특징점 기반 이미지 스티칭을 활용한 360도 공간 조회 기술

Space inquiry technology using feature point-based image stitching

김상현, 권민지, 변경탁

[Abstract]

The purpose of this paper is to improve the knowledge level of stitching-related algorithms and successful panoramic video production according to the recently rising interest of VR. The research method is to summarize image stitching techniques and algorithms used in the research method through research on paper-related data. In addition, Exploring OPEN CVs used for image stitching is too. As a result of the analysis, we thought it would be best to use Stitcher, one of the OPEN CVs provided by Python, and we decided to learn how to use it and apply it to the project. In conclusion, we learned deeply about SIFT and SURF, the two most commonly used algorithms in image stitching, and applied Python OPEN CV to successfully create 360-degree panoramic images and viewable on the Web page.

▸ **Key words** : Image-stitching, SURF, SIFT, Panorama, VR

[요 약]

이 논문의 연구목적은 최근 상승하고있는 VR의 관심도에 따라 이미지스티칭을 활용한 성공적인 파노라마 영상제작 및 스티칭 관련 알고리즘에 대한 지식수준 향상입니다. 연구방법은 논문 관련 자료조사를 통한 이미지스티칭 기법 및 해당 기법에 사용되는 알고리즘 정리와 이미지스티칭에 활용되는 OPEN CV 탐색입니다. 분석결과, 파이썬에서 제공하는 OPEN CV 중 하나인 Stitcher를 사용하는 것이 우리 프로젝트에 가장 적합하다고 생각되었으며 사용법을 익혀 프로젝트에 적용하기로 했습니다. 결론적으로 이미지 스티칭에서 가장 많이 사용하는 알고리즘 두 가지인 SIFT와 SURF에 대해 깊게 알게되었으며, 파이썬 OPEN CV를 적용하여 파노라마용 이미지를 여러장 받아 360도 조회가능한 파노라마 영상을 성공적으로 만들었으며 페이지에서 조회 가능하도록 하였습니다.

▸ **주제어** : 이미지스티칭, SURF, SIFT, 파노라마, VR

I. 서론

최근 전 세계적으로 VR (Virtual Reality) 기술이 빠른 발전과 함께 주목받기 시작하면서 생동감 넘치는 VR 컨텐츠를 볼 수 있는 360° 파노라마 사진과 영상이 많은 관심을 받고 있다. 이미지 스티칭 기술은 360° 파노라마 사진과 영상을 제작하는데 주요한 기술로서 많은 연구가 활발하게 이루어지고 있다. 이와 함께 상용화가 많이 되고 있는 기술이, 360도 camera이고, 또한 스마트폰의 panorama 촬영 기능이다. 360도 카메라는 근본적으로 panorama 촬영과 같은 원리를 갖는다. 360도 카메라는 2개 이상의 렌즈를 이용해 2개 이상의 방향으로 주변을 촬영한후, Image Stitching 기술을 통해, 이를 연결하여, 임의의 각도로 보아도 연속된 사진을 보여주게 하는 기술이다. 우리팀은 이번 프로젝트에서 Image Stitching 기술을 활용하여 Panorama영상을 만든 후 조회할 수 있는 기능을 구현해보고, 이를 개선하기 위한 아이디어에 대해 생각해보자 한다. 일반적으로 넓은 영역의 사진을 획득하는 방법은 크게 2가지로 나뉜다. 어안렌즈나 초광각 렌즈와 같은 넓은 화각을 지원하는 렌즈를 장착한 특수 카메라를 이용하는 방법과 여러 장의 사진에서 중첩된 부분을 찾아 정합하여 한 장의 사진으로 생성하는 방법이다. 일반 카메라 렌즈를 통해 얻어진 여러 개의 사진을 대상으로, 다양한 연산을 통해 같은 영역을 찾고 스티칭(Stitching) 과정을 거쳐 생성된 파노라마 영상은 넓은 시야각을 제공하여, 전시관 또는 스트리트 뷰(Street View) 등에서 점차 확대되고 있다. 일반적인 스티칭 알고리즘은 특징점 기반 이미지 스티칭을 기반으로 한다. 특징점 추출 연산을 기반하여 이미지 스티칭을 수행하는 과정에서 가장 중요한 것은 정확하고 빠르게 특징점을 뽑아내는 것이다. 이러한 특징점을 기준으로 매칭(Matching) 정합과정을 수행하기 때문에 수행 연산시간에 많은 영향을 미친다. 기존의 스티칭 알고리즘에서 동일 영역을 검출하는 방법으로, Shift Invariant Feature Transform(SIFT)과 Speeded Up Robust Feature(SURF) 기법이 가장 많이 사용되고 있다.

본 논문에서는 360도 공간조회 기술을 적용하기 위한 특징점 추출 방법을 기술하고자 한다. 이후 이러한 이미지 스티칭 기술의 한계와 보완점을 생각한 내용을 기술하고자 한다. 본 논문의 구성은 다음과 같다. 2절에서는 관련 기술 중 Shift Invariant Feature Transform(SIFT)과 Speeded Up Robust Feature(SURF) 기법에 대 기술한다. 3절에서는 2절에서 소개한 이미지스치팅 기법을 바탕으로 우리가 적용한 이미지 스티칭 기술을 작성한다. 이후 4절에서 우리 프로젝트에 활용한 결과를 제시하고, 5절에서 결론으로 마무리한다.

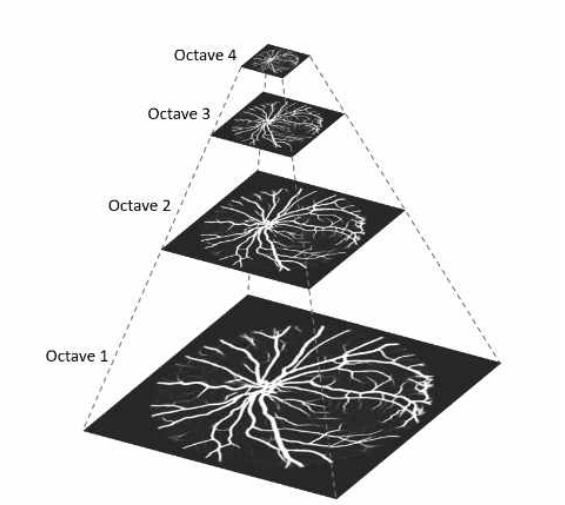
II. 관련 기법

1. SIFT Algorithm

SIFT Algorithm은 이미지의 크기 및 회전에 영향을 받지 않는 특징점을 추출하는 알고리즘이다. 따라서 이미지 유사도 평가나 이미지 정합에 활용할 수 있는 좋은 알고리즘이다. SIFT Algorithm은 Scale-space extrema detection, Keypoint localication, Orientation assignment, Keypoint descriptor 이 4단계로 이루어져 있다. 순서대로 이 4가지를 살펴보겠다.

1-1. Scale-space extrema detection

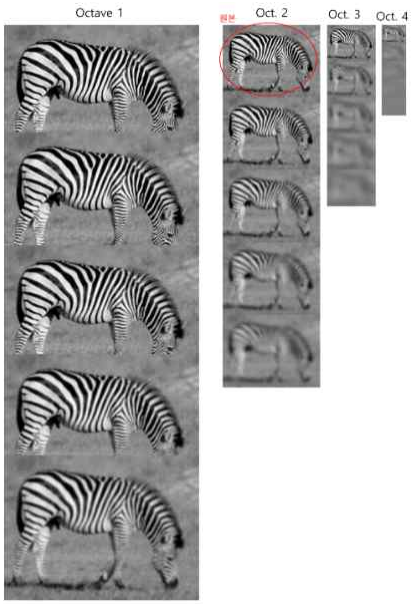
먼저 Scale-space (크기 공간)를 생성을 위해 원본 이미지를 다양한 크기로 resize해서 image pyramid를 만든다.



[image pyramid 예시]

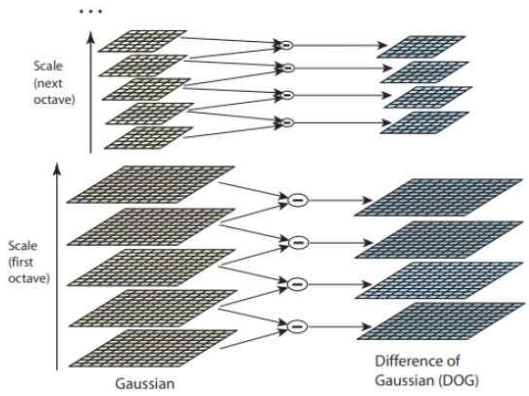
다음으로 image pyramid의 각 층 이미지(octave image)를 점점 더 커지는 σ(= gaussian blur scale factor)로 gaussian blurring를 적용한 이미지들을 얻는다. gaussian blurring이란 일반적으로 이미지 노이즈를 줄이고 세부 사항을 줄이기 위해 그래픽 소프트웨어에서 널리 사용되는 효과이다. 이렇게 Scale-space가 생성되며 Scale-space는 아래 이미

지들을 가리키는 용어다.



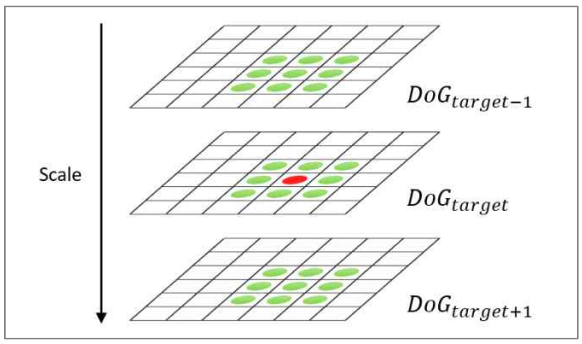
[Scale Space 만든 결과]

다음으로 Difference of Gaussian(DoG) 이미지를 구한다. DoG 이미지는 아래 그림처럼 같은 octave 내 서로 다른 두 개 gaussian blurred image로 빼기 연산을 수행하여 생성한다.



[Dog 구하는 과정]

마지막으로, 만들어진 DoG를 이용해 extrema detection을 수행한다. 해당 좌표가 극소점이거나 극대점이라 판단되면 이를 keypoint 후보군으로 분류한다. Extrema detection의 자세한 과정은 다음과 같다.



[Extrema detection 과정]

위 그림과 같이 DoG_(target)에서 target pixel (빨간점) 값을 총 26개의 주변 pixel 값과 비교해 극점인지 판단한다. 주변 pixel에는 다음이 해당된다.

(1) DoG_(target)에서 target pixel 주변의 8개 pixel

(2) DoG_(target-1)와 DoG_(target+1)에서 각 9개 pixel

1-2. Keypoint localization

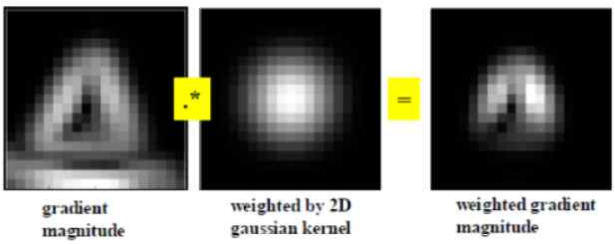
이렇게 뽑은 keypoint 후보군들 중에는 정확한 좌표계에 위치하지도 않고 활용가치가 떨어지는 것들도 있어서 most stable keypoint만을 선택하는 과정이 필요하다. 따라서 후처리로 keypoint selection이 진행된다. 이 단계까지 끝나면 keypoint는 scale-invariance를 갖게 된다.

1-3. Orientation assignment

Orientation assignment는 지금까지 찾은 keypoint의 방향을 결정하는 단계이다. 여기서 찾은 orientation을 이후에 keypoint descriptor에서 빼주면 rotation-invariance 성질을 갖게 된다. 먼저 keypoint를 중심으로 추출한 16x16 patch를 사용해서 Gradient magnitude와 Gradient orientation을 구한다 [Ref]. Gradient magnitude와 Gradient orientation을 구하는 식은 다음과 같다.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

이때 patch 추출에 사용되는 이미지는 해당 keypoint가 뽑힌 gaussian blurred image이다. 특정 Keypoint가 Octave2에서 k값(=scale factor)으로 gaussian blurring된 image에서 뽑힌 거라면, Gradient magnitude와 Gradient orientation 계산에도 Octave2에서 k값으로 gaussian blurring된 image의 pixel 값을 사용한다는 말이다. 이후 Keypoint에 방향 속성을 추가하기 위해 1 bin 당 10 degree을 담당하는 36개의 bin을 갖는 orientation histogram을 생성한다. 그런 다음 16x16 patch에서 구한 Gradient orientation 값들을 Gradient orientation이 해당되는 bin에 더하는데, 이때 Gradient magnitude에 gaussian-weighted circular window를 곱한 값을 bin에 더한다. 가중치를 부여하기 위함이다.



[그레이던트 크기에 가우시안 함수를 곱해준 결과]

bin들이 다 채워진 orientation histogram 상에서 가장 높은 값을 갖는 bin을 해당 keypoint의 방향으로 할당한다. 또한 가장 높은 bin의 80% 이상의 값을 갖는 bin이 추가로 존재한다면 그 방향을 갖는 keypoint를 새로 만든다.

1-4. Keypoint descriptor

마지막으로 Keypoint를 표현하기 위해 Descriptor를 생성해줘야 한다. 과정의 대부분은 Orientation assignment랑 비슷하고 약간의 차이가 있다. KeypointA에는 [X 좌표, Y 좌표, Scale factor, Octave, Orientation]의 정보가 담겨있다. Scale factor와 Octave 정보를 이용해 Scale-space에서 해당되는 gaussian blurred image를 가져오다. 이 image에서 KeypointA의 X 좌표, Y 좌표를 중심으로 16x16 patch를 추출한다. 이 patch를 16개의 4x4 sub-patch로 다시 분할한다. 한 개의 sub-patch에서는 8-bin orientation histogram을 생성해 한 sub-patch 당 8방향 정보를 담게 만든다. 결과적으로 한 개의 16x16 patch에서는 8방향 정보가 16개 생성되어 총 128개의 값을 얻게 되는데 이를 keypoint descriptor라 부른다. 후처리로 descriptor에서 keypoint orientation 값을 빼 rotation-invariance 성질을 부여하고 nomalization을 통해 밝기 의존성을 해결한다.

2. SURF Algorithm

SURF는 Speed-Up Robust Feature의 약자로 평면에서 회전에 불변하는 특징점 매칭 기법으로 노이즈에 강인하고 (robust to noise) 수행 시간이 빠른 알고리즘이다. SURF 알고리즘은 크게 특징점 추출, descriptor 추출, 특징점 매칭의 세 단계로 이루어져있다.

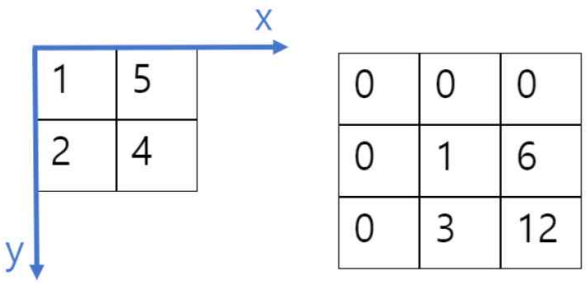
2-1. 특징점 추출

특징점 추출은 이미지의 적분 영상을 생성한 후 Hessian 검출기를 사용하여 진행한다. 적분 영상(integral image)이란 이미지 밀도 함수가 I(x,y)라 할 때, 픽셀 (x,y)에서의 픽셀 밝기값을 누적한 값이다. 적분 영상의 밀도함수를 나타내면 아래와 같다.

$$I_{\sum}(x,y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j)$$

[적분 영상 밀도함수]

적분영상은 다음곽 같이 구성된 이미지이다.

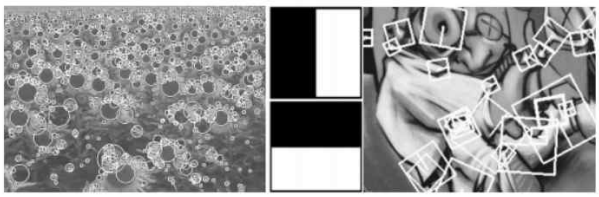


다음으로 Hessian 검출기를 사용하여 특징점을 추출한다. Hessian 행렬은 g(x,y,sigma)를 표준편차가 sigma인 가우시안 함수라 하고 함수 Lxy를 이 함수의 2차 미분값이라 할 때, 가우시안 필터의 2차 미분값이다. 앞서 설명한 헤시안 행렬식(Hessian determinant)이 특정 임계값보다 큰 경우 인접 픽셀 8개의 행렬식과 이 값을 비교해보고

이 값이 제일 크다면 (상하 박스 필터 3x3 크기 각 9개의 Hessian 행렬식 중 극값이면) 특징점으로 검출한다.

2-2. descriptor 추출

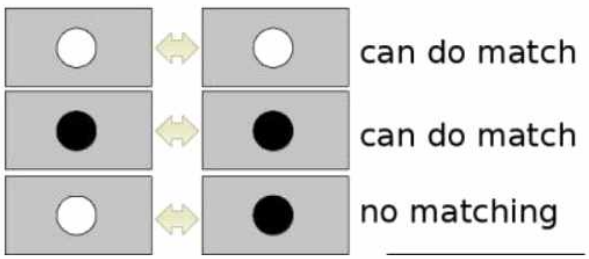
descriptor는 방향성을 갖는 벡터로 인접한 16개 픽셀 값에 대한 수평, 수직 방향의 Harr wavelet response을 사용한다. Haar wavelet은 특징점을 추출하는 방법 중 하나로, 이미지의 특정 패턴을 찾아내는 것이다. 먼저, 위 과정을 통해 검출된 특징점을 중심으로 해당 특징점이 검출된 크기 정보 s에 대한 반지름 6s 원안의 픽셀들에 대하여 수평, 수직 방향(x, y 방향)의 Haar wavelet response인 dx, dy를 아래 그림의 중간에 표현된 것처럼 구한다.



이미지를 일정 간격(-15 ~ +15도)으로 회전하면서, 60도 화면 내의 벡터 크기를 더해 가장 큰 크기를 갖는 방향을 특징점의 주요 방향(dominant orientation)으로 결정하면 특징점 주변 일정 역역 내 이웃 픽셀의 밝기 변화를 나타내는 descriptor을 생성한다. 위 그림의 가장 오른쪽 그림에 사각형 영역이 descriptor의 영역, 중심이 특징점, 중심에서 변으로 이어진 선분이 descriptor 벡터이다. 위에서 구한 descriptor 영역을 4x4 영역으로 위와 같이 분할하여 16개 영역으로 분할하면 그 안에서 중요한 공간 정보를 유지할 수 있다. 분할한 각 영역에 대해 다시 5x5 크기로 분할하여 각 픽셀을 x, y 방향의 Haar wavelet filter로 각 Haar wavelet response 값을 구한다. 여기서 구한 Haar wavelet response를 4x4 영역으로 분할된 한 영역 안에서 4차원의 벡터 v를 생성한다. 이렇게하면 이 4x4 영역의 한 영역 당 4개의 descriptor를 생성하여 한 descriptor 영역에서 총 64 길이의 descriptor을 생성하는 것이 된다.

2-3. 특징점 매칭

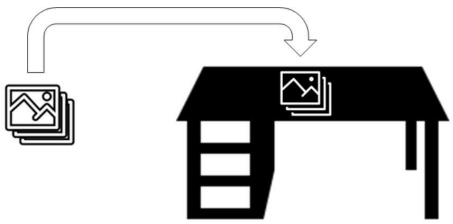
마지막 매칭 단계에서는 descriptor 영역의 라플라시안 부호(헤시안 행렬 H의 trace = Dxx+Dyy)를 사용한다. 따라서, descriptor 간의 거리를 이용하여 다음 그림과 같이 같은 유형의 contrast를 갖는다면, descriptor간 유클리디안 거리를 계산하여 최근접 이웃 탐색을 통해 매칭여부를 판단한다.



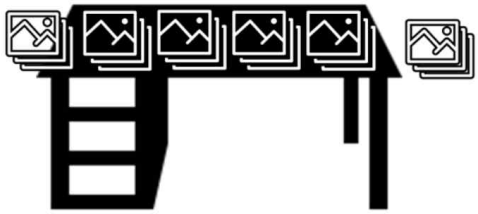
[특징점 매칭]

III. 적용한 이미지 스티칭 기법

이번 프로젝트에서 이미치 스티칭을 사용하기 위해 파이썬 open cv.Stitcher_create를 사용했다. 과정은 크게 두 가지인데 Image load와 Image stitching으로 이루어져있다. Image load에서는 'Image resize' 과정이 포함되고, Image stitching에는 '이미지1과 이미지2의 겹치는 부분(matching point) 찾기', '이미지1을 기준으로 이미지2의 겹치는 부분을 2D 변형하여 붙이기' 그리고 모든 이미지에 대해서 위 동작을 반복한다.



우선 이미지를 로드한다. 여기서 이미지의 용량이 크면 오래 걸리거나 한 작업대에서 처리하지 못 할 수도 있다. 그럴 경우, 아래 그림과 같이 이미지 리사이즈가 일어나는데 이미지 크기를 줄여서 빠르고 많은 양을 한 번에 처리하도록 한다.



다음 이미지스티칭 과정에서 공통부분 찾기를 먼저 진행한다. 두 이미지에서 공통부분을 찾았다면 2D 변형 후 연결한다. 같은 이미지라도 위치나 각도에 따라 다르게 보이기 때문에 2D 변형은 꼭 필요한 과정이다. 전체적인 과정을 조금 더 자세히 설명하겠다. 앞서 언급한 Stitcher_create를 사용하기 위해 우선 이미지 스티칭 객체를 생성한다. 우리 팀은 Stitcher 클래스에서 SURF 알고리즘을 사용했다.

cv2.Stitcher_create(, mode=None) -> retval

- mode: 스티칭 모드. cv2.PANORAMA 또는 cv2.SCANS 중 하나 선택. 기본값은 cv2.PANORAMA.
- retval: cv2.Stitcher 클래스 객체

일반적인 사진을 합성하기 때문에 mode의 인자에 cv2.PANORAMA를 입력해주었다.

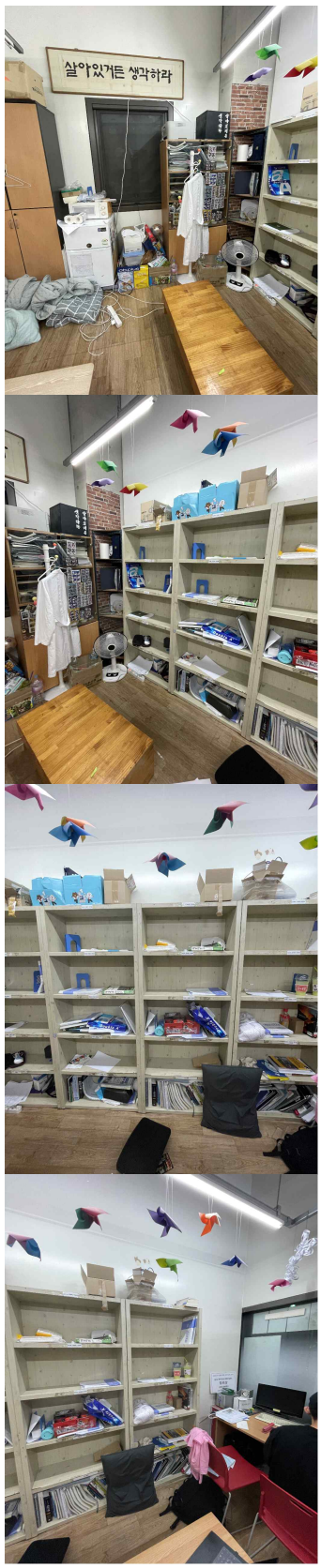
cv2.Stitcher.stitch(images, pano=None) -> retval, pano

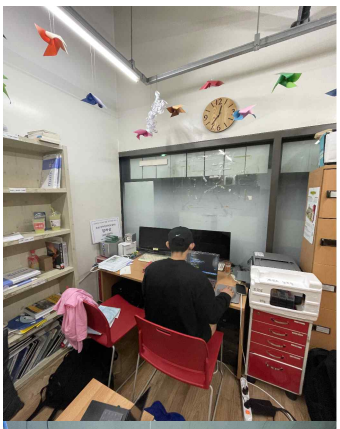
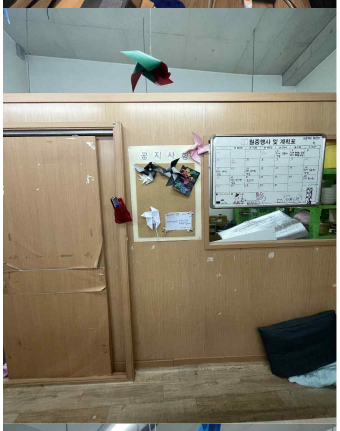
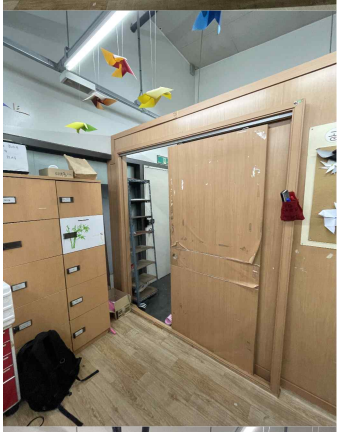
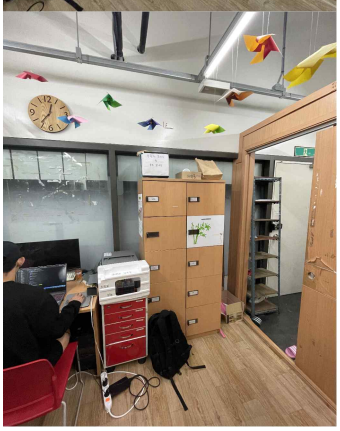
- images: 입력 영상 리스트
- retval: 성공하면 cv2.Stitcher_OK.
- pano: 파노라마 영상

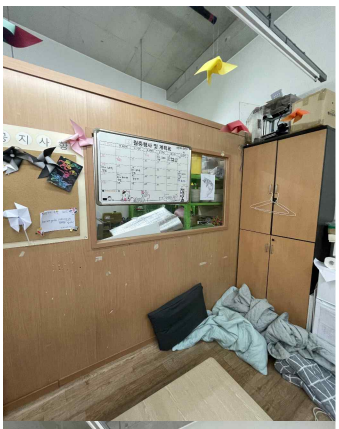
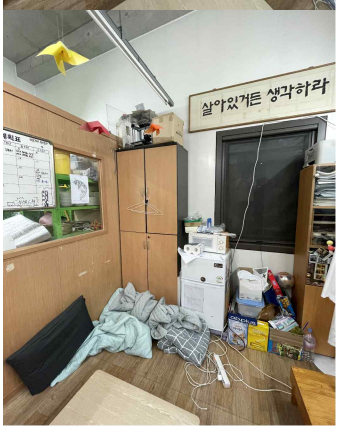
이후 이미지 스티칭 함수를 이용하여 파노라마 영상을 성공적으로 출력할 수 있었다.

IV. 결론

이미지스티칭 기법으로 여러 가지 방법들이 오픈소스화 되어있다. 대표적인 두 알고리즘은 SURF와 SIFT의 장단점을 비교하였다. 결정적으로 SIFT는 크기 변화에 따른 특징 검출 문제를 해결하기 위해 이미지 피라미드를 사용하므로 속도가 느리다는 단점이 존재한다. 바면에 SURF는 이미지 피라미드 대신 필터의 크기를 변화시키는 방식으로 성능을 개선한 알고리즘이기 때문에 우리팀은 SURF 알고리즘을 선택하였다. 그리고 이를 적용하기 위해 파이썬이 제공하는 OPEN CV인 Stitcher를 사용하였다. 아래 예 나오는 여러장의 여미지를 사용자가 등록하면 이미지들을 스티칭하여 하나의 파노라마 영상으로 성공적으로 보여줌을 확인할 수 있다.



REFERENCES

[1] ballentain, OpenCV
SIFT 알고리즘, https://ballentain.tistory.com/47

[2] Matthew Brown and David G. Lowe, “Automatic Panoramic Image Stitching using Invariant Features”, Department of Computer Science, University of British Columbia, Vancouver, Canada., pp.2~3