컴퓨터비전(AI 응용)

디지털 영상 기초

01

수업 목표

디지털 영상의 이해

디지털 영상의 개념과 데이터 표현 방식(행렬, 픽셀)을 확인합니다.

색공간 마스터

RGB, HSL, HSV 등 다양한 색공간을 이해하고 상호 변환할 수 있습니다.

픽셀 연산 활용

밝기·대비 조절 및 임계값 처리 등 픽셀 단위 연산을 실습합니다.

히스토그램 분석

영상의 밝기 분포를 분석하고, 정규화 및 평활화 기법을 적용합니다.

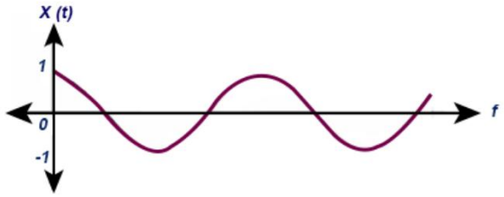
01 신호의 종류와 디지털화

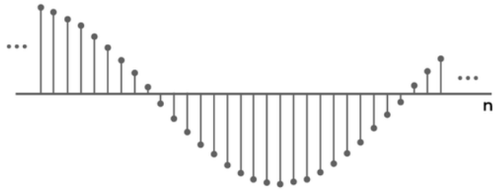
- 아날로그 신호 : 연속적인 신호, 자연계의 신호
- 디지털 신호 : 불연속 신호, 아날로그 신호로부터 '디지털화'를 수행한 결과물
- 아날로그 신호로 잡아낸 '영상' 을 디지털화 하기 위해 아래의 단계가 필요함
  - Sampling : 연속적인 신호를 이산 신호로 변환하는 과정으로, 특정 시간 지점에서 신호 값을 추출하는 것
  - Quantization : 연속적인 신호를 제한된 범위의 이산 값의 집합으로 표현하는 과정

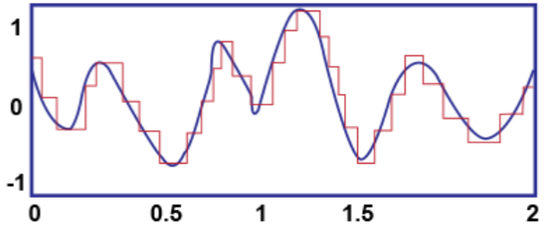
Sampler
샘플링

Quantization
양자화

Encode
인코딩







01

신호의 종류와 디지털화

아날로그 vs 디지털

Analog

아날로그 신호

자연계의 소리나 빛처럼 끊어지지 않고 **연속적(Continuous)**으로 변하는 신호입니다.

예: LP판, 수은 온도계, 사람의 목소리

Digital

디지털 신호

연속적인 아날로그 신호를 0과 1 같은 숫자로 변환하여 **불연속적(Discrete)**으로 끊어서 표현한 신호입니다.

예: MP3, JPG 이미지, 컴퓨터 데이터

디지털화 3단계

1

샘플링 (Sampling)

연속적인 신호를 일정한 간격(시간/공간)으로 잘게 쪼개어 표본을 추출하는 과정

Keywords: 격자 나누기, 픽셀화

2

양자화 (Quantization)

추출된 샘플 값을 정해진 범위의 대표값(정수)으로 근사하여 매핑하는 과정

Keywords: 밝기값 할당 (0~255)

3

인코딩 (Encoding)

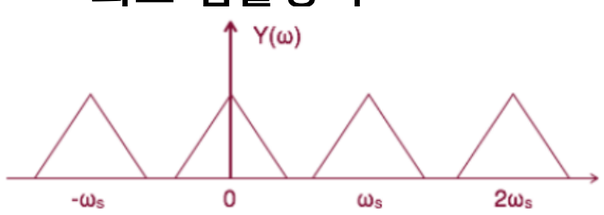
양자화된 값을 컴퓨터가 처리할 수 있는 이진수(0과 1) 형태의 부호로 변환하여 저장

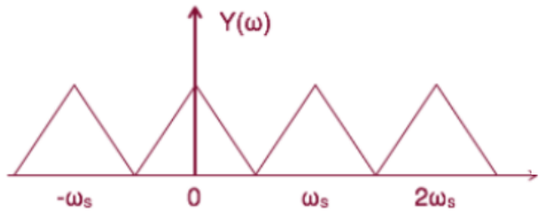
Keywords: 이진수 변환, 압축

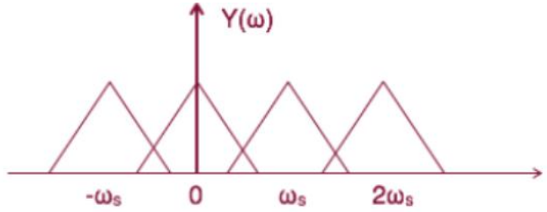
01 신호를 어떻게 '더 잘' 추출할 것인가?

- 더 촘촘하게 추출한다.
- Sampling Rate(Hz) : 아날로그 신호를 디지털 신호로 변환하기 위해 단위 시간당 추출되는 샘플의 수
- 신호의 손실 없이 추출한다.
- Nyquist Rate : 정보 손실 없이 아날로그 신호를 디지털 신호로 포착하기 위해 필요한

'최소 샘플링 수'







01 신호를 어떻게 '더 잘' 추출할 것인가?

Step 01. Quantity

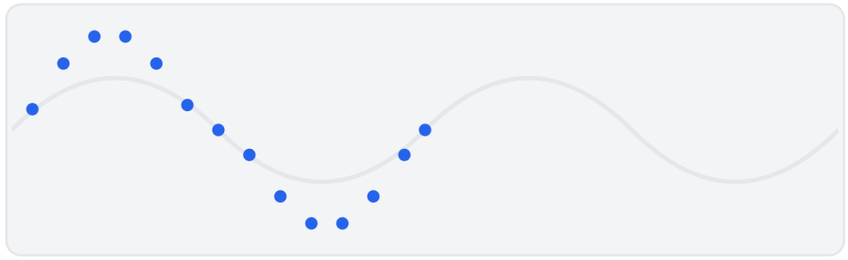
Sampling Rate

샘플링 주파수 (Hz)

"더 촘촘하게 추출한다"

아날로그 신호를 디지털로 변환하기 위해
단위 시간당 추출하는 샘플의 수를 의미합니다.

- 값이 클수록 원본 신호와 유사함 (고해상도)
- 데이터 용량이 커지고 처리 비용이 증가함



Step 02. Quality Rule

Nyquist Rate

나이퀴스트 레이트

"손실 없이 추출하는 최소 기준"

원래 신호를 완벽하게 복원하기 위해 필요한 **최소 샘플링 주파수**의 기준

$$F_s > 2 \times F_{max}$$

샘플링 주파수 > 신호 최고 주파수의 2배

02

색공간 개요

- 색공간(Color Space) : 색을 수치적으로 표현하기 위한 좌표계
- 색을 작업 목적에 따라 더 잘 추출, 표현하기 위해 적합한 색공간을 선택하여야 함

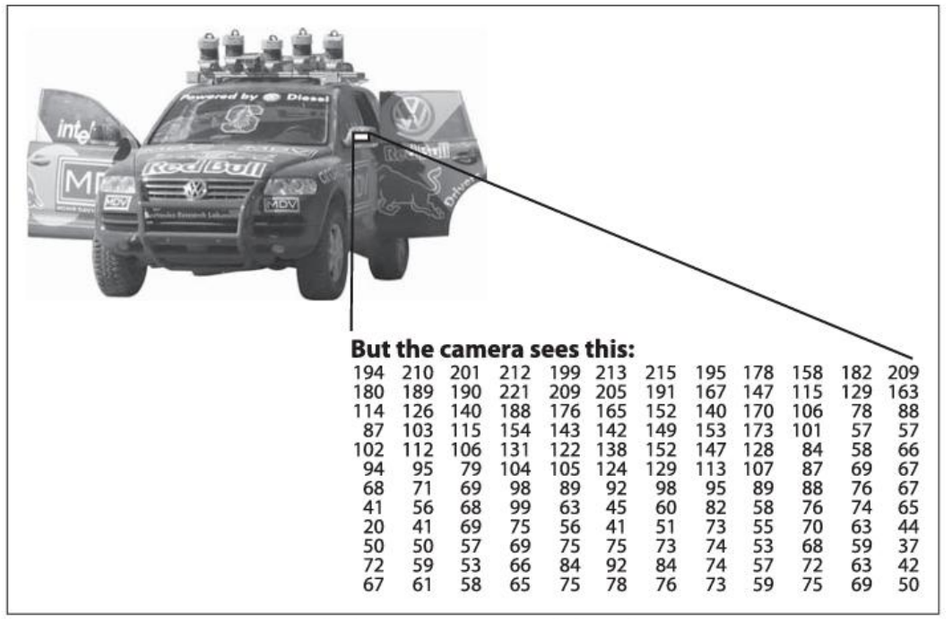
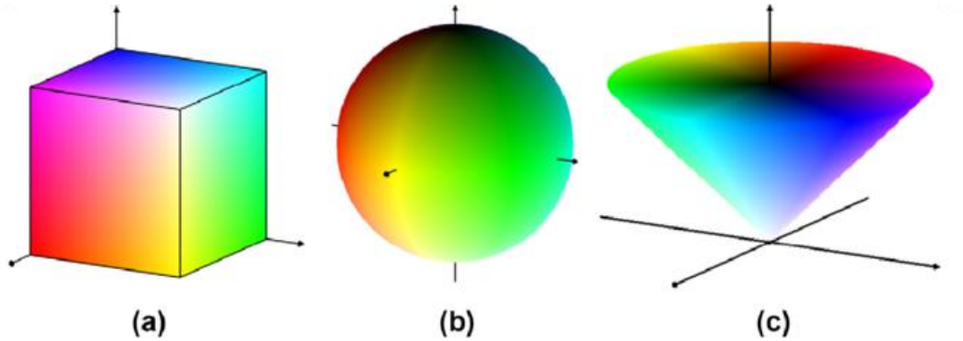


Figure 1-1. To a computer, the car's side mirror is just a grid of numbers



RGB
(Red-Green-Blue)

HSL
(Hue-Saturation-Lightness)

HSV
(Hue-Saturation-Value)

02

색공간

RGB (Red-Green-Blue)

빛의 삼원색을 조합하여 색을 표현하는 방식입니다. 주로 모니터, 카메라 등 디스플레이 장치에서 기본적으로 사용됩니다.

HSL (Hue-Saturation-Lightness)

색상(Hue), 채도(Saturation), 밝기(Lightness)로 구성되어 인간의 색 인지 방식과 유사하며 직관적입니다.

HSV (Hue-Saturation-Value)

HSL과 유사하지만 명도(Value)를 사용하여 조명 변화에 강하고, 컴퓨터 비전에서 색상 기반 분할에 자주 활용됩니다.

02

RGB vs HSV 비교

색공간 정의

Display Standard

RGB

R G B

빛의 삼원색(Red, Green, Blue)을 혼합하여 색을 표현하는 **가산 혼합** 방식
- 모니터나 카메라 등 하드웨어 장치에 주로 사용됩니다.

⚠ 조명 변화에 매우 민감함

Human Perception

HSV

H S V

인간이 색을 인지하는 방식(색상, 채도, 명도)과 유사한 모델입니다.
색상 정보(Hue)가 밝기(Value)와 분리되어 있습니다.

✓ 영상 처리 및 분석에 유리함

주요 특징 비교

표현 방식의 차이

RGB: 세 가지 색을 섞어서 결과물을 만듦 (기계적)
HSV: '어떤 색인지', '얼마나 진한지', '얼마나 밝은지'로 구분 (직관적)

조명 변화에 대한 강인함

RGB: 조명이 바뀌면 R, G, B 값이 모두 변하여 색 인식이 어려움
HSV: 조명이 바뀌어도 V(명도)만 변하고 H(색상)는 유지됨

활용 분야

RGB: 웹 디자인, 이미지 표시, 디스플레이 출력
HSV: 객체 추적, 피부색 검출, 특정 색상 영역 분리

Tip: 파란색 공을 찾으려면 RGB보다 HSV가 훨씬 쉽습니다!

03

픽셀 연산

연산의 개념 및 목적

0~255 범위의 픽셀 값에 수학적 연산을 수행하여 영상의 밝기, 대조, 색상을 조절하거나 필터 효과를 적용합니다.

산술 및 스칼라 연산

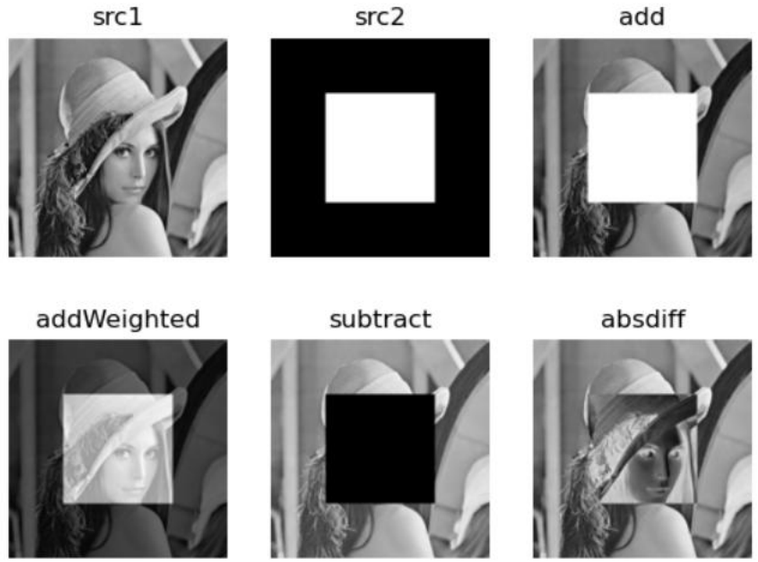
덧셈(밝게), 뺄셈(어둡게/차이), 곱셈, 나눗셈 등 픽셀 값 자체를 수치적으로 변화시키는 연산입니다.

논리 연산 (Bitwise)

AND, OR, XOR, NOT 등 비트 단위 연산을 통해 두 영상을 합성하거나 특정 영역을 반전시킵니다.

마스크 연산 (ROI)

관심 영역(Region of Interest)을 마스크로 지정하여 해당 부분에만 처리를 적용하는 기법입니다.



https://deep-learning-study.tistory.com/115

03

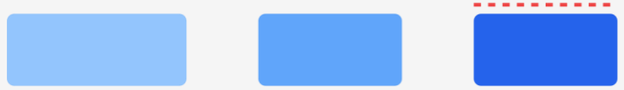
픽셀 연산(덧셈/뺄셈 연산)

덧셈 연산
Brightness Increase

```
cv2.add ( src1, src2 )
```

주요 목적

- 영상의 밝기를 증가시킬 때
- 두 영상을 겹쳐서 합성할 때



포화 연산 (Saturation)

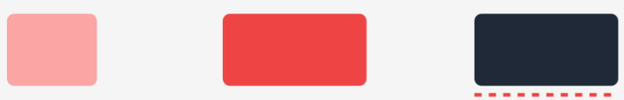
픽셀 값의 합이 255를 초과하면, 값을 **255(흰색)로 고정**합니다. (cf. numpy 덧셈은 256이 되면 0으로 순환됨)

뺄셈 연산
Difference Extraction

```
cv2.subtract ( src1, src2 )
```

주요 목적

- 영상의 밝기를 감소시킬 때
- 두 영상 간의 차이를 구할 때



0으로 고정 (Clamping)

결과가 음수가 되는 경우, 값을 **0(검은색)으로 고정**하여 이미지 데이터 형식을 유지합니다.

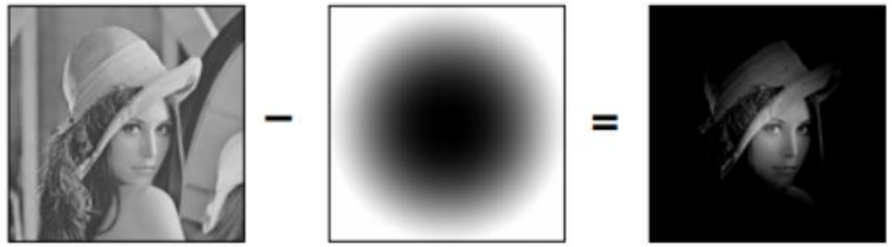
03

픽셀 연산(덧셈/뺄셈 연산)

덧셈 연산(cv2.add)

- **개념 : 두 영상, 혹은 영상과 상수 값을 더함**
- 목적 : 영상의 밝기 증가, 두 영상의 합성
- 특징 : 포화 연산(saturation)을 적용하여, 픽셀 값(0~255)을 넘길 시 255로 고정

뺄셈 연산(cv2.subtract)



- **개념 : 두 영상, 혹은 영상과 상수 값을 뺌**
- 목적 : 영상의 밝기 감소, 영상의 차이 추출
- 특징 : 픽셀 값(0~255)보다 작은 값(음수)인 경우, 0으로 고정

03 픽셀 연산(가중합/절대값 차이)

가중합 연산
Image Blending

```
cv2.addWeighted ( src1, α, src2, β, γ )
```

주요 목적

- 두 영상을 비율(투명도)에 맞춰 합성
- 장면 전환(Transition) 효과 구현

선형 결합 (Linear Combination)

수식: $dst = src1 \times \alpha + src2 \times \beta + \gamma$
보통 $\alpha + \beta = 1$이 되도록 설정하여 평균 밝기를 유지

절대값 차이 연산
Motion Detection

```
cv2.absdiff ( src1, src2 )
```

주요 목적

- 이전 프레임과 현재 프레임의 변화 감지
- CCTV 등에서 움직이는 물체 검출

절대값 계산 (Absolute)

단순 뺄셈은 음수가 되어 0이 되지만, 절대값 차이는 **변화량 그 자체**를 양수로 보존

03

픽셀 연산(가중합/절대값 차이)

가중합 연산(cv2.addWeighted)

- **개념 : 두 영상을 비율에 맞게(가중합) 섞어줌**
- 목적 : 블렌딩, 트랜지션 효과, 합성

절대값 차이 연산(cv2.absdiff)

- **개념 : 두 영상의 절대값 차이 계산**
- 목적 : 영상의 변화 검출, 모션 감지

03

마스크 연산 상세

마스크(Mask)란?

uint8(8비트 단일 채널) 흑백 이미지로, 픽셀 값이 0이면 '무시', 0이 아님은 '적용'을 의미합니다. (입력 영상과 크기가 동일해야 함)

Category	Function Name	Key Feature	Usage Purpose
생성 Mask Creation	cv2.inRange() cv2.threshold()	• 색상 범위(low~high) 지정하여 마스크 생성 • 임계값(Threshold)을 기준으로 이진화	색 기반 분할 배경/전경 분리
도형 Geometry	cv2.rectangle() cv2.circle()	빈 마스크(0) 위에 흰색(255) 도형을 그려서 관심 영역(ROI)을 직접 지정함	사용자 지정 영역
적용 Application	cv2.bitwise_and() cv2.bitwise_not()	• 마스크 영역(흰색)만 남기고 나머지는 제거 • 마스크 영역을 반전(흰색↔검은색)시킴	ROI 추출 선택 영역 반전
활용 Utilization	cv2.inpaint() Morphology / Blur	• 마스크 영역의 픽셀을 주변 색으로 복원 • 노이즈 제거, 구멍 메꾸기, 경계 부드럽게 처리	워터마크 제거 노이즈 정리

03

픽셀 연산(논리연산)

AND 연산
Intersection / Masking

```
cv2.bitwise_and(src1, src2)
```

✓두 영상의 픽셀 값이 모두 0이 아닐 때만 값을 유지

마스크 기반 영역 추출, 교집합 영역 확인

OR 연산
Union / Merging

```
cv2.bitwise_or(src1, src2)
```

✓하나라도 0이 아니면 결과는 켜짐(True)

두 영상 합치기, 합집합 영역 생성

XOR 연산
Difference

```
cv2.bitwise_xor(src1, src2)
```

✓두 값이 서로 다를 때만 켜짐

변화 감지, 서로 다른 영역(차집합) 강조

NOT 연산
Inversion

```
cv2.bitwise_not(src)
```

✓픽셀 값을 반전 (0→255, 255→0)

색상 반전(네거티브 필름), 마스크 뒤집기

03

픽셀 연산(논리연산)

논리 연산(AND, OR, XOR, NOT)

- cv2.bitwise_and(src1, src2)
  - 마스크 기반 영역 추출, 교집합 영역 강조
- cv2.bitwise_or(src1, src2)
  - 두 영상 합치기, 합집합 영역 강조
- cv2.bitwise_xor(src1, src2)
  - 차이나는 영역 강조
- cv2.bitwise_not(src)
  - 색 반전, 배경/전경 반전

AND 논리식

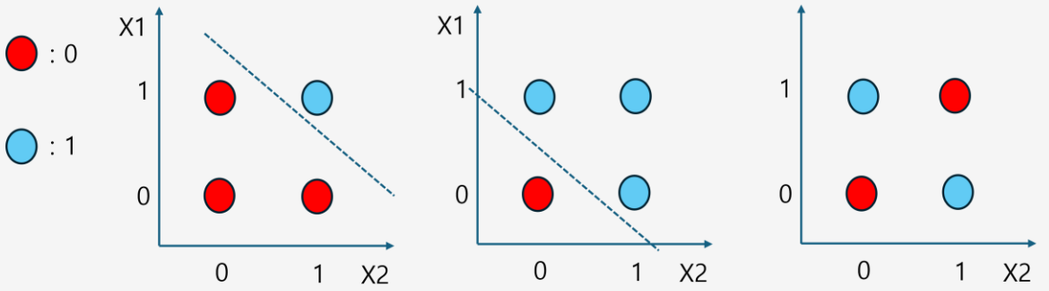
X1	X2	X1 AND X2
0	0	0
0	1	0
1	0	0
1	1	1

OR 논리식

X1	X2	X1 OR X2
0	0	0
0	1	1
1	0	1
1	1	1

XOR 논리식

X1	X2	X1 XOR X2
0	0	0
0	1	1
1	0	1
1	1	0



https://wikidocs.net/192917

04

히스토그램 정규화

개념: 분포 확장

좁은 범위에 몰려있는 영상의 픽셀 값(0~255) 분포를 전체 범위로 골고루 퍼뜨려 데이터의 표현 범위를 확장하는 과정입니다.

목적: 명암 대비 향상

흐릿한 영상의 대비(Contrast)를 높여 더 또렷하고 선명하게 만들거나, 컴퓨터 비전 알고리즘을 위한 전처리 단계로 활용합니다.

원리: Min-Max 변환

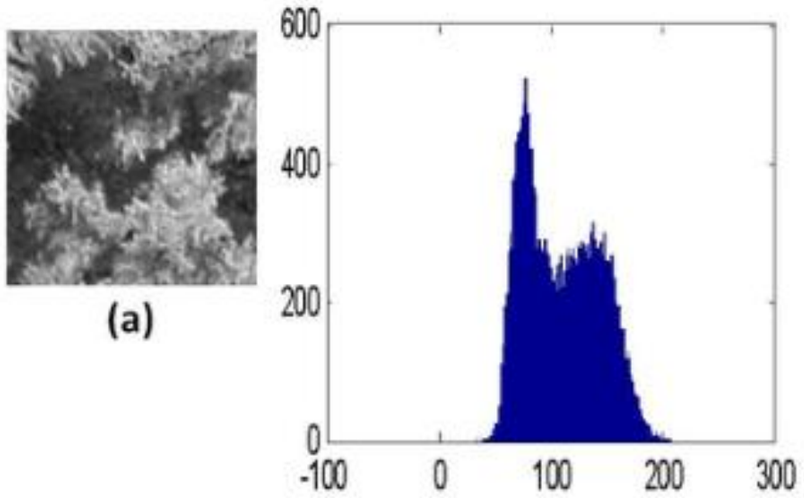
현재 영상의 최소값을 0, 최댓값을 255로 맞추고 나머지 값을 비율에 따라 선형 변환하는 공식을 사용합니다.

$$dst = (src - min) / (max - min) \times 255$$

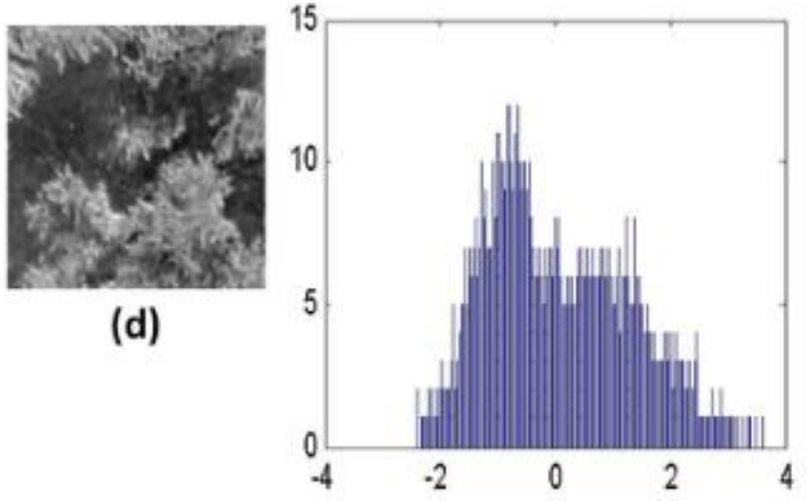
04

히스토그램 정규화

원본 이미지의 히스토그램이
중앙 부분에 집중되어 분포함을 알 수 있음



히스토그램 정규화(z-score)를 통해
특정 파트에 집중된 픽셀 값을 분산시켜 더 또렷한 대비를 얻음



https://medium.com/@rndayala/image-histograms-in-opencv-40ee5969a3b7

04

히스토그램 평활화(Equalization)

개념: 픽셀 분포 균등화

영상의 픽셀 값(0~255) 분포를 전체 범위에 걸쳐 수학적으로 재배치하여 균등하게 만드는 기법입니다.

목적: 명암 대비 향상

한쪽으로 치우친 명암 분포를 넓게 펼쳐, 특히 어두운 영역의 묻혀있던 디테일을 선명하게 살립니다.

예시 1: 역광 사진 보정

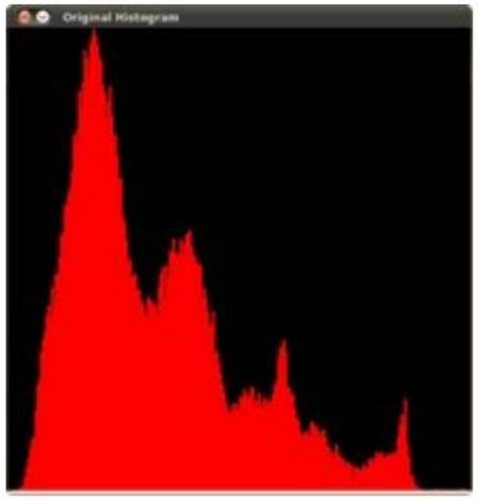
역광으로 인해 피사체가 검게 나온 사진의 명암비를 개선하여 피사체의 모습을 드러냅니다.

예시 2: 안개 낀 풍경 개선

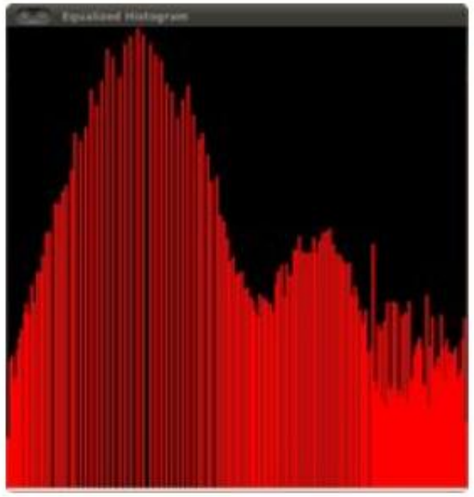
안개로 인해 전체적으로 뿌옇고 대비가 낮은 풍경 사진을 또렷하고 선명하게 복원합니다.

04 히스토그램 평활화(Equalization)

강아지 그림의 픽셀 값을 히스토그램으로 나타내면
히스토그램의 한 쪽에 값이 집중적으로 분포하는 것을 알 수 있음



히스토그램 정규화를 통해
특정 파트에 집중된 픽셀 값을 분산시켜 더 또렷한 대비를 얻음



https://docs.opencv.org/4.x/d4/d1b/tutorial_histogram_equalization.html
https://medium.com/@rndayala/image-histograms-in-opencv-40ee5969a3b7