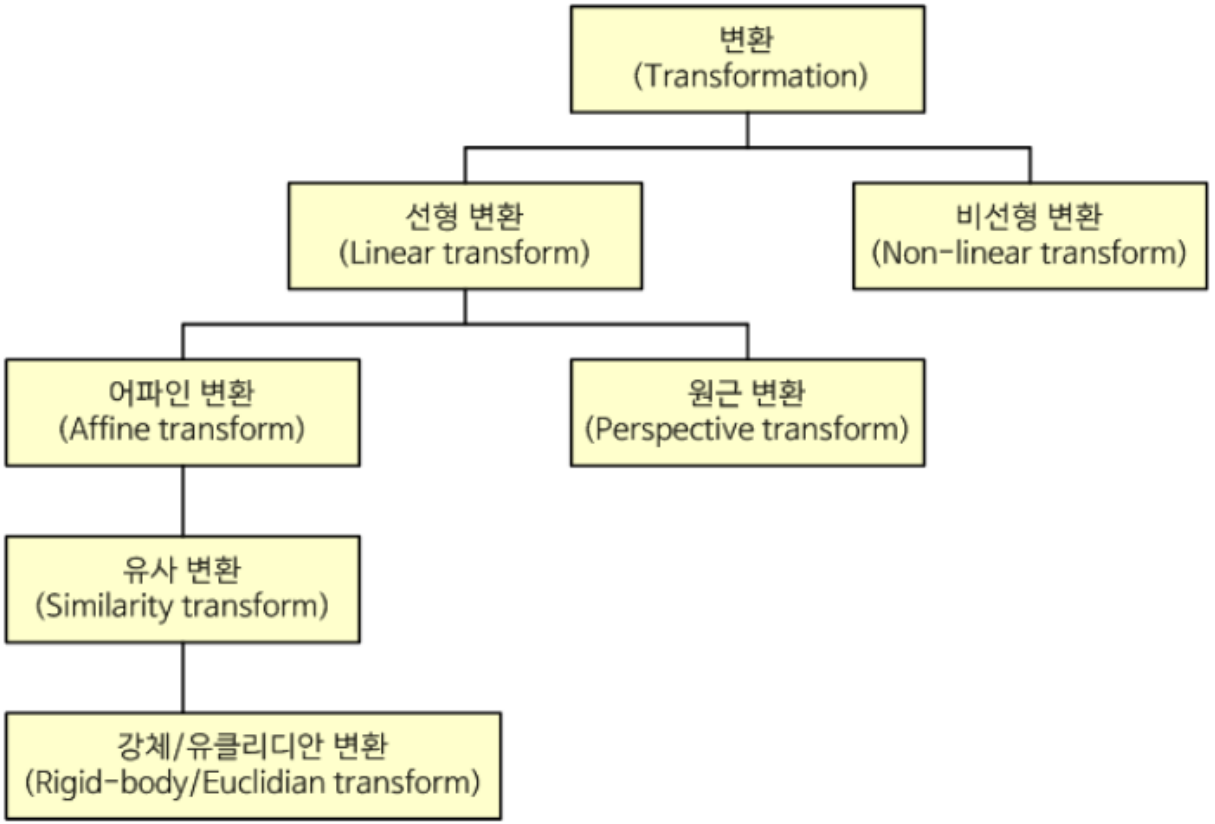
기하학적 변환(2)

3.4 어파인 변환(Affine)

- 어파인 변환 (Affine) 변환
  - 평행선이 유지되는 영상 변환
  - 어파인 변환에서는 이동, 확대, 크기 변환, 반전이 포함
  - OpenCV에서는 cv2.warpAffine() 함수를 사용하여 어파인 변환



3.4 어파인 변환(Affine)

- 다양한 어파인 변환을 수행하는 코드
  - 어파인 변환에 의한 영상 변환을 쉽게 확인하기 위하여 RGB 색상으로 각 변환 기준 점의 위치를 표시

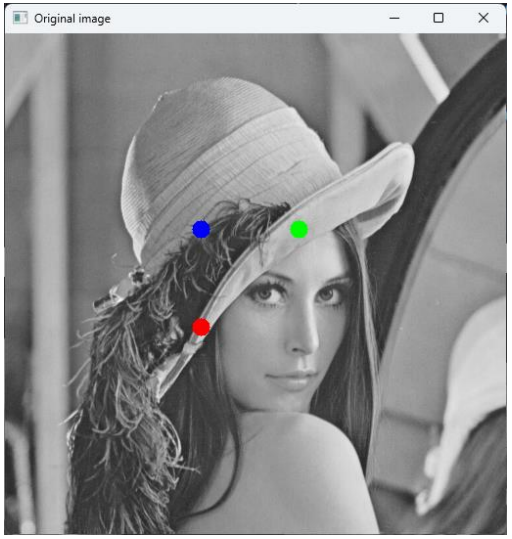
```
import numpy as np
import cv2


ori_img = cv2.imread("./images/Lena_g.jpg", cv2.IMREAD_UNCHANGED)
c_image = cv2.cvtColor(ori_img, cv2.COLOR_GRAY2BGR)
cv2.imshow('Original image', c_image)

rows, cols = ori_img.shape[:2] # channel 여부 무시

# pts1 좌표 표시
pts1 = np.float32([[200, 200], [300, 200], [200, 300]])

cv2.circle(c_image, (200, 200), 9, (255, 0, 0), -1)
cv2.circle(c_image, (300, 200), 9, (0, 255, 0), -1)
cv2.circle(c_image, (200, 300), 9, (0, 0, 255), -1)

pts2 = np.float32([[200, 200], [350, 200], [200, 250]])
Mat1 = cv2.getAffineTransform(pts1, pts2)
r_image1 = cv2.warpAffine(c_image, Mat1, (cols, rows))
```

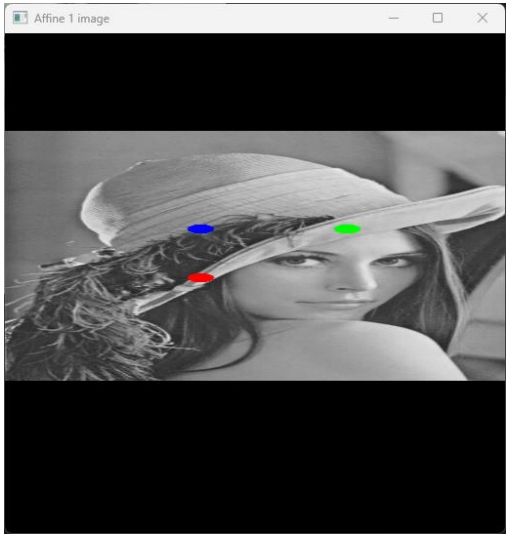
```
pts2 = np.float32([[200, 200], [300, 230], [260, 300]])
Mat2 = cv2.getAffineTransform(pts1, pts2)
r_image2 = cv2.warpAffine(c_image, Mat2, (cols, rows))

pts2 = np.float32([[200, 200], [100, 170], [170, 100]])
Mat3 = cv2.getAffineTransform(pts1, pts2)
r_image3 = cv2.warpAffine(c_image, Mat3, (cols, rows))

cv2.imshow('Affine 1 image', r_image1)
cv2.imshow('Affine 2 image', r_image2)
cv2.imshow('Affine 3 image', r_image3)

cv2.waitKey(0)
```

3.4 어파인 변환(Affine)

- 실행 결과

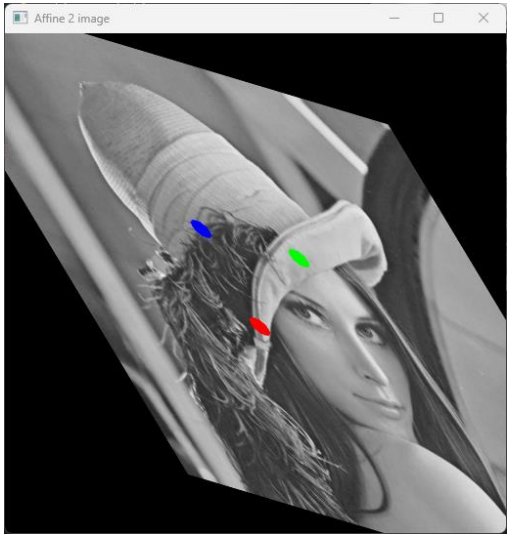
original

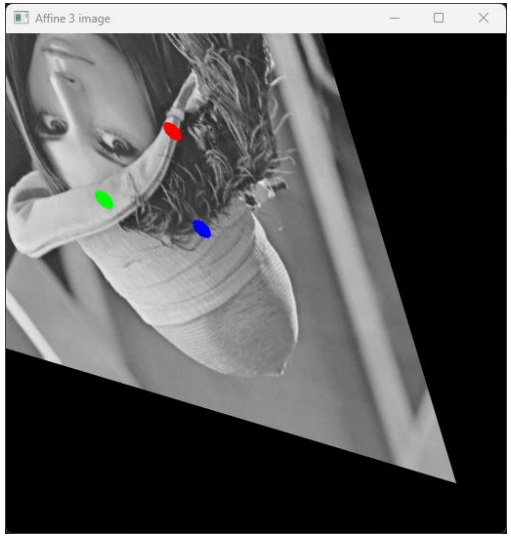




x, y축 크기 변환

어파인 변환

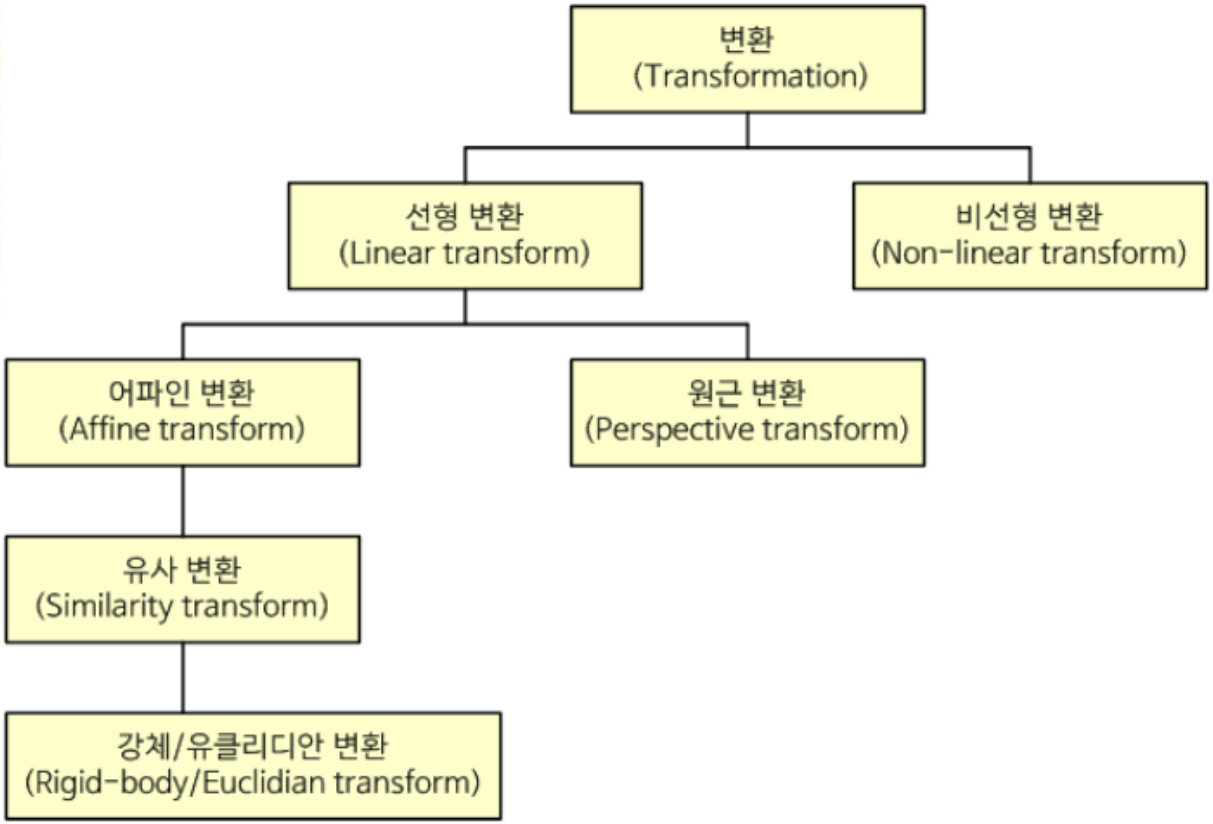




어파인 변환
(반전효과)

3.5 원근 변환(Perspective)

- 원근 변환
  - 직선의 성질만 유지되고 평행선이 유지되지 않는 영상 변환
  - OpenCV에서는 cv2.warpPerspective() 함수를 사용하여 원근 변환을 변환



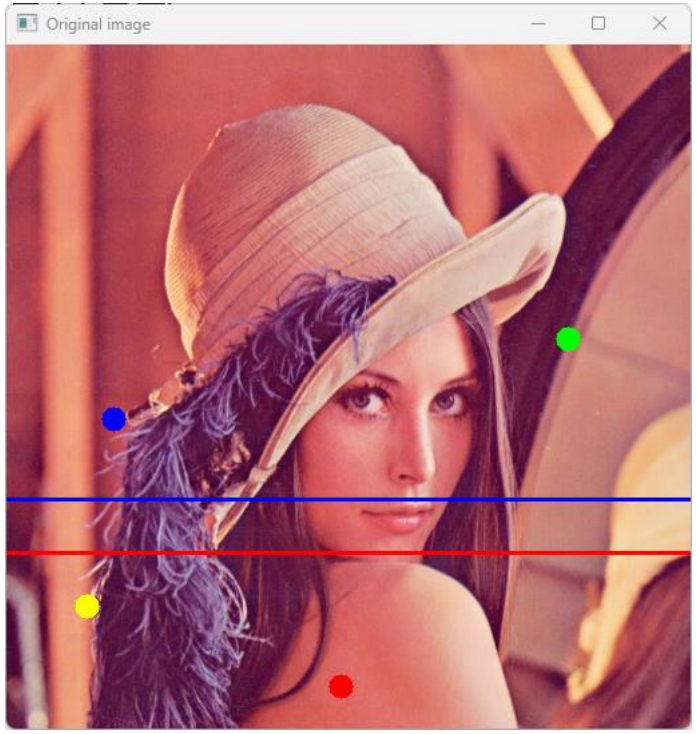
3.5 원근 변환(Perspective)

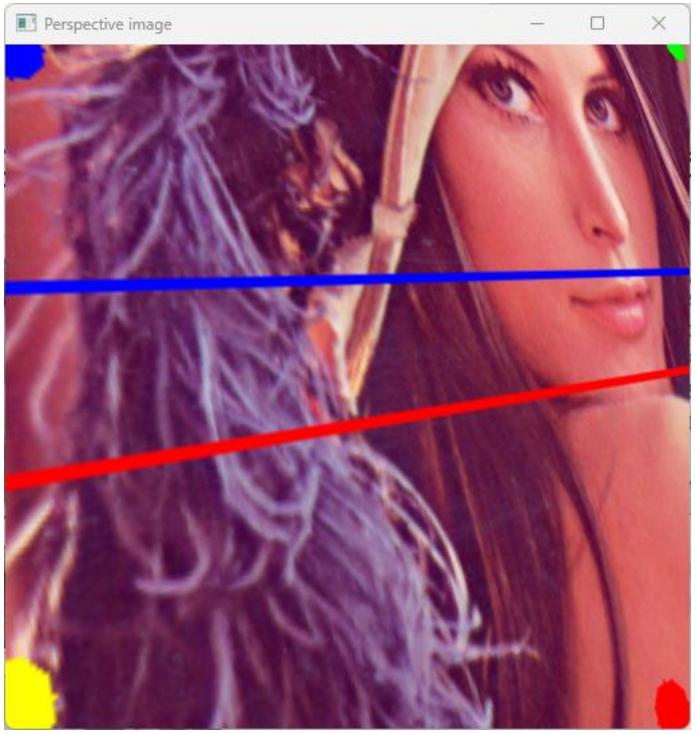
- 원근 변환을 수행

```
import numpy as np
import cv2
if __name__ == '__main__':
    ori_img = cv2.imread("./images/Lenna.jpg", cv2.IMREAD_COLOR)
    rows, cols = ori_img.shape[:2]  # channel 여부 무시
    # pts1 좌표 표시
    pts1 = np.float32([[80, 280], [220, 220], [250, 480], [60, 420]])
    cv2.circle(ori_img, (80, 280), 9, (255, 0, 0), -1)
    cv2.circle(ori_img, (220, 220), 9, (0, 255, 0), -1)
    cv2.circle(ori_img, (250, 480), 9, (0, 0, 255), -1)
    cv2.circle(ori_img, (60, 420), 9, (0, 255, 255), -1)
    cv2.line(ori_img, (0, 340), (511, 340), (255, 0, 0), 2)
    cv2.line(ori_img, (0, 380), (511, 380), (0, 0, 255), 2)
    pts2 = np.float32([[10, 10], [502, 10], [502, 502], [10, 502]])
    Mat1 = cv2.getPerspectiveTransform(pts1, pts2)
    print('Perspective matrix')
    print(Mat1)
    r_image = cv2.warpPerspective(ori_img, Mat1, (cols, rows))
    cv2.imshow('Original image', ori_img)
    cv2.imshow('Perspective image', r_image)
    cv2.waitKey(0)
```

3.5 원근 변환(Perspective)

- 실행결과
  - 평생선이 유지되지 않는 특성을 확인하기 위해 파란색 및 빨간색의 평행선 표시
  - 원근 변환으로 선의 왜곡되는 현상을 확인할 수 있음





3.5 원근 변환(Perspective)

- 영상 원근(Perspective) 변환 함수: cv2.warpPerspective() 함수
  - 입력된 영상 데이터에 대하여 지정한 원근 변환 행렬에 따른 변환을 수행

$$dst(x,y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right)$$

함수명	cv2.warpPerspective(src, M, dsize, dst, flags, borderMode, borderValue)
매개변수	- src (numpy.ndarray) 변환시킬 입력영상 데이터 - M (3x3) 변환 행렬 - dsize 출력 영상 데이터 크기 - dst 출력 영상 데이터 (dsize 크기로 조정됨) - flags 보간 방법과 역변환 방법 설정 - borderMode 픽셀 외삽(extracpolation) 방법 - borderValue BORDER_CONSTANT 경우 사용할 상수값 (기본값은 0)
리턴값	출력 영상 데이터 (numpy.ndarray)