AIoT AutoCar Prime 으로 배우는

온디바이스 AI 프로그래밍

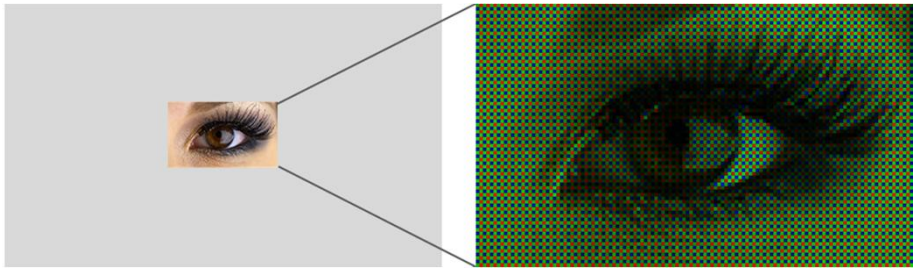
4 카메라 활용

카메라 활용

- 카메라 : 렌즈, 이미지 센서 및 프로세서로 구성
  - 렌즈로 들어오는 빛을 이미지 센서가 전기 신호로 변환
  - 프로세서가 신호를 분석해 하나의 디지털 이미지로 만듦
  - 이미지 센서
    - 잠자리의 눈처럼 수많은 센서 배열로 구성
    - 각 센서에서는 색상 필터에 빛을 투과 시켜 특정 색상의 밝기를 메모리에 저장
    - 센서 배열은 초록을 더 많이 배치하는 베이어 패턴을 사용해 초록빛을 더 많이 수용
      - 사람의 눈이 초록빛에 더 민감하게 반응

카메라 활용

- 이미지 신호처리 프로세서
  - 색상필터들이 감지한 밝기를 분석해 픽셀 만듦
  - 100% 풀 컬러를 만들려면 400% 베이어 어레이 이미지 센서 필요



OpenCV

- OpenCV
  - 인텔에서 개발된 실시간 이미지 프로세싱 전용 라이브러리
  - 노이즈 제어, 이미지 변환, 기계 학습 등 대부분의 알고리즘이 구현

OpenCV 설치

- 부록에서 설치 내용 확인 가능
- AutoCar에는 설치 되어 있음

이미지 활용

- 간단한 메소드로 이미지 제어
- imshow() 메소드
  - 창을 띄워 이미지 출력하는 OpenCV의 메소드
  - Jupyter 환경에서는 동작하지 않음
  - pop.Util 패키지의 enable_imshow() 메소드로 Jupyter에서 imshow 활성화

이미지 활용

- enable_imshow() 메소드
  - 창을 띄우는 것 대신 웹 그래픽으로 대체하여 표시
  - OpenCV의 imshow() 메소드를 변경

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
```

이미지 읽기

- imread(fileNmae, flag): 파일에서 이미지를 읽어 numpy의 ndarray 객체 반환
  - filenName: 이미지 파일 경로
  - flag: 읽기 옵션
    - cv2.IMREAD_COLOR: 기본값으로 컬러로 읽음
    - cv2.IMREAD_GRAYSCALE: 회색톤으로 읽음
    - cv2.IMREAD_UNCHAGED: 알파 채널을 포함해 읽음
  - 반환: numpy.ndarray 객체

테스트 이미지 생성

- pop.Util패키지의 createIMG() 메소드로 샘플 이미지 생성
  - ‘img.jpg’ 라는 이름으로 생성

```
01:     from pop import Util
02:
03:     Util.createIMG()
```

이미지 읽기

- 이미지의 너비, 높이, 채널 수 출력 예제
  - 동일 폴더 위치에 img.jpg파일 필요

```
01:     import cv2
02:
03:     image = cv2.imread("img.jpg", cv2.IMREAD_COLOR)
04:     height, width, channel = image.shape
05:
06:     print("width: %d, height: %d, channel: %d" % (width, height, channel))
```

이미지 보기

- imshow(title, image): 창에 이미지 표시
  - title: 창 제목
  - image: BGR 배열

이미지 보기

- 이미지 보기 예제
  - opencv 메소드 사용을 위한 cv2 라이브러리 import
  - Jupyter에서 imshow() 메소드 사용 불가능
  - Pop.Util의 enable_imshow() 메소드로 cv2의 imshow() 메소드를 변경

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
```

이미지 보기

- 불러올 이미지 파일명을 filename에 정의
- imread 메소드를 이용해 컬러, 회색톤 이미지로 imColor, imGray에 저장

```
05:     filename = "img.jpg"
06:
07:     imgColor = cv2.imread(filename, cv2.IMREAD_COLOR)
08:     imgGray = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
```

- 불러온 이미지를 ‘Color’, ‘GrayScale’란 이름의 창에 맞춰 출력

```
09:     cv2.imshow("Color", imgColor)
10:     cv2.imshow("GrayScale", imgGray)
```

이미지 보기

전체 코드

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
05:
06:     filename = "img.jpg"
07:
08:     imgColor = cv2.imread(filename, cv2.IMREAD_COLOR)
09:     imgGray = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
10:
11:     cv2.imshow("Color", imgColor)
12:     cv2.imshow("GrayScale", imgGray)
```

이미지 보기

- Jupyter 환경이 아닌 경우
  - imshow() 메소드는 즉시 닫히므로 창을 유지하려면 대기 필요
    - 이 때 waitKey() 메소드 사용
    - waitKey(n): 키 입력 대기
      - n: 밀리초 단위
        - 키 입력이 없어도 대기 제한 시간을 초과하면 반환
        - 0은 대기 제한 시간 사용 안함
      - 반환: 입력 키 값
        - 키 입력 없이 대기 제한 시간으로 반환되면 -1
  - 한 번에 모든 창을 닫을 때는 destroyAllWindows() 메소드를 사용
    - destroyAllWindows(): 모든 창 닫기

이미지 채널 변경하기

- BGR 순서의 채널을 RGB 순서로 변경하는 예제
  - 보편적인 이미지 뷰어는 BGR이 아닌 RGB 데이터 사용
  - Jupyter 환경에서 imshow를 사용할 수 있도록 함
  - 이미지를 가져옴

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
05:     imgOrigin = cv2.imread("img.jpg", cv2.IMREAD_COLOR)
```

이미지 채널 변경하기

- split() 메소드
  - 다차원 ndarray 객체를 1차원 ndarray 객체로 변경 -> 이미지 데이터를 채널별로 분리
- 분리된 채널을 R, G, B의 순서로 merge() 메소드를 통해 다시 묶음

```
06:     b, g, r = cv2.split(imgOrigin)
07:     imgNew = cv2.merge([r, g, b])
```

이미지 채널 변경하기

- 변경된 이미지를 imshow()로 출력하고 대기
- 키 입력이 들어올 경우 모든 창을 종료
  - Jupyter Lab에서는 Kernel Shutdown으로 종료

```
08:         cv2.imshow("Origin", imgOrigin)
09:         cv2.imshow("New", imgNew)
10:
11:         cv2.waitKey(0)
12:         cv2.destroyAllWindows()
```

이미지 채널 변경하기

전체 코드

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
05:
06:     imgOrigin = cv2.imread("img.jpg", cv2.IMREAD_COLOR)
07:
08:     b, g, r = cv2.split(imgOrigin)
09:     imgNew = cv2.merge([r, g, b])
10:
11:     cv2.imshow("Origin", imgOrigin)
12:     cv2.imshow("New", imgNew)
13:
14:     cv2.waitKey(0)
15:     cv2.destroyAllWindows()
```

이미지 저장하기

- OpenCV 라이브러리는 BGR 순서로 데이터 처리
- 파일로 저장 시에는 RGB 순서로 저장
- imwrite(filename, image): BGR 순서 ndarray 객체를 RGB 순서 이미지 파일로 저장
  - filename: 이미지 파일 경로
  - image: ndarray 객체

이미지 저장하기

- 이미지 파일을 읽어 원본과 흑백 이미지로 저장하는 예제
  - imshow() 메소드를 사용하지 않음
    - 같은 이미지를 Color와 Grayscale로 변환 후 저장
  - imgOrigin은 기존 이미지를 덮어쓰고, imgGray는 새로운 이미지 파일 생성

```
01:     import cv2
02:
03:     imgOrigin = cv2.imread("img.jpg", cv2.IMREAD_COLOR)
04:     imgGray = cv2.imread("img.jpg", cv2.IMREAD_GRAYSCALE)
05:
06:     cv2.imwrite("img.jpg", imgOrigin)
07:     cv2.imwrite("imgGray.jpg", imgGray)
```

카메라 활용

- OpenCV에서는 이미지, 영상 처리 가능
- GStreamer 프레임워크와 VideoCapture 클래스
  - 카메라나 파일에서 비디오 프레임을 읽을 때 사용

GStreamer 프레임워크

- GStreamer 프레임워크
  - OpenCV에서 카메라에 접근할 때 사용
  - 파이프라인 기반의 멀티미디어 프레임워크
  - 다양한 미디어를 프로그래머가 관리할 수 있는 기능을 제공
  - GStreamer를 사용하려면 해상도, 프레임, 색상 채널 등의 설정 필요

GStreamer 프레임워크

- 설정 내용을 VideoCapture 클래스 생성자로 입력하여 카메라 접근가능

```
01:     import cv2
02:
03:     cam = “nvarguscamerasrc ! video/x-raw(memory:NVMM), width=(int)640, height=(int)480,
format=(string)NV12, framerate=(fraction)30/1 ! nvvidconv flip-method=0 ! video/x-raw, width=(int)640,
height=(int)480, format=(string)BGRx ! videoconvert ! video/x-raw, format=(string)BGR ! appsink”
04:
05:     cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
```

- Pop.Util 라이브러리에 미리 정의되어 있는 gstrmer() 메소드로 대체 가능

```
01:     import cv2
02:     from pop import Util
03:
04:     cam = Util.gstrmer(width=640, height=480, fps=30, flip=0)
05:     cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
```

카메라 캡쳐

- 카메라 캡쳐를 위한 VideoCapture 클래스 주요 내용
  - VideoCapture([filename | device]): VideoCapture 객체 생성
    - filename: “video.avi” 와 같은 비디오 파일이름 또는 “img.jpg” 와 같은 이미지 파일
    - Device: 카메라 번호. 카메라가 연결된 순서대로 0 부터 1씩 증가
  - isOpened(): 카메라에 접근 가능한 지에 대한 여부
  - release(): 비디오 파일 또는 캡쳐 장치를 닫음

카메라 캡쳐

- read(): 다음 비디오 프레임을 캡쳐해 BGR로 변환한 ndarray 객체 반환
  - 반환: 튜플 타입 (retval, frame)
    - retval: 읽기에 성공하면 True. 아니면 False
    - frame: BGR 순서의 ndarray 객체
- get(propId): VideoCapture 속성 반환
- set(propId, value): VideoCapture 속성 설정

카메라 캡쳐

- 카메라 데이터를 640x480 해상도로 창에 표시하는 예제
  - 필요한 라이브러리 import
  - Util의 enable_imshow() 메소드로 imshow() 활성화

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
```

카메라 캡쳐

- Util의 gstrmer() 메소드로 카메라 해상도를 지정. VideoCapture 객체 생성

```
05:     cam = Util.gstrmer(width=640, height=480)
06:     camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
07:     if not camera.isOpened():
08:         print("Not found camera")
```

- 폭과 높이에 대한 정보를 얻고 출력

```
09:     width = camera.get(cv2.CAP_PROP_FRAME_WIDTH)
10:     height = camera.get(cv2.CAP_PROP_FRAME_HEIGHT)
11:     print("init width: %d, init height: %d" % (width,height))
```

카메라 캡쳐

- read() 메소드로 for문을 이용해 총 120개의 프레임 출력
  - read() 메소드는 한 프레임 씩 반환
- 만약 반환되는 데이터가 없으면 반복 종료

```
12:         for _ in range(120):
13:             ret, frame = camera.read()
14:             if not ret:
15:                 break
16:
17:             cv2.imshow("soda", frame)
```

- 반복이 종료 후 카메라 장치를 닫고 출력 창 닫음

```
18:         camera.release()
19:         cv2.destroyAllWindows()
```

카메라 캡쳐

전체 코드

```
01: import cv2
02: from pop import Util
03:
04: Util.enable_imshow()
05:
06: cam = Util.gstrmer(width=640, height=480)
07:
08: camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
09: if not camera.isOpened():
10:     print("Not found camera")
11:
12: width = camera.get(cv2.CAP_PROP_FRAME_WIDTH)
13: height = camera.get(cv2.CAP_PROP_FRAME_HEIGHT)
14: print("init width: %d, init height: %d" % (width,height))
15:
16: for _ in range(120):
17:     ret, frame = camera.read()
18:     if not ret:
19:         break
20:
21:     cv2.imshow("soda", frame)
22:
23: camera.release()
24: cv2.destroyAllWindows()
```

비디오 저장

- VideoWriter 클래스 : OpenCV로 비디오를 저장할 때 사용
  - VideoWriter(filename, fourcc, fps, frameSize): VideoWriter 객체 생성
    - Filename: 확장자 .avi인 파일 이름
    - fourcc: 프레임을 압축하는데 필요한 코덱 코드
    - fps: 프레임 속도
    - frameSize: (width, height): 프레임 크기 (튜플)
  - write(image): 비디오 프레임 쓰기
    - image: ndarray 객체
  - set(propId, value): VideoCapture 속성 설정

비디오 저장

- Camera로 읽은 프레임을 파일로 저장하는 예제
  - 필요한 모듈들을 불러오고 imshow() 메소드 활성화

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
```

  - Util의 gstrmer() 메소드로 카메라 해상도 지정, VideoCapture 객체 생성

```
05:     cam = Util.gstrmer(width=640, height=480)
06:     camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
07:     if not camera.isOpened():
08:         print("Not found camera")
```

비디오 저장

- H.264 코덱을 사용하는 VideoWriter 객체 생성
- 결과 영상은 640*480, 30FPS이며 soda.avi로 저장

```
09:         fourcc = cv2.VideoWriter_fourcc(*"X264")
10:         out = cv2.VideoWriter("soda.avi", fourcc, 30, (640,480))
```

- read() 메소드로 for문을 이용하여 120개의 프레임을 읽음
- 프레임을 회색톤으로 변환 후 출력하고 원본 프레임은 영상으로 저장

```
11:         for _ in range(120):
12:             ret, frame = camera.read()
13:             framGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
14:             out.write(frame)
15:
16:             cv2.imshow("soda", framGray)
```

비디오 저장

- 반복이 종료된 후 카메라 장치를 닫고 출력 창 닫음

```
17:     camera.release()
18:     cv2.destroyAllWindows()
```

비디오 저장

전체 코드

```
01: import cv2
02: from pop import Util
03:
04: Util.enable_imshow()
05:
06: cam = Util.gstrmer(width=640, height=480)
07: camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
08: if not camera.isOpened():
09:     print("Not found camera")
10:
11: fourcc = cv2.VideoWriter_fourcc(*"X264")
12: out = cv2.VideoWriter("soda.avi", fourcc, 30, (640,480))
13:
14: for _ in range(120):
15:     ret, frame = camera.read()
16:     framGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
17:     out.write(frame)
18:
19:     cv2.imshow("soda", framGray)
20:
21: camera.release()
22: cv2.destroyAllWindows()
```

외곽선 검출

- 외곽선
  - 밝기가 낮은 값에서 높은 값으로 변하거나 이와 반대로 변하는 지점
  - 외곽선은 영상안에 있는 객체들의 경계를 가리키는 것
    - 모양, 방향성 등을 탐지 가능

외곽선 검출

- OpenCV에는 여러가지 외곽선 검출 알고리즘이 포함
  - 캐니 엣지 검출 알고리즘
    - 잡음에 민감하지 않으며 명확한 외곽선을 검출하는데 목적을 두고 있는 알고리즘
    - 캐니 엣지 검출은 Canny() 메소드를 통해 사용 가능
  - Canny(image, threshold1, threshold2) : 이미지 외곽선을 검출 및 반환
    - image : 입력 이미지
    - threshold1 , threshold2 : 외곽선 검출을 위한 임계치
      - threshold1 : 최소값
      - threshold2 : 최대값

외곽선 검출

외곽선 검출 예제

필요한 모듈들을 불러오고 imshow() 메소드 활성화

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
```

Pop.Util의 gstrmer 메소드로 카메라 해상도 지정, VideoCapture 객체 생성

```
05:     cam = Util.gstrmer(width=640, height=480)
06:     camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
07:     if not camera.isOpened():
08:         print("Not found camera")
```

외곽선 검출

- read() 메소드로 for문을 이용해 총 120개의 프레임 출력
  - read() 메소드는 한 프레임 씩 반환
- Canny() 메소드를 통해 프레임에서 외곽선을 검출
  - Canny() 메소드에서는 8-bit로 변환 후 임계치 범위를 벗어나는 값들을 제거
- 변환된 이미지 출력

```
09:         for _ in range(120):
10:             ret, frame = camera.read()
11:             img = cv2.Canny(frame,100, 200)
12:
13:             cv2.imshow("soda", img)
```

외곽선 검출

- 반복이 종료된 이후에는 카메라 장치를 닫고 출력 창 닫음

```
14:     camera.release()
15:     cv2.destroyAllWindows()
```

외곽선 검출

전체 코드

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
05:
06:     cam = Util.gstrmer(width=640, height=480)
07:     camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
08:     if not camera.isOpened():
09:         print("Not found camera")
10:
11:     for _ in range(120):
12:         ret, frame = camera.read()
13:         img = cv2.Canny(frame,100, 200)
14:
15:         cv2.imshow("soda", img)
16:
17:     camera.release()
18:     cv2.destroyAllWindows()
```

얼굴 인식

- OpenCV 라이브러리
  - 얼굴인식 관련 데이터 모델과 얼굴 인식 알고리즘을 기본 제공
  - 카메라로 입력된 영상에서 사람 얼굴을 인지하는 프로그램을 작성 가능

얼굴 인식

- haar Cascades
  - 머신 러닝 기반의 객체 검출 알고리즘
  - 이미지나 비디오에서 객체를 검출할 때 사용
  - 직사각형 영역으로 구성되는 특징을 사용
    - 픽셀단위로 객체를 검출하는 방법보다 동작 속도 측면에서 검출 속도가 빠름
  - 검출하기 위한 객체가 포함된 이미지와 포함되지 않은 이미지를 활용
    - 특징 분류기를 통해 학습 진행
    - 학습이 완료되면 분류기를 활용하여 객체 검출

얼굴 인식

- haar Cascades 알고리즘의 4가지 분류
  - Haar Feature Selection : 특징 선택
    - 사각형 형태의 커널을 가지고 특징 계산을 위해 이미지 전체를 스캔
    - 이미지를 스캔하며 이동하는 인접한 사각 영역내에 있는 픽셀의 합의 차이를 활용
  - Integral Images : 적분 이미지
    - 사각 영역 내부의 픽셀들을 빠르게 더하고 연산하기 위해 적분 이미지를 사용

얼굴 인식

- Adaboost Training : 특징 학습
  - 선택한 특징을 활용하여 학습을 진행
  - 선택된 특징 중 객체를 검출하기 위한 특징을 선별
  - 선별된 특징을 이용하여 학습에 사용되는 이미지에 특징을 적용
  - 잘못 분류될 가능성이 있기 때문에 에러율이 낮은 특징을 선택
- Cascade Classifier : 특징 분류
  - 학습이 완료되면 입력 이미지를 통해 객체를 검출
  - 입력 이미지에서 객체가 있는 영역인지 단계별로 체크하여 검출

얼굴 인식

- OpenCV에서는 앞의 과정 후 생성된 얼굴 인식 분류기를 xml 파일로 제공
- 제공되는 분류기는 아래의 경로에 포함
  - /usr/local/share/opencv4/haarcascades/
- 분류기의 경로는 OpenCV를 설치 환경에 따라 다를 수 있음

얼굴 인식

- 설치된 분류기를 사용할 때는 CascadeClassifier 활용
  - 인자로 미리 학습된 분류기의 경로를 넣을 경우 로드 후 사용 가능
  - 많은 시간이 필요한 이미지 학습 없이 빠르게 활용 가능

얼굴 인식

- CascadeClassifier(cascPath): 분류기를 로드하여 반환
  - cascPath : 분류기 파일의 경로
- detectMultiScale(image, scaleFactor, minNeighbors, minSize):
  - image: 얼굴을 인식할 이미지 파일
  - scaleFactor: 스캔되는 이미지의 축소비율
  - minNeighbors: 얼굴로 판별되기 위한 주변 특징의 최소 개수
  - minSize: 얼굴로 판별되기 위한 개체의 최소 크기

얼굴 인식

얼굴 인식 예제

필요한 모듈들을 불러오고 imshow() 메소드 활성화

```
01:     import cv2
02:     from pop import Util
03:
04:     Util.enable_imshow()
```

OpenCV에서 제공하는 Haar Cascade 분류기 로드

```
05:     haar_face= '/usr/local/share/opencv4/haarcascades/haarcascade_frontalface_default.xml'
06:     face_cascade = cv2.CascadeClassifier(haar_face)
```

얼굴 인식

- Pop.Util의 gstrmer() 메소드로 카메라 해상도 지정, VideoCapture 객체 생성

```
07:         cam = Util.gstrmer(width=640, height=480)
08:         camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
09:         if not camera.isOpened():
10:             print("Not found camera")
```

얼굴 인식

- 입력받은 프레임을 회색톤으로 변환
- detectMultiScale() 메소드로 얼굴 검출
  - scaleFactor의 값을 줄일 경우
    - 정확도가 늘어날 수가 있지만 속도가 느려짐
  - minNeighbors의 값을 늘릴 경우
    - 정확도가 늘어날 수 있지만 해상도가 떨어지는 이미지에서는 검출 실패 가능성 있음

```
11:         for _ in range(300):
12:             ret, img = camera.read()
13:             gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
14:             faces= face_cascade.detectMultiScale(gray, scaleFactor=1.3 ,minNe
            ighbors=1,minSize=(100,100))
```

얼굴 인식

원본 이미지에서 찾은 객체들의 위치에 사각형 그리기

```
15:             for (x,y,w,h) in faces:
16:                 cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
```

완성된 이미지 출력

```
17:                         cv2.imshow('img',img)
18:
19:         cam.release()
20:         cv2.destroyAllWindows()
```

얼굴 인식

전체 코드

```
01: import cv2
02: from pop import Util
03:
04: Util.enable_imshow()
05:
06: haar_face=
'/usr/local/share/opencv4/haarcascades/haarcascade_frontalfa
ce_default.xml'
07: face_cascade = cv2.CascadeClassifier(haar_face)
08:
09: cam = Util.gstrmer(width=640, height=480)
10: camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
11: if not camera.isOpened():
12:     print("Not found camera")
13:
14: for _ in range(300):
15:     ret, img = camera.read()
16:     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
17:     faces= face_cascade.detectMultiScale(gray,
scaleFactor=1.3 ,minNeighbors=1,minSize=(100,100))
19:
20:     for (x,y,w,h) in faces:
21:         cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
22:         cv2.imshow('img',img)
23:
24: cam.release()
25: cv2.destroyAllWindows()
```

내용 정리

- 리눅스 커널은 카메라 하위 시스템인 V4L2로 카메라 장치를 추상화함
- OpenCV : 인텔에서 개발된 실시간 이미지 프로세싱 전용 라이브러리
- numpy의 ndarray 객체를 이용해 다차원 이미지 데이터를 관리
- imread() 메소드
  - 이미지를 읽기. 채널 순서를 RGB에서 BGR로 바꿔 읽음

내용 정리

- imshow() 메소드
  - ndarray 객체를 이미지로 바꿔 창에 표시
- imwrite() 메소드
  - BGR 순서의 ndarray 객체를 RGB로 바꿔 파일로 저장
- 주피터개발환경에서는 OpenCV의 imshow() 메소드가 사용 불가
  - 한백전자에서 제공하는 pop의Util 라이브러리 enable_imshow()로 사용 가능

내용 정리

- Gstreamer
  - 파이프라인 기반의 멀티미디어 프레임워크
- 다양한 미디어를 프로그래머가 관리할 수 있는 기능 제공
- VideoCapture 클래스
  - 카메라나 파일에서 비디오 프레임을 읽을 때 사용
- Haar Cascade
  - 머신 러닝 기반의 객체 검출 알고리즘

연습문제

- 문제 1. 다음 코드는 OpenCV 라이브러리를 이용하여 이미지를 읽어 너비, 높이, 채널을 출력하는 코드입니다. 질문을 읽고 답해보세요.

```
01:     import cv2
02:
03:     image = cv2.A(B)
04:     height, width, channel = image.shape
05:
06:     print("width: %d, height: %d, channel: %d"%(width, height, channel))
```

연습문제

- A. 이미지를 읽기위해 사용되는 메소드 A가 무엇인지 답해보세요.
- B. ‘Flash.jpg’ 라는 이미지 파일을 컬러로 읽으려 할 때 빈 칸 B에 들어갈 내용을 답해보세요.
- C. 이미지의 사이즈가 500x500일 때 코드의 출력을 답해보세요.

연습문제

- 문제 2. 다음 코드는 OpenCV 라이브러리를 이용하여 카메라 영상을 읽은 후 출력하는 코드입니다. 질문을 읽고 답해보세요.

```
01: import cv2
02: from pop import Util
03:
04: Util.enable_imshow()
05:
06: cam = Util.gstrmer(width=640, height=480)
07:
08: camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
09: if not camera.isOpened():
10:     print("Error!!")
```

```
11:
12: for _ in range(120):
13:     ret, frame = camera.read()
14:     if not ret:
15:         break
16:
17:     cv2.imshow("soda", frame)
18:
19: camera.release()
20: cv2.destroyAllWindows()
```

연습문제

- A. 코드를 실행했을 때, “Error!!” 가 출력되었다면 그 원인이 무엇인지 답해보세요.
- B. 코드를 실행했을 때, 총 몇 개의 프레임이 출력되는지 답해보세요.
- C. 영상을 회색톤으로 변환하여 출력하는 코드를 작성해보세요.

연습문제

- 문제 3. 다음 코드는 카메라를 읽어 얼굴을 인식하는 코드입니다. 질문을 읽고 답해보세요.

```
01: import cv2
02: from pop import Util
03:
04: Util.enable_imshow()
05:
06: haar_face=
'/usr/local/share/opencv4/haarcascades/haarcascade_fro
ntalface_default.xml'
07: face_cascade = cv2.CascadeClassifier(haar_face)
08:
09: cam = Util.gstrmer(width=640, height=480)
10: camera = cv2.VideoCapture(cam, cv2.CAP_GSTREAMER)
11: if not camera.isOpened():
12:     print("Not found camera")
13:
14: for _ in range(300):
15:     ret, img = camera.read()
16:     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
17:     faces= face_cascade.detectMultiScale(gray,
scaleFactor=1.3 ,minNeighbors =1,minSize=(100,100))
18:
19:     for (x,y,w,h) in faces:
20:         cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
21:         cv2.imshow('img',img)
22:
23: cam.release()
24: cv2.destroyAllWindows()
```

연습문제

- A. 코드를 실행했을 때 얼굴이 인식되는지 확인해보고 그 좌표를 출력해보세요.
- B. 다음 코드는 Pop.Pilot 라이브러리를 이용해 카메라를 움직이는 코드입니다. 이를 응용해 A의 좌표를 기반으로 카메라가 얼굴을 따라 움직이는 코드를 작성해보세요.

```
01:     from pop import Pilot
02:
03:     Car = Pilot.AutoCar()
04:
05:     Car.camTilt(90)
06:     Car.camPan(0)
```