

객체지향 C++로 구현한 Computer Vision (feat. MFC)

[인텔] 엣지 AI SW 아카데미
객체 지향 프로그래밍

최재혁

Contents

1 프로젝트 개요

2 화소 점 처리

3 기하학 처리

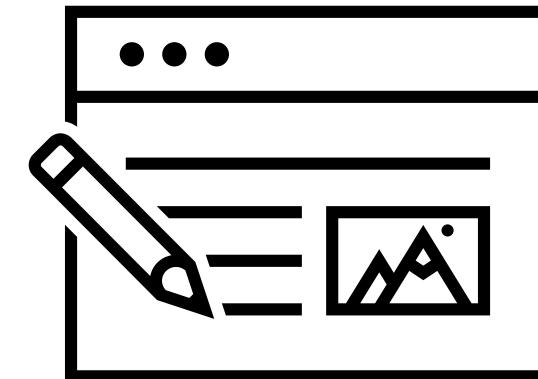
4 히스토그램 처리

5 화소 영역 처리

6 에지 검출

7 컬러이미지 효과

8 마무리



01 프로젝트 개요

목표

- C++을 이용하여 영상 처리 알고리즘을 수행하는 프로그램
- OpenCV 등 영상 처리 라이브러리의 기능을 직접 코딩하여 구현

구현 알고리즘

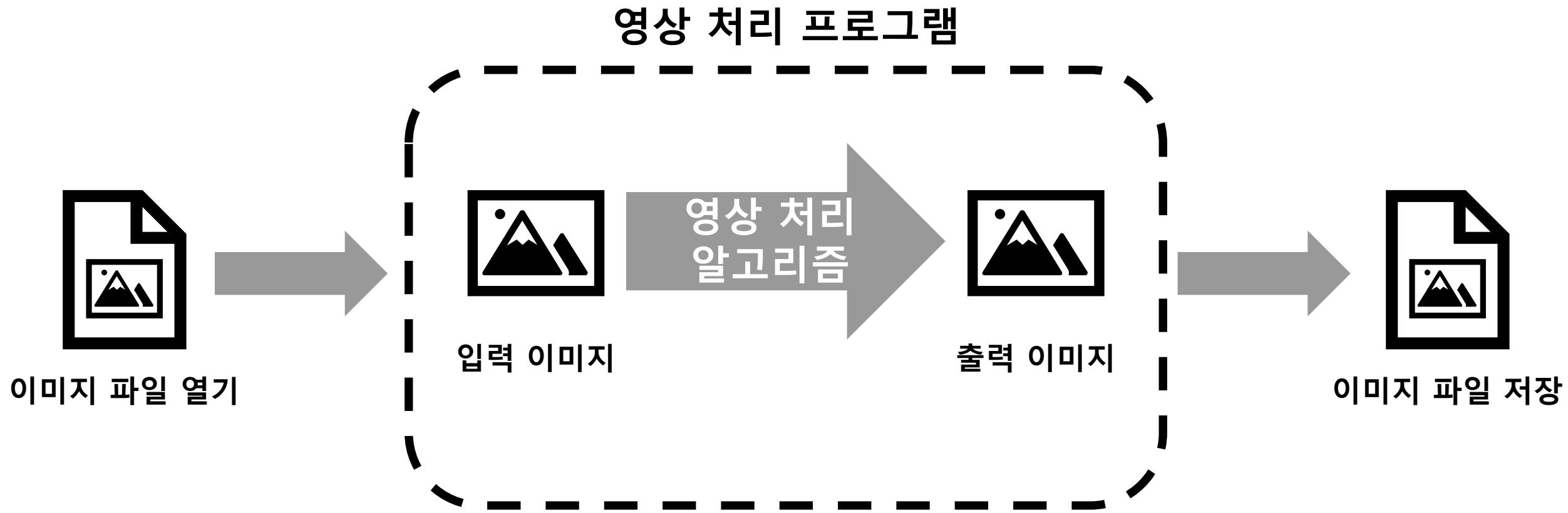
- 화소 점 처리
- 화소 영역 처리
- 기하학 처리
- 에지 검출
- 히스토그램 처리
- 컬러이미지 효과

개발 환경

- Windows10 x64
- VisualStudio2022
- MFC

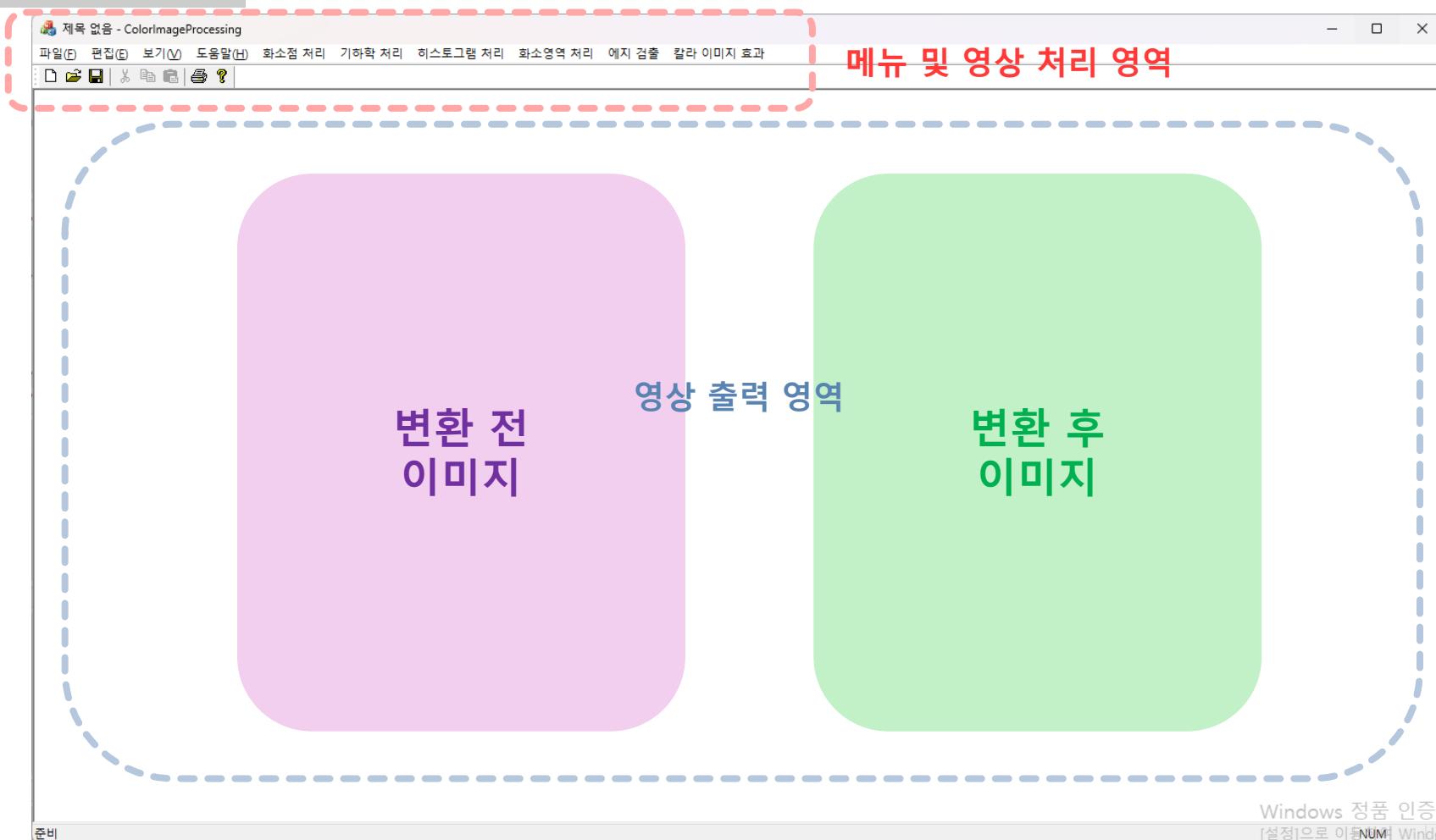
01 프로젝트 개요

프로그램 구조



01 프로젝트 개요

화면 구성



02 화소 점 처리

1 동일 이미지

2 밝게/어둡게

3 곱/나누기

4 그레이 스케일

5 비트 연산

6 흑백 변환

7 감마 보정

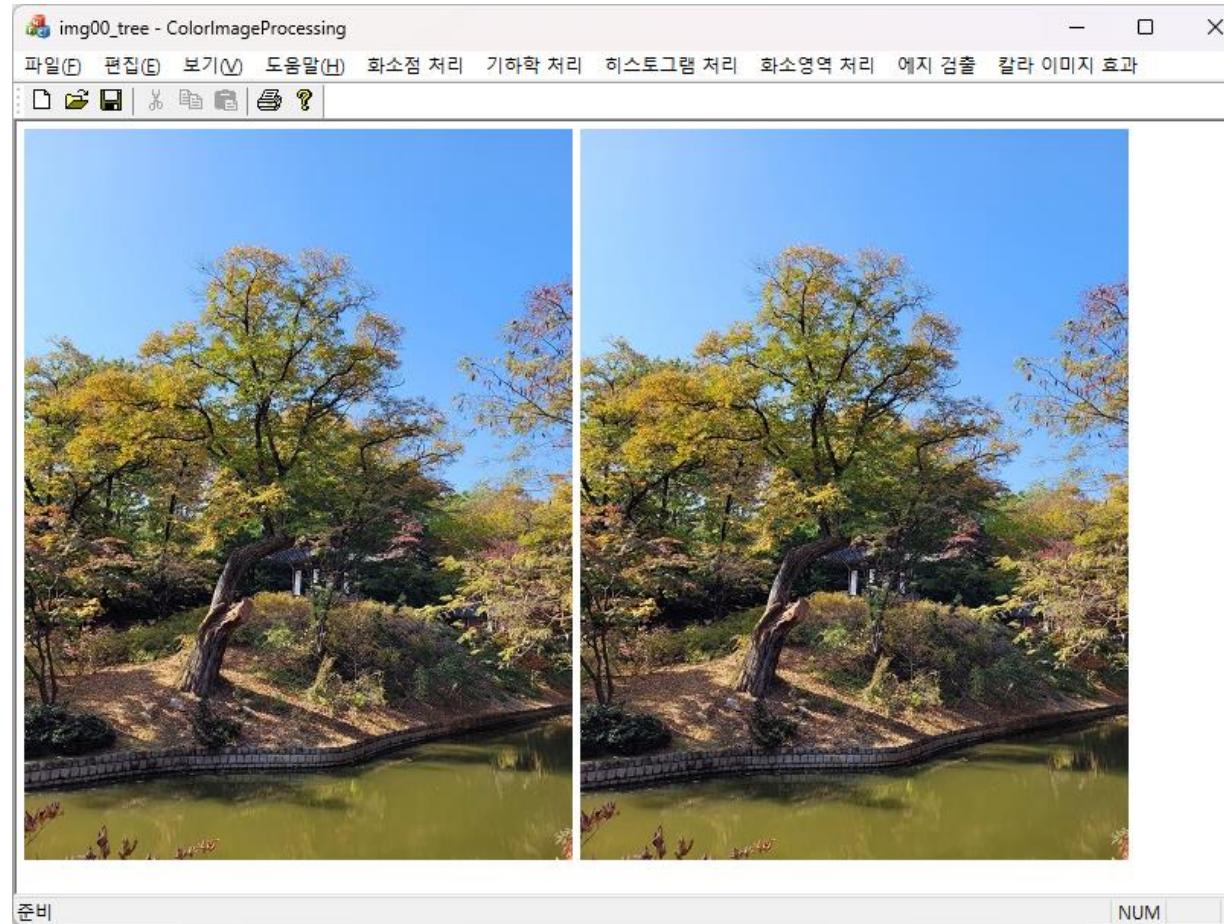
8 명암 대비 변환

9 포스터라이징

동일 이미지

동일 이미지

- 최초 열기를 통해 가져온 이미지를 출력



밝게/어둡게

밝게/어둡게

- 입력 이미지를 입력한 화소값 만큼 밝게/어둡게 하여 출력

+ 100



입력 이미지

- 100



출력 이미지

곱/나누기

곱/나누기

- 입력 이미지를 입력한 값만큼 곱하거나 나누어 화소값 간의 차이를 크거나 작게 함

X 2



입력 이미지

/ 2

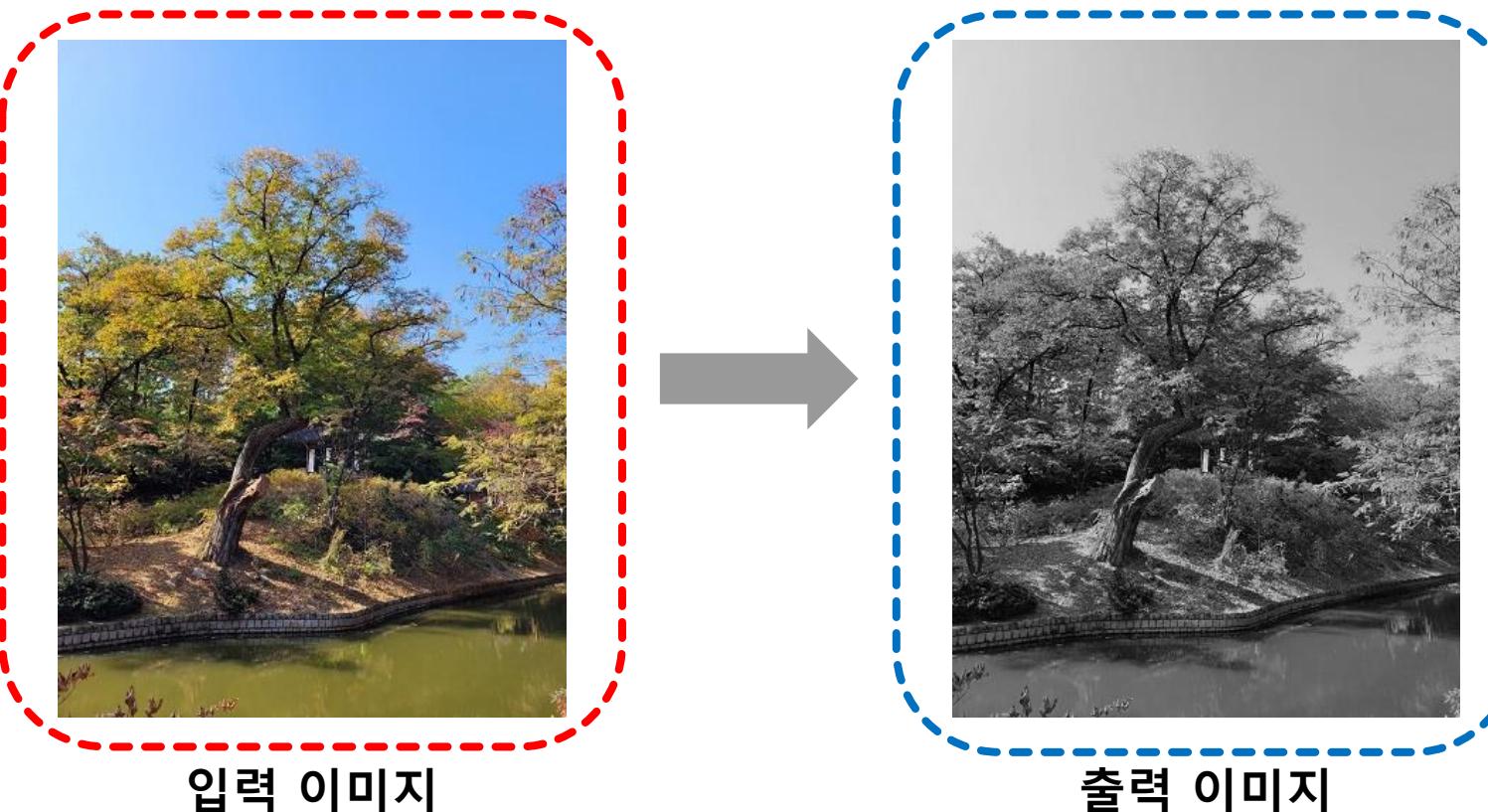


출력 이미지

그레이 스케일

그레이 스케일

- 입력 이미지의 RGB 값의 평균값으로 변경하여 단일색의 이미지로 변환

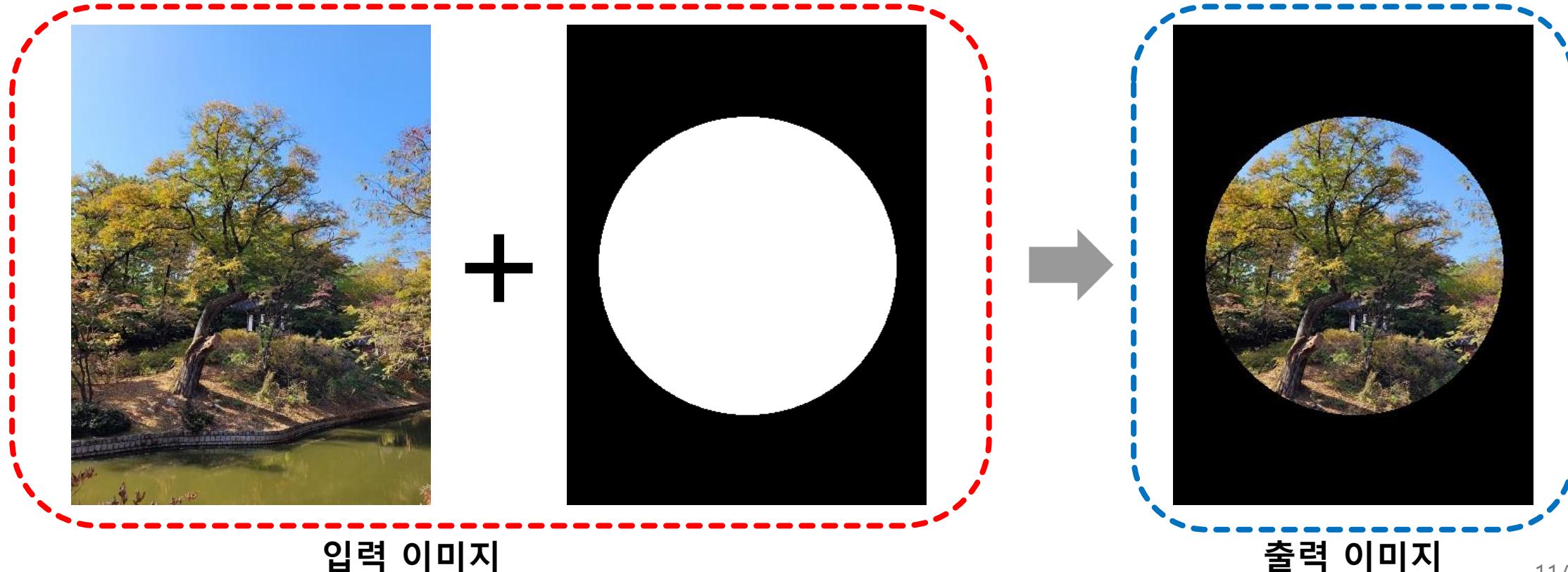


비트연산

비트연산(AND)

- 입력 이미지를 다른 이미지와 AND연산

A	0	0	1	1
B	0	1	0	1
(AND)	0	0	0	1

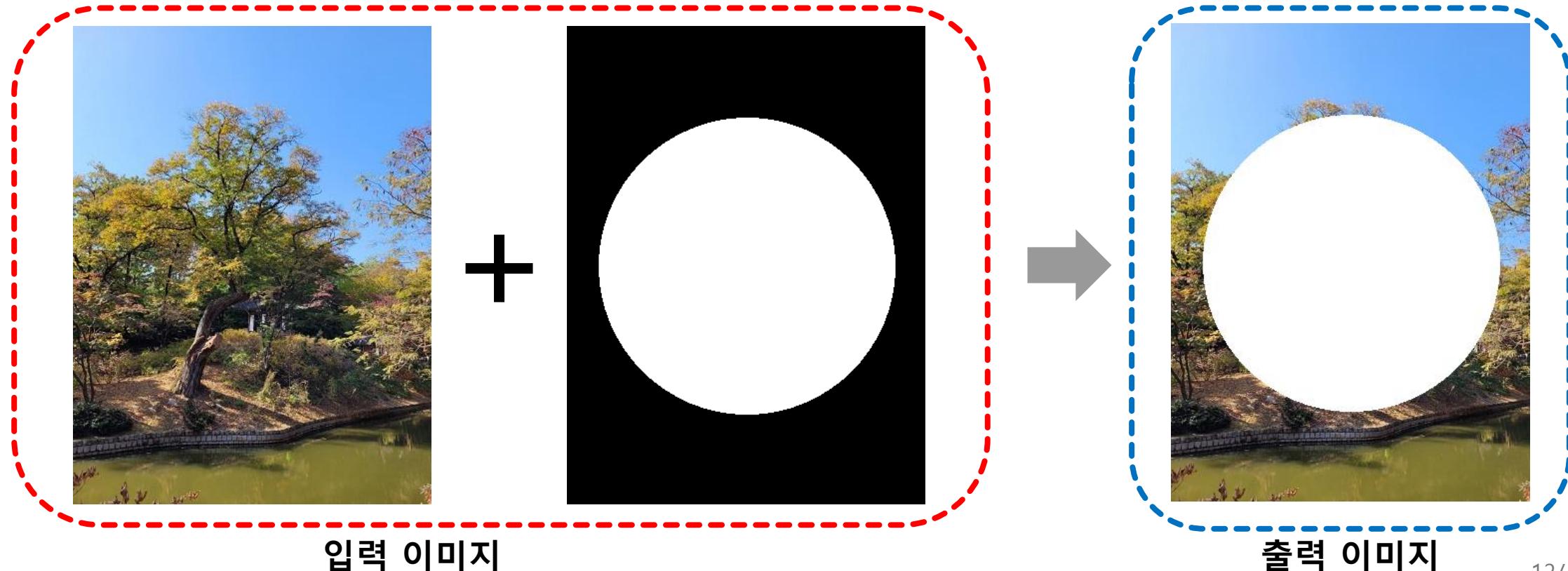


비트연산

비트연산(OR)

- 입력 이미지를 다른 이미지와 OR연산

A	0	0	1	1
B	0	1	0	1
(OR)	0	1	1	1

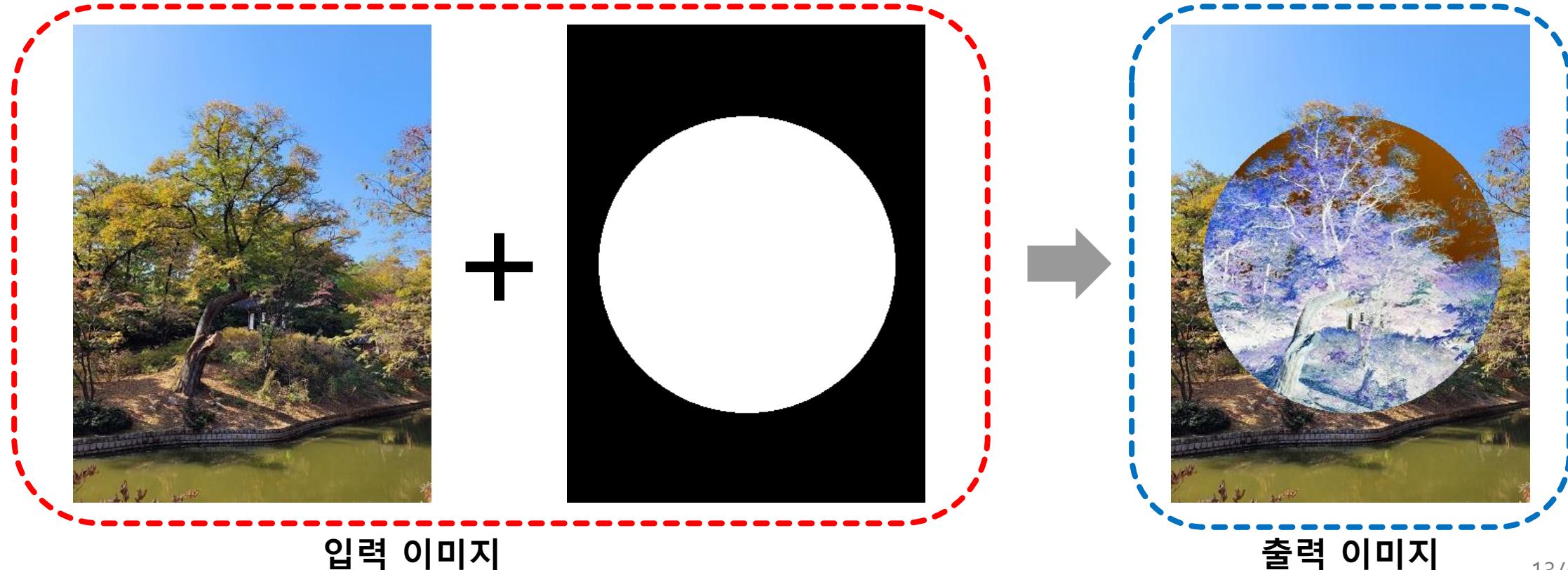


비트연산

비트연산(XOR)

- 입력 이미지를 다른 이미지와 XOR연산

A	0	0	1	1
B	0	1	0	1
(XOR)	0	1	1	0



비트 연산

비트 연산(NOT)

- 입력 이미지에 대해 NOT연산
- 반전 효과와 동일함

A	0	1
(NOT)	1	0



입력 이미지

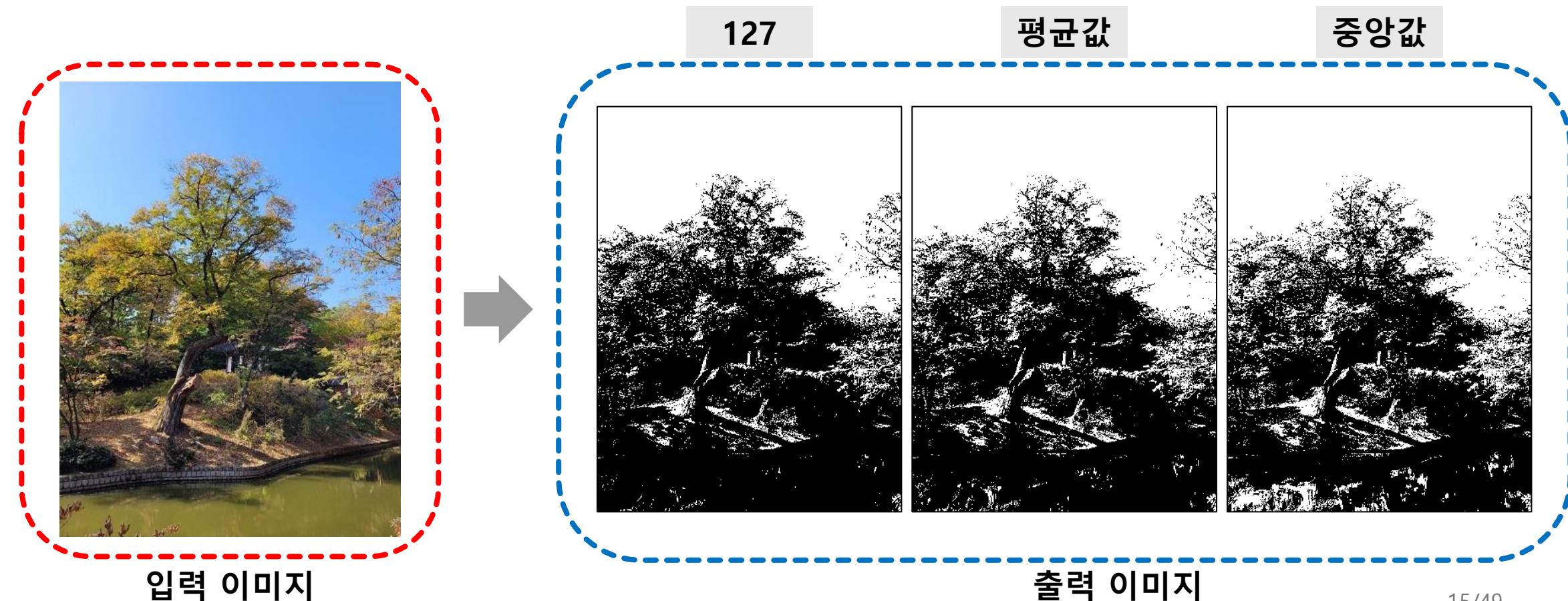


출력 이미지

흑백 변환

흑백 변환

- 입력 이미지를 특정 화소값을 기준으로 2가지 색(흰색, 검은색)으로 변환



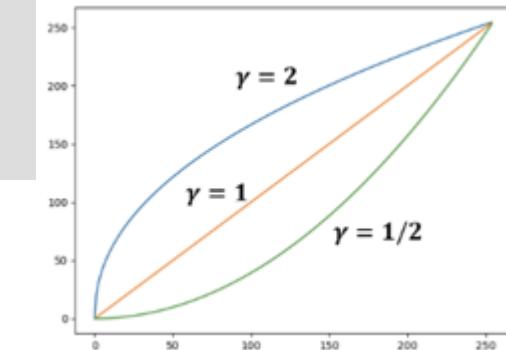
02 화소 점 처리 감마보정

감마 보정

- 감마 보정 함수를 적용하여 이미지의 밝기를 개선
- 감마 보정 함수 : $Output/M = (Input/M)^{1/\gamma}$

2.2

0.5

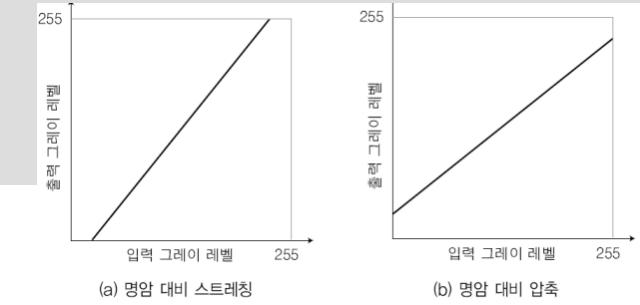


02 화소 점 처리

명암 대비 변환

명암 대비 변환

- 명암 대비 변환 함수의 그래프의 기울기를 조절하여 이미지의 밝기 차이를 조절



스트레칭

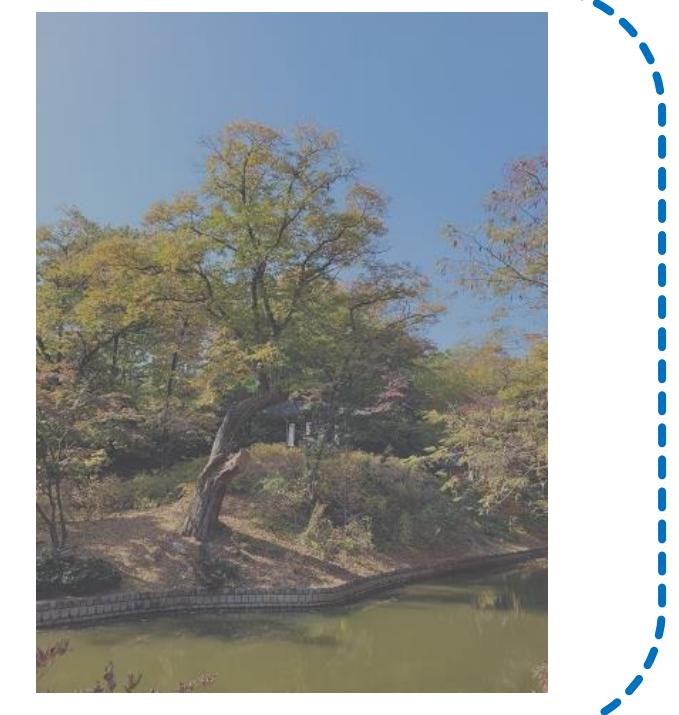
압축



입력 이미지



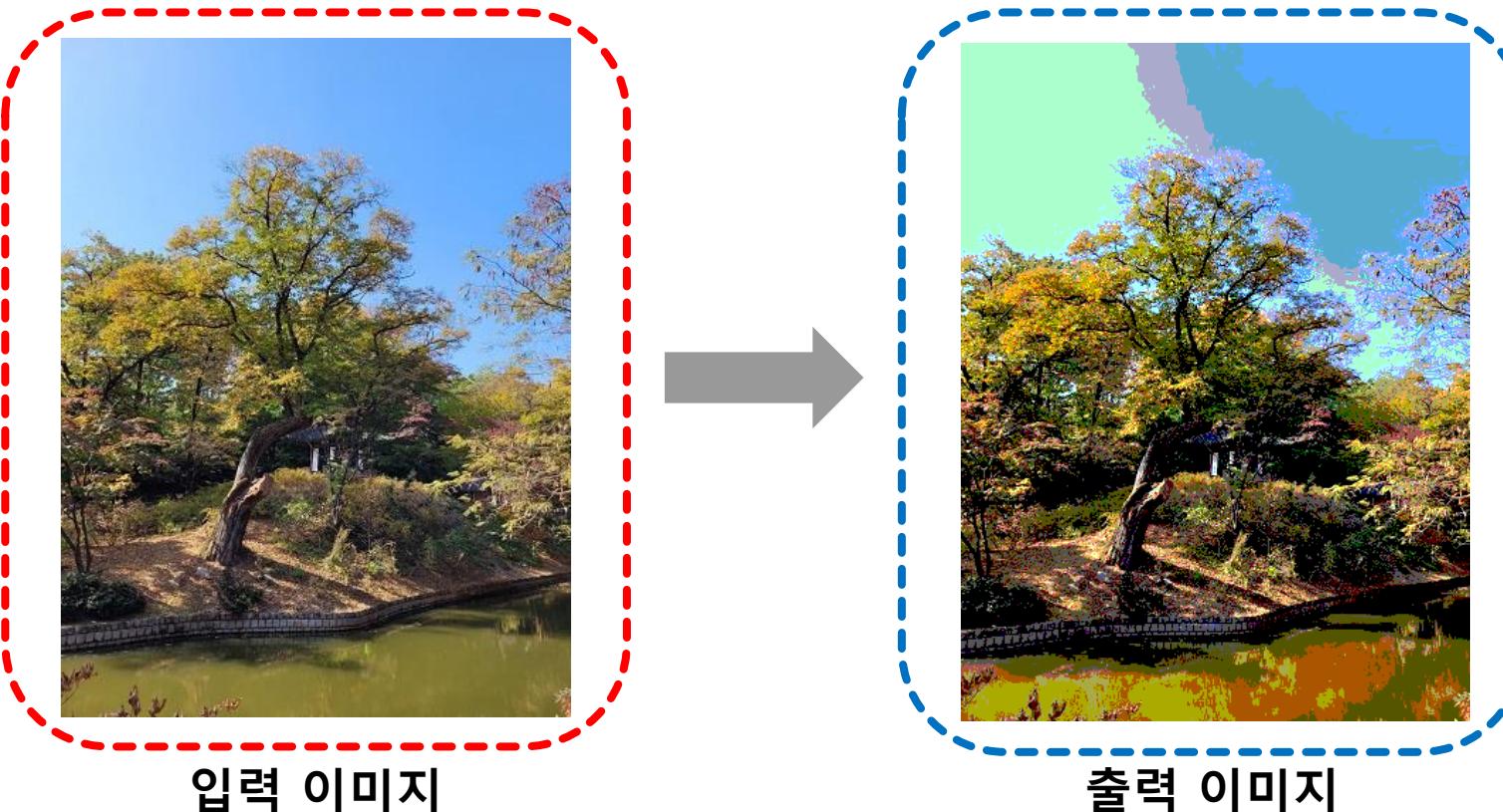
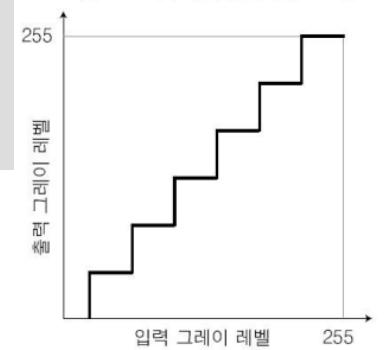
출력 이미지



포스터라이징

포스터라이징

- 이미지의 화소값의 범위를 경계값으로 축소하여 계단 그래프 상의 화소값으로 변경



03 기하학 처리

1 확대

2 축소

3 회전

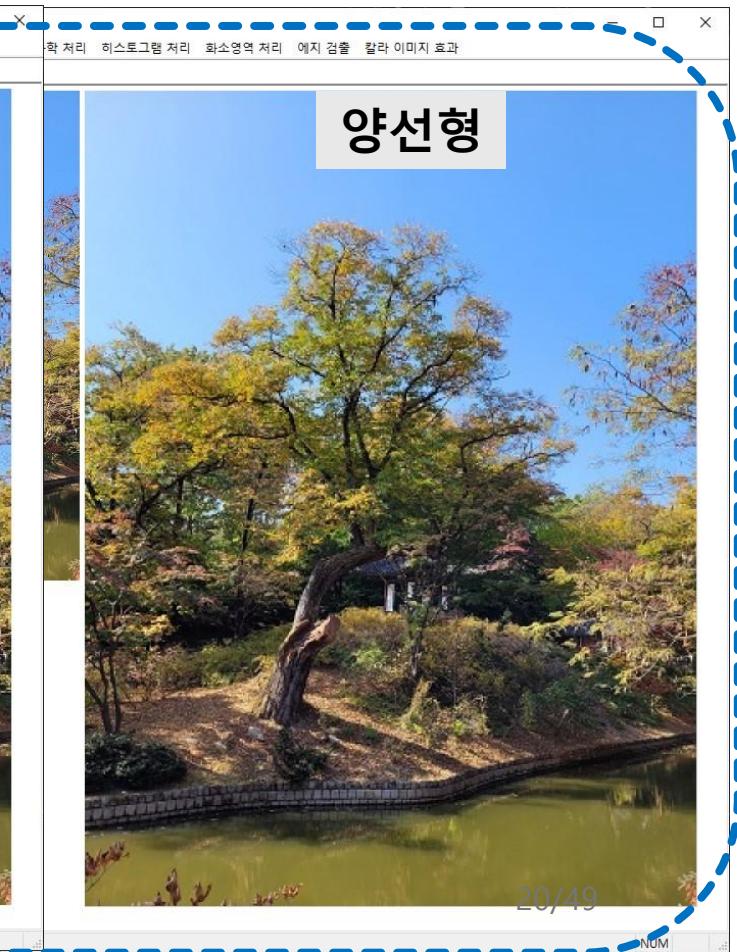
4 이동

5 미러링

확대

확대

- 이미지의 크기를 특정 비율로 확대



축소

축소

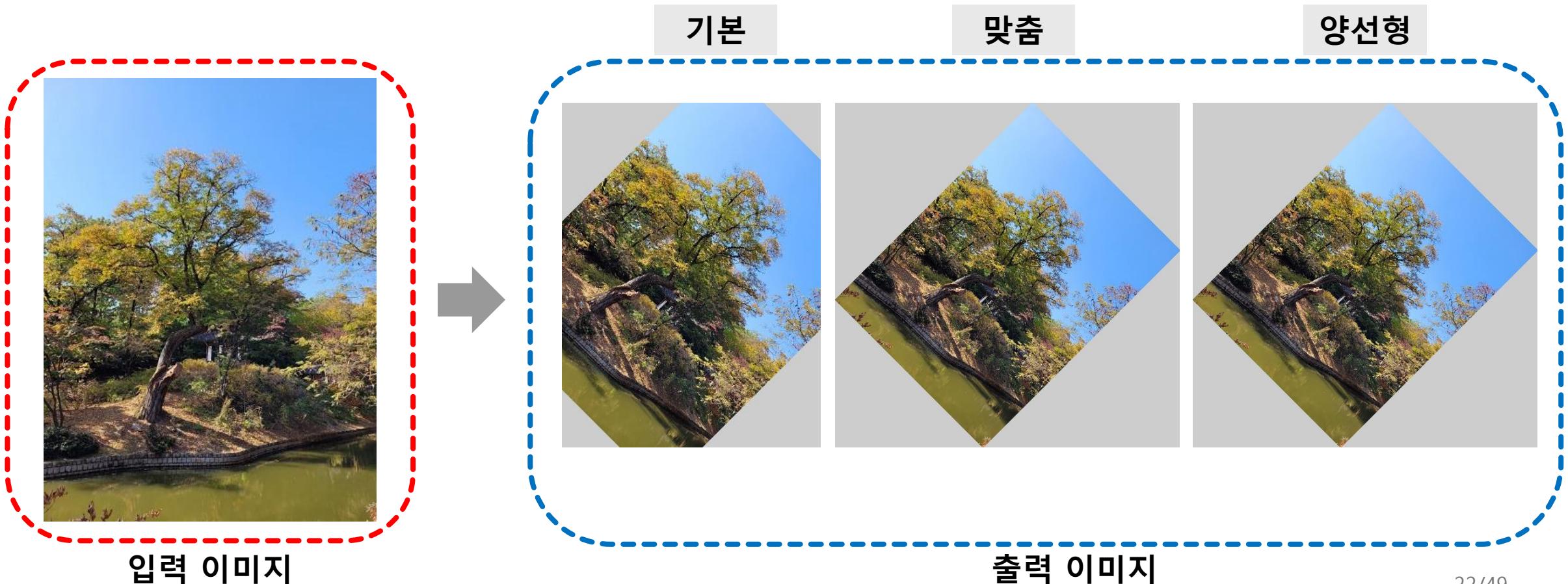
- 이미지의 크기를 특정 비율로 축소



회전

회전

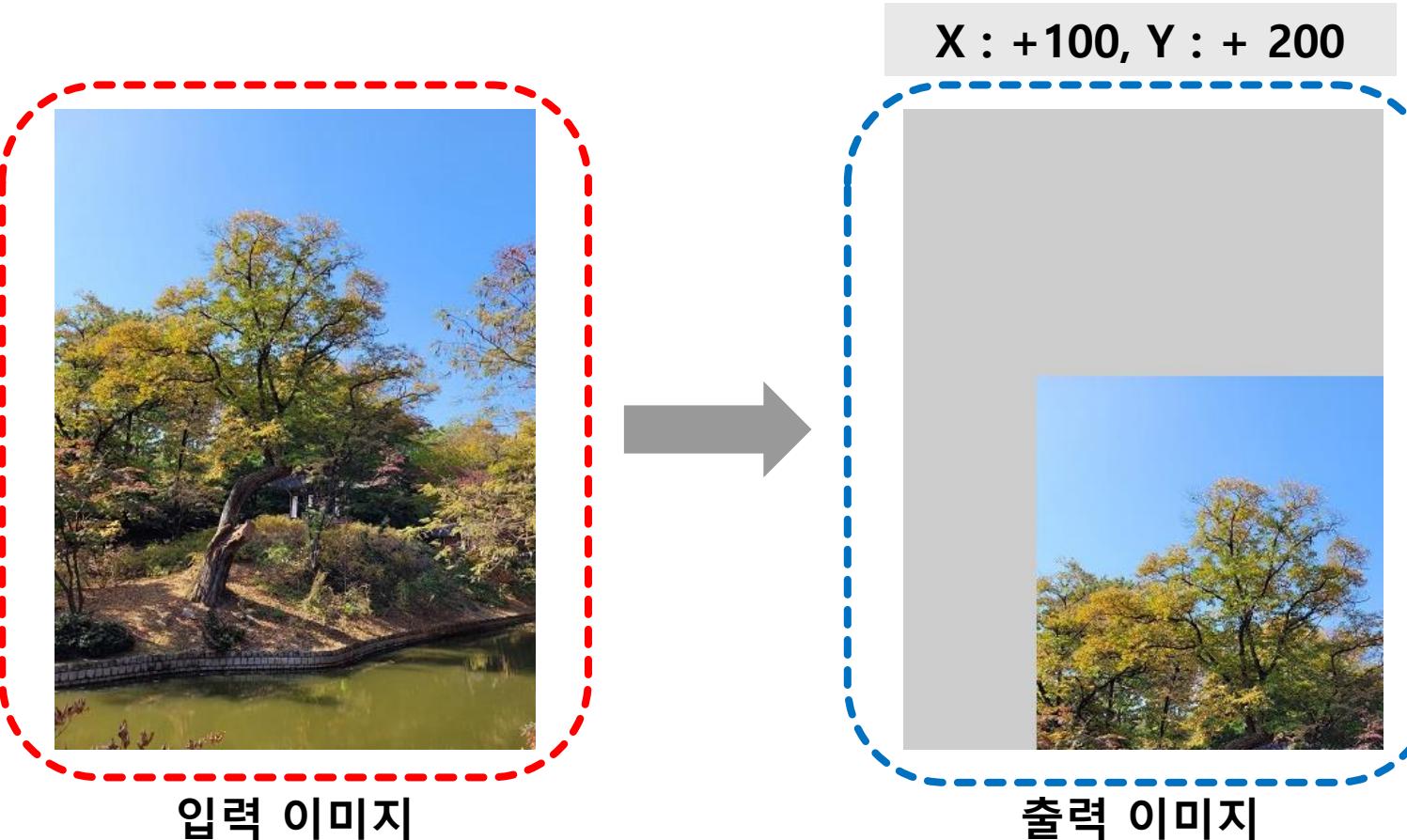
- 이미지의 중앙을 기준으로 입력 degree만큼 시계방향 회전



이동

이동

- 이미지의 화소값의 범위를 경계값으로 축소하여 계단 그래프 상의 화소값으로 변경



미러링

미러링

- 입력 이미지를 상하, 좌우로 대칭 변환



입력 이미지



상하



좌우

출력 이미지

04 히스토그램 처리

1 스트레칭

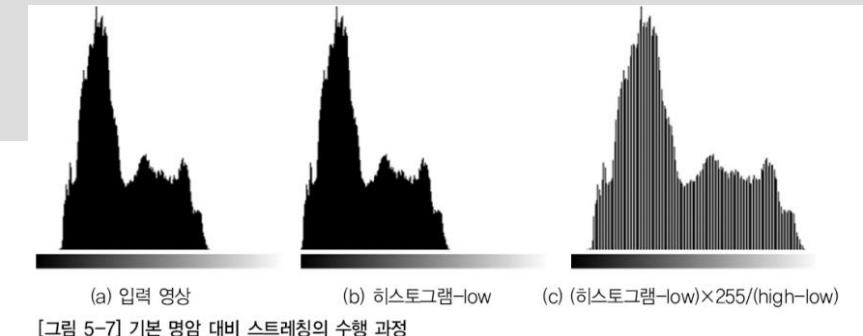
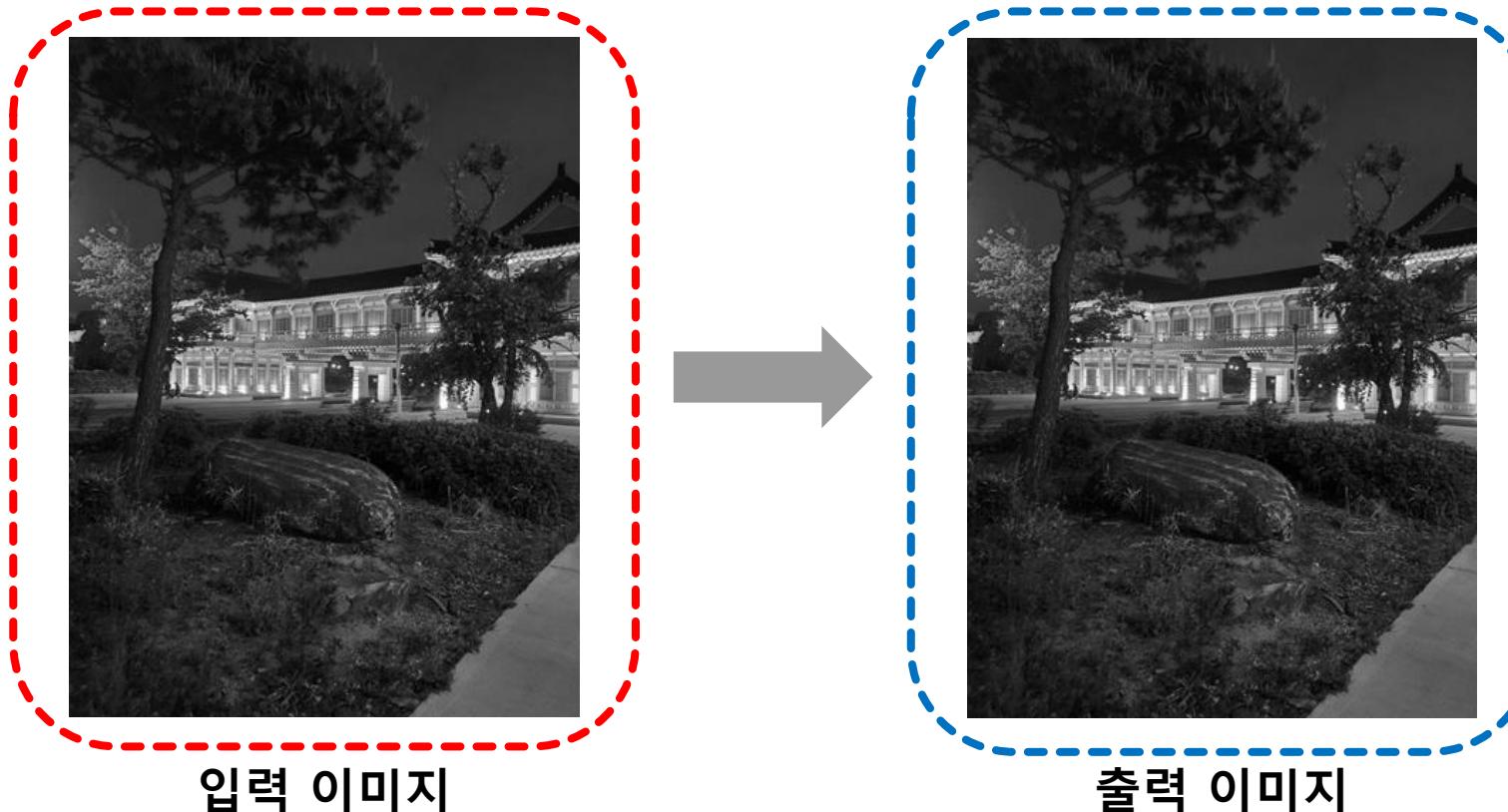
2 엔드인

3 평활화

스트레칭

스트레칭

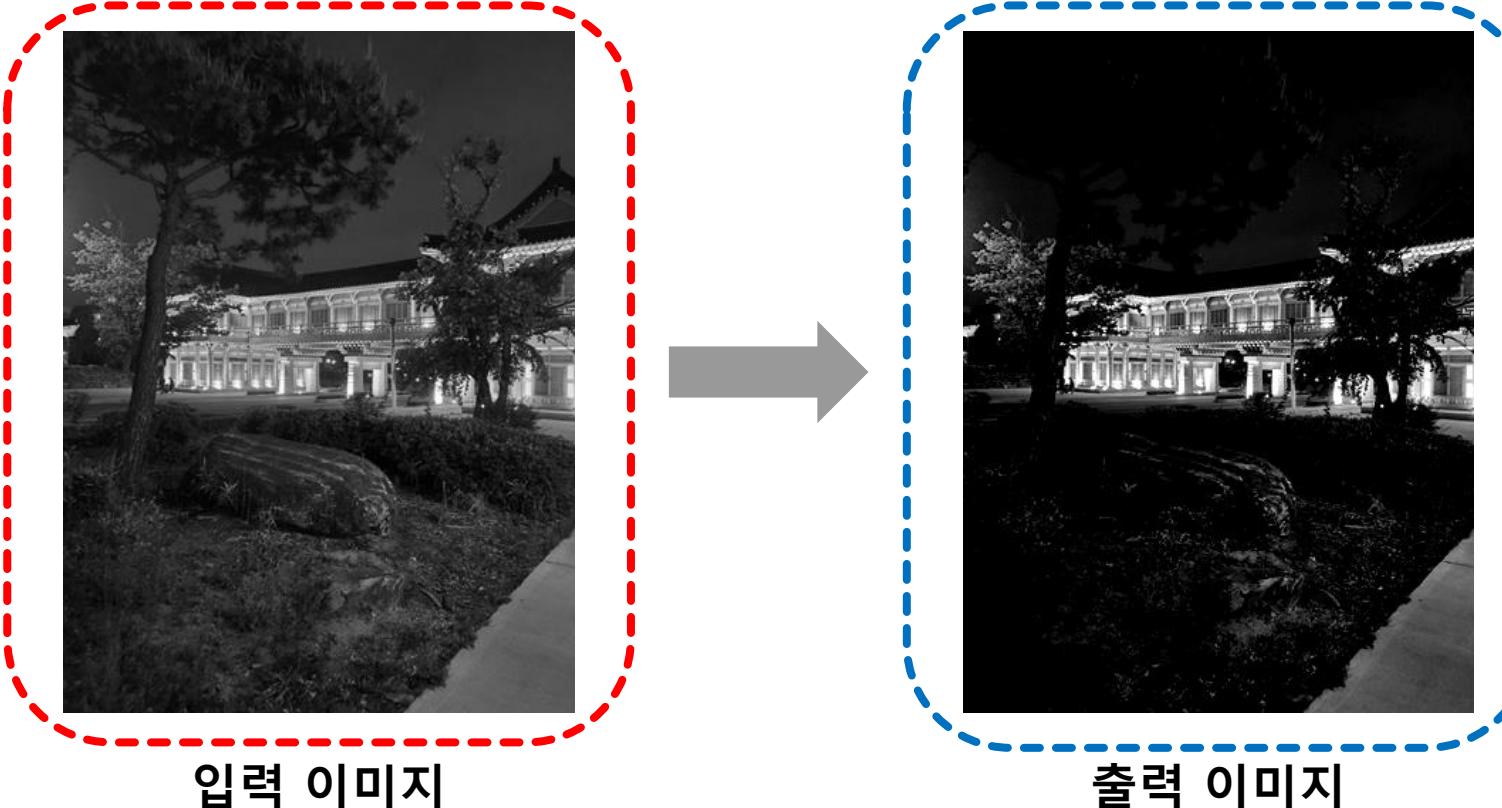
- 집중된 히스토그램을 모든 영역으로 확장하여 명암 대비 개선
- 스트레칭 수행 공식 : $new = (old - low)/(high - low) \times 255$



엔드인

엔드인

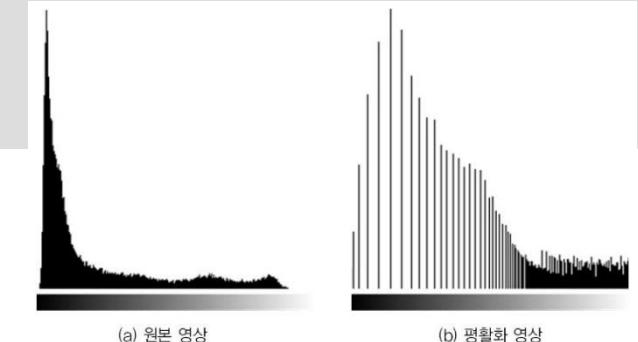
- 히스토그램 스트레칭에서 특정 화소를 흰색 또는 검은색으로 지정
- 기존 스트레칭 함수에서 $high/low$ 에 $+-$ 를 적용



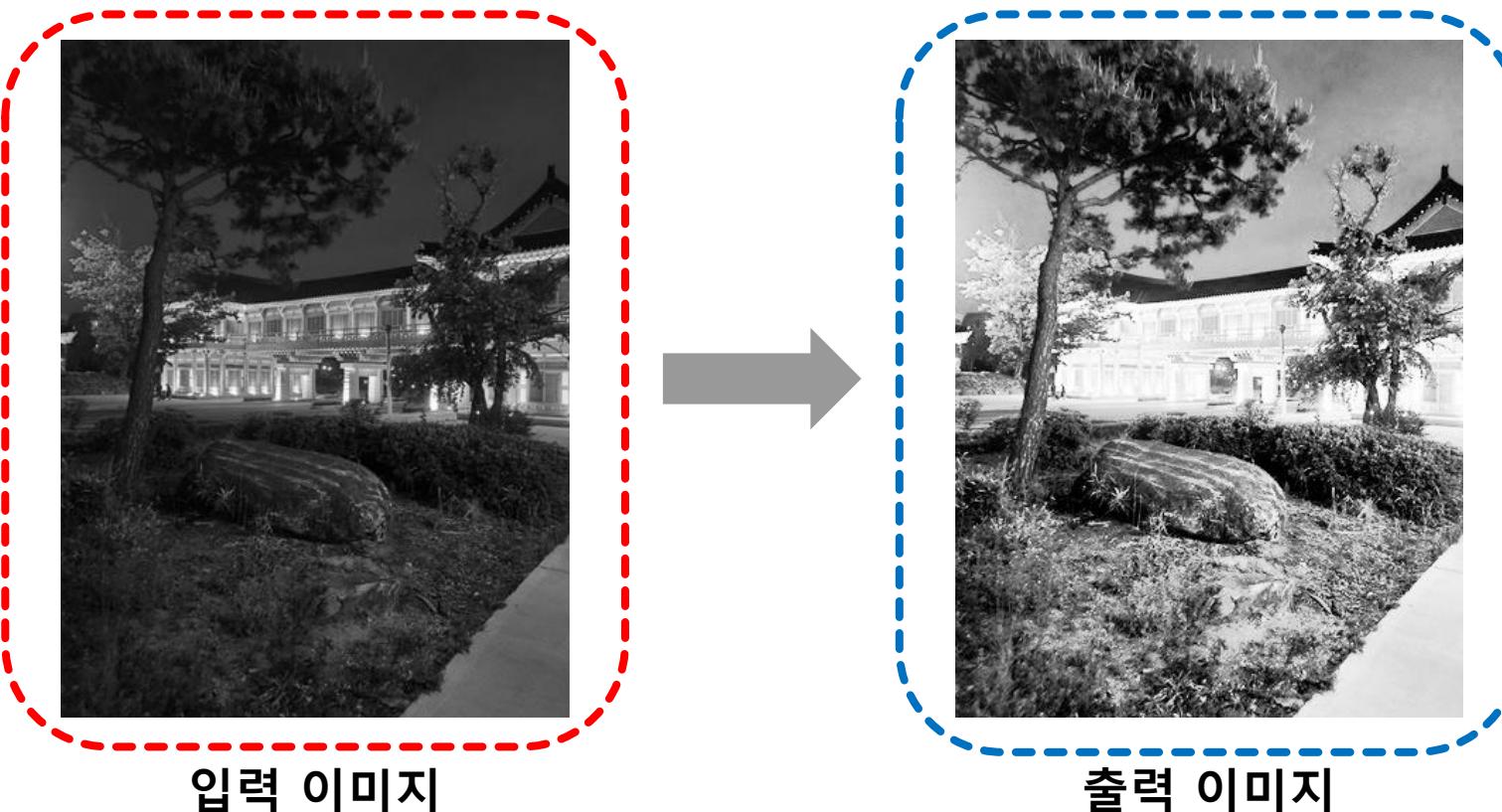
평활화

평활화

- 히스토그램 누적에 따라 분포를 조절하여 명암 대비를 개선



[그림 5-16] 명암 대비가 낮은 영상의 평활화와 평활화한 영상의 히스토그램



05 화소 영역 처리

1 엠보싱

2 블러링

3 샤프닝

4 스무딩

엠보싱

엠보싱

- 회선 처리를 통해 이미지를 양각 형태로 보이게 하는 기술

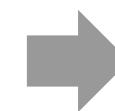
RGB 기반



HSI 기반



입력 이미지



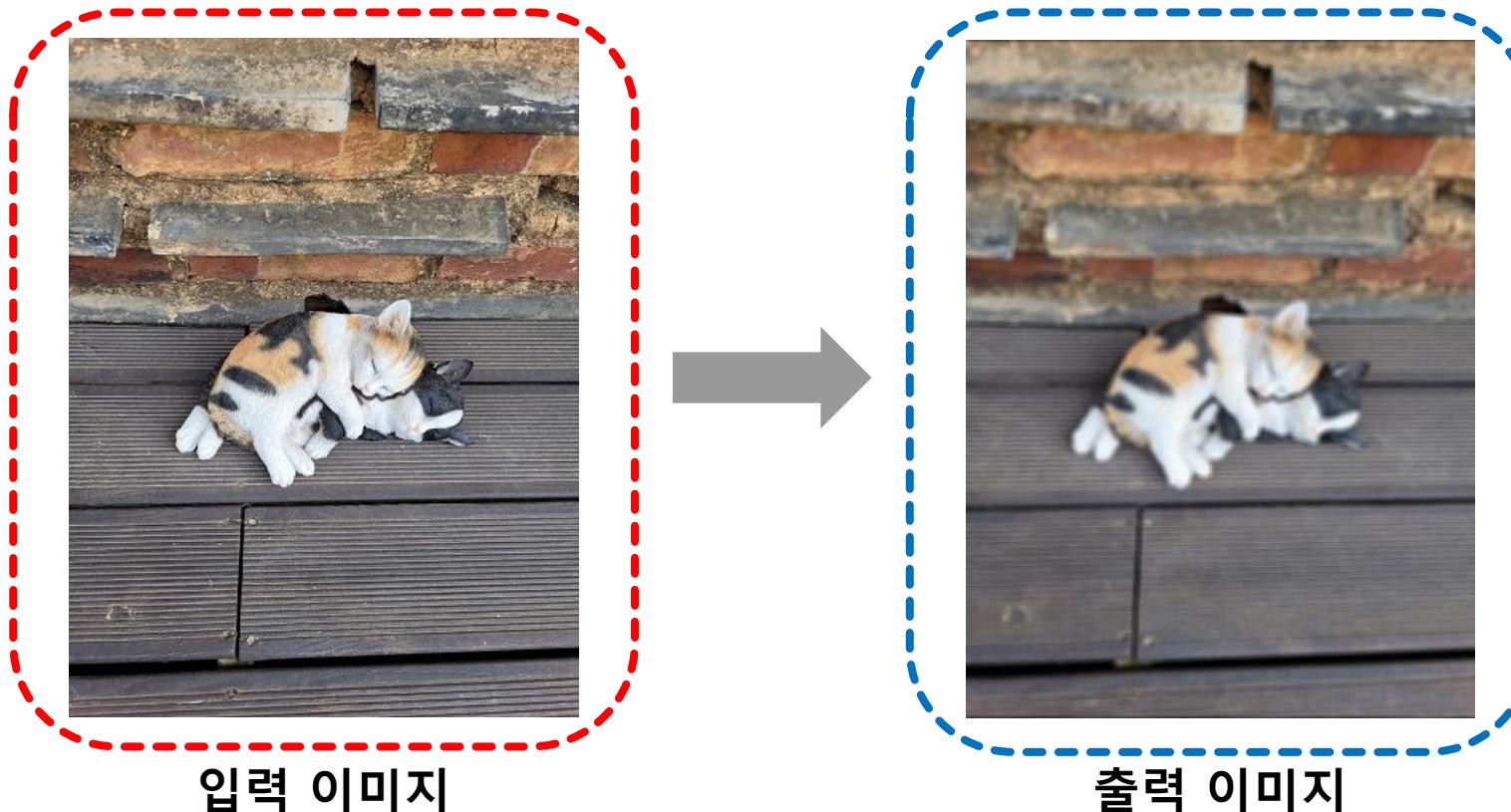
출력 이미지

05 화소 영역 처리

블러링

블러링

- 이미지의 세밀한 부분을 제거하여 흐리게 하는 기술
- 저역 통과 필터를 사용



[그림 6-15] 블러링 마스크의 회선 계수

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

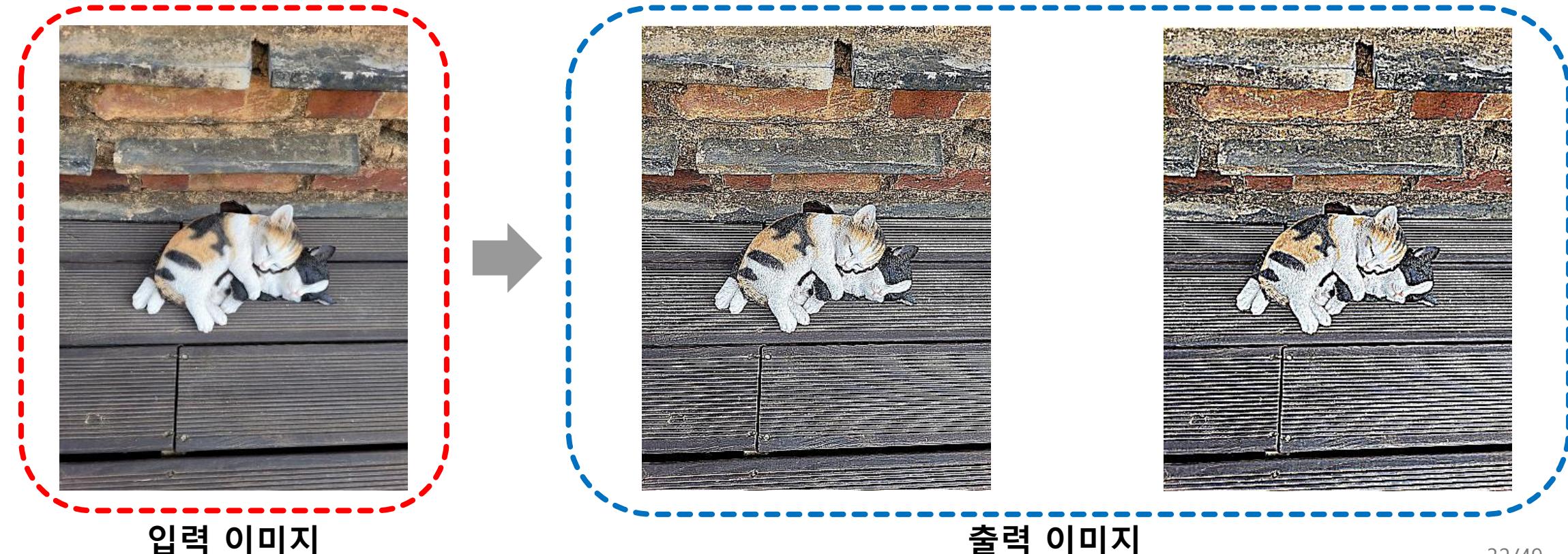
샤프닝

샤프닝

- 블러링과 반대로 이미지의 상세한 부분을 더 선명하게 강조하는 기술
- 고주파 필터 또는 저주파 필터를 사용

고주파

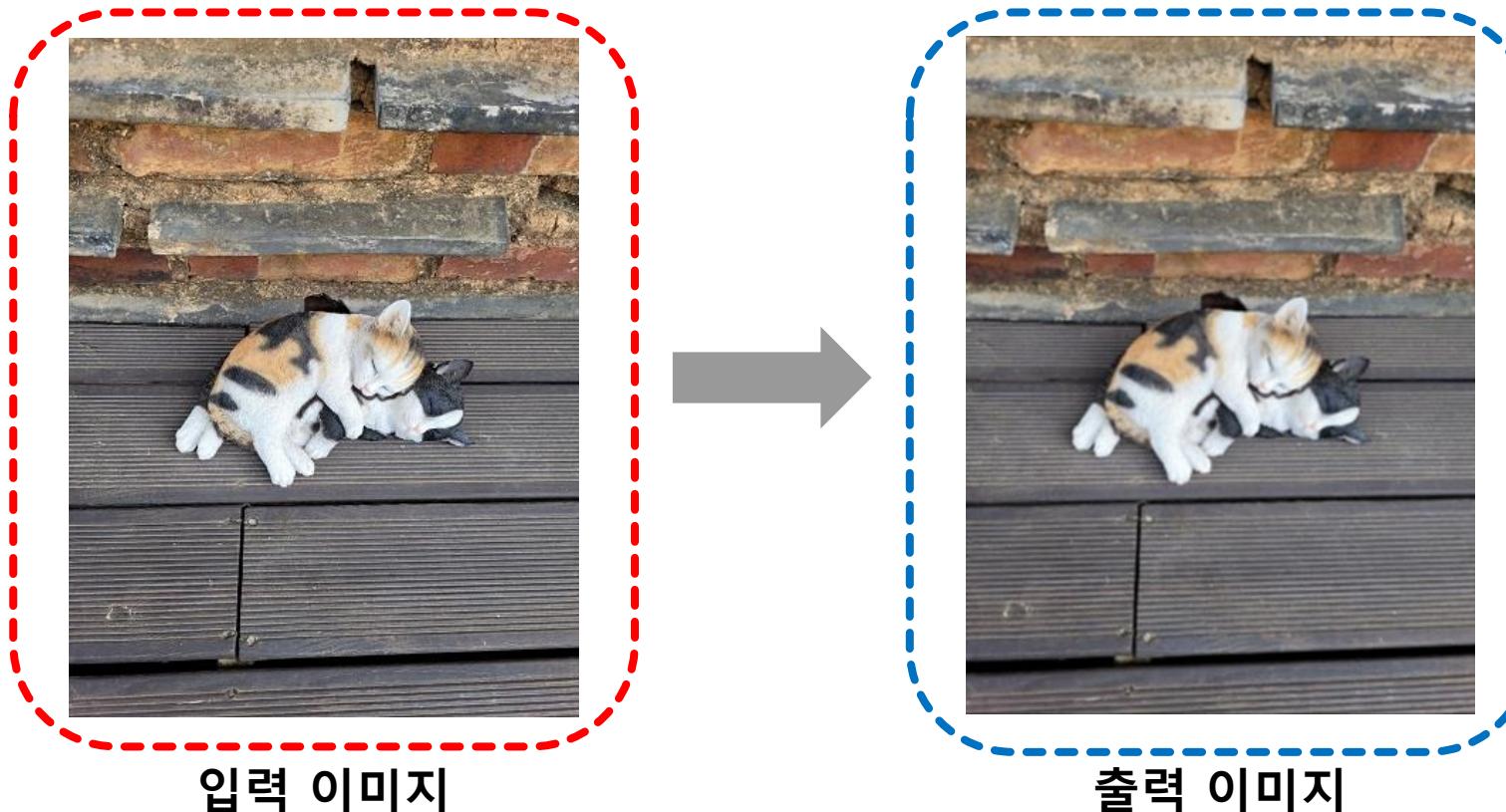
저주파



스무딩

스무딩

- 가우시안 필터를 이용한 블러링
- 2차 가우시안 함수를 이용하여 필터를 생성



3x3 가우시안 필터 예시

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

$$G[x, y] = \frac{e^{\frac{-(x^2+y^2)}{2\sigma^2}}}{2\pi\sigma^2}$$

06 에지 검출

1 이동/차분

2 유사 연산자

3 차 연산자

4 1차 미분 연산자

5 2차 미분 연산자

06 에지 검출

이동/차분

이동/차분

- 가장 기본적인 마스크를 이미지에서 한칸씩 이동하여 계산

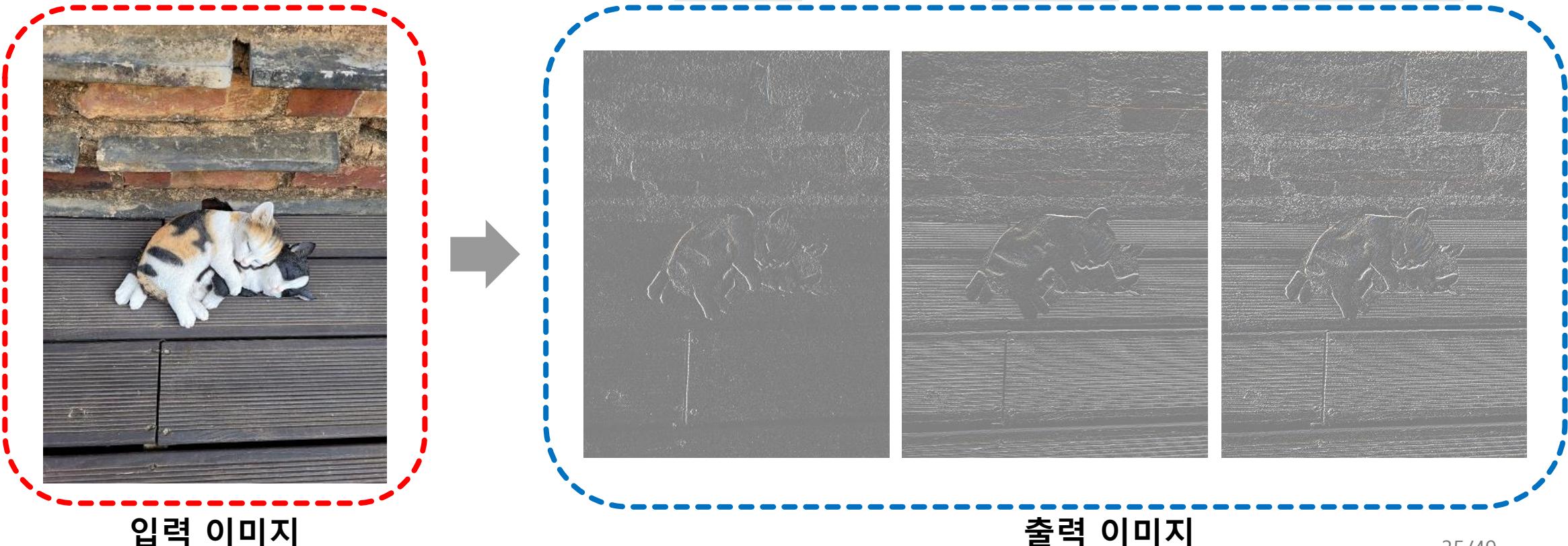
수직 에지 검출 마스크		
0	0	0
-1	1	0
0	0	0

수평 에지 검출 마스크		
0	-1	0
0	1	0
0	0	0

수직

수평

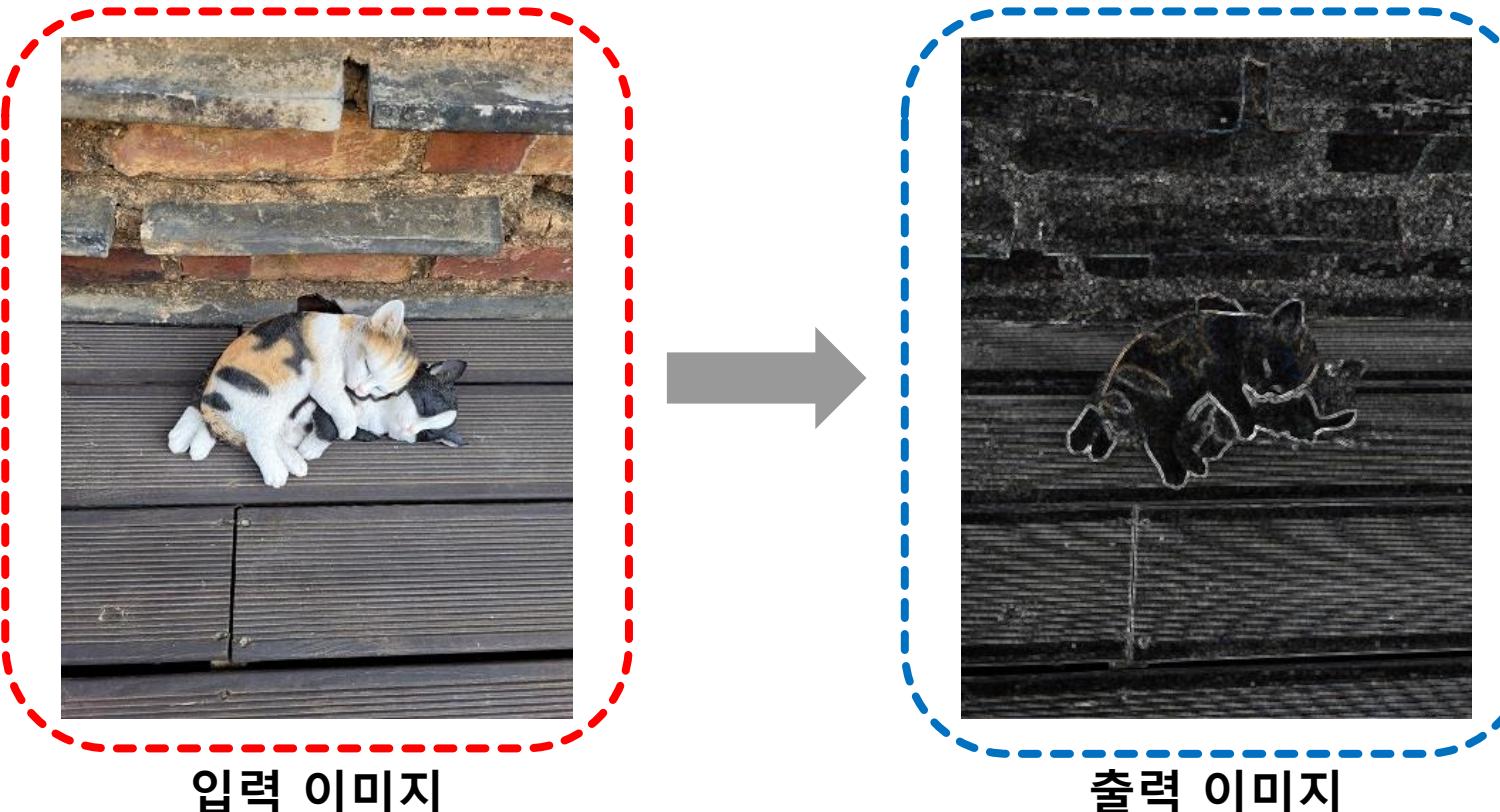
수직+수평



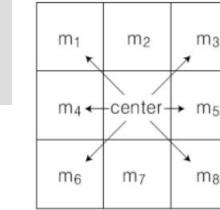
유사 연산자

유사 연산자

- 입력 이미지의 RGB 값의 평균값으로 변경하여 단일색의 이미지로 변환



유사 연산자

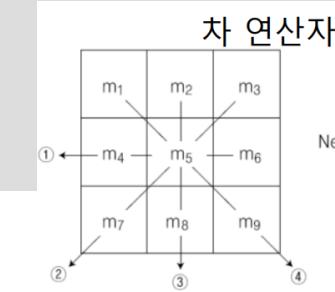


New Pixel = $\max(|\text{center} - m_1| \dots |\text{center} - m_8|)$
총 8번 계산

차 연산자

차 연산자

- 입력 이미지의 RGB 값의 평균값으로 변경하여 단일색의 이미지로 변환



입력 이미지



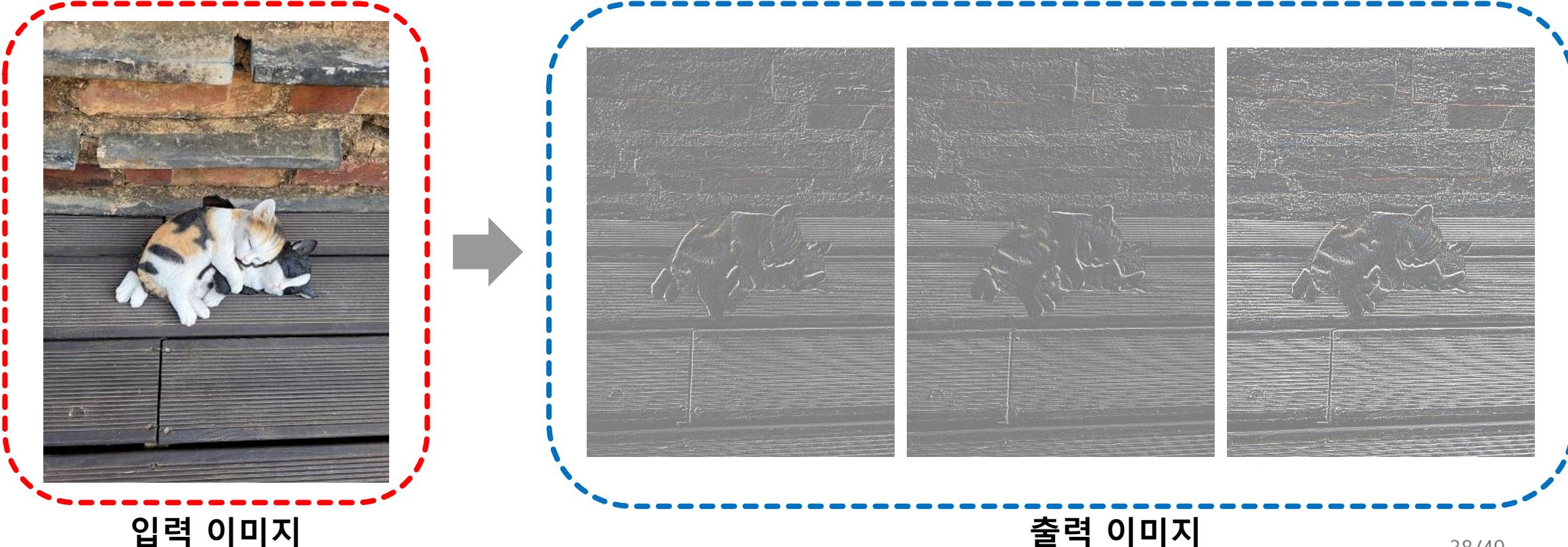
출력 이미지

1차 미분 연산자

로버트 마스크

- 미분 연산을 이용하여 에지를 검출
- 로버트 마스크는 돌출된 값을 잘 평균화

	행 검출 마스크	열 검출 마스크
로버츠	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

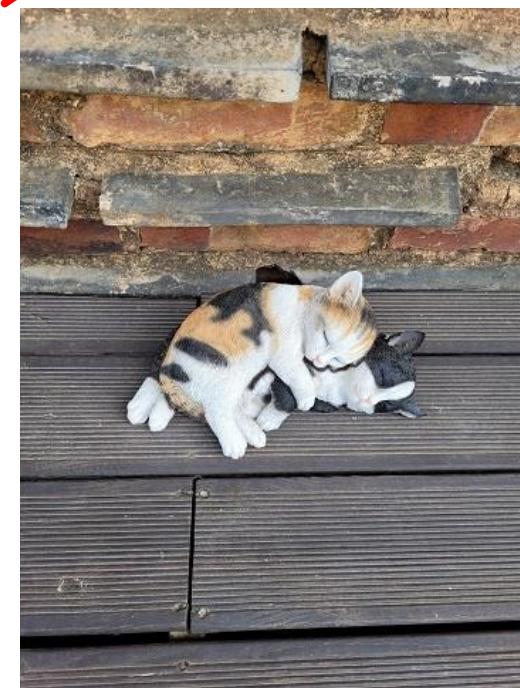


1차 미분 연산자

프리윗 마스크

- 미분 연산을 이용하여 에지를 검출
- 프리윗 마스크는 수평, 수직 에지에 민감

	행 검출 마스크	열 검출 마스크
프리윗	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$



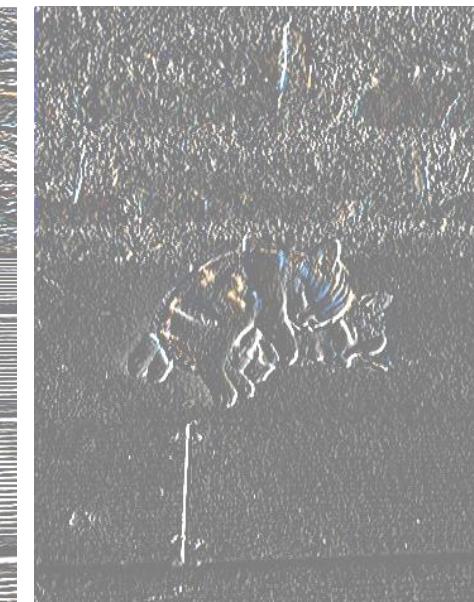
입력 이미지



수직



수평



수직+수평



출력 이미지

1차 미분 연산자

소벨 마스크

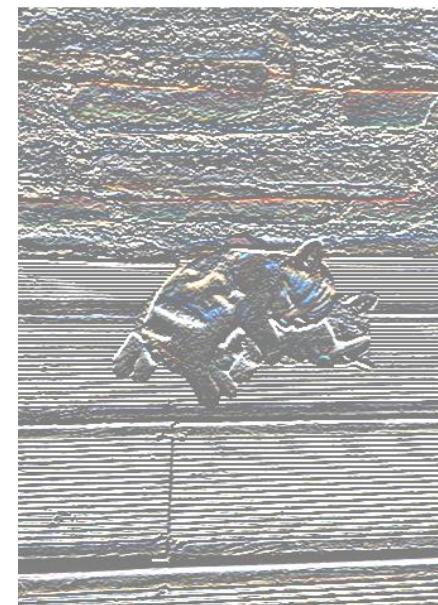
- 미분 연산을 이용하여 에지를 검출
- 소벨 마스크는 대각선 에지에 민감



입력 이미지



수직



출력 이미지

수평



수직+수평

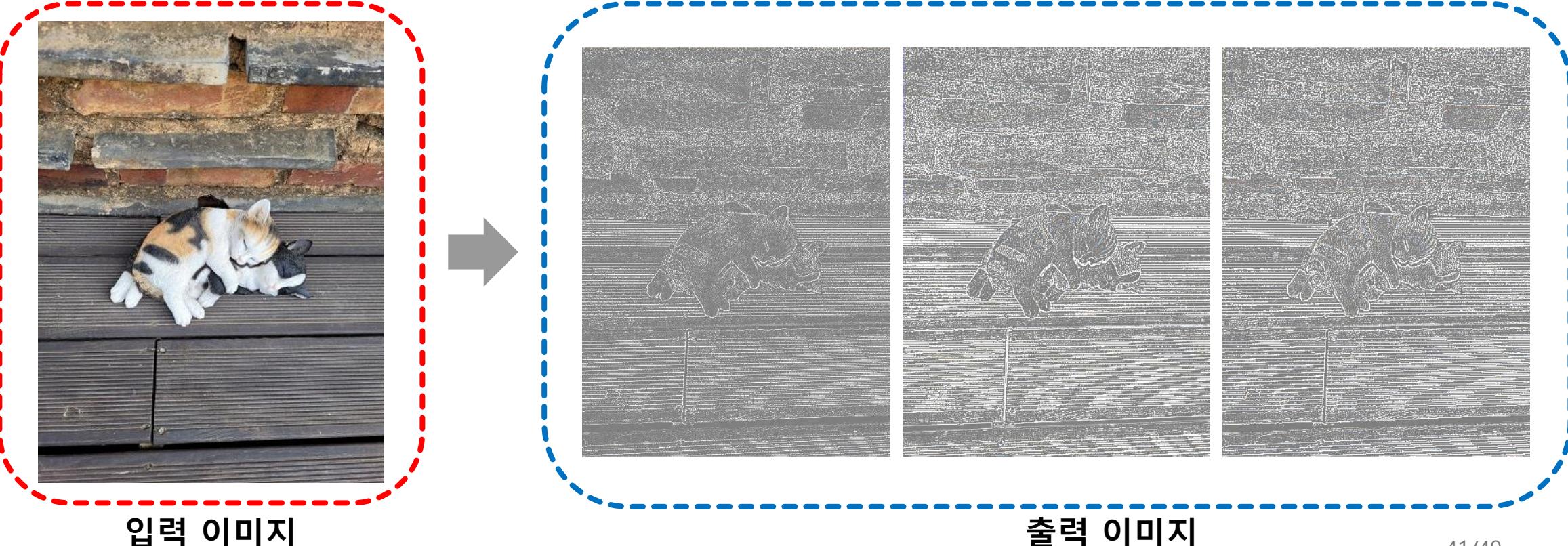


	행 검출 마스크	열 검출 마스크
소벨	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

2차 미분 연산자

라플라시안

- 1차 미분 마스크에 미분을 한번더 수행하여 1차 미분의 민감성 완화
- 모든 방향의 에지 강조

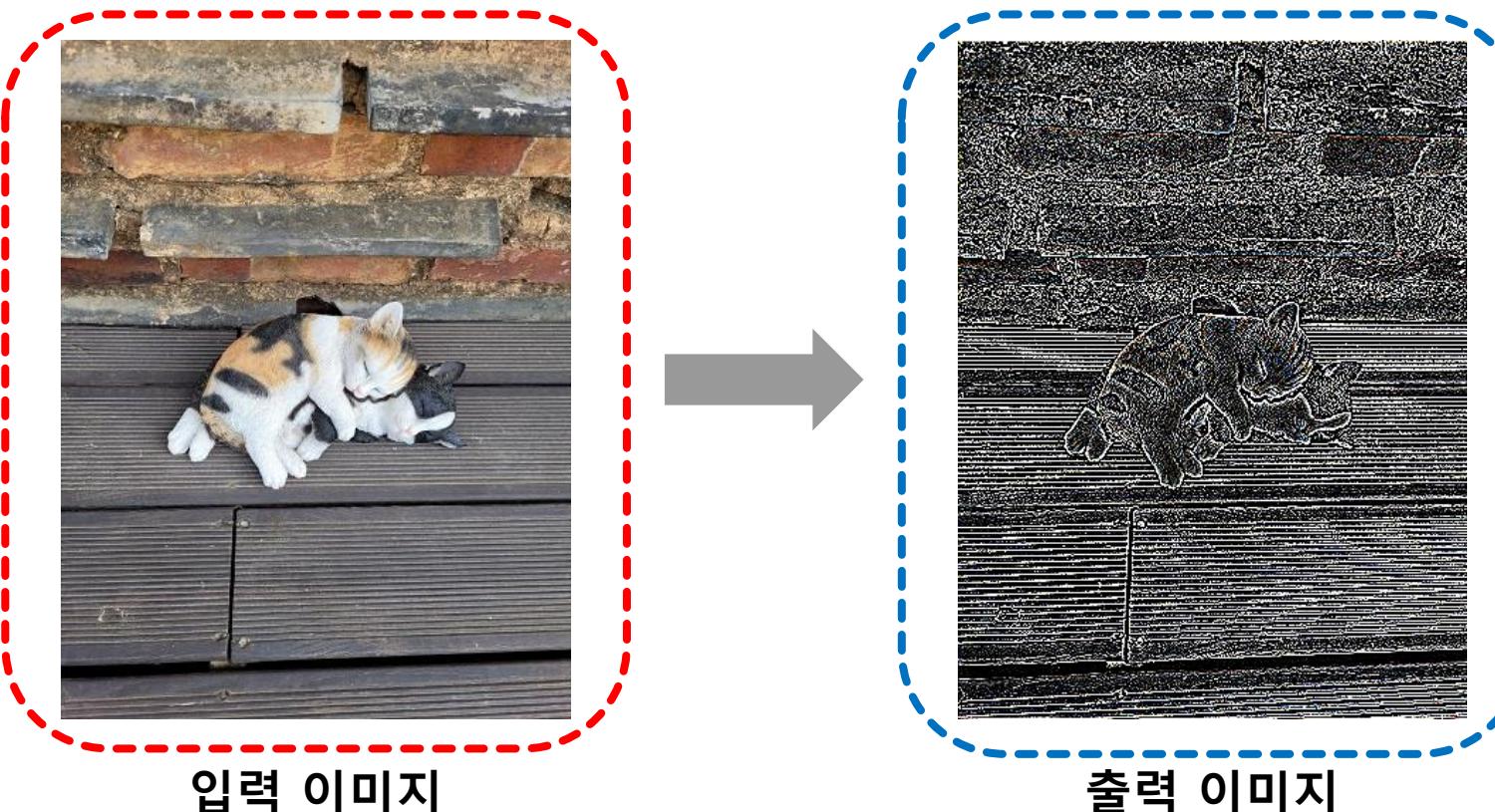


라플라시안 마스크		
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
(a)	(b)	(c)

2차 미분 연산자

LoG

- 1차 미분 마스크에 미분을 한번더 수행하여 1차 미분의 민감성 완화
- 가우시안 스무딩을 통해 잡음 제거



$$\text{LoG}(x, y) = \frac{1}{\pi\sigma^4} \left[1 - \frac{(x^2+y^2)}{2\sigma^2} \right] e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

5x5 LoG 회선 마스크

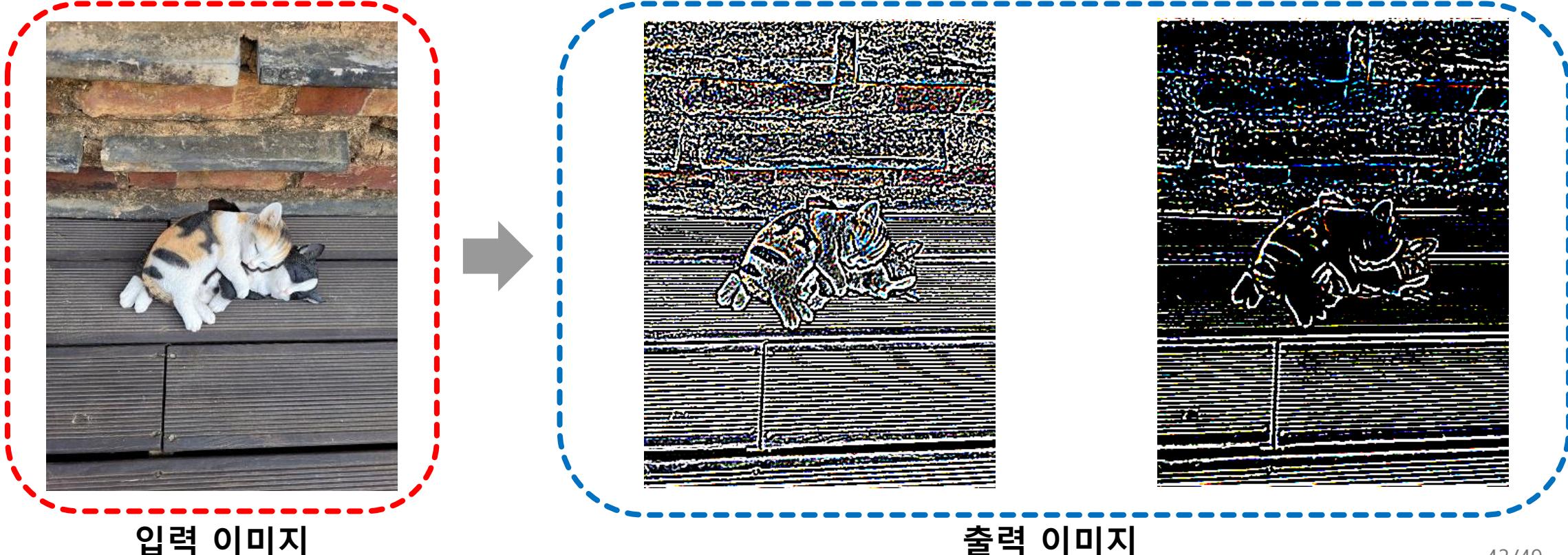
06 에지 검출

2차 미분 연산자

$$DoG(x, y) = \frac{e^{\frac{-(x^2+y^2)}{2\sigma_1^2}}}{2\pi\sigma_1^2} - \frac{e^{\frac{-(x^2+y^2)}{2\sigma_2^2}}}{2\pi\sigma_2^2}$$

DoG

- 1차 미분 마스크에 미분을 한번더 수행하여 1차 미분의 민감성 완화
- LoG 연산의 계산시간 보완



0	0	0	-1	-1	-1	0	0	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	-3	-2	-1	-1	-1	-2	-3	0
-1	-3	-1	9	9	9	-1	-3	-1
-1	-3	-1	9	19	9	-1	-3	-1
-1	-3	-1	9	9	9	-1	-3	-1
0	-3	-2	-1	-1	-1	-2	-3	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	0	0	-1	-1	-1	0	0	0

(a) 7×7 DoG 마스크

[그림 7-20] DoG의 회선 마스크

0	0	0	-1	-1	-1	0	0	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	-3	-2	-1	-1	-1	-2	-3	0
-1	-3	-1	9	9	9	-1	-3	-1
-1	-3	-1	9	19	9	-1	-3	-1
-1	-3	-1	9	9	9	-1	-3	-1
0	-3	-2	-1	-1	-1	-2	-3	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	0	0	-1	-1	-1	0	0	0

(b) 9×9 DoG 마스크

43

07 컬러이미지 효과

1 RGB 변환

2 HSI 변환

3 색상 추출

RGB 변환

RGB 변환

- 입력 이미지의 RGB 개별 영역에 입력 화소값 만큼 더하거나 감산하여 변환

R : +127

G : +127

B : +127

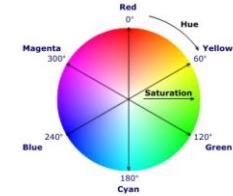
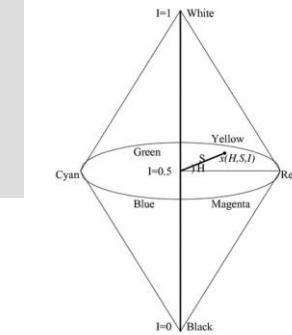


입력 이미지



출력 이미지

07 컬러이미지 효과 HSI 변환



HSI 변환

- 입력 이미지의 HSI 개별 영역에 입력값 만큼 더하거나 감산하여 변환

H : + 180

I : -0.2

S : -50

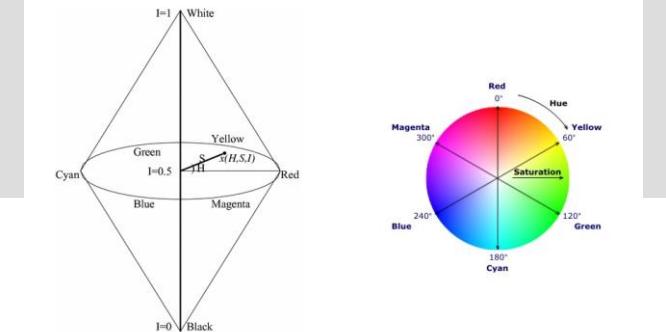


입력 이미지



출력 이미지

07 컬러이미지 효과 색상 추출



색상 추출

- HSI에서 H를 기준으로 특정 영역의 색만 남기고 나머지는 그레이스케일로 변환

H : $60^\circ \sim 180^\circ$

H : $180^\circ \sim 300^\circ$



입력 이미지

출력 이미지

08 마무리

한계점

- 용량이 큰 이미지에 대해서 로딩 속도가 느림
- 이미지 변환 중 이전단계로 되돌아가는 기능 미비

향후 발전 방향

- 코드 최적화를 통한 이미지 로딩 속도 개선
- undo/redo 기능과 기타 영상 처리 알고리즘 추가

느낀점

- 영상 처리 알고리즘의 적용 원리를 배우고 직접 구현하는 과정이 매우 흥미로웠다.
- MFC를 통해 GUI를 구현함으로써 실제 영상 처리 프로그램을 구현했다는 것을 느꼈다.

감사합니다