Python OpenCV

고급 교육 자료

영상 처리와 고급 얼굴 인식

1. 비디오 처리 기초

1.1 비디오 읽기와 재생

```
import cv2

# 비디오 파일 열기
cap = cv2.VideoCapture('video.mp4')

# 비디오 속성 확인
fps = cap.get(cv2.CAP_PROP_FPS)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

print(f"FPS: {fps}, 크기: {width}x{height}, 총 프레임: {total_frames}")

while cap.isOpened():
    ret, frame = cap.read()

    if not ret:
        break

    cv2.imshow('Video', frame)

    # FPS에 맞춰 대기
    if cv2.waitKey(int(1000/fps)) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

1.2 비디오 저장

```
import cv2

cap = cv2.VideoCapture(0)  # 웹캠

# 비디오 저장 설정
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))

while cap.isOpened():
    ret, frame = cap.read()

    if not ret:
        break

    # 프레임 저장
    out.write(frame)

    cv2.imshow('Recording', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

> **주요 코덱:**
>
> - **XVID**: 압축률 좋음, AVI 형식
> - **MJPG**: JPEG 압축, 빠름
> - **MP4V**: MP4 형식
> - **X264**: H.264 코덱, 고품질

1.3 프레임 간 차이 검출

```
import cv2
import numpy as np

cap = cv2.VideoCapture('video.mp4')
```

```
# 첫 프레임 읽기
ret, frame1 = cap.read()
ret, frame2 = cap.read()

while cap.isOpened():
    # 차이 계산
    diff = cv2.absdiff(frame1, frame2)
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)

    # 팽창하여 작은 구멍 채우기
    dilated = cv2.dilate(thresh, None, iterations=3)

    # 컨투어 찾기
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE,
                                   cv2.CHAIN_APPROX_SIMPLE)

    for cnt in contours:
        if cv2.contourArea(cnt) < 900:  # 작은 변화 무시
            continue

        x, y, w, h = cv2.boundingRect(cnt)
        cv2.rectangle(frame1, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow('Motion Detection', frame1)

    frame1 = frame2
    ret, frame2 = cap.read()

    if not ret:
        break

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

2. 객체 추적 (Object Tracking)

2.1 Meanshift 추적

색상 히스토그램을 이용한 객체 추적 방법입니다.

```
import cv2
import numpy as np

cap = cv2.VideoCapture('video.mp4')

# 첫 프레임에서 추적할 영역 선택
ret, frame = cap.read()
x, y, w, h = 300, 200, 100, 50  # 추적할 영역
track_window = (x, y, w, h)

# ROI 설정
roi = frame[y:y+h, x:x+w]
hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_roi, np.array((0., 60., 32.)),
                   np.array((180., 255., 255.)))
roi_hist = cv2.calcHist([hsv_roi], [0], mask, [180], [0, 180])
cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)

# 종료 조건
term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)

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

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)

    # Meanshift 적용
    ret, track_window = cv2.meanShift(dst, track_window, term_crit)

    # 추적 결과 그리기
    x, y, w, h = track_window
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow('Meanshift Tracking', frame)

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

2.2 CamShift 추적

Meanshift의 개선 버전으로, 객체의 크기와 회전도 추적합니다.

```
import cv2
import numpy as np

cap = cv2.VideoCapture('video.mp4')

ret, frame = cap.read()
x, y, w, h = 300, 200, 100, 50
track_window = (x, y, w, h)

roi = frame[y:y+h, x:x+w]
hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_roi, np.array((0., 60., 32.)),
                   np.array((180., 255., 255.)))
roi_hist = cv2.calcHist([hsv_roi], [0], mask, [180], [0, 180])
cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)

term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)

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

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)

    # CamShift 적용
    ret, track_window = cv2.CamShift(dst, track_window, term_crit)

    # 회전된 사각형 그리기
    pts = cv2.boxPoints(ret)
```

```
    pts = np.int0(pts)
    cv2.polylines(frame, [pts], True, (0, 255, 0), 2)

    cv2.imshow('CamShift Tracking', frame)

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

2.3 OpenCV 추적 API

OpenCV 4.x에서 제공하는 다양한 추적 알고리즘입니다.

```
import cv2

# 추적기 생성
# 다양한 추적 알고리즘: BOOSTING, MIL, KCF, TLD, MEDIANFLOW, MOSSE, CSRT
tracker = cv2.TrackerCSRT_create()

cap = cv2.VideoCapture('video.mp4')
ret, frame = cap.read()

# 추적할 영역 선택 (마우스로 선택 가능)
bbox = cv2.selectROI('Select Object', frame, False)
cv2.destroyWindow('Select Object')

# 추적기 초기화
tracker.init(frame, bbox)

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

    # 추적 업데이트
    success, bbox = tracker.update(frame)

    if success:
        x, y, w, h = [int(v) for v in bbox]
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    else:
        cv2.putText(frame, 'Lost', (50, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    cv2.imshow('Tracking', frame)

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

추적 알고리즘 비교:

알고리즘	속도	정확도	특징
BOOSTING	느림	보통	오래된 알고리즘
MIL	느림	좋음	부분 가림에 강함
KCF	빠름	좋음	균형 잡힌 성능
CSRT	느림	매우 좋음	가장 정확함
MOSSE	매우 빠름	보통	실시간 처리에 적합

3. 고급 얼굴 인식

3.1 얼굴 랜드마크 검출 (dlib 사용)

```
# dlib 설치: pip install dlib
import cv2
import dlib

# dlib의 얼굴 검출기와 랜드마크 예측기
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

img = cv2.imread('face.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 얼굴 검출
faces = detector(gray)

for face in faces:
    # 랜드마크 검출
    landmarks = predictor(gray, face)

    # 68개의 랜드마크 포인트 그리기
    for n in range(68):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        cv2.circle(img, (x, y), 2, (0, 255, 0), -1)
        cv2.putText(img, str(n), (x, y),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.3, (255, 0, 0), 1)

cv2.imshow('Facial Landmarks', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.2 얼굴 인식 (Face Recognition)

```
# face_recognition 설치: pip install face_recognition
import cv2
import face_recognition

# 기준 얼굴 이미지 로드
known_image = face_recognition.load_image_file('known_person.jpg')
known_encoding = face_recognition.face_encodings(known_image)[0]

# 테스트 이미지
test_image = cv2.imread('test_image.jpg')
rgb_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)

# 얼굴 위치와 인코딩 찾기
face_locations = face_recognition.face_locations(rgb_image)
face_encodings = face_recognition.face_encodings(rgb_image, face_locations)

# 각 얼굴 확인
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
    # 얼굴 비교
    matches = face_recognition.compare_faces([known_encoding], face_encoding)
    name = "Unknown"

    if matches[0]:
        name = "Known Person"

    # 사각형과 이름 그리기
    cv2.rectangle(test_image, (left, top), (right, bottom), (0, 255, 0), 2)
    cv2.putText(test_image, name, (left, top - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

cv2.imshow('Face Recognition', test_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.3 얼굴 감정 분석

```
import cv2
from deepface import DeepFace

img = cv2.imread('face.jpg')

try:
    # 얼굴 분석
    analysis = DeepFace.analyze(img, actions=['emotion'])

    # 감정 정보 추출
    dominant_emotion = analysis[0]['dominant_emotion']
```

```
        emotions = analysis[0]['emotion']

        # 결과 표시
        y = 30
        for emotion, score in emotions.items():
            text = f"{emotion}: {score:.1f}%"
            cv2.putText(img, text, (10, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)
            y += 25

        # 주요 감정 크게 표시
        cv2.putText(img, f"Dominant: {dominant_emotion}", (10, img.shape[0]-20),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 3)

        cv2.imshow('Emotion Analysis', img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

    except Exception as e:
        print(f"Error: {e}")
```

4. 광학 흐름 (Optical Flow)

4.1 Lucas-Kanade 광학 흐름

특징점의 움직임을 추적합니다.

```
import cv2
import numpy as np

cap = cv2.VideoCapture('video.mp4')

# 첫 프레임 읽기
ret, old_frame = cap.read()
old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)

# Shi-Tomasi 코너 검출
p0 = cv2.goodFeaturesToTrack(old_gray, maxCorners=100,
                             qualityLevel=0.3, minDistance=7,
                             blockSize=7)

# 랜덤 색상 생성
color = np.random.randint(0, 255, (100, 3))

# 궤적을 그릴 마스크
mask = np.zeros_like(old_frame)

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

    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Lucas-Kanade 광학 흐름 계산
    p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray,
                                           p0, None)

    # 좋은 포인트 선택
    if p1 is not None:
        good_new = p1[st==1]
        good_old = p0[st==1]

    # 궤적 그리기
    for i, (new, old) in enumerate(zip(good_new, good_old)):
        a, b = new.ravel()
        c, d = old.ravel()
        a, b, c, d = int(a), int(b), int(c), int(d)

        mask = cv2.line(mask, (a, b), (c, d), color[i].tolist(), 2)
        frame = cv2.circle(frame, (a, b), 5, color[i].tolist(), -1)

    img = cv2.add(frame, mask)
    cv2.imshow('Optical Flow', img)

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

    # 업데이트
    old_gray = frame_gray.copy()
    p0 = good_new.reshape(-1, 1, 2)

cap.release()
cv2.destroyAllWindows()
```

4.2 Dense 광학 흐름 (Farneback)

모든 픽셀의 움직임을 계산합니다.

```
import cv2
import numpy as np

cap = cv2.VideoCapture('video.mp4')

ret, frame1 = cap.read()
prev_gray = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)

# HSV 이미지 초기화
hsv = np.zeros_like(frame1)
hsv[..., 1] = 255

while True:
    ret, frame2 = cap.read()
    if not ret:
        break
```

```
    gray = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)

    # Dense 광학 흐름 계산
    flow = cv2.calcOpticalFlowFarneback(prev_gray, gray, None,
                                        0.5, 3, 15, 3, 5, 1.2, 0)

    # 극좌표로 변환
    mag, ang = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    hsv[..., 0] = ang * 180 / np.pi / 2
    hsv[..., 2] = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX)

    # BGR로 변환
    bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

    cv2.imshow('Dense Optical Flow', bgr)
    cv2.imshow('Original', frame2)

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

    prev_gray = gray

cap.release()
cv2.destroyAllWindows()
```

5. 실전 프로젝트

프로젝트 1: 실시간 차선 검출 시스템

```
import cv2
import numpy as np

def region_of_interest(img, vertices):
    mask = np.zeros_like(img)
    cv2.fillPoly(mask, vertices, 255)
    return cv2.bitwise_and(img, mask)

def draw_lines(img, lines):
    if lines is None:
        return img

    for line in lines:
        x1, y1, x2, y2 = line[0]
        cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 3)

    return img

cap = cv2.VideoCapture('road_video.mp4')

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

    # 그레이스케일 변환
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 가우시안 블러
    blur = cv2.GaussianBlur(gray, (5, 5), 0)

    # Canny 경계선 검출
    edges = cv2.Canny(blur, 50, 150)

    # 관심 영역 설정
    height, width = edges.shape
    vertices = np.array([[(0, height), (width/2, height/2),
                          (width, height)]], dtype=np.int32)
    roi = region_of_interest(edges, vertices)

    # Hough 변환으로 선 검출
    lines = cv2.HoughLinesP(roi, rho=1, theta=np.pi/180,
                            threshold=50, minLineLength=100,
                            maxLineGap=50)

    # 선 그리기
    line_image = np.zeros_like(frame)
    if lines is not None:
        line_image = draw_lines(line_image, lines)

    # 원본과 합성
    result = cv2.addWeighted(frame, 0.8, line_image, 1, 0)

    cv2.imshow('Lane Detection', result)

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

프로젝트 2: 사람 카운터 시스템

```
import cv2

# HOG + SVM 사람 검출기
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

cap = cv2.VideoCapture('people_walking.mp4')

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

    # 크기 조정 (처리 속도 향상)
```

```
    frame = cv2.resize(frame, (640, 480))

    # 사람 검출
    boxes, weights = hog.detectMultiScale(frame, winStride=(8, 8),
                                          padding=(4, 4), scale=1.05)

    # 검출된 사람 그리기
    for (x, y, w, h) in boxes:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # 카운트 표시
    cv2.putText(frame, f'People Count: {len(boxes)}', (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    cv2.imshow('People Counter', frame)

    if cv2.waitKey(30) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

프로젝트 3: QR 코드 스캐너

```
import cv2
import numpy as np

# QR 코드 검출기 생성
qr_detector = cv2.QRCodeDetector()

cap = cv2.VideoCapture(0)

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

    # QR 코드 검출 및 디코딩
    data, bbox, _ = qr_detector.detectAndDecode(frame)

    if bbox is not None:
        # QR 코드 영역 그리기
        bbox = bbox[0].astype(int)
        for i in range(len(bbox)):
            pt1 = tuple(bbox[i])
            pt2 = tuple(bbox[(i+1) % len(bbox)])
            cv2.line(frame, pt1, pt2, (0, 255, 0), 3)

        # 데이터 표시
        if data:
            cv2.putText(frame, f'Data: {data}', (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)

    cv2.imshow('QR Scanner', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

6. 최종 과제

최종 프로젝트: 종합 보안 시스템

다음 기능을 모두 포함하는 보안 시스템을 구현하세요:

- **모션 감지**: 프레임 간 차이로 움직임 감지
- **얼굴 인식**: 등록된 사람 식별
- **객체 추적**: 감지된 움직임 추적
- **이벤트 기록**: 감지 시 이미지와 로그 저장
- **알림 시스템**: 미등록 얼굴 감지 시 경고

고급 과제 1: 자율주행 시뮬레이션

비디오에서 차선을 검출하고, 주행 경로를 시각화하며, 장애물을 감지하는 시스템을 구현하세요.

고급 과제 2: 증강 현실 애플리케이션

마커 기반 또는 얼굴 기반 AR 애플리케이션을 구현하세요. 예: 얼굴에 필터 추가, 3D 객체 오버레이

고급 과제 3: 실시간 제스처 인식

손 동작을 인식하여 마우스나 키보드를 제어하는 시스템을 구현하세요.

프로젝트 구현 팁

- **성능 최적화**: 프레임 크기 조정, ROI 사용, 멀티스레딩
- **에러 처리**: try-except로 예외 상황 대비
- **로깅**: 중요한 이벤트는 파일로 기록
- **설정 파일**: 파라미터는 별도 설정 파일로 관리
- **테스트**: 다양한 환경에서 충분히 테스트

7. 추가 학습 자료

딥러닝과 OpenCV

- **YOLO**: 실시간 객체 검출
- **Mask R-CNN**: 인스턴스 세그멘테이션
- **OpenPose**: 포즈 추정
- **DeepFace**: 얼굴 인식 및 분석

유용한 라이브러리

- **dlib**: 얼굴 랜드마크, 얼굴 인식
- **face_recognition**: 간단한 얼굴 인식
- **mediapipe**: 손/얼굴/포즈 추적
- **scikit-image**: 추가 이미지 처리 기능

학습 리소스

- OpenCV 공식 문서: docs.opencv.org
- PyImageSearch 블로그
- OpenCV GitHub 저장소
- Kaggle 컴퓨터 비전 대회

축하합니다!

Python OpenCV 고급 과정을 모두 완료하셨습니다!

이제 실전 프로젝트를 시작하여
여러분만의 컴퓨터 비전 애플리케이션을 만들어보세요.

계속해서 학습하고 실험하며 성장하시길 바랍니다!