Affine/Perspective Transformation

2020-08-11 기술보고

Affine/Perspective

- Affine : 선형 변환과 이동 변환이 포함
  선의 수평성과 평행성 유지
  평행사변형으로의 변환

- Perspective : Affine 변환의 특징 포함
  수평성과 평행성이 유지되지 않음
  원근 변환

Affine Transformation

- Translation
- Scale
- Rotation
- Euclidean : Translation + Rotation
- Similarity : Euclidean + Scale
- Shear

Affine Matrix

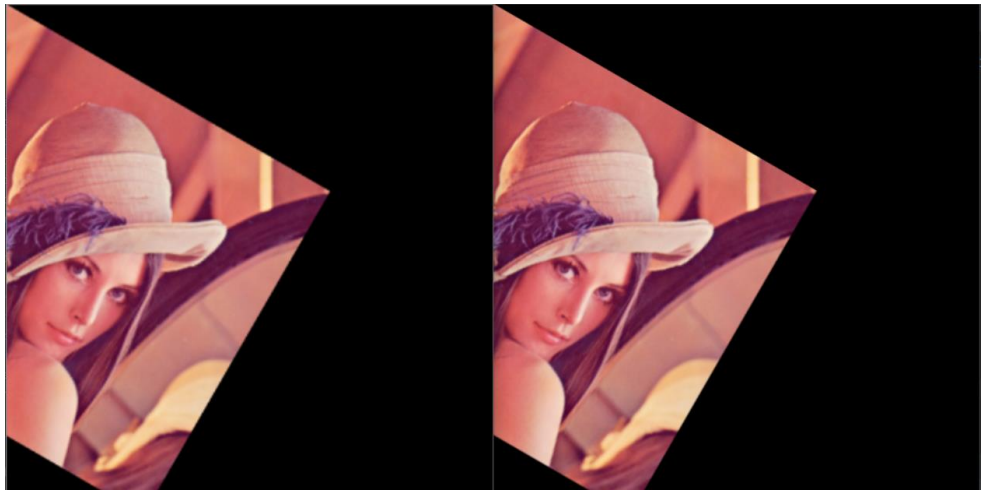
- Translation $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Scale $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Rotation $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Rotation @ Scale == Scale @ Rotation
- $(s_x\,(\cos(\theta)^* x - \sin(\theta) * y),\ s_y(\sin(\theta) * x + \cos(\theta) * y))$ ==

  $((\cos(\theta) * s_x * x - \sin(\theta) * s_y * y),\ (\sin(\theta) * s_x * x + \cos(\theta) * s_y * y))$
- Rotation @ Translation != Translation @ Rotation
- $((\cos(\theta)^* x - \sin(\theta) * y\ + dx),\ (\sin(\theta) * x + \cos(\theta) * y\ + dy))\ ! =$

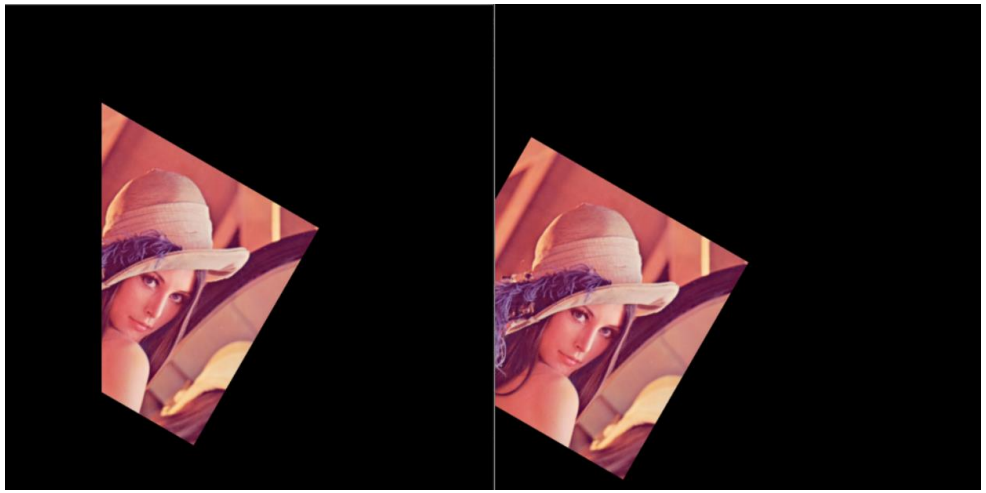
  $((\cos(\theta)^*(x + dx) - \sin(\theta) * (y + dy)),\ (\sin(\theta) * (x + dx) + \cos(\theta) * (y + dy)))$

Affine Matrix

- Rotation(30) -> Scale(1.5) Scale(1.5) -> Rotation(30)



- Rotation(30) -> Translation(100,100) Translation(100,100) -> Rotation(30)



Affine Matrix

- Euclidean $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \varepsilon\cos\theta & -\sin\theta & t_x \\ \varepsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

  ε = ±1
  ε = 1, 방향 유지
  ε = -1, 반전

- Similarities $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \varepsilon s\cos\theta & -s\sin\theta & t_x \\ \varepsilon s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

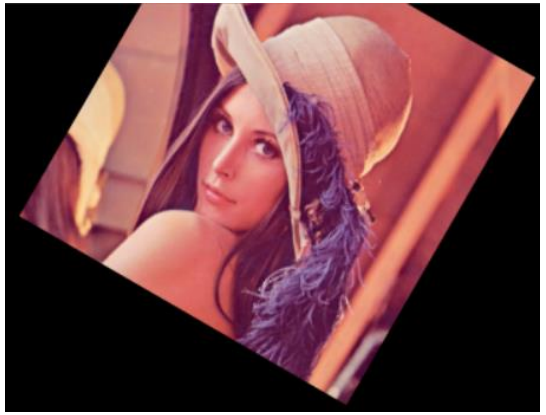
- Shear $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \lambda_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

  $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \lambda_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$(x', y') = (x + \lambda_x * y, \lambda_y * x + y)$
=> 일정 비율의 x 또는 y좌표의 값이 추가가 되므로 평행성과 평행선의 길이의 비율 유지

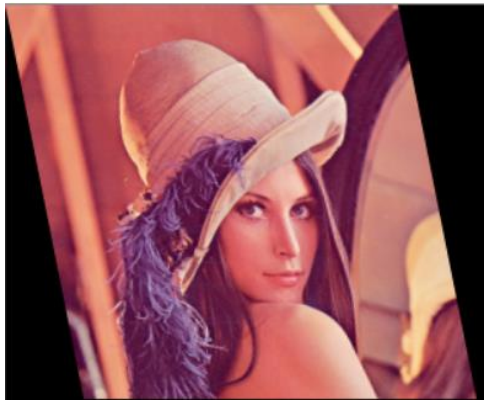
Affine Matrix

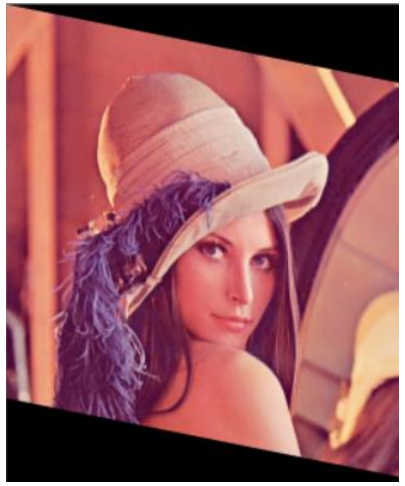
- Euclidean – R(30), T(130, 50)

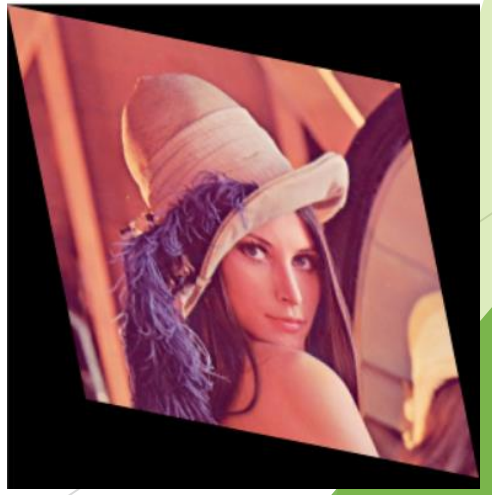


ε : -1

- Shear(0.2)







Affine Matrix

- Affinities

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos(-\Phi) & -\sin(-\Phi) \\ \sin(-\Phi) & \cos(-\Phi) \end{bmatrix} \cdots$$

$$\cdots \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos\Phi & -\sin\Phi \\ \sin\Phi & \cos\Phi \end{bmatrix}$$

$$x' = H_A x = \begin{bmatrix} A & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} x$$

$$A = R(\theta) R(-\Phi) D(\lambda_1, \lambda_2) R(\Phi)$$

Φ의 값 불명확

src : [[57, 90], [250, 45], [200, 250]]

dst : [[100, 100], [240, 90], [150, 172]]

```
[[ 6.60592255e-01 -2.77904328e-01  8.73576310e+01]
[ 4.39501541e-02  4.10719550e-01  6.05300817e+01]]
```

src : [[57, 90], [250, 45], [200, 250]]
dst : [[120, 110], [240, 150], [100, 250]]

```
[[  0.4904194   -0.56331234 142.74420474]
[  0.34034571   0.57081603  39.22685247]]
```

OpenCV – Affine Functions

- cv2.getRotationMatrix2D() : 중심점과 각도, 크기를 입력받아 matrix 반환
- cv2.getAffineTransform() : 세 개의 좌표정보 쌍을 입력받아 Affine matrix 반환
- cv2.warpAffine() : 2x3의 matrix를 이용하여 Affine Transformation 시행
  - Matrix는 float형
- cv2.invertAffineTransform() : Affine 변환의 역변환 matrix 생성
  - Affine Matrix는 non-singular matrix이므로 역행렬 존재

Perspective Transformation

- Perspective Matrix : linear, non-singular => 역행렬 존재

$$k_{p2}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} k_{p1}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \frac{k_{p1}}{k_{p2}}\begin{bmatrix} a_{11}/a_{33} & a_{12}/a_{33} & a_{13}/a_{33} \\ a_{21}/a_{33} & a_{22}/a_{33} & a_{23}/a_{33} \\ a_{31}/a_{33} & a_{32}/a_{33} & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = k\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & v \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} s\cos\theta & -\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & k & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\cdots$$

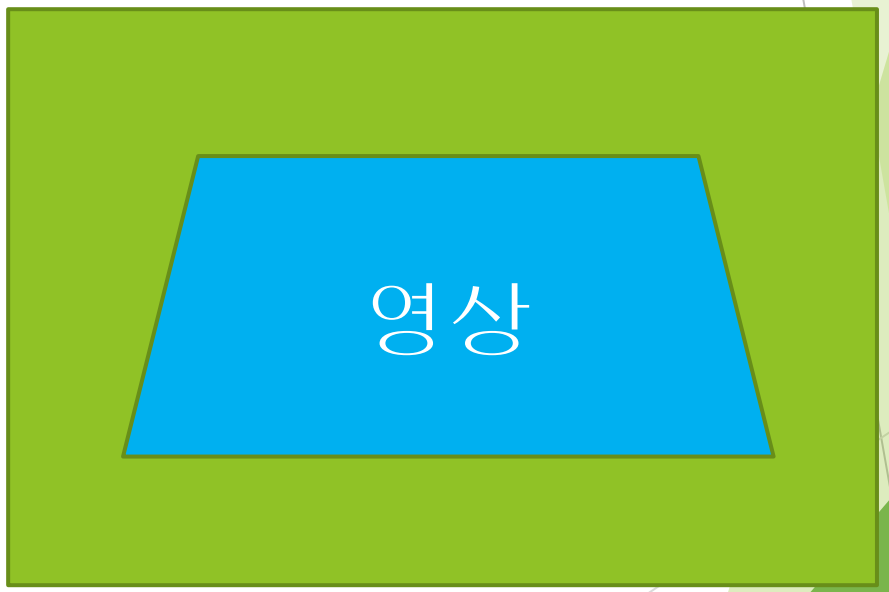
$$\cdots\begin{bmatrix} \lambda & 0 & 0 \\ 0 & 1/\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ v_1 & v_2 & v \end{bmatrix}$$

$$x' = \frac{x}{v_1 x + v_2 y + v}$$

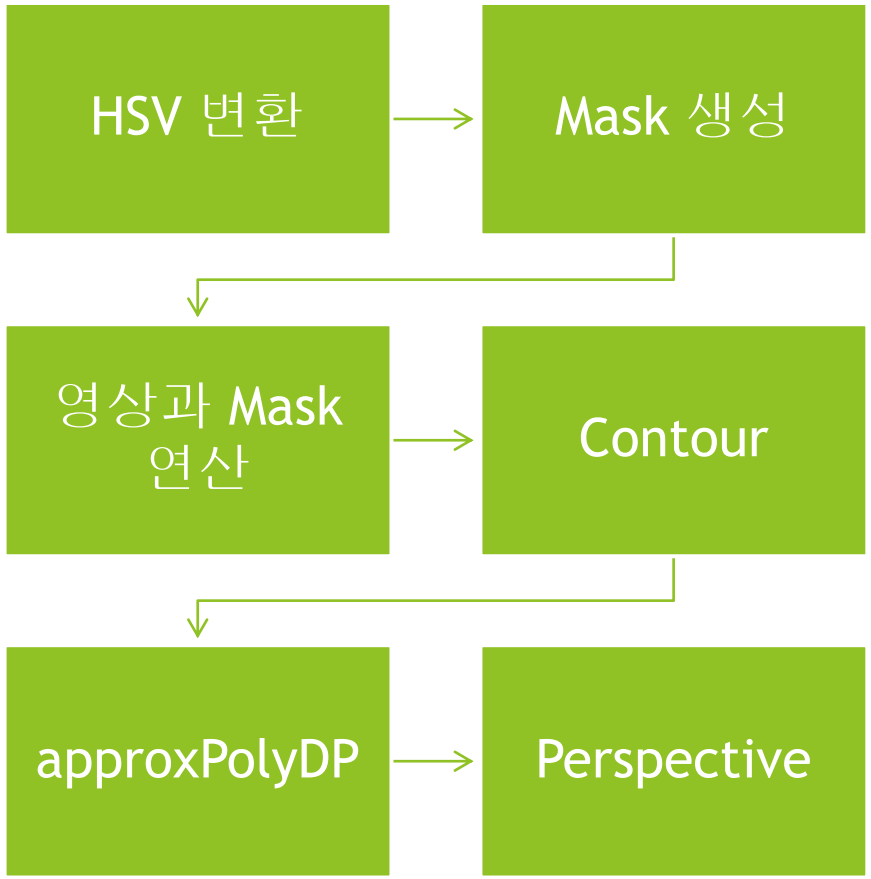
$$y' = \frac{y}{v_1 x + v_2 y + v}$$

mission

- 특정 색의 사각형 물체를 이용하여 영상을 Perspecitve 변환하여 합성하기



미션 진행 과정


HSV 변환
Mask 생성
영상과 Mask 연산
Contour
approxPolyDP
Perspective

세부과정 – Mask, Contour

- inRange() : 원하는 색상의 범위 mask 생성
- Dilate() : background를 팽창시켜 흐릿한 경계부분을 이미지로 만든다
- 이미지를 합치기 위한 역변환된 mask_inv 생성
- Bitwise_and 연산으로 만들어진 물체만의 이미지를 이용하여 Contour 생성

세부과정 - approxPolyDP

- 만들어진 contou로 approxPolyDP 연산
- Perspective 연산을 위해서는 네 점이 필요하므로 approxPolyDP가 4개의 점을 가지도록 계수 조절
- 4개의 점을 가지면 해당 4개의 점으로 Perspective matrix를 구한 후 변환
  - 4개의 점이 순서대로 만들어지지 않음 => 4개의 점을 시계방향으로 정렬
  - 합성될 영상과의 크기가 맞지 않음 => approxPolyDP의 크기로 resize
  - Perspective된 영상과의 위치가 맞지 않음 => approxPolyDP의 첫 번째 점의 위치로 이동

세부과정 - 영상 합성

- Mask를 이용하여 원본 영상에 mask를 적용한다.
- Mask_inv를 이용하여 합성될 영상에 mask를 적용한다.
- 사용될 영상을 제외한 화소값들은 0이므로 + 연산을 사용한다.