두산 Rokey Boot Camp

스터디 주간 활동 보고서

<table>
<tr><td>팀명</td><td>벌꿀오소리</td><td>제출자
성명</td><td>강인우</td></tr>
<tr><td>참여 명단</td><td colspan="3">정찬원, 임소정, 송주훈, 강인우</td></tr>
<tr><td>모임 일시</td><td colspan="3">2025 년 2 월 20 일 21 시 ~ 22 시 (총 1 시간)</td></tr>
<tr><td>장소</td><td>Google meet 화상회의</td><td>출석 인원</td><td>4 / 6</td></tr>
<tr><td>학습목표</td><td colspan="3">컴퓨터비전 및 openCV 관련 개별 주제 선정 후 발표</td></tr>
<tr><td>학습내용</td><td colspan="3">
▶주제선정
<table>
<tr><td>서준원</td><td>가우시안블러</td></tr>
<tr><td>정찬원</td><td>Camera Calibration</td></tr>
<tr><td>강인우</td><td>CNN</td></tr>
<tr><td>임소정</td><td>Canny Edge Detection</td></tr>
<tr><td>송주훈</td><td>intensity transform</td></tr>
<tr><td>정민섭</td><td>기하학적 변환 및 크기 변환</td></tr>
</table>
▶ intensity transform

○로그 변환 (Log Transformation)
</td></tr>
</table>

	어두운 부분 → 변화 많이 줌, 밝은 부분 → 변화 적게 줌 대비가 낮은 이미지에서 어두운 영역을 강조하는 데 유용 ○감마 변환 (Gamma Correction) 밝기 조절을 위한 비선형 변환 γ(감마 값)에 따라 어두운 부분과 밝은 부분 강조 조절 가능 ○Monotonic Increasing 변환 입력 값이 증가할수록 출력 값도 증가하는 변환 영상의 계조(gradation)를 보존하며 밝기 변화를 적용하는 특징 ○히스토그램 평활화 (Histogram Equalization) 이미지의 픽셀 값 분포를 고르게 만들어 대비 향상 밝기 값의 균형을 맞춰 어두운 이미지나 밝은 이미지를 보정 OpenCV 에서 cv2.equalizeHist() 함수 사용 ▶ Camera Calibration 카메라 영상은 3 차원 공간(3D)의 점을 2 차원 평면(2D 이미지)에 투사하여 얻어진다. 하지만 렌즈와 센서의 특성상 왜곡이 발생하므로, 이를 보정하기 위해 카메라 캘리브레이션이 필요 ○카메라 내부 파라미터 카메라 자체의 특성을 나타내며, 이미지 센서와 렌즈의 관계를 정의 초점 거리(Focal Length): 광학 중심에서 이미지 센서까지의 거리

광학 중심(Principal Point/axis): 카메라의 광축이 교차하는 이미지 센서의 좌표

비대칭 계수(Distortion Coefficients): 렌즈 왜곡 보정을 위한 계수

○카메라 좌표계 (Camera Coordinate, 3D)

카메라 좌표계 원점: 핀홀(Pinhole) 기준

좌표 변환:

카메라 좌표계 → 이미지 좌표계 변환 시 단위 변환 필요 (mm → 픽셀 단위)

변환 과정에서 Scaling 및 Translation 적용

○카메라 외부파라미터

카메라 좌표계와 월드 좌표계 사의 변환관계를 설명하는 파라미터

Rotation, Translation 활용

카메라 캘리브레이션 예시)체커보드 패턴 이미지들을 사용하여 카메라의 내부 파라미터 및 왜곡 계수를 계산

```
import glob

# 패턴 정보
nH = 9
nV = 6
# 실제 크기에 맞게 쓰세요
nSize = 26.1

#  image names
images = glob.glob('/content/drive/MyDrive/Colab
Notebooks/Calibration/captures/*.jpg')

CHECKERBOARD = (nV,nH)
criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
# 각 체커보드 이미지에 대한 3D 점 벡터를 저장할 벡터 생성
objpoints = []
# 각 체커보드 이미지에 대한 2D 점 벡터를 저장할 벡터 생성
imgpoints = []
# 3D 점의 세계 좌표 정의
objp = np.zeros((1, CHECKERBOARD[0] * CHECKERBOARD[1], 3),
np.float32)
objp[0,:,:2] = np.mgrid[0:CHECKERBOARD[0],
0:CHECKERBOARD[1]].T.reshape(-1, 2) * nSize
```

```
img = None

for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    ret, corners = cv2.findChessboardCorners(gray,
                                             CHECKERBOARD,
                                             cv2.CALIB_CB_ADAP
TIVE_THRESH + cv2.CALIB_CB_FAST_CHECK +
cv2.CALIB_CB_NORMALIZE_IMAGE)

    if ret == True:
        objpoints.append(objp)
        corners2 = cv2.cornerSubPix(gray, corners, (11,11),(-
1,-1), criteria)
        imgpoints.append(corners2)
        img = cv2.drawChessboardCorners(img, CHECKERBOARD,
corners2, ret)
    cv2_imshow(cv2.resize(img,  dsize=(0, 0), fx=0.4, fy=0.4,
interpolation=cv2.INTER_LINEAR))

h,w = img.shape[:2]
# 알려진 3D 점(objpoints) 값과 감지된 코너의 해당 픽셀
좌표(imgpoints) 전달, 카메라 캘리브레이션 수행
ret, intrinsic, dist, rvecs, tvecs =
cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],
None, None)
```

▶Canny Edge Detection

1.컬러 이미지를 흑백(Grayscale) 으로 변환하여 엣지 검출 성능 향상

노이즈 제거 (Gaussian Blur)

2.가우시안 블러를 적용하여 노이즈를 줄여 잘못된 엣지 검출 방지

Gradient 계산 (Sobel 필터 사용)

3.Sobel 필터를 적용하여 엣지의 방향과 강도 계산

수평(X), 수직(Y) 방향의 밝기 변화를 측정

4.강한 엣지 추출 (NMS, 비최대 억제)

엣지 방향을 따라 가장 강한 픽셀만 남기고 나머지 제거

5.강한/약한 엣지 구분 (이중 임계값 적용)

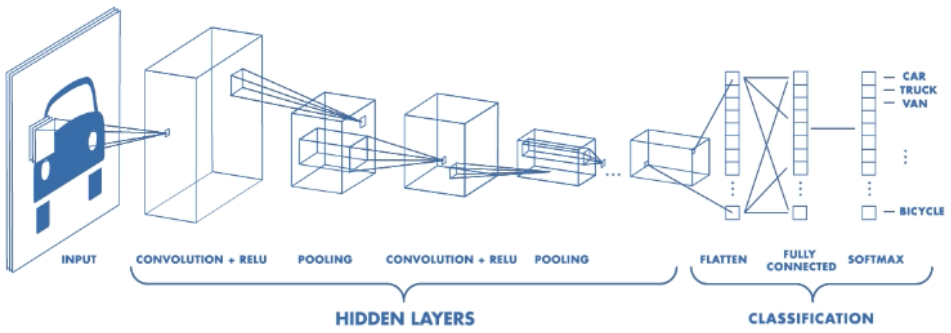
높은 임계값 이상 → 확실한 엣지 (강한 엣지)

낮은 임계값 ~ 높은 임계값 사이 → 약한 엣지

```python
import cv2
import numpy as np

# 1. 이미지 불러오기
img = cv2.imread(r'python_study/week5/sample.jpg')

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

# 3. 가우시안 블러 적용
blur = cv2.GaussianBlur(gray, (5,5), 1.5)

# 4. Canny Edge Detection 실행 (이중 임계값 적용)
edges = cv2.Canny(blur, 100, 200)

# 5. 결과 출력
cv2.imshow('Original', img)
cv2.imshow('Canny Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

▶CNN

학습과정



○ **합성곱 연산 (Convolutions)**

- 여러 개의 필터를 사용하여 **이미지의 중요한 특징(Edges, Shapes, Patterns 등)을 추출

- 입력 이미지에서 작은 영역(Patch)을 선택하고, 각 필터와 곱셈 연산을 수행하여 특징 맵(Feature Maps)을 생성

○ **풀링(Subsampling, Pooling)****

- 특징 맵의 크기를 줄이는 과정 (다운샘플링).

- 주로 Max Pooling(최댓값 풀링)이나 Average Pooling(평균 풀링)을 사용함.

○ **완전연결층 (Fully Connected, FC Layer)**

- 최종적으로 Flatten(1 차원 벡터로 변환)하여 완전연결층(FC Layer)에 전달.

- FC Layer 는 일반적인 신경망 구조이며, 모든 뉴런이 연결됨.

- `Softmax` ,`Sigmoid` 등의 활성화 함수를 통해 최종 클래스 확률을 계산

○ torch.nn.Conv2d

PyTorch 에서 2 차원 합성곱 연산을 수행하는 모듈

```
import torch
import torch.nn as nn
```

```
# 입력: 배치 크기=1, 채널=3, 높이=32, 너비=32
input_tensor = torch.randn(1, 3, 32, 32)

# Conv2d 레이어 생성
conv_layer = nn.Conv2d(
    in_channels=3,       # 입력 채널 수
    out_channels=16,     # 출력 채널 수
    kernel_size=3,       # 3x3 필터
    stride=1,            # 스트라이드 1
    padding=1            # 패딩 1 (출력 크기를 입력과 동일하게
유지)
```

○nn.MaxPool2d

```
import torch
import torch.nn as nn

input_tensor = torch.tensor([[[[1.0, 2.0, 3.0, 4.0],
                               [5.0, 6.0, 7.0, 8.0],
                               [9.0, 10.0, 11.0, 12.0],
                               [13.0, 14.0, 15.0, 16.0]]]])

maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
output = maxpool(input_tensor)

print(output)

<출력>
tensor([[[[ 6.,  8.],
          [14., 16.]]]])
```

▶ 기하학적 변환 및 크기 변환 요약

기하학적 변환(Geometric Transformation): 이미지의 픽셀 위치를 변경하여

영상의 모양을 바꾸는 작업

주요 변환: 크기 변환(Scaling), 회전(Rotation), 대칭(Flipping),

이동(Translation)

	○ OpenCV 를 이용한 크기변환 예제
	
	`import cv2`
	`import numpy as np`
	
	`# 이미지 불러오기`
	`image = cv2.imread('이미지 경로')`
	
	`# 크기 변환 (고정 크기)`
	`resize1 = cv2.resize(image, (200,200))  # 가로, 세로 200px`
	
	`# 크기 변환 (비율 조정)`
	`resize2 = cv2.resize(image, None, fx=2, fy=2)  # 2 배 확대`
	
	`# 보간법 적용한 크기 변환`
	`result1 = cv2.resize(image, (100,100),`
	`interpolation=cv2.INTER_AREA) # 축소 시 가장 좋은 방법`
	`result2 = cv2.resize(image, None, fx=0.5, fy=0.5,`
	`interpolation=cv2.INTER_AREA)`
	`result3 = cv2.resize(image, None, fx=2, fy=2,`
	`interpolation=cv2.INTER_NEAREST) # 최근접 이웃`
	
	`# 결과 출력`
	`cv2.imshow('Resize Fixed', resize1)`
	`cv2.imshow('Resize Scaled', resize2)`
	`cv2.imshow('Interpolation AREA', result1)`
	`cv2.imshow('Interpolation NEAREST', result3)`
	`cv2.waitKey(0)`
	`cv2.destroyAllWindows()`

활동평가	컴퓨터 비전관련 개념과 OpenCV 등 주요 라이브러리 등의 사용법 설명

과제	스터디 자료 업로드 및 복기
향후 계획	파트 분할 후 정규수업내용 복습
첨부 자료	*ex) 사진, 구글 드라이브 링크, git 주소 등*