1주차 요약본

작성자 이의진(엘텍공과대학 휴먼기계바이오공학부)

Chapter 02 OpenCV로 시작하는 컴퓨터 비전

2.1 OpenCV 소개

- OpenCV: 인텔 사에서 만들어 공개한 컴퓨터 비전 라이브러리. 함수와 클래스는 C,C++ 언어로 개발되었다.

2.3 객체지향 잘 활용하기

- 객체 확인
  - type: 객체의 자료형 반환
  - dir: 객체의 모든 속성과 method 목록 반환

2.4 영상을 읽고 표시하기

2-2. 영상파일을 읽고 윈도우에 디스플레이한다.

```
import cv2 as cv
import sys
## sys: 시스템 변수를 설정/ exit함수로 프로그램 종료.

img=cv.imread('soccer.jpg')
if img is None:
    sys.exit('파일을 찾을 수 없습니다.')

## cv.imshow(윈도우의 이름, 디스플레이할 영상)
cv.imshow('Soccer Image Display',img)

## 키 입력을 기다렸다가 윈도우를 닫고 프로그램 종료.
cv.waitKey()
cv.destroyAllWindows()
```

1) 이미지 읽기

```
cv2.imread(fileName, flag)
```

2) 이미지 보기

```
cv2.imshow(윈도우의 이름, 디스플레이할 영상)
```

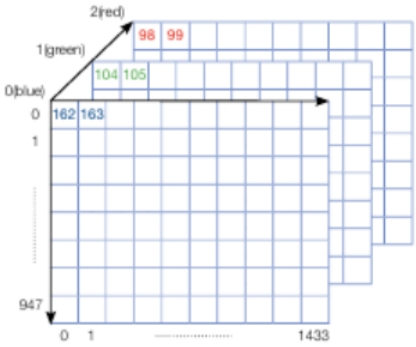
3) 키 입력을 기다리다가, 키가 눌리면 해당 키의 유니코드 값을 반환

```
cv.waitKey()
```

4) 모든 윈도우 닫고 프로그램 종료

```
cv.destroyAllWindows()
```

- **OpenCV는 numpy로 영상을 표현한다.**
- 이미지는 행(y) X 열(x) X 채널(BGR) 3차원 배열
- pixel: 영상을 구성하는 한 점
- pixel의 위치: (r,c)
- CV에서는 왼쪽 위가 원점
    - ex) img.shape = (948,1434,3)



2.5 영상 형태 변환하고 크기 축소하기

2-3. 영상을 명암 영상으로 변환하고 반으로 축소하기

```
gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY) # BGR 컬러 영상을 명암 영상으로 변환
gray_small=cv.resize(gray,dsize=(0,0),fx=0.5,fy=0.5)  # 반으로 축소

cv.imwrite('soccer_gray.jpg',gray)  # 영상을 파일에 저장
cv.imwrite('soccer_gray_small.jpg',gray)

cv.imshow('Color image',img)
cv.imshow('Gray image',gray)
cv.imshow('Gray image small',gray_small)

cv.waitKey()
cv.destroyAllWindows()
```

1) 컬러영상을 명암 영상으로 변환

```
cv2.cvtColor(img,cv.COLOR_BGR2GRAY)
```

2) 영상 저장

```
cv2.imwrite(fileName, image)
```

3) 영상 크기 조절

```
cv2.resize(입력 영상, dsize=(결과 영상), fx=가로방향 크기 변환 비율, fy=세로방향 크기 변환 비율)
```

- cvtColor함수가 BGR 영상을 명암 영상으로 변환하는 공식

$$I = \text{round}(0.299 \times R + 0.587 \times G + 0.114 \times B)$$

2.6 웹 캠에서 비디오 읽기

2-4. 웹 캠으로 비디오 획득하기

```
import cv2 as cv
import sys

cap = cv.VideoCapture(0,cv.CAP_DSHOW)  # 카메라와 연결 시도

if not cap.isOpened():
    sys.exit('카메라 연결 실패')

while True:
    ret, frame = cap.read()# 비디오를 구성하는 프레임 획득

    if not ret:
        print('프레임 획득에 실패하여 루프를 나납니다.')
        break

    cv.imshow('Video display', frame)

    key = cv.waitKey(1) # 1밀리초 동안 키보드 입력 기다림
    if key==ord('q'):#'q'입력 시 루프 탈출
        break

cap.release() # 카메라와 연결을 끊음
cv.destroyAllWindows() # 윈도우를 모두 닫음
```

1) 웹 캠과 연결 시도→ 연결 실패 시 cap.isOpened()가 False

```
cap = cv.VideoCapture(0,cv.CAP_DSHOW)
# 카메라와 연결 시도하는 결과를 cap에 저장
```

2) 동영상 입력 무한 반복

```
ret, frame = cap.read()
```

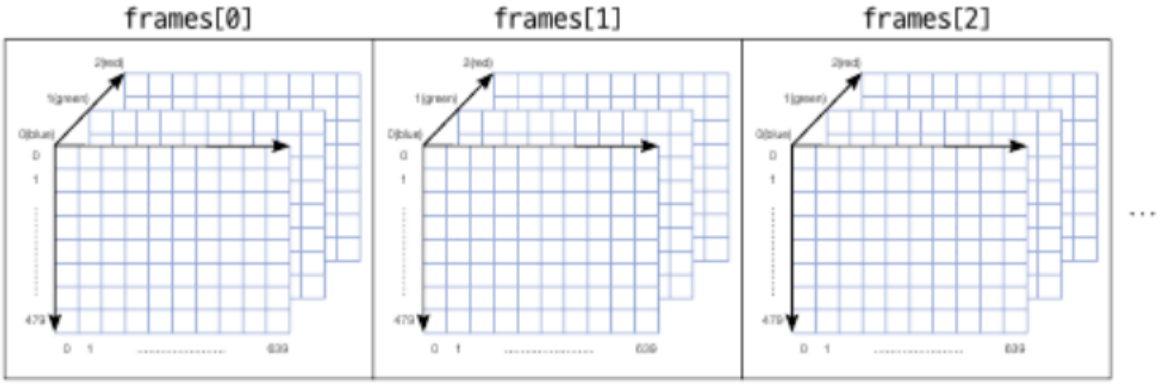
- read함수: 호출한 순간의 프레임을 획득하고 성공 여부, 프레임 반환
- ret객체(성공 여부), frame객체(호출한 순간의 영상 한 장, 프레임) 저장

3) 영상 디스플레이

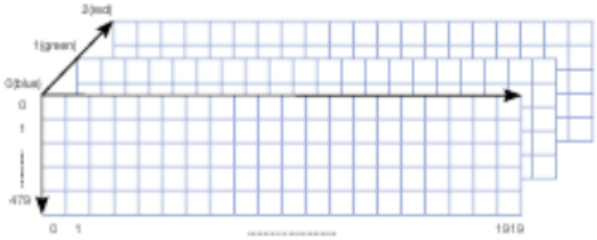
2.5 비디오에서 수집한 영상을 이어 붙이기

```
import cv2 as cv
import numpy as np
import sys

cap=cv.VideoCapture(0,cv.CAP_DSHOW) #카메라와 연결 시도

if not cap.isOpened():
  sys.exit('카메라 연결 실패)

frames=[]
while True:
  ret,frame cap.read() #비디오를 구성하는 프레임 획득

  if not ret:
    print('프레임 획득에 실패하여 루프를 나갑니다.)
    break

  cv.imshow("Video display', frame)

  key=cv.waitKey(1)#1밀리초 동안 키보드 입력 기다림
  if key==ord('c'):# 'C' 키가 들어오면 프레임을 리스트에 추가
    frames.append(frame)#'q'키가 들어오면 루프를 빠져나감
  elif key==ord('q'):
    break

cap.release()#카메라와 연결을 끊음
cv.destroyAllWindows()

if len(frames)>0:#수집된 영상이 있으면
  imgs=frames[0]
  for i in range(1,min(3, len(frames))): #최대 3개까지 이어 붙임.
    imgs=np.hstack((imgs, frames[i]))

  cv.imshow("collected images',imgs)

  cv.waitKey()
  cv.destroyAllWindows()
```

- np.hstack: 넘파이 배열을 가로로 결합



(a) frames 리스트



(b) imgs 배열

그림 2-11 [프로그램 2-5]의 자료 구조

2.7 그래픽 기능과 사용자 인터페이스 만들기

1) 영상에 도형 그리기

```
cv2.rectangle(img, (직사각형의 왼쪽 위 구석점의 x좌표, y좌표), (직사각형의 오른쪽 아래 구석점의 x좌표, y좌표), (B,G,R), 선의 두께)
```

2) 영상에 글씨 쓰기

```
cv2.putText(image, 'text', 문자열의 왼쪽 아래 구석점의 위치, font, fontScale, color, thickness)
```

2.7 마우스 클릭한 곳에 직사각형 그리기

```
def draw(event,x,y, flags, param):#콜백 함수
  if event==cv.EVENT_LBUTTONDOWN:# 왼쪽 버튼 클릭
    cv.rectangle(img, (x,y), (x+200, y+200), (0,0,255), 2)
  elif event==cv.EVENT_RBUTTONDOWN: #오른쪽 버튼 클릭
    cv.rectangle(img, (x,y), (x+100, y+100), (255,0,0),2)

  cv.imshow('Drawing',img)

cv.namedWindow('Drawing')
cv.imshow('Drawing',img)

cv.setMouseCallback('Drawing',draw) #Drawing 윈도우에 draw 콜백 함수 지정
# 마우스 이벤트 발생 -> draw 호출

while(True): #마우스 이벤트가 언제 발생할지 모름-> 무한 루프를 돌며 프로그램 실행 지속
  if cv.waitKey(1)==ord('q'):
    cv.destroyAllWindows()
    break
```

- callback function: 다른 함수의 인자로써 이용되는 함수

Chapter 03 영상 처리

3.1 디지털 영상 기초

아날로그 신호→디지털 변환

- 어떠한 과정을 통해 아날로그 신호를 받았다→ 샘플링(sampling)과 양자화(quantization)를 통해 디지털 영상으로 변환
- 샘플링: 2차원 영상 공간을 가로 방향 N X 세로 방향 M개 구간으로 나눔
  - 화소(pixel): 샘플링을 통해 형성된 한 점
  - **M** X **N**: 영상 크기(size), 해상도(resolution)
- 양자화: 화소의 명암을 L개 구간으로 나눔.

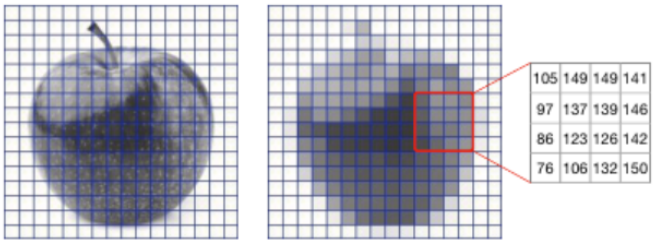


그림 3-3 피사체가 반사하는 빛 신호를 샘플링과 양자화를 통해 디지털 영상으로 변환

디지털 영상 좌표계의 특징

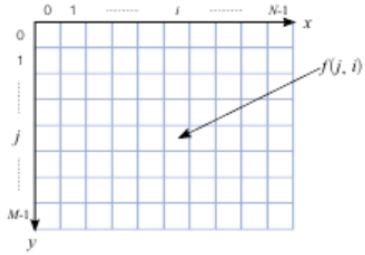


그림 3-4 디지털 영상의 좌표계

- 원점이 왼쪽 위이다
- (x,y)대신 (y,x)표기를 사용
  - 화소의 위치를 지정할 때→ (y,x)표기
  - 그 외의 경우→(x,y)

다양한 종류의 영상

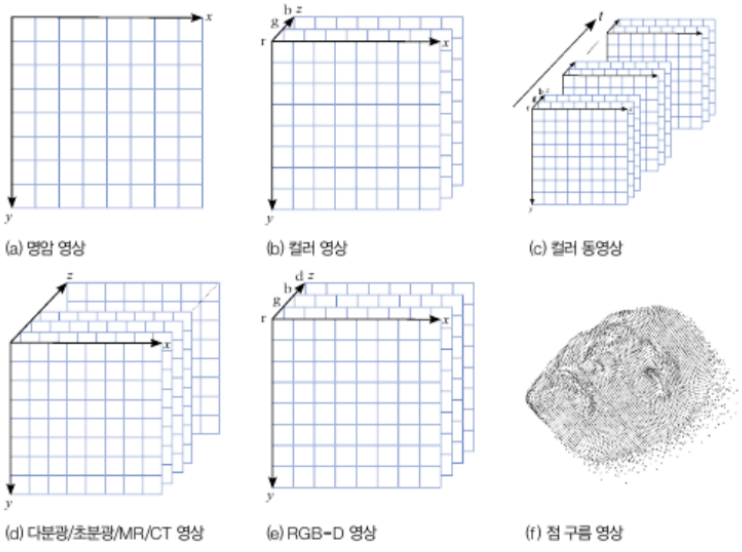


그림 3-5 다양한 형태의 디지털 영상

(a) 명암 영상(grayscale image): 2차원 구조의 배열. 채널 하나

(b) 컬러 영상: 3차원 배열. 3개 채널(R,G,B)로 구성

(c) 컬러 동영상: 4차원 구조의 배열.

(d) 다분광, 초분광, 의료영상 : 3차원 구조의 배열. 세 번째 축의 크기가 확장됨

ex)MRI-xy평면이 신체의 한 단면. z축은 신체의 위에서 아래로 진행

(e) RGB-D 영상: 물체까지의 거리를 측정한 영상. RGB컬러 센서 + 깊이 센서로 획득.

RGB 채널별로 디스플레이

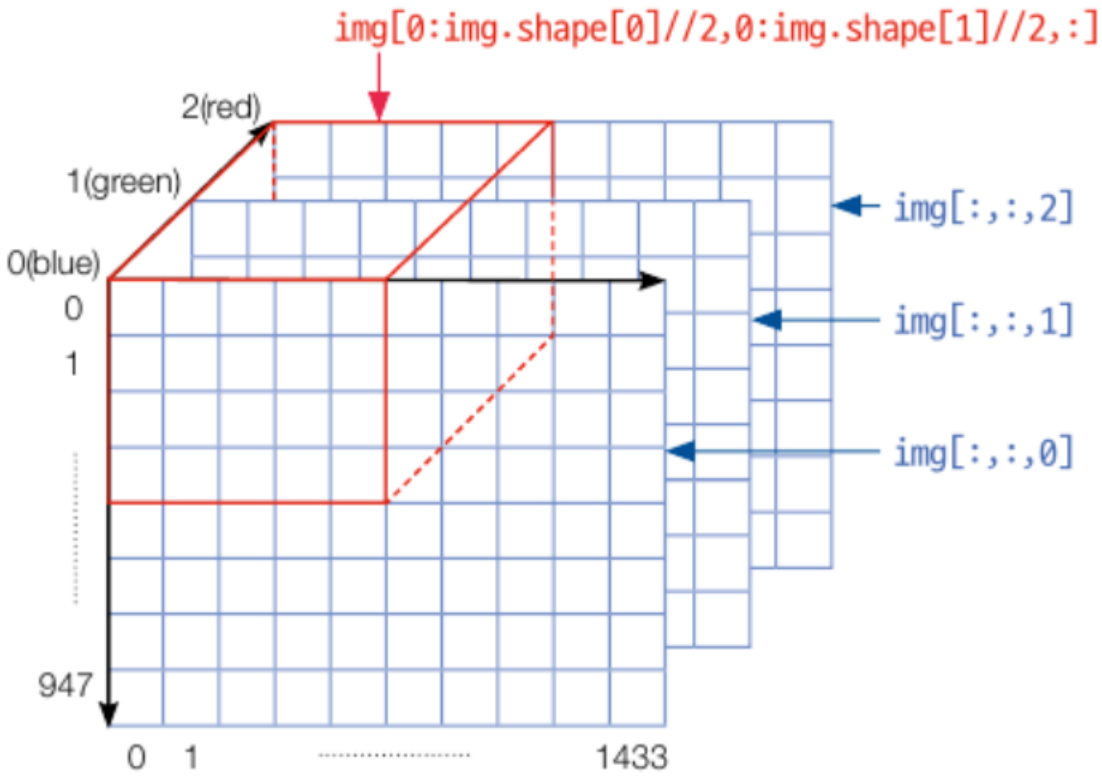


그림 3-8 numpy.ndarray의 슬라이싱을 이용한 영상 일부분 자르기([프로그

img.shape=(948,1434,3)를 슬라이싱하고 채널별로 분리해 디스플레이.

```
import cv2 as cv import sys
img=cv.imread('soccer.jpg')
if img is None:
sys.exit('파일을 찾을 수 없습니다.')

cv.imshow('original_RGB', img)
cv.imshow('Upper left half', img[0:img.shape[0]//2,0:img.shape[1]//2,:])
cv.imshow('Center half',img[img.shape[0]//4:3*img.shape[0]//4, img.shape[1]//4:3*img.shape[1]//4,:])

cv.imshow('R channel',img[:,:,2])
cv.imshow('G channel',img[:,:,1])
cv.imshow('B channel',img[:,:,0])

cv.waitKey()
cv.destroyAllWindows()
```

3.2 이진 영상

- 이진영상(binary image): 화소가 0(흑) 또는 1(백)
- CV에서의 활용: 에지만 1로 표현하기, 물체 검출한 후 물체_1 배경_0으로 표시하기

이진화

- binary image: 임곗값(threshold) T 보다 크면 1, 아니면 0. T의 결정이 중요함.

$$b(j,i)=\begin{cases}1, f(j,i)\geq T\\ 0, f(j,i)<T\end{cases} \qquad (3.1)$$

- 히스토그램의 계곡을 T로 결정하는게 Best!
- 계곡이 불분명하거나 많을 경우에는 T 결정이 어렵다.

→ 이를 해결하고자하는게 오츄 알고리즘임.

오츄 알고리즘

- 이진화는 최적화 문제로 공식화하여 푼다.

$$\hat{t}=\underset{t\in\{0,1,2,\cdots,L-1\}}{\operatorname{argmin}} J(t) \qquad (3.2)$$

t: 명암값(pixel value)

J: 목적 함수

- 목적 함수 J(t): 0이 되는 pixel value의 분산과 1이 되는 pixel value의 부산의 weighted sum.

$$J(t)=n_0(t)v_0(t)+n_1(t)v_1(t)$$

- exhaustive search algorithms: 모든 해(t)를 다 검사하여 최소가 되는 t를 결정한다. → 현실성이 없어 역전파 알고리즘 등을 사용함.

연결요소

- 화소의 연결성

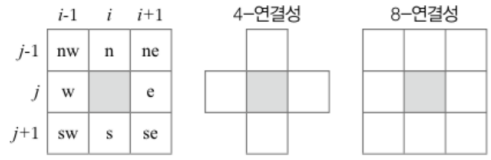


그림 3-10 화소의 연결성

- binary image에서 1의 값을 가진 연결된 pixel value의 집합