CHAPTER 08
기하학 처리

8.6 행렬 연산을 통한 기하학 변환 - 어파인 변환
8.7 원근 투시(투영) 변환

29

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

◆기하학 변환 수식이 행렬의 곱으로 표현 가능

❖ 회전

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

❖ 크기변경

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

❖ 평행이동

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

❖ 어파인 변환 수식

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

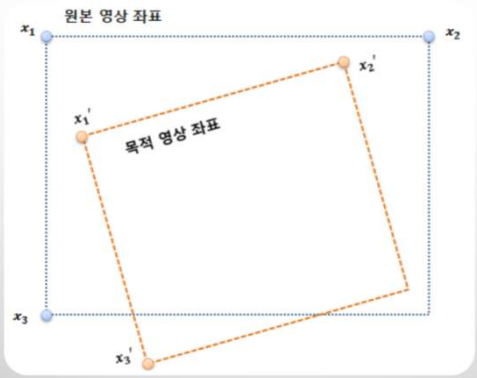
❖ 각 변환 행렬을 행렬 곱으로 구성가능 → 최종 행렬곱 후에 마지막 행 삭제

$$\text{어파인 변환행렬} = \begin{bmatrix} \cos\theta & -sin'\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

30

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

❖ 입력영상의 좌표 3개(x_1, x_2, x_3)와 변환이 완료된 목적영상에서 상응하는 좌표 3개(x_1, x_2, x_3)를 알면 → **어파인 행렬 구성 가능**



31

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

❖ OpenCV에서도 어파인 변환 수행 함수 제공

함수 및 인수 구조		
cv2.warpAffine (src, M, dsize [, dst [, flags [, borderMode [, borderValue]]]]) → dst ■ 설명: 입력 영상에 어파인 변환을 수행해서 반환한다.		
인수 설명	■ src	입력 영상
	■ dst	반환 영상
	■ M	어파인 변환 행렬
	■ dsize	반환 영상의 크기
	■ flags	보간 방법
	■ borderMode	경계지정 방법
cv2.getAffineTransform(src, dst) → retval ■ 설명: 3개의 좌표쌍을 입력하면 어파인 변환 행렬을 반환한다.		
인수 설명	■ src	입력 영상 좌표 3개 (행렬로 구성)
	■ dst	목적 영상 좌표 3개 (행렬로 구성)
cv2.getRotationMatrix2D(center, angle, scale) → retval ■ 설명: 회전 변환과 크기 변경을 수행할 수 있는 어파인 행렬을 반환한다.		
인수 설명	■ center	회전의 중심점
	■ angle	회전각도, 양수 각도가 반시계 방향 회전 수행
	■ scale	변경할 크기
cv2.invertAffineTransform(M [, iM]) → iM ■ 설명: 어파인 변환 행렬의 역 행렬을 반환한다.		
인수 설명	■ M	어파인 변환 행렬
	■ iM	어파인 역변환 행렬

32

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

예제 8.6.1 어파인 변환 – 07.affine_transform.py

```
01  import numpy as np, cv2
02  from Common.functions import contain                    # 함수 임포트
03  from Common.interpolation import bilinear_value         # 화소 선형 변환 함수 임포트
04
05  def affine_transform(img, mat):                         # 어파인 변환 수행 함수
06      rows, cols = img.shape[:2]
07      inv_mat = cv2.invertAffineTransform(mat)            # 어파인 변환의 역행렬
08      ## 리스트 생성 방식
09      pts = [np.dot(inv_mat, (j, i, 1)) for i in range(rows) for j in range(cols)]
10      dst = [bilinear_value(img, p) if contain(p, size) else 0 for p in pts]
11      dst = np.reshape(dst, (rows, cols)).astype('uint8') # 1차원 → 2차원
12
13      ## 반복문 방식
14      # dst = np.zeros(img.shape, img.dtype)              # 목적 영상 생성
15      # for i in range(rows):                             # 목적 영상 순회– 역방향 사상
16      #     for j in range(cols):
17      #         pt = np.dot(inv_mat, (j, i, 1))           # 행렬 내적 계산
18      #         if contain(pt, size): dst[i, j] = bilinear_value(img, pt)      # 양선형 보간
19
20      return dst
21
22  image = cv2.imread("images/affine.jpg", cv2.IMREAD_GRAYSCALE)
23  if image is None: raise Exception("영상파일 읽기 에러")
```

역방향 사상을 위해 어파인 변환의 역행렬 계산

행렬 곱을 위해 3차원 좌표 선언

33

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

```
25  center = (200, 200)                                     # 회전 변환 기준 좌표
26  angle, scale = 30, 1                                    # 30도 회전. 크기 변경 안 함
27  size = image.shape[::-1]                                # 크기는 행태의 역순
28
29  pt1 = np.array([( 30, 70),(20, 240), (300, 110)], np.float32)
30  pt2 = np.array([(120, 20),(10, 180), (280, 260)], np.float32)
31  aff_mat = cv2.getAffineTransform(pt1, pt2)              # 3개 좌표쌍 어파인 행렬 생성
32  rot_mat = cv2.getRotationMatrix2D(center, angle, scale)         # 어파인 행렬
33
34  dst1 = affine_transform(image, aff_mat)                 # 어파인 변환 수행
35  dst2 = affine_transform(image, rot_mat)                 # 회전 변환 수행
36  dst3 = cv2.warpAffine(image, aff_mat, size, cv2.INTER_LINEAR)
37  dst4 = cv2.warpAffine(image, rot_mat, size, cv2.INTER_LINEAR)
38
39  image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
40  dst1 = cv2.cvtColor(dst1, cv2.COLOR_GRAY2BGR )
41  dst3 = cv2.cvtColor(dst3, cv2.COLOR_GRAY2BGR )
42
43  for i in range(len(pt1)):                               # 3개 좌표 표시
44      cv2.circle(image, tuple(pt1[i].astype(int)), 3, (0, 0, 255), 2)
45      cv2.circle(dst1 , tuple(pt2[i].astype(int)), 3, (0, 0, 255), 2)
46      cv2.circle(dst3 , tuple(pt2[i].astype(int)), 3, (0, 0, 255), 2)
47
48  cv2.imshow("image", image)
49  cv2.imshow("dst1_affine", dst1);          cv2.imshow("dst2_affine_rotate", dst2)
50  cv2.imshow("dst3_OpenCV_affine", dst3);   cv2.imshow("dst4_OpenCV_affine_rotate", dst4)
51  cv2.waitKey(0)
```

입력영상 3개 좌표

목적영상 3개 좌표

사용자 정의 함수로 어파인 변환 수행

OpenCV 함수로 어파인 변환 수행

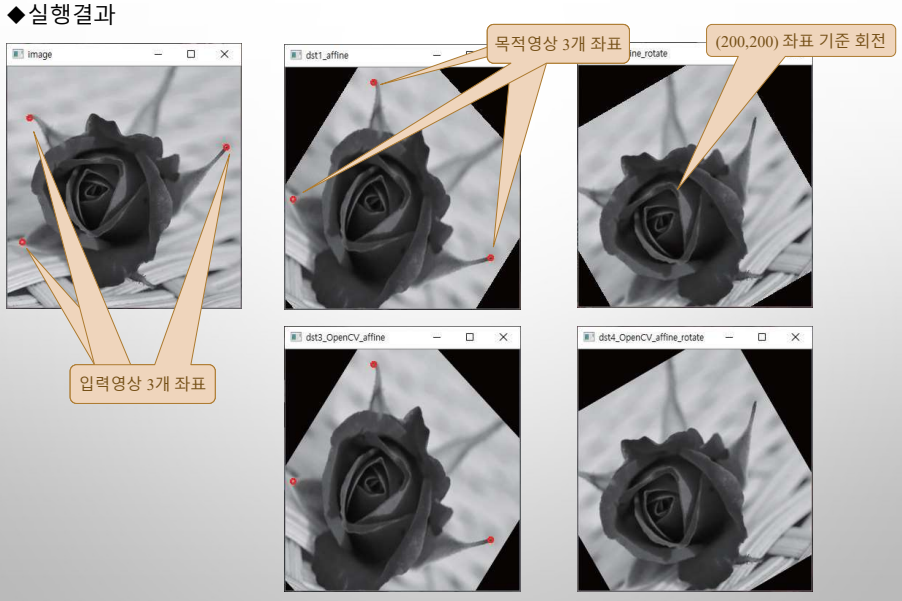
빨간점 표시 위해 명암도 영상을 BGR 컬러 변환

입력영상에 3개 좌표와 목적영상에 3개 좌표 표시

34

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

◆실행결과



35

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

심화예제 8.6.2 어파인 변환의 연결 – 07.affine_combination.py

```
01  import numpy as np, math, cv2
02  from Common.interpolation import affine_transform    # 저자 구현 어파인변환 함수 임포트
03
04  def getAffineMat(center, degree, fx=1, fy=1, translate=(0,0)):   # 변환 행렬 합성 함수
05      scale_mat = np.eye(3, dtype=np.float32)          # 크기 변경 행렬
06      cen_trans = np.eye(3, dtype=np.float32)          # 중점 평행 이동
07      org_trans = np.eye(3, dtype=np.float32)          # 원점 평행 이동
08      trans_mat = np.eye(3, dtype=np.float32)          # 좌표 평행 이동
09      rot_mat   = np.eye(3, dtype=np.float32)          # 회전 변환 행렬
10
11      radian = degree / 180 * np.pi                    # 회전 각도– 라디안 계산
12      rot_mat[0] = [ np.cos(radian), np.sin(radian), 0]  # 회전행렬 0행
13      rot_mat[1] = [-np.sin(radian), np.cos(radian), 0]  # 회전행렬 1행
14
15      cen_trans[:2, 2] = center                        # 중심 좌표 이동
16      org_trans[:2, 2] = -center[0], -center[1]        # 원점으로 이동
17      trans_mat[:2, 2] = translate                     # 평행 이동 행렬의 원소 지정
18      scale_mat[0, 0], scale_mat[1, 1] = fx, fy        # 크기 변경 행렬의 원소 지정
19
20      ret_mat = cen_trans.dot(rot_mat.dot(trans_mat.dot(scale_mat.dot(org_trans))))
21      # ret_mat = cen_trans.dot(rot_mat.dot(scale_mat.dot(trans_mat.dot(org_trans))))
22      return np.delete(ret_mat, 2, axis=0)             # 마지막행 제거 ret_mat[0:2,]
23
```

np.eye(n, dtype)
$n \times n$ 단위행렬 생성
n=3인 경우, $\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$

rot_mat = $\begin{bmatrix}cos&sin&0\\-sin&cos&0\\0&0&1\end{bmatrix}$

cen_trans = $\begin{bmatrix}1&0&center\\0&1&center\\0&0&1\end{bmatrix}$

org_trans = $\begin{bmatrix}1&0&-center_0\\0&1&-center_1\\0&0&1\end{bmatrix}$

trans_mat = $\begin{bmatrix}1&0&trans\\0&1&trans\\0&0&1\end{bmatrix}$

scale_mat = $\begin{bmatrix}fx&0&0\\0&fy&0\\0&0&1\end{bmatrix}$

변환 순서에 따라 각 변환 행렬을 곱하여 최종 어파인 변환 행렬 계산

36

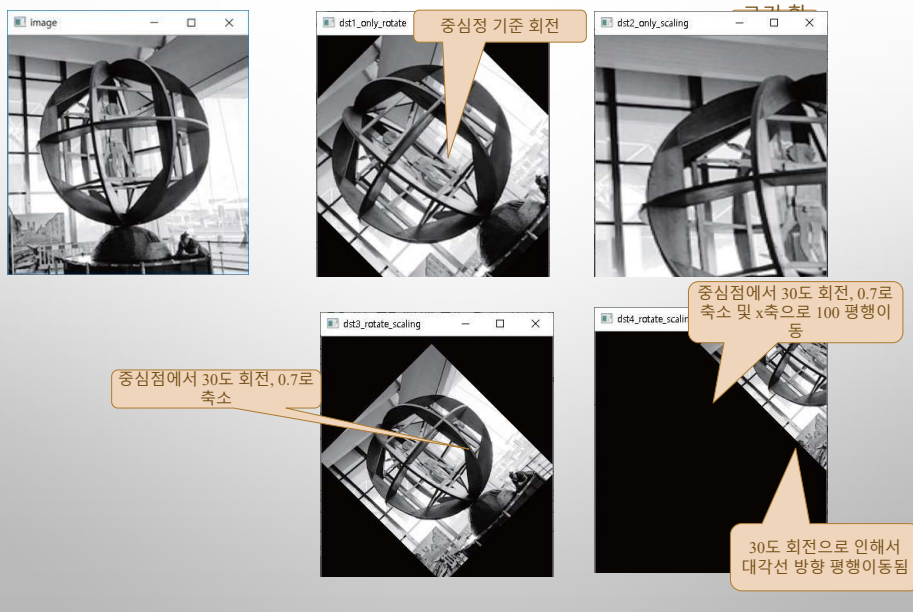
8.6 행렬 연산을 통한 기하학 변환–어파인 변환

```
24  image = cv2.imread("images/affine2.jpg", cv2.IMREAD_GRAYSCALE)
25  if image is None: raise Exception("영상파일 읽기 에러")
26
27  size = image.shape[::-1]
28  center = np.divmod(size, 2)[0]                    # 회전 중심 좌표
29  angle, tr = 45, (200, 0)                          # 각도와 평행이동 값 지정
30
31  aff_mat1 = getAffineMat(center, angle)            # 중심 좌표 기준 회전
32  aff_mat2 = getAffineMat((0,0), 0, 2.0, 1.5)       # 크기 변경– 확대
33  aff_mat3 = getAffineMat(center, angle, 0.7, 0.7)  # 회전 및 축소
34  aff_mat4 = getAffineMat(center, angle, 0.7, 0.7, tr)  # 복합 변환
35
36  dst1 = cv2.warpAffine(image, aff_mat1, size)      # OpenCV 함수
37  dst2 = cv2.warpAffine(image, aff mat2, size)
38  dst3 = affine_transform(image, aff_mat3)      # 사용자 정의 함수
39  dst4 = affine_transform(image, aff_mat4)
40
41  cv2.imshow("image", image)
42  cv2.imshow("dst1_only_rotate", dst1)
43  cv2.imshow("dst2_only_scaling", dst2)
44  cv2.imshow("dst3_rotate_scaling", dst3)
45  cv2.imshow("dst4_rotate_scaling_translate", dst4)
46  cv2.waitKey(0)
```

37

8.6 행렬 연산을 통한 기하학 변환–어파인 변환

◆실행결과



38

심화예제

심화예제 8.6.3 마우스 드래그로 영상 회전하기 – 09.affine_event.py

```
01  import numpy as np, cv2
02
03  def contain_pts(p, p1, p2):                          # p가 2개 좌표 범위 내 검사
04      return p1[0] <= p[0] < p2[0] and p1[1] <= p[1] < p2[1]
05
06  def draw_rect(title, img, pts):                      # 4개 사각형 표시 함수
07      rois = [(p - small, small * 2) for p in pts]     # 좌표 사각형 관심 영역들
08      for (x, y), (w, h) in np.int32(rois):
09          cv2.rectangle(img, (x, y, w, h), (0, 255, 0), 2)        # 사각형 그리기
10      cv2.imshow(title, img)
11
12  def affine(img):                                     # 어파인 변환 수행 함수
13      aff_mat = cv2.getAffineTransform(pts1, pts2)
14      dst = cv2.warpAffine(img, aff_mat, image.shape[1::-1], cv2.INTER_LINEAR)
15      draw_rect('image', np.copy(image), pts1)         # 입력 영상에 좌표 표시
16      draw_rect('dst', dst, pts2)                      # 목적 영상에 좌표 표시
17
```

39

심화예제

```
18  def onMouse(event, x, y, flags, param):              # 마우스 이벤트 처리 함수
19      global check
20      if event == cv2.EVENT_LBUTTONDOWN:               # 좌버튼 다운
21          for i, p in enumerate(pts1):
22              p1, p2 = p - small, p + small            # small 2개 크기 좌표
23              if contain_pts((x, y), p1, p2): check = i        # 범위 내인지 확인
24
25      if event == cv2.EVENT_LBUTTONUP: check = -1      # 우버튼 업
26
27      if check >= 0:                                   # 좌표 사각형이 클릭 되면
28          pts1[check] = (x, y)                         # 해당 위치 저장
29          affine(np.copy(image))                       # 어파인 변환 수행
30
31  image = cv2.imread("images/affine1.jpg", 1)          # 컬러 영상 읽기
32  if image is None: raise Exception("영상파일 읽기 에러")
33
34  small = np.array([12, 12])                           # 좌표 사각형 크기
35  check = -1                                           # 선택된 좌표 사각형
36  pts1 = np.float32([(30, 30), (450, 30), (200, 370)]) # 입력 영상 3개 좌표
37  pts2 = np.float32([(30, 30), (450, 30), (200, 370)]) # 목적 영상 3개 좌표
38
39  draw_rect('image', np.copy(image), pts1)             # 입력 영상에 좌표 사각형 그리기
40  draw_rect('dst', np.copy(image), pts2)               # 결과 영상에 좌표 사각형 그리기
41  cv2.setMouseCallback("image", onMouse, 0)
42  cv2.waitKey(0)
```

40

심화예제

◆실행결과

41

8.7 원근 투시(투영) 변환

◆아테네 학당

❖ 라파엘로 산치오 작품 / 바티칸 사도궁전의 방들 중에서 서명실에 그린 벽화

❖ 벽면에 그려진 그림에서 이렇게 입체감을 느끼는 이유 → 원근법

❖ 원근법

●눈에 보이는 3차원의 세계를 2차원의 그림(평면)으로 옮길 때에 관찰자가 보는 것 그대로 사물과의 거리를 반영하여 그리는 방법

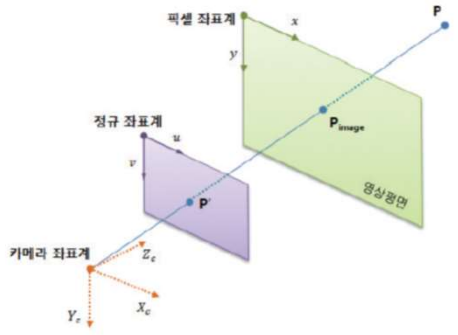
42

8.7 원근 투시(투영) 변환

◆원근 투시 변환(perspective projection transformation)

❖ 이 원근법을 영상 좌표계에서 표현하는 것

❖ 3차원의 실세계 좌표를 투영 스크린상의 2차원 좌표로 표현할 수 있도록 변환해 주는 것



(그림 8.7.2) 3차원의 실세계 좌표를 2차원 좌표계에 표현 방법

43

8.7 원근 투시(투영) 변환

◆동차 좌표계(homogeneous coordinates)

❖ 모든 항의 차수가 동일하기 때문에 붙여진 이름으로서 n차원의 투영 공간을 n+1개의 좌표로 나타내는 좌표계

- ●직교 좌표인 (x, y)를 (x, y, 1)로 표현하는 것
- ●일반화해서 0이 아닌 상수 w에 대해 (x, y)를 (wx, wy, w)로 표현
- ●상수 w가 무한히 많기 때문에 (x, y)에 대한 동차 좌표 표현은 무한히 많이 존재

❖ 동차 좌표계에서 한 점(wx, wy, w)을 직교 좌표로 나타내면

- ●각 원소를 w로 나우어 어서 (x/w, y/w)가 됨
- ●예, 동차 좌표계에서 한 점(5, 7, 5) → 직교 좌표에서(5/5, 7/5) 즉, (1 , 1.4)

◆원근 변환을 수행하는 행렬

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

44

8.7 원근 투시(투영) 변환

심화예제 8.7.1 원근 왜곡 보정 – 10.perspective_transform.py

```
01  import numpy as np, cv2
02
03  image = cv2.imread("images/perspective.jpg", cv2.IMREAD_COLOR)
04  if image is None: raise Exception("영상파일 읽기 에러")
05
06  pts1 = np.float32([(80, 40), (315, 133), (75, 300), (335, 300)])    # 입력 영상 4개 좌표
07  pts2 = np.float32([(50, 60), (340, 60), (50, 320), (340, 320)])     # 목적 영상 4개 좌표
08
09  perspect_mat = cv2.getPerspectiveTransform(pts1, pts2)              # 원근 변환 행렬
10  dst = cv2.warpPerspective(image, perspect_mat, image.shape[1::-1], cv2.INTER_CUBIC)
11  print("[perspect_mat] = \n%s\n" % perspect_mat )
12
13  ## 변환 좌표 계산 – 행렬 내적 이용 방법
14  ones = np.ones((4,1), np.float64)
15  pts3 = np.append(pts1, ones, axis=1)                  # 원본 좌표→동차 좌표 저장
16  pts4 = cv2.gemm(pts3, perspect_mat.T, 1, None, 1)     # 좌표 변환값 계산
17
```

4개 좌표쌍을 이용해 원근 변환 행렬 계산

원근 변환 행렬로 원근 변환 수행

원근 변환 in OpenCV

cv2.**getPerspectiveTransform**() 함수
4개의 좌표쌍으로부터 원근변환 행렬을 계산

cv2.**warpPerspective**() 함수
원근변환 행렬에 따라서 원근변환 수행

45

8.7 원근 투시(투영) 변환

```
18  ## 변환 좌표 계산 – cv2.transform() 함수 이용방법
19  # pts3 = np.expand_dims(pts1, axis=0)                  # 차원 증가
20  # pts4 = cv2.transform(pts3, m=perspect_mat)
21  # pts4 = np.squeeze(pts4, axis=0)                      # 차원 감소
22  # pts3 = np.squeeze(pts3, axis=0)                      # 차원 감소 – 결과 표시 위해
23
24  print(" 원본 영상 좌표 \t 목적 영상 좌표 \t\t 동차 좌표 \t\t 변환 결과 좌표")
25  for i in range(len(pts4)):
26      pts4[i] /= pts4[i][2] # 동차 좌표→직교 좌표
27      print("%i : %-14s %-14s %-18s %-18s" % (i, pts1[i], pts2[i], pts3[i], pts4[i]))
28      cv2.circle(image, tuple(pts1[i].astype(int)), 3, (0, 255, 0), -1)
29      cv2.circle(dst, tuple(pts2[i].astype(int)), 3, (0, 255, 0), -1)
30
31  cv2.imshow("image", image)
32  cv2.imshow("dst_perspective", dst)
33  cv2.waitKey(0)
```

상수ω로 나누어서 동차좌표를 직교좌표로 변환

변환 좌표 위치 표시

원근 변환 in OpenCV

cv2.**transform**() 함수
입력영상의 4개 좌표와 원근 행렬을 인수로 입력하면 원근 변환된 좌표를 반환

46

8.7 원근 투시(투영) 변환

◆실행결과

```
Run: 10.perspective_transform
C:\Python\python.exe D:/source/chap08/10.perspective_transform.py
[perspect_mat] =
[[ 6.25789284e-01  3.98298577e-02 -6.88839366e+00]
 [-5.02676539e-01  1.06358288e+00  5.13923399e+01]
 [-1.57086418e-03  5.25700042e-04  1.00000000e+00]]

   원본 영상 좌표    목적 영상 좌표      동차 좌표          변환 결과 좌표
0 : [80. 40.]     [50. 60.]     [80. 40.  1.]     [50. 60.  1.]
1 : [315. 133.]   [340. 60.]    [315. 133.  1.]   [340. 60.  1.]
2 : [ 75. 300.]   [ 50. 320.]   [ 75. 300.  1.]   [ 50. 320.  1.]
3 : [335. 300.]   [340. 320.]   [335. 300.  1.]   [340. 320.  1.]
```

47

8.7 원근 투시(투영) 변환 – 심화예제

◆마우스 드래그로 선택된 영역에 원근 변환 제거하기

심화예제 8.7.2 마우스 이벤트로 원근 왜곡 보정 – 11.perspective_event.py

```
01  import numpy as np, cv2
02  from Common.functions import imread, contain          # 좌표로 범위 확인 함수 임포트
03
04  def draw_rect(img):                                   # 좌표 사각형 그리기 함수
05      rois = [(p - small, small * 2) for p in pts1]     # 좌표 사각형 관심 영역
06      for (x, y), (w, h) in np.int32(rois):
07          roi = img[y:y+h, x:x+w]                       # 좌표 사각형 범위 가져오기
08          val = np.full(roi.shape, 80, np.uint8)        # 컬러(3차원) 행렬 생성
09          cv2.add(roi, val, roi)                        # 관심영역 밝기 증가
10          cv2.rectangle(img, (x, y, w, h), (0, 255, 0), 1)
11      cv2.polylines(img, [pts1.astype(int)], True, (0, 255, 0), 1)  # 4개 좌표 잇기
12      cv2.imshow("select rect", img)
13
14  def warp(img):                                        # 원근 변환 수행 함수
15      perspect_mat = cv2.getPerspectiveTransform(pts1, pts2)
16      dst = cv2.warpPerspective(img, perspect_mat, (350, 400), cv2.INTER_CUBIC) # 원근 변환
17      cv2.imshow("perspective transform", dst)
18
```

4개좌표 잇는 직선

4개 좌표 - 작은 사각형으로 표시

마우스 드래그로 선택된 4개 좌표와 목적영상 4개 좌표로 원근 변환 행렬 계산

48

8.7 원근 투시(투영) 변환

```
19  def onMouse(event, x, y, flags, param):                # 마우스 이벤트 처리 함수
20      global check
21      if event == cv2.EVENT_LBUTTONDOWN:
22          for i, p in enumerate(pts1):
23              p1, p2 = p - small, p + small              # p 좌표의 우상단, 좌하단
24              if contain((x,y), p1, p2): check = i       # 클릭 좌표로 좌표 사각형
25
26      if event == cv2.EVENT_LBUTTONUP:  check = -1       # 마우스 업시 좌표번호 초
27
28      if check >= 0 :                                    # 좌표 사각형 선택 시
29          pts1[check] = (x, y)
30          draw_rect(np.copy(image))
31          warp(np.copy(image))
32
33  image = cv2.imread('images/perspective2.jpg', cv2.IMREAD_COLOR)
34  if image is None: raise Exception("영상파일 읽기 에러")
35
36  small = np.array([12, 12])                             # 좌표 사각형 크기
37  check = -1                                             # 선택 좌표 사각형 번호 초기화
38  pts1 = np.float32([(100, 100), (300, 100), (300, 300), (100, 300)])  # 4개 좌표 초기화
39  pts2 = np.float32([(0, 0), (400, 0), (400, 350), (0, 350)])          # 목적 영상 4개 좌표
40
41  draw_rect(np.copy(image))
42  cv2.setMouseCallback("select rect", onMouse, 0)
43  cv2.waitKey(0)
```

4개 사각형 중 선택된 사각형의 번호 체크

49

단원 요약

◆2×3 크기의 어파인 변환 행렬을 이용해서 회전, 크기변경, 평행이동 등을 복합적으로 수행할 수 있다. OpenCV에서는 cv::getAffineTransform()와 getRotationMatrix2D() 함수로 어파인 변환 행렬을 만들며, cv::warpAffine() 함수로 어파인 변환을 수행한다.

◆원근법은 눈에 보이는 3차원의 세계를 2차원의 평면으로 옮길 때에 관찰자가 보는 것 그대로 사물과의 거리를 반영하여 그리는 방법을 말한다. 그리고 이 원근법을 영상 좌표계에서 표현하는 것이 원근 투시 변환이다. 원근 변환에서는 주로 동차 좌표계를 사용하는 것이 편리하다.

◆OpenCV에서는 cv::getPerspectiveTransform() 함수로 원근 변환 행렬을 계산하며, cv::warpPerspective() 함수는 원근변환 행렬에 따라서 원근변환을 수행한다. 또한 cv::transform() 함수는 입력영상의 4개 좌표와 원근 행렬을 인수로 입력하면 원근변환된 좌표를 반환해 준다.

50