

Basic image processing techniques

Part 2

Byeongjoon Noh

Dept. of AI and Bigdata, SCH Univ.

powernoh@sch.ac.kr

Contents

1. Sampling, Quantization, Encoding
2. Histogram
3. Binary image
4. Operations in image processing
5. Multi resolution

4. Operations in image processing

Operation in image processing

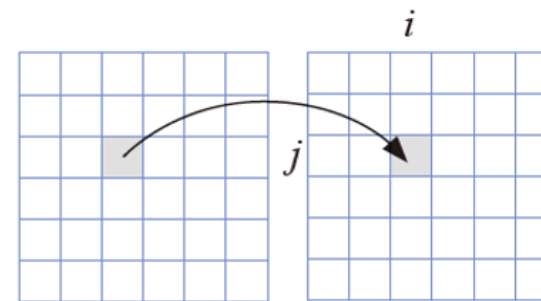
화소가 새로운 값을 어디서 갱신(획득)하느냐에 따라 세 가지 유형으로 구분됨

- Point operation (점 연산)
 - 자기 자신으로부터 획득
 - 히스토그램 평활화 등

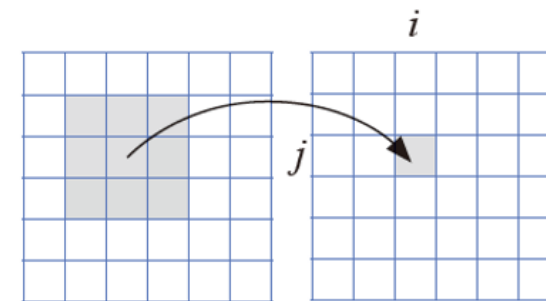
- Area operation (영역 연산)
 - 이웃 화소들로부터 받음

- Geometric operation

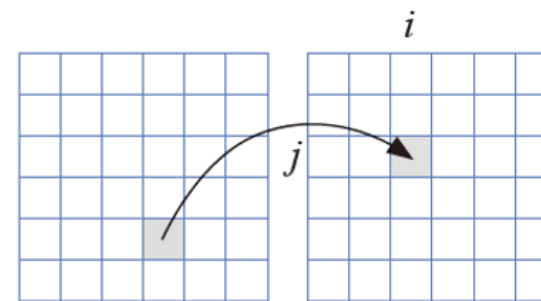
- 기하학적 변환이 정해주는 곳에서 받음



(a) 점 연산



(b) 영역 연산



(c) 기하 연산

Point operation

Expression

$$f_{out}(j, i) = t(f_1(j, i), f_2(j, i), \dots, f_k(j, i))$$

- 대부분 $K = 1$ (한 장의 영상을 변환)

Linear operation

- 모든 픽셀 값을 동일한 비율만큼 증가 또는 감소

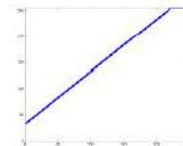
$$f_{out}(j, i) = \begin{cases} \min(f(j, i) + a, L - 1) & \text{(밝게)} \\ \max(f(j, i) + a, 0) & \text{(어둡게)} \\ (L - 1) - f(j, i) & \text{(반전)} \end{cases}$$



(a) 원래 영상



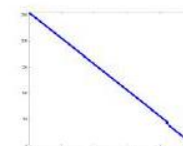
(b) 밝게(a=32)



(c) 어둡게(a=32)



(d) 반전

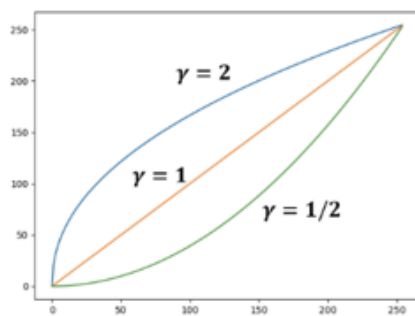


Point operation

Non-linear operation

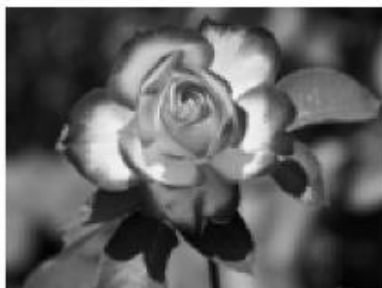
- Gamma (γ) correction (감마 보정)
 - $\gamma > 1$
 - 어두운 영역의 변화폭 \uparrow , 밝은 영역의 변화폭 \downarrow
 - ➔ 밝은 영역이 급격하게 밝아지지 않고 유지, 어두운 영역만 상대적으로 밝게
 - 모니터나 프린터 색상 조절에 사용

$$f_{out}(j, i) = (L - 1) * \left(\hat{f}(j, i) \right)^{1/\gamma} ; \hat{f}(j, i) = \frac{f(j, i)}{L - 1}$$

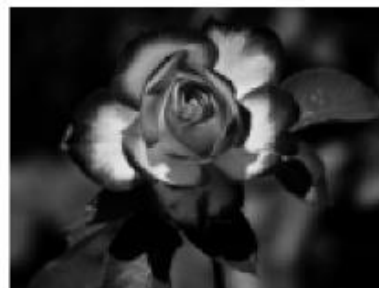


gamma 에 따른 pixel 값 변화

Original



Gamma = 0.5



Gamma = 2.2



Point operation

Non-linear operation

- 1_5_gammaCorrection.py
 - (실행 전) matplotlib 라이브러리 설치 필요
 - > pip install matplotlib
 - 이전 슬라이드의 Gamma correction 공식과 비교 해보기!

```
def gamma_correction(image, gamma=1.0):  
    inv_gamma = 1 / gamma  
    output = np.uint8(((image / 255) ** inv_gamma) * 255)  
    return output
```

Point operation

Non-linear operation

- Dissolve
 - 녹다, 용해시키다 → 영상(이미지)를 합치다
 - 영상(video)에서 앞 영상의 끝~그 다음 영상의 시작 간의 fade in/out 효과

$$f_{out}(j, i) = \alpha * f_1(j, i) + (1 - \alpha) * f_2(j, i)$$



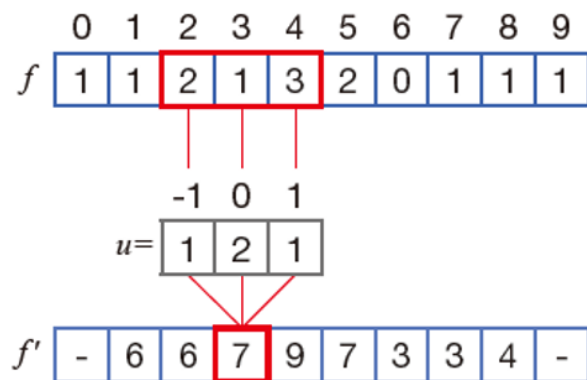
Area operation

이웃 화소를 고려하여 새로운 값을 결정

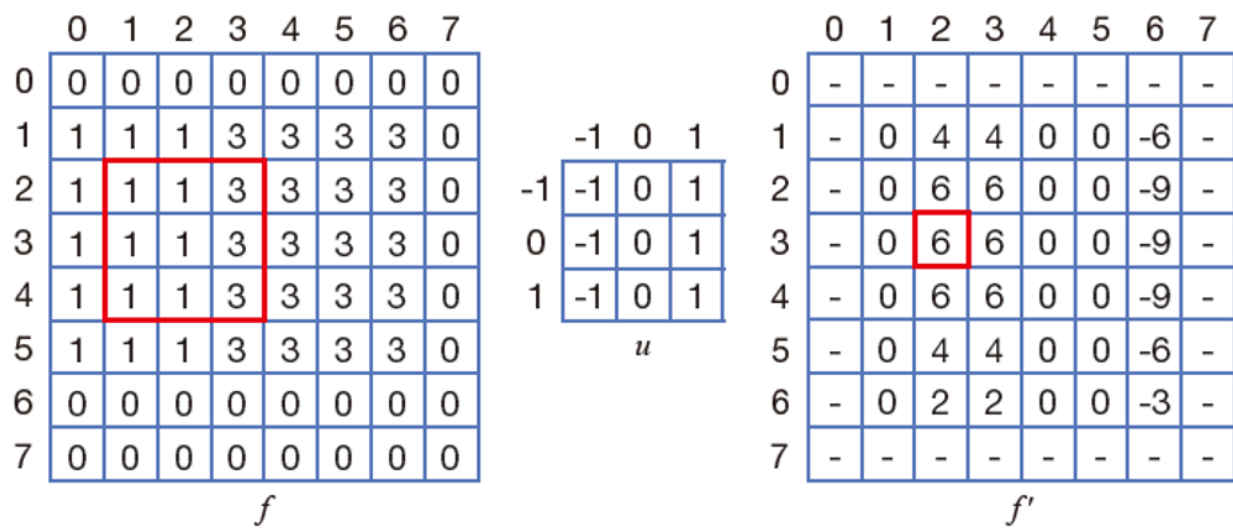
- Convolution (컨볼루션)

$$f'(x) = \sum_{i=-(w-1)/2}^{(w-1)/2} u(i) f(x+i)$$

$$f'(y, x) = \sum_{j=-(h-1)/2}^{(h-1)/2} \sum_{i=-(w-1)/2}^{(w-1)/2} u(j, i) f(y+j, x+i)$$



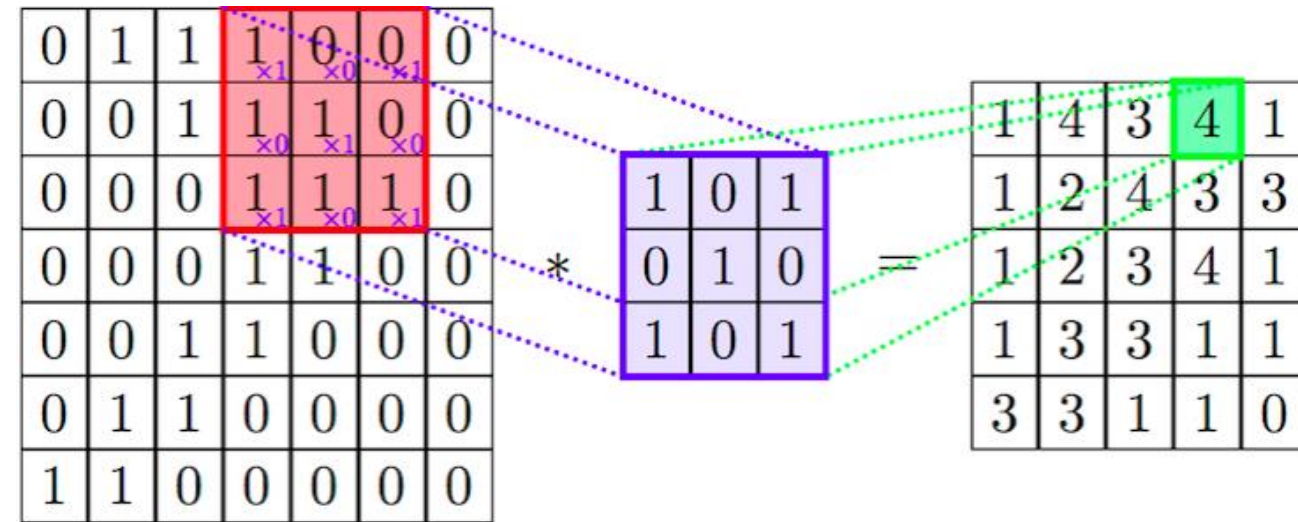
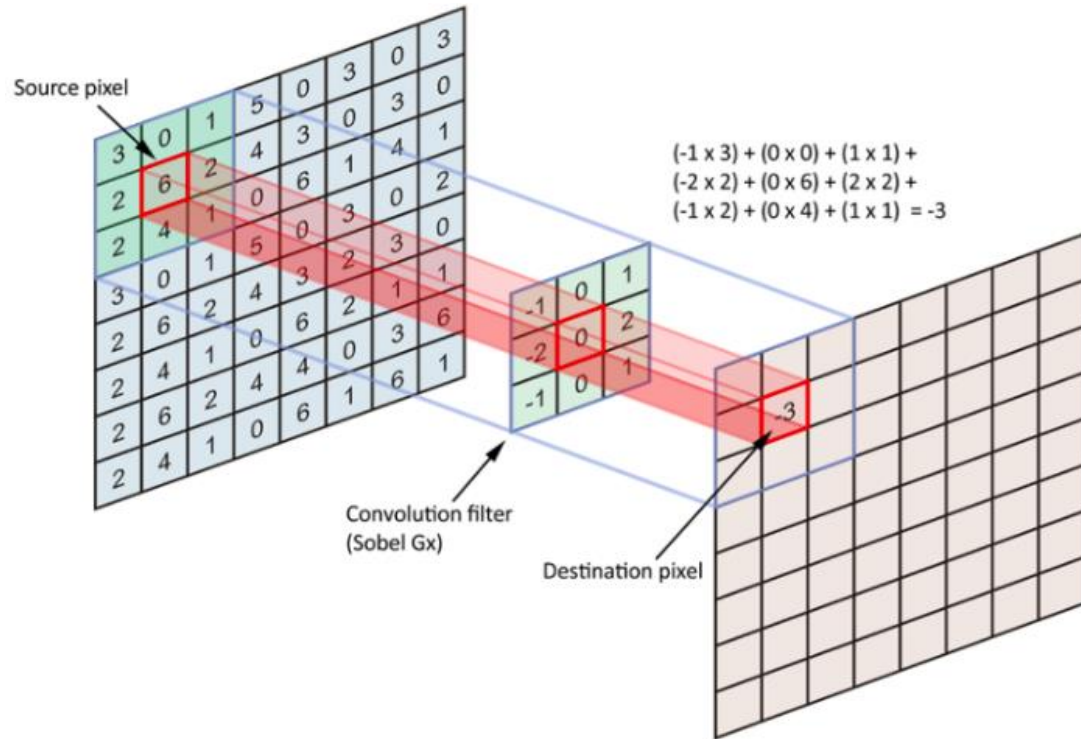
(a) 1차원 영상에 컨볼루션 적용



(b) 2차원 영상에 컨볼루션 적용

Area operation

Convolution 연산 방법 (자세히)



Area operation

Filter의 역할

- 연구를 통해 다양한 목적에 맞는 필터 matrix가 발견(발명??)

1/9	1/9	1/9	0.0030	0.0133	0.0219	0.0133	0.0030
1/9	1/9	1/9	0.0133	0.0596	0.0983	0.0596	0.0133
1/9	1/9	1/9	0.0219	0.0983	0.1621	0.0983	0.0219
			0.0133	0.0596	0.0983	0.0596	0.0133
			0.0030	0.0133	0.0219	0.0133	0.0030

(a) 스무딩 필터

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

(b) 샤프닝 필터

-1	0	0	-1	-1	0
0	0	0	-1	0	1
0	0	1	0	1	1

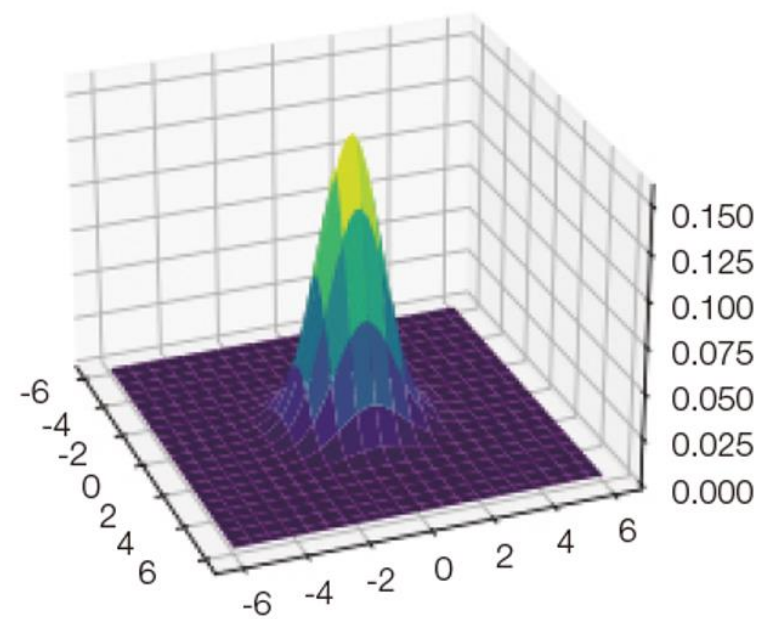
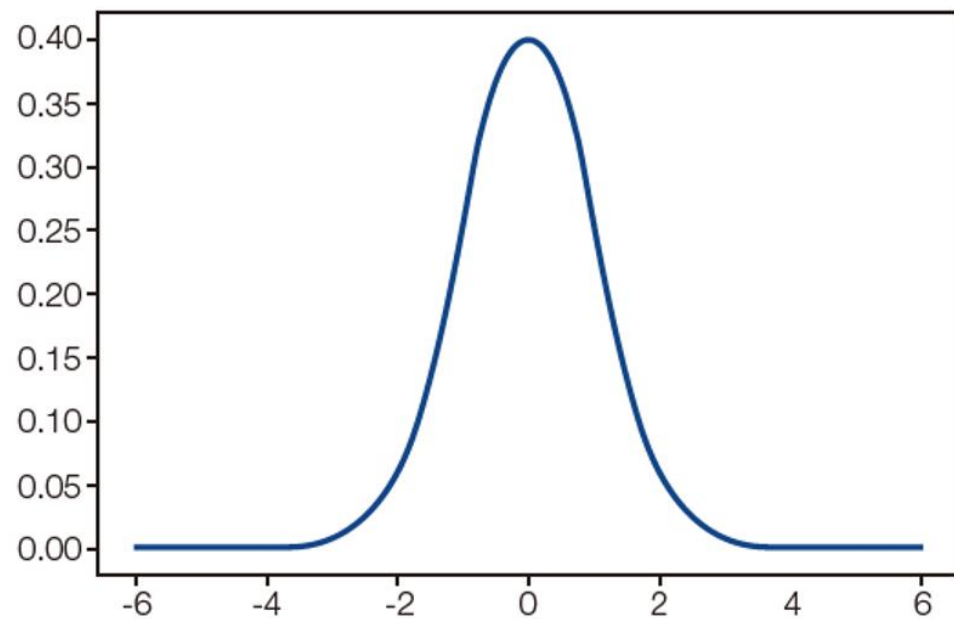
(c) 엠보싱 필터

Area operation

Gaussian filter

1차원 가우시안: $g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$

2차원 가우시안: $g(y, x) = \frac{1}{\sigma^2 2\pi} e^{-\frac{y^2+x^2}{2\sigma^2}}$



Area operation

필터 생성 및 활용 예제

- 1_6_filter.py

가우시안 필터 예제

- 1_7_Gaussianfilter.py, 1_8_Gaussianfilter2.py

```
g_kernel = cv2.getGaussianKernel(3, 0)
g_blur1 = cv2.filter2D(image, -1, g_kernel*g_kernel.T)
```

```
# 2) 가우시안 블러자체를 opencv 함수를 호출하여 활용
g_blur2 = cv2.GaussianBlur(image, (3, 3), 0)
```

Matplotlib 라이브러리 활용하여 사진 한번에 출력 및 비교하기 예제

- 1_9_subPlotView.py

Area operation

** (참고) OpenCV 사용 시 연산 결과를 저장하는 변수의 유효 값 범위

- In OpenCV, 영상 화소 = numpy.uint8 데이터 타입으로 표현 ([0, 255])

```
img = cv2.imread(path)
print(type(img[0, 0, 0]))
```

➔ <class 'numpy.uint8'>

- [0, 255] 범위를 벗어나는 경우 문제 발생

```
a = np.array([-3, -2, -1, 0, 1, 254, 255, 256, 257, 258], dtype = np.uint8)
print(a)
```

➔ [253 254 255 0 1 254 255 0 1 2]

Geometric operation

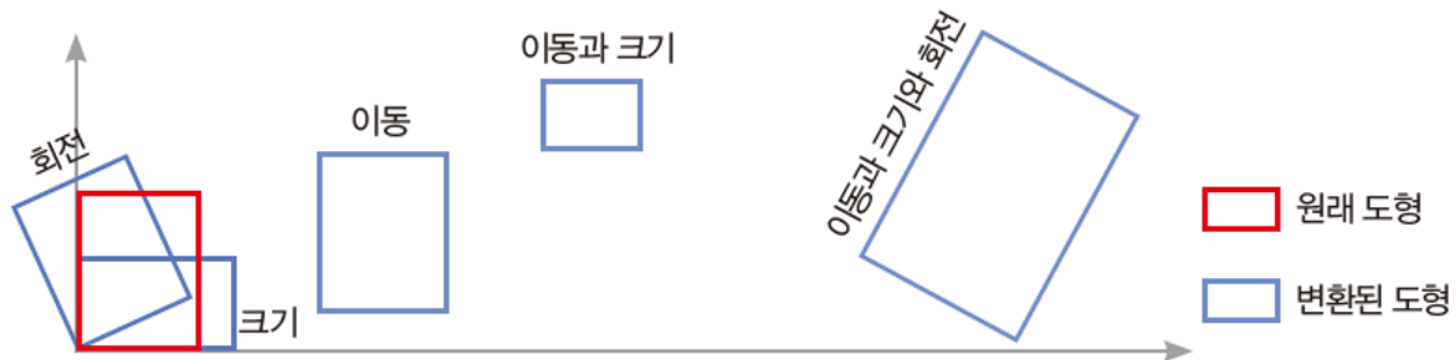
특정 위치의 화소에서 값을 획득함

- 특정 위치 = 기하 연산의 결과
- 주로 물체의 이동, 크기, 회전에 따른 기하 변환을 수행

Geometric operation

동차 좌표 (Homogeneous coordinate)

- 2차원 좌표를 3차원 벡터로 표현
 - $p = (x, y) \rightarrow \bar{p} = (x, y, 1)$
- 3개 요소에 같은 값을 곱하면 같은 좌표
 - $(-2, 4, 1), (-4, 8, 2) \rightarrow (-2, 4)$



Geometric operation

동차 좌표 (Homogeneous coordinate)

- 동차 행렬 (Homogeneous matrix, \hat{H}) 표현법 및 그 원리
 - 이동 (Translation)

$$x' = x + t_x$$

$$y' = y + t_y$$

- (x, y) 를 각각 t_x, t_y 만큼 이동
- 행렬 표현

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

Geometric operation

동차 좌표 (Homogeneous coordinate)

- 동차 행렬 (Homogeneous matrix) 표현법 및 그 원리
 - 축척 (Scaling)

원점기준

$$\begin{aligned}x' &= s_x x \\y' &= s_y y\end{aligned}$$

임의의 점 기준

$$\begin{aligned}x' &= s_x(x - p_x) + p_x = s_x x + p_x(1 - s_x) \\y' &= s_y(y - p_y) + p_y = s_y y + p_y(1 - s_y)\end{aligned}$$

- (x, y) 를 각각 s_x, s_y 만큼 축소/확대
- 행렬 표현

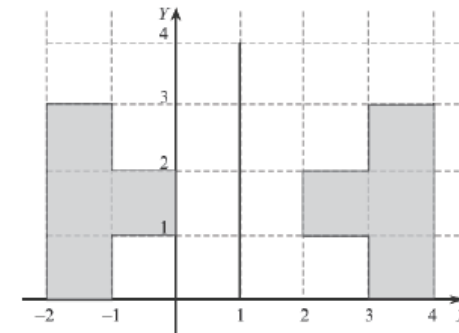
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & p_x(1 - s_x) \\ 0 & s_y & p_y(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Geometric operation

동차 좌표 (Homogeneous coordinate)

- 동차 행렬 (Homogeneous matrix) 표현법 및 그 원리
 - 반사 (Reflections)



y축 반사

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned}$$

x축 반사

$$\begin{aligned} x' &= x \\ y' &= -y \end{aligned}$$

임의의 y축 반사
($x = a_x$)

$$\begin{aligned} x' &= -(x - a_x) + a_x = -x + 2a_x \\ y' &= y \end{aligned}$$

- 행렬 표현

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

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

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2a_x \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Geometric operation

동차 좌표 (Homogeneous coordinate)

- 동차 행렬 (Homogeneous matrix) 표현법 및 그 원리
 - 회전 (Rotation)
 - $p(x, y)$ 가 각도 β 만큼 (반시계방향) 회전하여 $p'(x', y')$

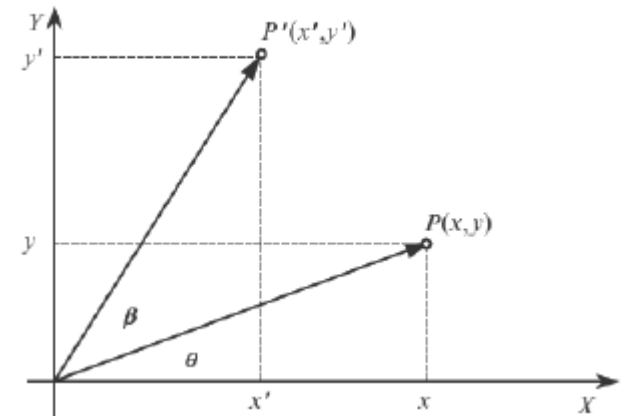
$$R(p, \beta): p(x, y) \rightarrow p'(x', y')$$

$$x' = \cos(\theta + \beta) = \cos \theta \cos \beta - \sin \theta \sin \beta = x \cos \beta - y \sin \beta$$

$$y' = \sin(\theta + \beta) = \sin \theta \cos \beta + \cos \theta \sin \beta = y \cos \beta + x \sin \beta$$

- 행렬 표현

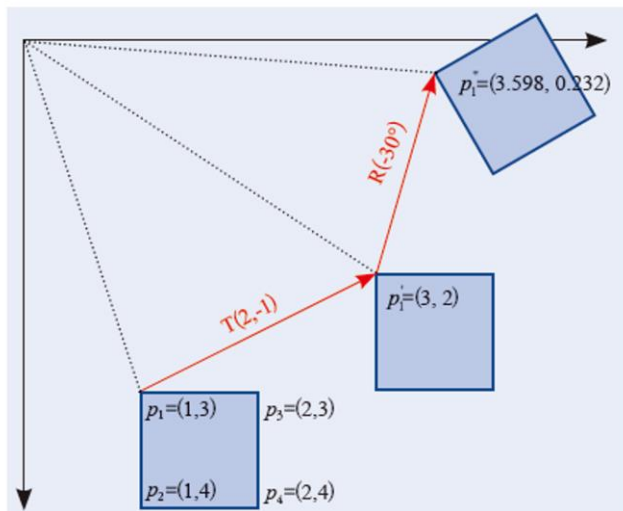
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \beta & -\sin \beta & 0 \\ \sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Geometric operation

동차 행렬을 이용한 기하 변환 예시

정사각형을 x 방향으로 2, y 방향으로 -1만큼 이동한 다음 반시계 방향으로 30도 회전



[그림 2-26]의 삼각형을 y방향으로 3, x방향으로 2만큼 이동시킨 후 30° 회전시켜보자.

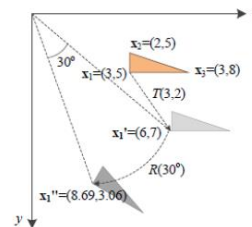


그림 2-26 기하 변환의 예

먼저, 이동 변환을 구하려면 $T(3,2)$ 가 필요하다. 꼭지점 $x_1=(3,5)$ 를 동차 좌표로 확장하여 $\hat{x}_1=(3,5,1)$ 을 만들고 식 (2.16)의 연산을 적용한다.

$$T(3,2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix}, \quad \hat{x}'_1 = \begin{pmatrix} 3 & 5 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 6 & 7 & 1 \end{pmatrix}$$

연산의 결과로 $\hat{x}'_1 = (6, 7, 1)$ 을 얻었는데, 마지막 요소를 제거하여 2차원 좌표로 바꾸면 $x'_1 = (6, 7)$ 이 된다. 나머지 두 점 x_2 와 x_3 도 같은 과정으로 변환한 후 이동한 삼각형을 그려보면 가운데 삼각형과 같다. 이제 이동한 삼각형을 30° 회전시켜보자. 회전을 계산하는 데 필요한 행렬 $R(30^\circ)$ 를 꼭지점 x' 에 적용하면, 다음과 같이 $x''_1 = (8.6962, 3.0622)$ 를 얻는다. 나머지 두 점을 계산하고 결과를 그려보면 맨 아래 삼각형과 같다.

$$\begin{pmatrix} 6 & 7 & 1 \end{pmatrix} \begin{pmatrix} \cos 30^\circ & -\sin 30^\circ & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 6 & 7 & 1 \end{pmatrix} \begin{pmatrix} 0.866 & -0.500 & 0 \\ 0.500 & 0.866 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 8.6962 & 3.0622 & 1 \end{pmatrix}$$

동차 행렬 사용 → 계산 효율화

$$A = R(30^\circ)T(2,-1) = \begin{pmatrix} 0.8660 & 0.5000 & 0 \\ -0.5000 & 0.8660 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.8660 & 0.5000 & 1.232 \\ -0.5000 & 0.8660 & -1.866 \\ 0 & 0 & 1 \end{pmatrix}$$

$$A\bar{p}_1^T = \begin{pmatrix} 0.8660 & 0.5000 & 1.232 \\ -0.5000 & 0.8660 & -1.866 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 3.598 \\ 0.232 \\ 1 \end{pmatrix}$$

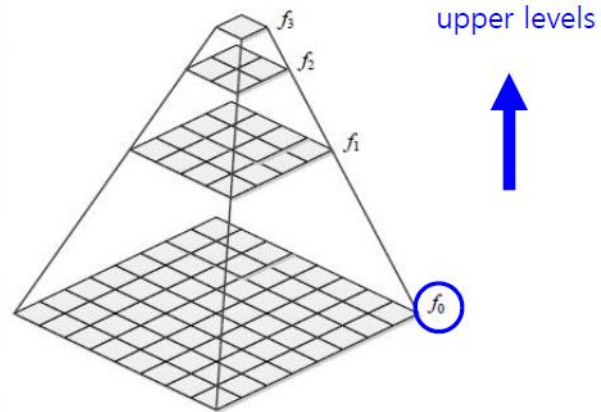
5. Multi resolution

Multi-resolution

해상도를 줄이거나 늘리는 연산

- Down-sampling, up-sampling
- 멀티미디어 장치 내 디스플레이
- 물체 크기 변환에 강인한 인식

Pyramid



End of slide