x

K-Digital Training 스마트 팩토리 3기

OpenCV 3

트랙바

- cv2.createTrackbar(trackbarname, winname, value, count, onChange=0, userdata=0)
  - 슬라이더를 생성해서 window에 붙여줌
  - trackbarname : 트랙바 이름
  - winname : 윈도우 이름
  - value : default 값
  - count : 최대 값(0~count)
  - hist : 출력 히스토그램
  - onChange : 슬라이더의 값이 바뀔 때마다 호출되는 callback 함수
  - userdata : callback 함수에 들어갈 데이터

트랙바

- cv2.setTrackbarPos(trackbarname, winname, pos)
  - 슬라이더에 값을 설정
  - trackbarname : 트랙바 이름
  - winname : 윈도우 이름
  - pos : 설정할 값

윤곽선 검출

- cv2.findContours(image, mode, method, contours[, hierarchy[,offset]]]) -> contours, hierarchy
  - image : src, 8비트 싱글 채널 이미지, 0->0, non-zero ->1,
  - contours : 검출된 윤곽정보
  - hierarchy : 이미지 위상에 대한 정보를 포함하는 벡터
  - mode : 윤곽 검출 모드
  - method : 윤곽 근사 모드
  - offset : (optional) 윤곽 이동 offset, 이미지 ROI 에서 윤곽선을 추출 후에 전체 이미지 컨텍스트에서 분석해야 하는 경우 유용

윤곽선 검출

- cv::RetrievalModes {
  cv::RETR_EXTERNAL = 0, # 가장 바깥쪽 윤곽선만 검색
  cv::RETR_LIST = 1, # 모든 윤곽선을 검색
  cv::RETR_CCOMP = 2, # 모든 윤곽선을 검색하여 2단계 계층 구조로 구성
  cv::RETR_TREE = 3, # 모든 윤곽선을 검색하고 중첨된 윤곽선의 전체 계층 구조를 재구성
  cv::RETR_FLOODFILL = 4
  }

윤곽선 검출

- cv::ContourApproximationModes {
  cv::CHAIN_APPROX_NONE = 1, #모든 윤곽점 저장
  cv::CHAIN_APPROX_SIMPLE = 2, #수평, 수직,대각선 세그먼트 압축 후 끝점만 남김
  cv::CHAIN_APPROX_TC89_L1 = 3, # Teh-Chin 체인 근사 알고리즘 중 하나 적용
  cv::CHAIN_APPROX_TC89_KCOS = 4 # Teh-Chin 체인 근사 알고리즘 중 하나 적용
  }

그림 그리기

- cv2.drawContours(img, contours, contourIdx, color[, thickness[,lineType[,hierarchy[,maxLevel[,offset]]]]]) -> image
  - img : 출력 이미지
  - contours : 모든 input 윤곽들
  - contourIdx : 그려질 윤곽의 index 파라미터, -1이면 모든 윤곽
  - color : 윤곽의 색깔
  - thickness : 윤곽선의 두께, 음수인 경우에(FILLED) 윤곽선 내부가 그려짐
  - lineType : 라인 유형
  - hierarchy :
  - maxLevel : 그려질 윤곽선의 최대 level, 0이면 지정된 윤곽선만 그림. 1이면 윤곽선과 모든 중첩 윤곽선, 2이면 윤곽선, 모든 중첩 윤곽선, 모든 중첩의 중첩 윤곽선 등을 그림, hierarchy가 가능한 경우에만 적용
  - offset : (dx, dy)

코너 검출

- cv2.goodFeaturesToTrack(img, maxConers, qualityLevel, minDistance[, corners[,mask[,blockSize[,useHarrisDetector[,k]]]]]) -> corners
  - img : 입력 이미지, 8bit 혹은 32bit single 채널 이미지
  - corners : 출력 coner 벡터
  - maxCorners : 반환할 최대 코너 수, 음수면 모든 모서리 반환
  - qualityLevel : 이미지 모서리의 최소 허용 품질을 특성화
  - minDistance : 반환된 모서리 사이의 가능한 최소 유클리드 거리
  - mask: ROI, 모서리가 감지되는 영역
  - blockSize : 평균 블록의 크기
  - useHarrisDetector : Harris 검출기 사용 여부를 나타내는 매개변수
  - k : Harris 검출기의 자유 매개변수

히스토그램

- cv2.calcHist(images, channels, mask, histSize, ranges[, hist[,accumulate]]) -> hist
  - 배열의 히스토그램을 계산
  - img : 입력 배열, 모두 다 같은 depth여야 한다.
  - nimages : 소스 이미지 개수
  - channels : 히스토그램을 계산하는 데 사용되는 채널 목록
  - mask : 선택적 마스크
  - hist : 출력 히스토그램
  - dims: 히스토그램의 차원
  - histSize : 히스토그램 크기 배열
  - ranges : 히스토그램 Bin 경계의 희미한 배열
  - uniform : 히스토그램이 균일한지 여부
  - accumulate : 누적 플래그

템플릿 매칭

- cv2.matchTemplate(image, templ, method[,result[,mask]])
  -> result
  - 이미지 에서 template 을 찾기
  - image : template을 찾을 이미지, 8bit나 32비트 이미지
  - templ : 찾을 template, data type이 동일해야 하고 원본보다는 클 수 없다
  - result : 비교 결과값, 원본이미지가 W*H 이고 템플릿이 w*h 이면
    결과는 (W-w+1)*(H-h+1)
  - method : 비교 방법
  - mask : 선택적 마스크, templ과 동일한 크기여야함

Hugging face

1. https://huggingface.co/arnabdhar/YOLOv8-Face-Detection
2. Hugging face에서 모델 사용법 복사
3. 새 폴더 생성후에 ipynb 로 저장
4. 해당 폴더에서 visual studio code 열기

Hugging face

4. 가상 환경 생성
   - Kernel 선택 -> Select another kernel -> Python Env… -> Create Python Env… -> Venv
5. 필요한 패키지 설치
   - %pip install opencv-python
   - %pip install huggingface_hub
   - %pip install ultralytics
   - %pip install supervision
6. 얼굴있는 이미지 넣어서 moel 실행

Hugging face

7. 결과 확인

```
print(results)
```

✓ 0.0s Python

```
Detections(xyxy=array([[     1065.4,       355.8,      1229.2,       554.1],
       [     1324.2,      88.624,        1453,      298.12],
       [     267.84,      104.12,      404.61,      312.53],
       [     423.26,      377.83,      586.45,      590.99],
       [     769.53,      102.92,       890.6,      285.95]], dtype=float32), mask=None, confidence=array
```

8. results.xyxy 에 좌표정보들이 있다는것을 알 수 있음

9. 해당 정보 이용하여 얼굴 위치에 사각형 표시

```
for x, y, x2, y2 in arr_int:
    cv2.rectangle(src, (x, y), (x2, y2), (0, 255, 0), 2)
```