exact Order”, 방송공학회논문지, Vol.18 No.1, 2013