6주차

Computer Vision

이미지 밝기 조절

Goal

- `cv.convertScaleAbs()`를 사용해 이미지 밝기를 조절하는 방법 학습
- `cv.createTrackbar()`와 `cv.getTrackbarPos()`로 실시간 밝기 조절

Code Demo

```python
import cv2 as cv
import numpy as np

def nothing(x):
    pass

# 이미지 로드
img = cv.imread('lena.jpg')
img = cv.resize(img, (500, 500))
if img is None:
    raise FileNotFoundError('lena.jpg 파일을 찾을 수 없습니다.')

cv.namedWindow('Brightness Adjust')

# 트랙바 생성: 밝기 오프셋 β 범위 [ -100, +100 ]
cv.createTrackbar('Beta', 'Brightness Adjust', 100, 200, nothing)

while True:
    # 트랙바 값(0~200)에서 100을 빼면 [-100~100]
    beta = cv.getTrackbarPos('Beta', 'Brightness Adjust') - 100
    # 밝기 조절: dst = src * 1.0 + beta
    adjusted = cv.convertScaleAbs(img, alpha=1.0, beta=beta)

    cv.imshow('Brightness Adjust', adjusted)
    if cv.waitKey(1) & 0xFF == 27:  # Esc 키로 종료
        break

cv.destroyAllWindows()
```

Explanation

- `alpha`: 대비(contrast) 계수 (여기서는 1.0 고정)
- `beta`: 밝기(brightness) 오프셋
  - `beta < 0` → 어둡게
  - `beta > 0` → 밝게
- 트랙바를 움직여 `beta` 값을 실시간으로 변경하며 이미지 밝기를 확인

Exercises

- 대비(`alpha`)도 트랙바로 조절하도록 확장
- 영상 파일(`.mp4`)에 동일한 밝기 조절 기능 적용

OpenCV: Blurring and Sharpening

Goal

- 이미지 흐림(Blurring)과 선명화(Sharpening) 효과를 적용하는 방법 학습
- OpenCV 필터링 함수 사용: `cv.GaussianBlur()`, `cv.MedianBlur()`, `cv.BilateralFilter()`, `cv.filter2D()`

Blurring (흐림 효과)

흐림 효과는 이미지의 세부 사항을 제거하고 부드러운 결과를 만듭니다. 이를 위해 여러 가지 필터를 사용할 수 있습니다.

✅ Gaussian Blur

Gaussian Blur는 이미지의 각 픽셀을 주변 픽셀과 결합하여 부드러운 흐림 효과를 만듭니다.

Sample

```
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg')
img = cv.resize(img, (500, 500))

# GaussianBlur 적용

blurred = cv.GaussianBlur(img, (15, 15), 0)

# 결과 보여주기

cv.imshow('Gaussian Blurred Image', blurred)
cv.waitKey(0)
cv.destroyAllWindows()
```

Median Blur

Median Blur는 이미지에서 각 픽셀을 주변 픽셀들의 중간값으로 교체하여 흐림 효과를 만듭니다. 일반적으로 노이즈 제거에 유용합니다.

```
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg')
img = cv.resize(img, (500, 500))

# MedianBlur 적용

blurred = cv.medianBlur(img, 15)

# 결과 보여주기

cv.imshow('Median Blurred Image', blurred)
cv.waitKey(0)
cv.destroyAllWindows()
```

Bilateral Filter

Bilateral Filter는 이미지의 엣지를 유지하면서 흐림 효과를 적용합니다. 엣지를 보존하면서 노이즈를 제거하는데 유용합니다.

```
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg')
img = cv.resize(img, (500, 500))

# Bilateral Filter 적용

blurred = cv.bilateralFilter(img, 15, 75, 75)

# 결과 보여주기

cv.imshow('Bilateral Filtered Image', blurred)
cv.waitKey(0)
cv.destroyAllWindows()
```

Sharpening (선명화 효과)

선명화는 이미지의 엣지를 강조하여 더 뚜렷한 결과를 만듭니다. 이를 위해 특정 커널을 사용하여 이미지를 필터링합니다.

✅ Sharpening Mask

이미지에서 엣지를 강조하는 샤프닝 효과를 위해, 고유의 샤프닝 마스크를 사용할 수 있습니다.

Sample

```
import cv2 as cv
import numpy as np

# 이미지 로드

img = cv.imread('lena.jpg')
img = cv.resize(img, (500, 500))

# 샤프닝 마스크 정의

kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])

# 필터 적용

sharpened = cv.filter2D(img, -1, kernel)

# 결과 보여주기

cv.imshow('Sharpened Image', sharpened)
cv.waitKey(0)
cv.destroyAllWindows()
```

Exercises

- 다른 흐림 효과들을 비교하여 적용해보세요.
- 다양한 샤프닝 커널을 사용하여 효과를 실험해보세요.

OpenCV: 이진화

Goal

- 이미지를 이진화하여 배경과 객체를 분리하는 방법 학습
- OpenCV 함수 `cv.threshold()`와 `cv.adaptiveThreshold()` 사용

이진화란?

이진화는 이미지를 흑백(0과 255)으로 변환하는 과정입니다. 주로 객체를 분리하거나 전처리 단계에서 사용됩니다.

✅ 기본 이진화

`cv.threshold()` 함수는 픽셀 값이 특정 임계값을 초과하면 255(흰색), 그렇지 않으면 0(검정색)으로 변환합니다.

Sample

```
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg', cv.IMREAD_GRAYSCALE)

# 기본 이진화

_, binary_img = cv.threshold(img, 127, 255, cv.THRESH_BINARY)

# 결과 보여주기

cv.imshow('Binary Image', binary_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

적응형 이진화

`cv.adaptiveThreshold()` 함수는 지역적인 임계값을 사용하여 이미지의 각 영역을 이진화합니다. 조명이 고르지 않은 이미지에서 효과적입니다.

```
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg', cv.IMREAD_GRAYSCALE)

# 적응형 이진화

binary_img = cv.adaptiveThreshold(img, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C,
cv.THRESH_BINARY, 11, 2)

# 결과 보여주기

cv.imshow('Adaptive Binary Image', binary_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Thresholding 종류

OpenCV에서 제공하는 이진화 방식은 여러 가지가 있습니다.

- **THRESH_BINARY**: 픽셀 값이 임계값을 초과하면 255(흰색), 그렇지 않으면 0(검정색)
- **THRESH_BINARY_INV**: 반대로, 픽셀 값이 임계값을 초과하면 0, 그렇지 않으면 255
- **THRESH_OTSU**: Otsu의 이진화 방법으로, 자동으로 임계값을 설정해줍니다.

Sample

```
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg', cv.IMREAD_GRAYSCALE)

# Otsu's 이진화

_, binary_img = cv.threshold(img, 0, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)

# 결과 보여주기

cv.imshow('Otsu Binary Image', binary_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Exercises

- 다양한 이진화 기법들을 실험하여 가장 적합한 방법을 찾아보세요.
- 다른 이미지를 사용하여 Otsu 이진화 효과를 비교해보세요.

OpenCV: 히스토그램

Goal

- 이미지의 히스토그램을 생성하고 분석하는 방법 학습
- OpenCV 함수 `cv.calcHist()`, `cv.histCalc()`, `cv.equalizeHist()` 사용

히스토그램이란?

히스토그램은 이미지의 픽셀 값 분포를 시각적으로 나타낸 것입니다. 각 픽셀 값의 빈도수를 계산하여 이미지의 밝기나 대비를 분석할 수 있습니다.

✅ 히스토그램 계산

`cv.calcHist()` 함수는 이미지의 히스토그램을 계산합니다. 이 함수는 각 픽셀 값의 발생 빈도를 계산하고 히스토그램을 반환합니다.

Sample

```python
import cv2 as cv
import matplotlib.pyplot as plt

# 이미지 로드
img = cv.imread('lena.jpg', cv.IMREAD_GRAYSCALE)

# 히스토그램 계산
hist = cv.calcHist([img], [0], None, [256], [0, 256])

# 히스토그램을 막대 그래프로 표시
plt.bar(range(256), hist.ravel(), width=1)
plt.title('Histogram (Bar)')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()
```

히스토그램 균등화

히스토그램 균등화는 이미지의 대비를 향상시키는 기법입니다. `cv.equalizeHist()` 함수는 이미지의 히스토그램을 평탄하게 만들어서 더 균일한 밝기 분포를 만듭니다.

```
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg', cv.IMREAD_GRAYSCALE)

# 히스토그램 균등화

equalized_img = cv.equalizeHist(img)

# 결과 보여주기

cv.imshow('Equalized Image', equalized_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

컬러 이미지의 히스토그램

컬러 이미지는 각각의 채널(RGB)에 대해 히스토그램을 계산할 수 있습니다. 아래는 RGB 채널 각각에 대해 히스토그램을 계산하는 예시입니다.

```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np

# 이미지 로드
img = cv.imread('lena.jpg')

# 채널 분리
channels = cv.split(img)

# 컬러 설정
colors = ('b', 'g', 'r')

# 그래프 크기 설정
plt.figure(figsize=(10, 4))

# 각 채널에 대해 히스토그램 계산 및 bar plot
for i, color in enumerate(colors):
    hist = cv.calcHist([channels[i]], [0], None, [256], [0, 256])
    plt.bar(np.arange(256), hist.ravel(), color=color, alpha=0.4, label=f'{color.upper()} channel', width=1)

# 결과 플로팅
plt.title('RGB Histogram (Bar)')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

Exercises

- 컬러 이미지에서 각 채널의 히스토그램을 비교하여 어떤 채널이 가장 밝거나 어두운지 분석해보세요.
- 여러 이미지를 대상으로 히스토그램 균등화 효과를 비교해보세요.

OpenCV: 침식(Erosion)과 팽창(Dilation)

Goal

- 영상 처리에서 **침식(Erosion)** 과 **팽창(Dilation)** 개념 학습
- OpenCV의 `cv.erode()` 와 `cv.dilate()` 함수 사용법 익히기

침식 (Erosion)

침식은 이미지에서 객체의 외곽을 깎아내는 연산입니다. 주로 노이즈 제거, 객체 분리 등에 사용됩니다.

- 밝은(흰색) 영역이 줄어듭니다.
- 커널이 적용된 영역 내에 모든 픽셀이 1일 때만 중심 픽셀이 유지됩니다.

Sample

```
import cv2 as cv
import numpy as np

# 이미지 로드 (이진 이미지)

img = cv.imread('i.png', 0)

# 커널 생성

kernel = np.ones((5,5), np.uint8)

# 침식 적용

erosion = cv.erode(img, kernel, iterations=1)

# 결과 보여주기

cv.imshow('Original', img)
cv.imshow('Eroded', erosion)
cv.waitKey(0)
cv.destroyAllWindows()
```

팽창 (Dilation)

팽창은 이미지의 객체 외곽을 확장시키는 연산입니다. 객체를 강조하거나 구멍을 메우는 데 사용됩니다.

- 밝은(흰색) 영역이 확장됩니다.
- 커널 영역 내에 하나라도 1이 있다면 중심 픽셀을 1로 설정합니다.

Sample

```
import cv2 as cv
import numpy as np

# 이미지 로드 (이진 이미지)

img = cv.imread('i.png', 0)

# 커널 생성

kernel = np.ones((5,5), np.uint8)

# 팽창 적용

dilation = cv.dilate(img, kernel, iterations=1)

# 결과 보여주기

cv.imshow('Original', img)
cv.imshow('Dilated', dilation)
cv.waitKey(0)
cv.destroyAllWindows()
```

침식과 팽창의 응용

- **노이즈 제거**: 침식 후 팽창 (Opening)
- **구멍 채우기**: 팽창 후 침식 (Closing)
- **객체 경계 추출**: 팽창과 침식 결과의 차이로 객체 외곽 강조

Sample

```python
import cv2 as cv
import numpy as np

# 이미지 로드 (이진 이미지)
img = cv.imread('i.png', 0)
img = cv.resize(img, (200, 300))

# 커널 생성
kernel = np.ones((5,5), np.uint8)

# Opening (침식 -> 팽창)
opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)

# Closing (팽창 -> 침식)
closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)

# 경계 추출
gradient = cv.morphologyEx(img, cv.MORPH_GRADIENT, kernel)

cv.imshow('Original', img)
cv.imshow('opening', opening)
cv.imshow('closing', closing)
cv.imshow('gradient', gradient)
cv.waitKey(0)
cv.destroyAllWindows()
```

Exercises

- 다양한 커널 크기와 반복 횟수로 실험해보세요.
- 침식과 팽창을 조합하여 객체 분할, 잡음 제거 효과를 관찰해보세요.
- Opening과 Closing의 차이를 시각적으로 비교해보세요.

OpenCV: Canny 경계선 검출

Goal

- Canny Edge Detection 알고리즘의 원리를 이해하고 OpenCV로 구현해보기
- 경계선(Edge)을 효과적으로 추출하는 방법 학습

Canny Edge Detection이란?

Canny는 **정확한 경계선 검출**을 위한 알고리즘입니다.

Sample Code

```python
import cv2 as cv

# 이미지 로드

img = cv.imread('lena.jpg')
img = cv.resize(img, (500, 500))
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

# GaussianBlur로 노이즈 제거
blur = cv.GaussianBlur(gray, (5, 5), 1.4)

# Canny 경계선 검출
edges = cv.Canny(blur, 50, 150)

# 결과 출력

cv.imshow('Canny Edge Detection', edges)
cv.waitKey(0)
cv.destroyAllWindows()
```

Threshold 설정

- `cv.Canny()`의 두 번째, 세 번째 인자 = **하한 임계값**, **상한 임계값**
- 픽셀의 그래디언트가 상한 이상 → 엣지로 확정
- 하한 미만 → 무시
- 하한과 상한 사이 → 연결된 경우에만 엣지로 간주

장점

- 노이즈에 강하고, 얇은 엣지를 정확하게 검출
- 다른 엣지 검출보다 결과 품질 우수

Exercises

- 임계값을 조절하며 엣지 결과 비교해보기
- Sobel, Laplacian 등 다른 엣지 검출 방법과 비교 실습

영상 압축: I-Frame, P-Frame, B-Frame

I-Frame (Intra-coded Frame)

- **완전한 이미지 정보**를 갖는 프레임
- 다른 프레임과 독립적으로 디코딩 가능
- 주로 **GOP(Group of Pictures)** 의 시작에 위치
- 압축률은 낮지만, 랜덤 액세스나 탐색에 필수

P-Frame (Predictive-coded Frame)

- 이전의 I-Frame이나 P-Frame을 참조하여 인코딩
- **시간적 예측(Temporal Prediction)** 을 통해 차이 만 저장
- 압축률이 높고, 디코딩 시 참조 프레임 필요

B-Frame (Bi-directional Predictive Frame)

- **양방향 참조**를 사용하는 프레임
- 이전 및 이후의 I/P 프레임을 동시에 참조해 예측
- 가장 높은 압축률을 제공하지만, 디코딩은 복잡함

프레임 간 관계

- 일반적인 GOP 구조: `I B B P B B P B B I ...`
- 디코딩 순서: I → P → B (B는 앞뒤 프레임이 있어야 복원 가능)

비교 표

항목	I-Frame	P-Frame	B-Frame
참조 여부	참조 안함	이전 프레임 참조	이전 + 이후 프레임 참조
압축률	낮음	중간	가장 높음
디코딩 난이도	쉬움	보통	어려

OpenCV: MP4 영상 재생

Goal

- OpenCV를 이용하여 `.mp4` 영상을 불러오고 재생하는 방법 학습
- `cv.VideoCapture()`와 `cv.imshow()` 함수 사용

MP4 영상 재생 기본

OpenCV의 `VideoCapture` 객체를 사용하면 동영상 파일을 프레임 단위로 읽고 처리할 수 있습니다.

```
import cv2 as cv

# 동영상 파일 열기
cap = cv.VideoCapture('video1.mp4')

# 파일이 열렸는지 확인
if not cap.isOpened():
    print("Error: 동영상을 열 수 없습니다.")
    exit()

# 프레임 반복 재생
while True:
    ret, frame = cap.read()

    # 프레임 읽기 실패 시 종료
    if not ret:
        break

    # 프레임 출력
    cv.imshow('MP4 Video', frame)

    # 'q' 키 누르면 종료
    if cv.waitKey(30) & 0xFF == ord('q'):
        break

# 종료
cap.release()
cv.destroyAllWindows()
```

주요 함수 설명

- `cv.VideoCapture(path)` : 동영상 파일 열기
- `cap.read()` : 다음 프레임 읽기 (성공 여부, 프레임 반환)
- `cv.imshow()` : 프레임 출력

Tips

- `cv.waitKey()`의 값이 작을수록 재생 속도가 빨라짐
- 웹캠 입력도 `cv.VideoCapture(0)`으로 처리 가능
- 동영상 프레임을 처리하거나 저장하는 것도 가능

Exercises

- 영상 속 객체를 추적하는 기능을 추가해보세요.
- 영상에서 특정 프레임만 추출하여 저장해보세요.
- 키 입력에 따라 영상 속도나 밝기를 조절해보세요.

색상 추출

Goal

- 이미지를 한 색상 공간에서 다른 색상 공간으로 변환하는 방법. (예: `BGR` ↔ `Gray`, `BGR` ↔ `HSV` 등)
- 추가로, 비디오에서 특정 색상의 객체만 추출하는 간단한 애플리케이션을 만들어 봅니다.
- 사용 함수: `cv.cvtColor()`, `cv.inRange()` 등.

Changing Color-space

OpenCV에는 150개 이상의 색상 공간 변환 메서드가 있지만, 여기서는 가장 널리 쓰이는 두 가지를 다룹니다:

- BGR ↔️ Gray
- BGR ↔️ HSV

```
# 예: BGR → Gray, BGR → HSV 변환
gray  = cv.cvtColor(input_image, cv.COLOR_BGR2GRAY)
hsv   = cv.cvtColor(input_image, cv.COLOR_BGR2HSV)
```

HSV 범위

- **Hue**: `[0, 179]`
- **Saturation**: `[0, 255]`
- **Value**: `[0, 255]`

다른 소프트웨어와 비교 시 범위가 다를 수 있으므로 **정규화**가 필요.

Object Tracking

HSV 색상 공간에서는 특정 색을 표현하고 추출하기가 더 쉽다.
예시: 파란색 객체를 추적

1. 비디오의 각 프레임 읽기
2. BGR → HSV 변환
3. `cv.inRange()`로 파란색 범위(threshold) 지정
4. `cv.bitwise_and()`로 마스크 처리하여 파란 객체만 추출

```python
import cv2 as cv
import numpy as np

# 비디오 파일 경로 지정
video_path = 'video1.mp4'
cap = cv.VideoCapture(video_path)

if not cap.isOpened():
    print(f"Error: 비디오 파일을 열 수 없습니다: {video_path}")
    exit()

while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv.resize(frame, (400, 300))

    # BGR → HSV
    hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

    # 파란색 범위 정의
    lower_blue = np.array([110, 50,  50])
    upper_blue = np.array([130, 255, 255])

    # 파란색만 추출
    mask = cv.inRange(hsv, lower_blue, upper_blue)
    res  = cv.bitwise_and(frame, frame, mask=mask)

    # 원본·마스크·결과 창에 표시
    cv.imshow('Original', frame)
    cv.imshow('Mask',     mask)
    cv.imshow('Result',   res)

    # 'Esc' 키를 누르면 조기 종료
    if cv.waitKey(50) & 0xFF == 27:
        break

cap.release()
cv.destroyAllWindows()
```

How to find HSV values to track?

특정 BGR 색상의 HSV 값을 알아내려면, BGR 픽셀을 `cv.cvtColor()`에 넘기면 됩니다:

```
import cv2 as cv
import numpy as np

green = np.uint8([[[0, 255, 0]]]) # BGR 형식
hsv_green = cv.cvtColor(green, cv.COLOR_BGR2HSV)
print(hsv_green) # → [[[ 60 255 255]]]
```

얻은 Hue 값 H를 기준으로:

- **Lower bound**: `[H-10, 100, 100]`
- **Upper bound**: `[H+10, 255, 255]`

Exercises

- 빨강, 파랑, 초록 등 **여러 색상** 객체를 동시에 추출해 보세요.