

왼쪽/오른쪽 이미지에 세 개의 점이 필요한 이유(getAffineTransform)

20221673 이승미

[1. Affine transformation 이해]

: 아핀 변환(Affine Transformation) 행렬 계산은 한 이미지(소스)를 다른 이미지(대상)의 좌표계에 일치시켜 정렬(Alignment)하고 겹치게(Mapping) 하기 위해 사용.

1.1 기존 문제점

기존의 Geometric Transformations(Rotation, Scaling, Shear)은 원점을 기준으로 이루어지며 모두 행렬 곱셈으로 표현됨. 그런데 평행 이동(Translation)은 좌표 자체에 값을 더해 주는 연산이 필요함. 그래서 여러 변환을 연속적으로 적용할 때마다 곱셈과 덧셈을 번갈아 해야 하는 문제 발생.

1.2 해결책: 차원 추가

이러한 문제를 해결하기 위해 차원을 하나 추가함.

A. 동차 좌표계로의 변환

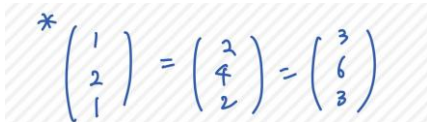
- 2차원 점 $(x, y) \rightarrow$ 3차원 동차 좌표 $(x, y, 1)$ 로 변환

B. 모든 변환의 통합 (행렬 곱셈 단일화)

차원을 늘리면, 이제 Affine 변환의 모든 요소(회전, 크기 조절, 평행 이동)를 단일 3×3 행렬 T의 곱셈으로 표현할 수 있음.

$$\begin{bmatrix} x_{new} \\ y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & b_1 \\ A_{21} & A_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

+) 동차 좌표계의 특성: 스케일 불변성

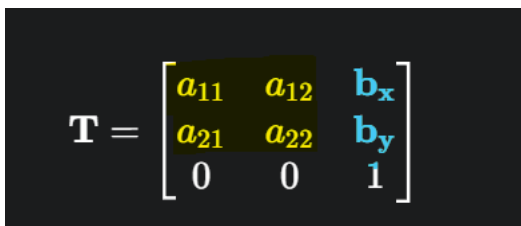

$$* \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 6 \\ 3 \end{pmatrix}$$

동차 좌표계에서는 위의 세 좌표가 같은 공간상의 점을 나타냄.

-> 최종 좌표 추출: 변환 행렬을 곱한 후 최종 결과를 얻으면, 실제 이미지 좌표로 되돌리기 위해 마지막 성분을 1로 맞춰주는 과정(정규화)이 필요함.

[2. 왼쪽/오른쪽 이미지에 세 개의 점이 필요한 이유]

2.1 수학적 필요성: 6개의 미지수 결정


$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & b_x \\ a_{21} & a_{22} & b_y \\ 0 & 0 & 1 \end{bmatrix}$$

Affine 변환을 정의하는 변환 행렬(T)에는 이미지를 변형시키는 총 6개의 미지수가 존재함. 이 미지수는 회전, 크기 조절, 기울이기(전단) 등 모양을 담당하는 4개(a)와 평행 이동을 담당하는 2개(b)임.

- 6개의 미지수 값을 유일하게 계산하기 위해서는 최소 6개의 독립적인 선형 방정식(힌트)이 반드시 필요함.
- 3쌍의 점의 역할: 이미지에서 대응점 1쌍을 찍을 때마다 x축과 y축 방향에 대한 2개의 힌트가 발생함. 따라서 3쌍의 점으로 6개의 미지수를 풀 수 있게 됨.

2.2 cv2.getAffineTransform 코드 연관성

제시된 get_mapping 함수는 OpenCV의 cv2.getAffineTransform 함수가 요구하는 입력 조건을 명시적으로 따름.

```
def get_mapping(self):
    if not (self.left_image_widget.has_image() and self.right_image_widget.has_image()):
        return None
    left = self.left_image_widget.get_clicked_points_in_image_coordinates()
    right = self.right_image_widget.get_clicked_points_in_image_coordinates()

    num_points = min(len(left), len(right))
    if num_points != 3:
        error('Please click on at exactly three corresponding points.')
        return None

    left = np.array([[x, y] for y, x in left[:num_points]], np.float32)
    right = np.array([[x, y] for y, x in right[:num_points]], np.float32)
    return cv2.getAffineTransform(right, left)
```

- 강제 조건: 코드에서 if num_points != 3: 을 통해 정확히 3쌍의 점을 요구하는 것은 아핀 변환의 수학적 필요성을 코드로 구현한 것임.
- 기능: cv2.getAffineTransform(right, left)는 오른쪽(소스) 이미지의 3점을 왼쪽(대상) 이미지의 3점으로 정렬하는 변환 규칙(행렬)을 이 6개의 힌트를 바탕으로 계산함.

+) 최적의 점 찍는 위치

변환 행렬의 안정성과 정확도 확보를 위해, 찍는 점은 같은 지점을 정확하게 가리키는 세 쌍의 대응점이어야 함. 특히, 이 세 점은 한 직선 위에 있지 않고 (선형 독립), 이미지 영역에 넓게 퍼져 삼각형을 이루도록 찍는 것이 가장 정확함.