동영상 처리
(Video Processing)

담당교수: 김민기

비디오 처리

❖동영상 파일

- 초당 30프레임 저장, 압축 필요 → 압축 코덱(codec) 사용함
- 동영상을 읽어올 수 있는 VideoCapture 클래스

  - 생성자 3가지 VideoCapture(),
    VideoCapture(filename)
    VideoCapture(device)

비디오 처리

❖동영상 파일

- 카메라의 주요 속성 식별자 ➔ 교재 <표 4.5.1>

- VideoCapture.isOpened()
  - 동영상 파일이나 캡처 장치의 연결 여부를 반환
- VideoCapture.released()
  - 동영상 파일이나 캡처 장치를 해제
- VideoCapture.get() / VideoCapture.set()
  - 비디오 캡처의 지정된 속성값 반환 / 속성값 설정
- VideoCapture.grab()
  - 동영상 파일이나 캡처 장치에서 다음 프레임을 가져옴
- VideoCapture.read()
  - 가져온 프레임을 디코드해서 image 행렬로 반환

예제 4.5.1 카메라 프레임 읽기 – 17.read_pccamera.py

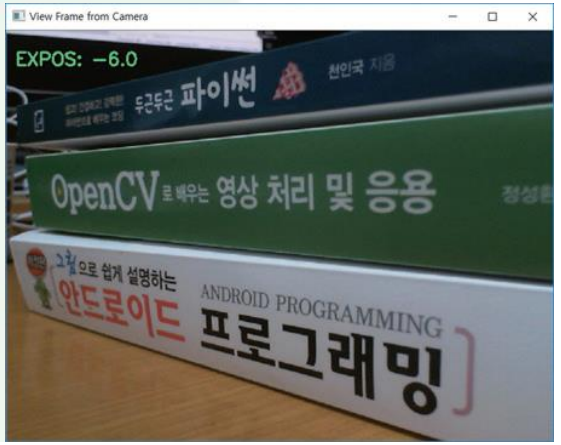
```
01  import cv2
02
03  def put_string(frame, text, pt, value, color=(120, 200, 90) ):  # 문자열 출력 함수
04      text += str(value)
05      shade = (pt[0] + 2, pt[1] + 2)
06      font = cv2.FONT_HERSHEY_SIMPLEX
07      cv2.putText(frame, text, shade, font, 0.7, (0, 0, 0), 2)      # 그림자 효과
08      cv2.putText(frame, text, pt    , font, 0.7, color, 2)         # 글자 적기
09
10  capture = cv2.VideoCapture(0)                                    # 0번 카메라 연결
11  if capture.isOpened() == False:                                  # 카메라 연결 예외처리
12       raise Exception("카메라 연결 안됨")
13
14  ## 카메라 속성 획득 및 출력
15  print("너비 %d" % capture.get(cv2.CAP_PROP_FRAME_WIDTH))
16  print("높이 %d" % capture.get(cv2.CAP_PROP_FRAME_HEIGHT))
17  print("노출 %d" % capture.get(cv2.CAP_PROP_EXPOSURE))
18  print("밝기 %d" % capture.get(cv2.CAP_PROP_BRIGHTNESS))
19
```

```python
## 카메라 속성 획득 및 출력
print("너비 %d" % capture.get(cv2.CAP_PROP_FRAME_WIDTH))
print("높이 %d" % capture.get(cv2.CAP_PROP_FRAME_HEIGHT))
print("노출 %d" % capture.get(cv2.CAP_PROP_EXPOSURE))
print("밝기 %d" % capture.get(cv2.CAP_PROP_BRIGHTNESS))

while True                                          # 무한 반복
    ret, frame = capture.read()                     # 카메라 영상 받기
    if not ret: break
    if cv2.waitKey(30) >= 0: break                  # 종료 조건 - 스페이스바 키

    exposure = capture.get(cv2.CAP_PROP_EXPOSURE)   # 노출 속성 획득
    put_string(frame, 'EXPOS: ', (10, 40), exposure)
    title = "View Frame from Camera"
    cv2.imshow(title, frame)                        # 윈도우에 영상 띄우기
capture.release()
```

```
Run:  17.read_pccamera
C:\Python\python.exe D:/source/chap04/17.
너비 640
높이 480
노출 -6
밝기 143
```



비디오 처리

❖동영상 파일

- ▪동영상을 파일로 저장할 수 있는 VideoWriter 클래스
  - 생성자 VideoWriter(filename, fourcc, fps, frameSize, isColor)
    - Fourcc: 프레임 압축에 사용되는 코덱의 4문자
    - 주요 코덱 문자 ➔ 교재 <표 4.5.2>
  - VideoWriter.isOpened() 동영상 파일이나 캡처 장치의 연결 여부를 반환
  - VideoWriter.write(img) img 프레임을 파일로 저장

예제 4.5.3 카메라 프레임을 동영상파일로 저장 – 19.write_camera_frame.py

```
01  import cv2
02
03  capture = cv2.VideoCapture(0)                          # 0번 카메라 연결
04  if capture.isOpened() == False: raise Exception("카메라 연결 안됨")
05
06  fps = 29.97                                            # 초당 프레임 수
07  delay = round(1000/fps)                                # 프레임 간 지연 시간
08  size = (640, 360)                                      # 동영상파일 해상도
09  fourcc = cv2.VideoWriter_fourcc(*'DX50')               # 압축 코덱 설정
10
11  ## 카메라 속성 실행창에 출력
12  print("width × height: ", size )
13  print("VideoWriterfourcc: %s" % fourcc)
14  print("delay: %2d ms" % delay)
15  print("fps: %.2f" % fps)
16
17  capture.set(cv2.CAP_PROP_ZOOM, 1)                      # 카메라 속성 지정
```

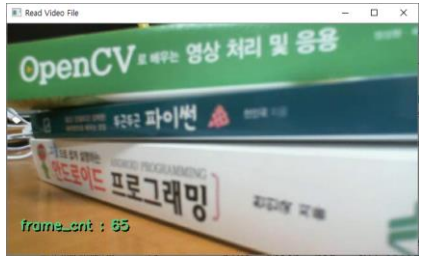
```
18  capture.set(cv2.CAP_PROP_FOCUS, 0)
19  capture.set(cv2.CAP_PROP_FRAME_WIDTH, size[0]) # 해상도 설정
20  capture.set(cv2.CAP_PROP_FRAME_HEIGHT, size[1])
21
22  ## 동영상파일 개방 및 코덱, 해상도 설정
23  writer = cv2.VideoWriter("images/video_file.avi", fourcc, fps, size)
24  if writer.isOpened() == False: raise Exception("동영상파일 개방 안됨")
25
26  while True:
27      ret, frame = capture.read()              # 카메라 영상 받기
28      if not ret: break
29      if cv2.waitKey(delay) >= 0: break
30
31      writer.write(frame)                      # 프레임을 동영상으로 저장
32      cv2.imshow("View Frame from Camera", frame)
33
34  writer.release()
35  capture.release()
```

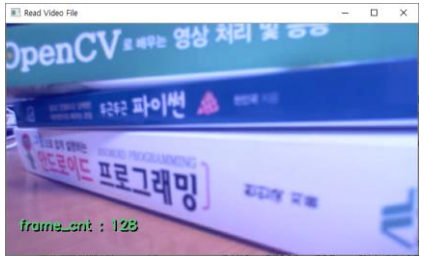
예제 4.5.4 동영상파일 읽기 – 20.read_video_file.py

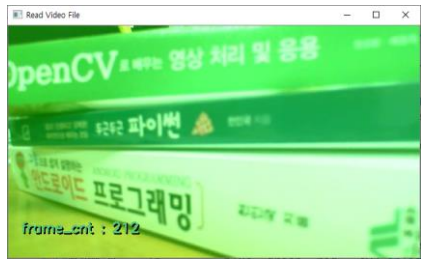
```
01  import cv2
02  from Common.utils import put_string                    # 글쓰기 함수 임포트
03
04  capture = cv2.VideoCapture("images/video_file.avi")    # 동영상파일 개방
05  if not capture.isOpened(): raise Exception("동영상파일 개방 안됨")          # 예외 처리
06
07  frame_rate = capture.get(cv2.CAP_PROP_FPS)             # 초당 프레임 수
08  delay = int(1000 / frame_rate)                         # 지연 시간
09  frame_cnt = 0                                          # 현재 프레임 번호
```

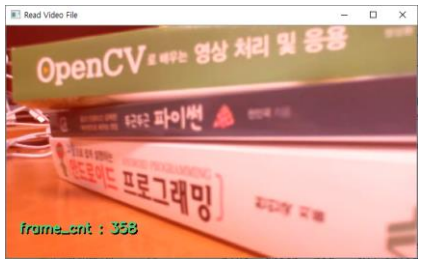
```
11  while True:
12      ret, frame = capture.read()
13      if not ret or cv2.waitKey(delay) >= 0: break      # 프레임 간 지연 시간 지정

14      blue, green, red = cv2.split(frame)               # 컬러 영상 채널 분리
15      frame_cnt += 1
16
17      if 100 <= frame_cnt < 200: cv2.add(blue, 100, blue)        # blue 채널 밝기 증가
18      elif 200 <= frame_cnt < 300: cv2.add(green, 100, green)    # green 채널 밝기 증가
19      elif 300 <= frame_cnt < 400: cv2.add(red   , 100, red)     # red 채널 밝기 증가
20
21      frame = cv2.merge( [blue, green, red] )           # 단일채널 영상 합성
22      put_string(frame, 'frame_cnt: ', (20, 30), frame_cnt)
23      cv2.imshow("Read Video File", frame)
24  capture.release()
```









〈표 4.5.3〉 프레임 번호에 따른 영상처리 예시

프레임 번호	영상처리
1 ~ 99	아무런 영상처리를 적용하지 않음
100 ~ 199	프레임별 화소의 파란색 성분에 100을 더해서 영상을 더 푸르게 만듦
200 ~ 299	프레임별 화소의 녹색 성분에 100을 더해서 영상을 더 녹색으로 만듦
300 ~ 399	프레임별 화소의 빨간색 성분에 100을 더해서 영상을 더 빨갛게 만듦

히스토그램 응용 (역투영)

❖히스토그램 역투영

- 명도 값을 해당 명도의 히스토그램 빈도로 변환

Ex) 8 levels gray image

2	4	4	3
2	1	3	3
1	0	1	2
0	1	1	2

h[i] = { 2, 5, 4, 3, 2, 0, 0, 0 }

histogram back projection →

4	2	2	3
4	5	3	3
5	2	5	4
2	5	5	4

✓ 히스토그램 빈도가 큰 명도 값이 역투영 결과 큰 값을 갖는다.
즉 역투영한 값이 크다는 것은 원래 히스토그램에 자주 나타난 명도 값을 의미함

히스토그램 역투영

h[i] = { 2, 5, 4, 3, 2, 0, 0, 0 }

2	4	4	3
2	1	3	3
1	0	1	2
0	1	1	2

Patch A

6	4	4	6
6	7	6	6
7	7	7	6
1	7	5	5

histogram back projection
by Patch A

0	2	2	0
0	0	0	0
0	0	0	0
5	0	0	0

1	4	4	6
1	7	2	2
3	7	2	2
1	3	1	1

histogram back projection
by Patch A

5	2	2	0
5	0	4	4
0	0	4	4
5	3	5	5

히스토그램 역투영

```
calcBackProject (
        const Mat* images,              // 1개 이상의 영상
        int nimages,                    // 영상의 개수
        const int* channels,            // 역투영에 사용할 채널
        const SparseMat& hist,          // 입력으로 사용되는 히스토그램
        OutputArray backProject,        // images 영상과 같은 크기와
                                        // 같은 깊이를 갖는 1-채널 역투영 행렬
        const float** ranges,
        double scale=1, bool uniform=true)
);
```

```python
import cv2

src = cv2.imread("./images/logo.jpg")
cv2.imshow("Image", src)

# 관심영역 추출
x, y, w, h = cv2.selectROI(src)
print(x, y, w, h)

hsv = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)
roi = hsv[y:y+h, x:x+w]

channels = [0, 1]   # 명도는 사용하지 않음
h_bins = 128        # 색상(hue)의 범위 0~127
s_bins = 256        # 밝기(value)의 범위 0~127
histSize = [h_bins, s_bins]
ranges = [0, 128] + [0, 256]
hist = cv2.calcHist([roi], channels, None, histSize, ranges)

# 히스토그램 역투영으로 마스크 생성
backproj = cv2.calcBackProject([hsv], channels, hist, ranges, 1)
cv2.imshow('backprj', backproj)

# 마스크 연산
dst = cv2.copyTo(src, mask=backproj)
cv2.imshow('dst', dst)

cv2.waitKey(0)
cv2.destroyAllWindows()
```