Homography를 이용한 Warping 영상 합성

DLT 이용과 aliasing 해결을 중심으로

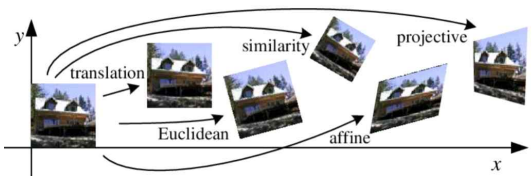
유승욱
20191181
소프트웨어학부 3학년
bokju128@cau.ac.kr

이나혁*
20194538
소프트웨어학부 3학년
nahyuk0113@cau.ac.kr

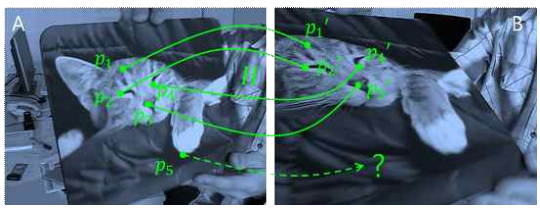
이하윤
20193418
소프트웨어학부 3학년
hayun0406@cau.ac.kr

Ⅰ. 서론

영상의 기하학적 변환이란 영상을 구성하는 픽셀이 배치된 구조를 변경함으로써 전체 영상의 모양을 바꾸는 작업을 뜻한다. 이는 어떤 픽셀의 좌표가 다른 좌표로 이동하는 경우를 의미한다.



그중에서도 Homography는 서로 다른 두 평면 간의 매핑 관계를 모델링하는 변환으로 planar surface가 다른 위치로 투영되었을 때 전후 이미지 관계를 표현한다. 이번 레포트에서는 임의의 네 대응쌍을 이용해서 구한 호모그래피 행렬 H를 다른 모든 점에 대해서도 동일하게 적용할 수 있는 성질을 이용하여 와핑 영상을 합성하는 과정을 담았다.



특히 호모그래피 행렬을 구하고자 Direct Linear Transformation를 이용하는 부분과 Backward mapping을 통해 에일리어싱 현상을 해결하는 부분을 중점적으로 작성했다.

Ⅱ. 본론

진행된 알고리즘은 다음과 같다.

(1) 두 개의 영상을 입력
(2) 각 영상마다 대응점 4개를 설정
(3) 4개의 대응관계(총 8개의 포인트)로부터 호모그래피 계산
(4) 호모그래피를 이용해 와핑 영상 합성

알고리즘에 따른 진행 과정에서 중점적인 부분들의 작동 원리를 알아보았으며 이를 구현한 코드 구현을 통해 호모그래피에 따른 와핑 영상 합성 결과물을 확인했다.

가. Homography란 무엇인가

호모그래피는 평면물체의 2D 이미지 변환 관계를 설명할 수 있는 일반적인 모델로서 perspective transformation이라고도 칭한다. 기존 이미지에서 4개의 꼭짓점이 자유롭게 변환될 수 있어야 하므로 네 쌍의 변환 전 좌표와 변환 후 좌표를 알아야 한다. Homogeneous 좌표계에서 정의되는 호모그래피 일반식은 다음과 같다.

$$w\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

이때 $(x, y, 1)$과 (wx', wy', w)는 homogeneous 좌표에서 scale이 결정될 수 없으므로 호모그래피 행렬은 8의 자유도를 가진다.

나. DLT를 이용한 Homography 계산

두 이미지 사이의 매칭 관계를 표현하는 호모그래피를 계산하고자 Direct Linear Transformation을 사용했다. DLT는 실제 3차원 좌표와 카메라 2차원 좌표 사이의 관계를 식으로 표현한 것으로 호모그래피 일반식을 다른 형태의 행렬 곱으로 표현한다. 새로운 행렬 곱 형태에서 각 행렬은 다음과 같다.

$$\mathbf{A}_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\boldsymbol{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^\top$$

따라서 호모그래피 일반식은 $A_i h = 0$로 표현되며, 이를 확장하여 4개의 좌표 쌍을 사용하면 아래와 같이 표현할 수 있다.

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y_1' & y_1y_1' & y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y_2' & y_2y_2' & y_2' \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' & x_3' \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y_3' & y_3y_3' & y_3' \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x_4' & y_4x_4' & x_4' \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y_4' & y_4y_4' & y_4' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

위 식에서 행렬 A에 속한 값들은 실제 구현에서 입력 값이 되는 좌푯값으로, 구해야 하는 미지수는 행렬 h가 된다. 따라서 이는 Homogeneous Linear Least Squares 문제가 되며 SVD를 이용해 해결할 수 있다.

다. SVD를 이용한 행렬 h 계산

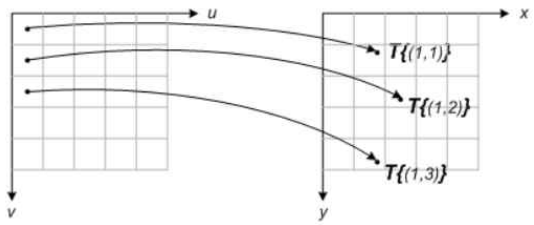
행렬 A를 Singular Value Decomposition을 통해 분해하면 다음과 같다.

$$A = U\Sigma V^T$$

이때 행렬 V에서의 각 열은 $Ah = 0$의 해를 가지므로 eigenvalue 값이 최소인 열을 선택해 호모그래피 행렬 h를 구했다.

라. Forward mapping에서 aliasing 발생

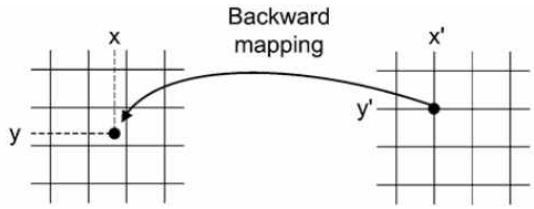
SVD를 통해 구한 호모그래피 행렬로 기존 이미지의 꼭짓점을 변환한 좌표 쌍을 구해 매핑 과정을 진행했다. Forward mapping은 기본적인 매핑 방식으로 source의 좌표에 대해 target의 좌표를 계산한 뒤 값을 채우는 방식이다.



이때 각 픽셀은 정수 단위지만 연산을 거치면 실수로 할당되어 결과적으로 target에 값이 매핑되지 못해 벌어진 공간이 생길 수 있다. 이렇게 hole이 생기는 것을 방지하고자 Backward mapping을 사용했다.

마. Backward mapping으로 aliasing 해결

Backward mapping은 target의 좌표에 대해 source의 좌표를 계산한 뒤 값을 채우는 방식이다. Forward mapping과 계산 방향이 반대인 셈이다. 실제 코드에서는 후방 변환을 거침으로써 Forward mapping에서 발생할 수 있는 hole을 미리 방지했다.

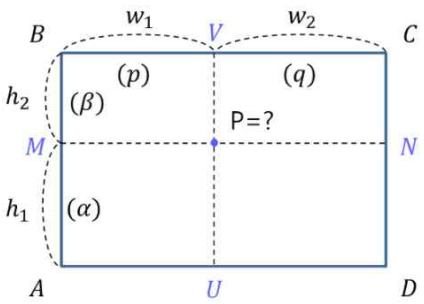


바. 보간에 의한 anti-aliasing

Forward mapping을 하면 경우에 따라 값이 매핑되지 못해 hole이 생기고, 단순 Backward mapping을 하면 이미지의 화질 저하가 발생할 수 있는데 이때 보간을 사용하면

결과 이미지를 더욱 개선할 수 있다.

실수 좌표를 반올림해 정수로 변환하는 과정에서 발생한 aliasing을 해결하고자 주위 화소 값을 이용해 보간을 진행한다.



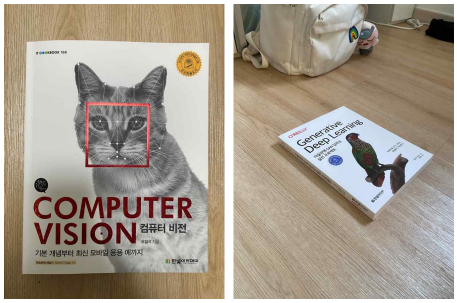
이때 주로 사용되는 Bilinear Interpolation은 원본 이미지에서 인접한 네 개의 픽셀값을 이용해 실수 좌표상의 픽셀값을 계산하는 방법이다. 위와 같이 사각형 내부 임의의 점에서의 값을 추정한다고 할 때,

$$P=q(\beta A+\alpha B)+p(\beta D+\alpha C)$$
$$=q\beta A+q\alpha B+p\beta D+p\alpha C$$

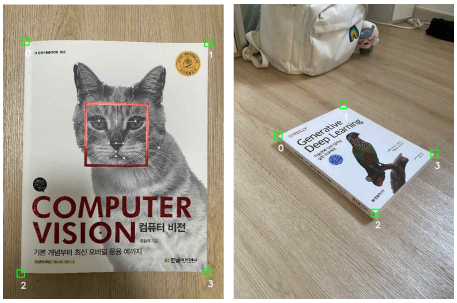
추정할 점에서 사각형까지 x축과 y축 각 방향으로의 거리를 사용하여 추정할 점의 좌표를 계산할 수 있다.

Ⅲ. 결론

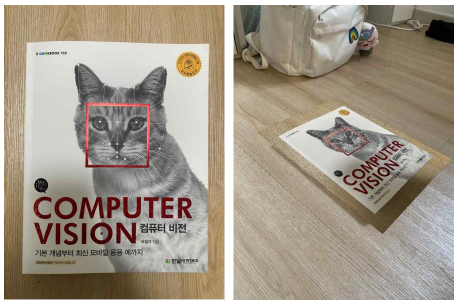
코드 구현 후 테스트에 사용된 Source(좌)와 Target(우) 이미지는 다음과 같다.



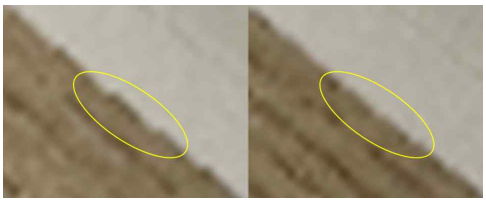
각 이미지에서 동차 좌표 내 호모그래피 행렬을 계산하기 위해 설정한 4개의 대응점 관계는 다음과 같다.



안티 에일리어싱을 수행하고자 변환 역행렬을 사용해 최종 Result(우) 이미지를 생성했다. Source 이미지에서 책의 표지가 정상적으로 와핑된 모습을 보인다.



아래는 단순 후방 변환을 적용한 이미지(좌)와 후방 변환에 양선형 보간까지 적용한 이미지(우)의 모습이다. 인접한 픽셀값을 이용해 추정할 점의 좌표를 계산한 양선형 보간을 적용으로써 매핑된 이미지를 더욱 매끄럽게 표현할 수 있었다.



양선형 보간을 적용하는 과정에서 각 점의 외부 지점이 존재하지 않는 점에는 경계 픽셀과 같은 값으로 채우는 extrapolation을 진행했다.

<Reference>

- OpenCV API(https://docs.opencv.org/3.4/d2/de8/group__core_array.html#gab477b5b7b39b370bb03e75b19d2d5109)
- Homography(https://docs.opencv.org/3.4/d2/de8/group__core_array.html#gab477b5b7b39b370bb03e75b19d2d5109)
- Homogeneoous Coordinates(https://blog.daum.net/shksjy/229)

<역할 분담>

최대한 균등하게 역할을 배분하기 위해 상호 협의하 노력하였음. 그 결과 좋은 팀워크와 함께 좋은 결과를 이룰 수 있었다고 판단함.

- 유승욱

[개발] MouseCallBack 함수 개발, 전체 코드 구조(Pipeline 기반) 구성 및 결과 테스트

[과제물] 보고서(Full Report) 작성

- 이나혁(조장)

[개발] 후방 기하 변환, 양선형 보간 파트 개발

[과제물] 보고서, README 작성 보조

- 이하윤

[개발] Homography 계산 파트 개발

[과제물] GitHub README.md 작성

<전체 코드 (GitHub Repository)>

https://github.com/NahyukLEE/Computer_Vision_Assignments/tree/main/6%EC%A3%BC%EC%B0%A8%20Assignment

: 코드 구현에 대한 간략한 소개를 포함하고 있음 (전체 링크를 직접 복사해서 웹 브라우저에 붙여넣기)