종합설계1 과제 수행 보고서

작품과제명	OpenCV 필터 카메라
과제 개요	○ 과제 선정 배경 - 엔터테인먼트: 카메라 애플리케이션에 동물 필터 및 왜곡 효과를 추가하면 사용자에게 재미있는 경험을 제공할 수 있습니다. 이를 통해 실시간으로 자신의 외모를 변형하고 창의적이고 매력적인 콘텐츠를 저장하여 다른 사람과 공유 - 기술: OpenCV 및 얼굴 표정 인식 기술을 사용하여 필터 카메라를 구축하여 컴퓨터 비전 개념에 대한 이해와 적용을 보여줍니다. OpenCV와 같은 강력한 라이브러리를 활용하여 실시간 이미지 처리 및 증강 현실 경험을 만드는 능력을 보여준다. - 학습: 동물 필터와 왜곡 효과가 있는 필터 카메라를 개발하는 것은 귀중한 학습 경험이 될 수 있다. 얼굴 랜드마크 감지, 표정 인식, 이미지 조작과 같은 다양한 컴퓨터 비전 기술을 탐색할 수 있다. OpenCV 작업에 대한 실용적인 지식을 얻고 이미지 처리 알고리즘에 대한 이해를 심화할 수 있다. ○ 과제의 필요성 - 개인화 및 자기 표현: 필터 카메라는 사용자에게 독창성과 고유한 스타일을 표현할 수 있는 기회를 제공, 특히 동물 필터는 사용자가 다양한 동물과 캐릭터를 구현할 수 있도록 하여 콘텐츠에 장난기와 자기 표현의 요소를 추가 - 소셜 공유: 현재 카메라 필터는 여러 소셜 미디어 플랫폼에서의 공유 가능성으로 인해 대중화 되었음 특히 tiktok, snow, instagram등에서 많이 활용, 해당 sns 모바일 앱, 프로그램으로 인해 수 많은 사용자들이 종종 필터링된 사진과 비디오를 친구, 팔로워 및 온라인 커뮤니티와 공유하는 것을 즐기는 것을 확인 - 원격 협업 및 커뮤니케이션: AR은 가상 요소를 실제 시나리오에 오버레이하여 원격 협업 및 커뮤니케이션을 촉진할 수 있습니다. AR 지원 장치를 사용하여 개인은 자신의 관점을 공유하고 지침을 제공하며 마치 실제로 있는 것처럼 원격 동료와 상호 작용할 수 있습니다. 이것은 원격 작업, 원격 의료, 원격 교육 및 실시간 공동 작업이 중요한 기타 시나리오에 엄청난 잠재력을 가지고 있습니다.

과제 내용

○ 과제 구성

- 사용한 라이브러리

```
import cv2 # opencv, 영상처리
import mediapipe as mp # mediopipe, 얼굴 인식 라이브러리 사용
import os # 파일 생성을 위해 사용
import datetime # 파일 이름을 지정하기 위해 그리고 최대 녹화시간을 카
import pyaudio # 오디오 녹화
import wave # 오디오 녹화
import numpy as np # 오디오 연산, 표정 인식 얼굴 크기 계산
from moviepy.editor import * # 비디오, 오디오 합성
import time # 음성 파일이 생성 될 때 까지 delay 발생시키기 위해 사용
import threading # 쓰레드, 두 가지 이상의 일을 하기 위해 사용 (opencv
import dlib
from keras.models import load_model
```

- 카메라 입력: 카메라 입력은 실시간 이미지 또는 비디오를 캡처하기 위한 소스 역할

```
# 웹캠 객체
cap = cv2.VideoCapture(0)

# 웹캠 입력 창 크기 변수 저장
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
print(width, height) # 640, 480
SCREEN_REGION = (0, 0, width, height)
```

- 동물 필터 오버레이: 얼굴을 인식하여 동물 요소가 될 눈, 코, 입등 적절한 위치에 이미지를 배치하는 형태

```
#region 동물 이미지 불러오기
# cv2.IMREAD_UNCHANGED, 이미지파일을 alpha channel(누끼)까지 포함하여 읽는다.
# 이미지를 쉽게 보기 위해 dictionary 를 사용
imageList = {
    'panda' : [cv2.imread(animal_path+'panda/right_eye_cutout.png', cv2.IMREAD_U
               cv2.imread(animal_path+'panda/nose_tip_cutout.png', cv2.IMREAD_UNCHA
    'cat' : [cv2.imread(animal_path+'cat/right_eye2.png', cv2.IMREAD_UNCHANGED),
             cv2.imread(animal_path+'cat/nose_tip2.png', cv2.IMREAD_UNCHANGED)],
    'dog' : [cv2.imread(animal_path+'dog/right_eye3.png', cv2.IMREAD_UNCHANGED),
             cv2.imread(animal_path+'dog/nose_tip3.png', cv2.IMREAD_UNCHANGED)]
}
#endregion

# 처음 이미지 기본값, 판다 적용
image_right_eye = imageList['panda'][0]
image_left_eye = imageList['panda'][1]
image_nose_tip = imageList['panda'][2]
```

```python
#region 캠에 이미지 덮어 씌우는 함수
def overlay(image, x, y, w, h, overlay_image): # 대상 이미지, x, y 좌표, width, heig
    alpha = overlay_image[:, :, 3] #BGRA, A값을 가져옴
    mask_image = alpha / 255 # 0~255 ->255로 나누면 0~1의 값을 가짐, 1: 불투명, 0: 투
    # print(x, y, w, h)

    # 얼굴이 창 크기를 벗어나면 오류가 생기므로 예외처리
    try:
        for c in range(0, 3): #BGR 처리
            image[y-h: y+h, x-w: x+w, c] = (overlay_image[:, :, c] * mask_image)
    except Exception as e:
        print(e)
        pass
#endregion
```

- 화면 왜곡: 얼굴을 인식하고 사용자의 표정을 인식하여 사용자에게 보여지는 화면을 왜곡 시켜 보여준다, 여러 기능 있는데 표정에 따라 렌즈 효과, 색상 효과 등이 있다.

```python
#region 볼록 렌즈 필터 함수
def lens(exp, scale, frame):
    rows, cols = frame.shape[:2]
    # 매핑 배열 생성 ---②
    mapy, mapx = np.indices((rows, cols),dtype=np.float32)
    # 좌상단 기준좌표에서 -1~1로 정규화된 중심점 기준 좌표로 변경 ---③
    mapx = 2*mapx/(cols-1)-1
    mapy = 2*mapy/(rows-1)-1
    # 직교좌표를 극 좌표로 변환 ---④
    r, theta = cv2.cartToPolar(mapx, mapy)
    # 왜곡 영역만 중심확대/축소 지수 적용 ---⑤
    r[r< scale] = r[r<scale] **exp
    # 극 좌표를 직교좌표로 변환 ---⑥
    mapx, mapy = cv2.polarToCart(r, theta)
    # 중심점 기준에서 좌상단 기준으로 변경 ---⑦
    mapx = ((mapx + 1)*cols-1)/2
    mapy = ((mapy + 1)*rows-1)/2
    return mapx, mapy
#endregion
```

- 영상, 이미지 저장: 동영상으로 저장할 때에는 영상과 소리를 따로 저장 후 library를 이용해 두 개의 파일의 싱크를 맞춘 후 mp4 형태의 파일로 저장

```python
#region 녹화 종료 클래스
class record():

    def __init__(self):
        pass

    def exit(out, stream, audio, audio_frames):
        global recording
        out.release()
        stream.stop_stream()
        stream.close()
        audio.terminate()
        waveFile = wave.open(save_path + f"{fileName}.wav", 'wb')
        waveFile.setnchannels(channels)
        waveFile.setsampwidth(audio.get_sample_size(format))
        waveFile.setframerate(rate)
        waveFile.writeframes(b''.join(audio_frames))
        waveFile.close()
        audio_frames = []
        recording = False
        print('녹화종료')
#endregion
# 녹음 시간이 duration 을 넘으면 녹화 종료
if keycode == ord('b') and recording:
    # 녹화 종료
    record.exit(out, stream, audio, audio_frames)
    video_save.start()
```

```python
#region 이미지 저장 함수
def displayCapture(image):
    # 저장 폴더, 없는 경우 생성
    if not os.path.exists(save_path):
        os.makedirs(save_path)

    try:
        # 현재 시간을 파일 이름으로 지정하여 png 파일로 저장
        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H-%M-%S")
        file_name = f"{save_path}/{current_time}.png"
        cv2.imwrite(file_name, image) # 이미지 저장
        print(f"{file_name} 저장 완료") # 출력
    except:
        print("에러 발생")
#endregion
```

○ 과제 주요 특징

- 얼굴 랜드마크 감지: 동물 필터 오버레이를 사용자의 얼굴에 정확하게 배치하려면 얼굴 랜드마크 감지가 중요합니다. 이 구성 요소는 사용자의 눈, 코, 입등을 감지

```python
#region 영상, 소리 합성, 저장 함수
def videoSink():
    print('영상, 소리 합성중')
    # 음성 파일은 녹화가 끝나고 만들어 지므로 살짝 지연
    time.sleep(2)

    # 합성할 비디오, 오디오 객체 생성
    videoclip = VideoFileClip(save_path + f"{fileName}.avi")
    audioclip = AudioFileClip(save_path + f"{fileName}.wav")

    # 비디오에 오디오 삽입후 파일 생성
    videoclip.audio = audioclip
    video = f"{fileName}.mp4"
    videoclip.write_videofile(save_path + video)
    print(f'영상, 소리 합성 완료, {video} 생성 완료')
```

```python
#region 녹화 시작
if keycode == ord('v') and recording == False:
    print('녹화 시작')
    # 오디오 생성
    stream = audio.open(format=format, channels=channels, rate=rate, input=True, frames

    # 비디오 저장 객체 생성
    fileName = datetime.datetime.now().strftime("%Y-%m-%d %H-%M-%S")
    out = cv2.VideoWriter(save_path + f'{fileName}.avi',fourcc, fps, (width, height))

    # 녹화 시작
    recording = True
    start_time = datetime.datetime.now()
#endregion
# 이미지 적용
# 이미지의 크기를 동적으로 하기위해 얼굴의 크기 값을 연산하는 부분을 이용
box_wh = detection.location_data.relative_bounding_box
box_w = int(round(box_wh.width, 2) * 100)
box_h = int(round(box_wh.height, 2) * 100)

# 이미지 위치 지정, 얼굴크기에 맞게 위치를 동적으로 지정 함
w, h = width, height
right_eye = (int(right_eye.x * w)-box_w, int(right_eye.y * h)-(box_h*3))
left_eye = (int(left_eye.x * w)+box_w, int(left_eye.y * h)-(box_h*3))
nose_tip = (int(nose_tip.x * w), int(nose_tip.y * h)+box_h)

#region 이미지 대입
# operands could not be broadcast together with shapes을 방지하기 위해 기존
# 해당 에러는 현재 입히는 이미지의 크기와 opencv 상에서 적용되는 이미지의 해상
# 둘의 이미지를 같아지게 하도록 cv2.resize()를 적용

# 오른쪽 귀
overlay_right_eye = cv2.resize(image_right_eye, (box_w*2, box_h*2))
overlay(image, *right_eye, box_w, box_h, overlay_right_eye)

# 왼쪽 귀
overlay_left_eye = cv2.resize(image_left_eye, (box_w*2, box_h*2))
overlay(image, *left_eye, box_w, box_h, overlay_left_eye)
```

```
# 얼굴 인식
# scaleFactor이 1에 가까울수록 표정 인식이 잘 되고 멀 수록 잘 안됨
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeig

#region 얼굴이 인식되면 표정을 인식
for (x, y, w, h) in faces:
    # 얼굴 크기에 알맞도록 사각형 그리기
    # cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # 얼굴 크기 반환
    face_roi = gray[y:y+h, x:x+w]

    # 표정을 인식하기 위해 표정 dataset과 똑같은 사이즈 변환
    # dataset 이미지와 입력된 얼굴의 크기가 다르면 error 발생
    face_roi = cv2.resize(face_roi, (64, 64))
    face_roi = np.expand_dims(face_roi, axis=-1)
    face_roi = np.expand_dims(face_roi, axis=0)
    face_roi = face_roi / 255.0

    # 모델을 통해 표정 분석
    output = model.predict(face_roi)[0]

    # 해당 표정의 값 반환
    expression_index = np.argmax(output)

    # 표정에 따른 label 값 저장
    expression_label = expression_labels[expression_index]
```

- 얼굴 표정 감지: 사용자의 얼굴을 인식, 감지하고 턱, 입의 위치 및 모양 눈을 뜬 정도와 같은 사용자의 얼굴 특징을 분석하여 표정을 결정하도록 만들었습니다. 표정에 따라 dataset을 구축하여 인공지능을 통해 학습

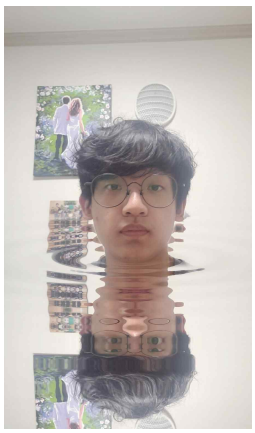
```
# 얼굴 인식
face_cascade = cv2.CascadeClassifier('c:/j/haarcascade_frontalface_default.xml')

# 표정 인식을 위한 눈, 코, 입등의 위치 반환
predictor = dlib.shape_predictor('c:/j/shape_predictor_68_face_landmarks.dat')

# 표정 라벨링
expression_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']

# 표정 가중치 모델
model = load_model('c:/j/emotion_model.hdf5')

expression_label = None
```

결과물의 활용방안 및 기대효과

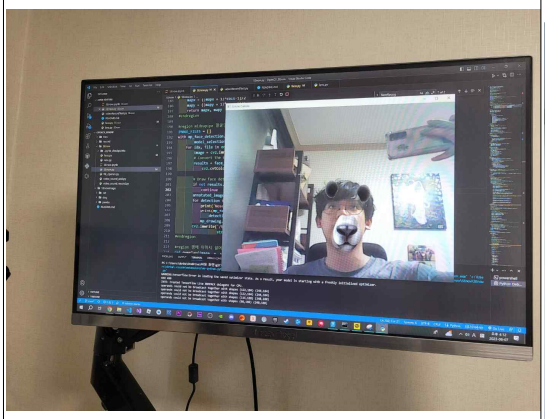
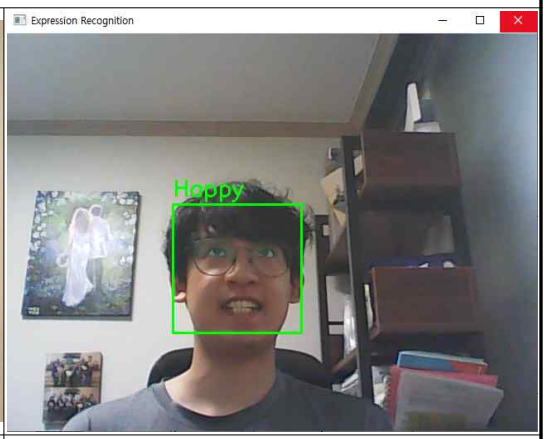
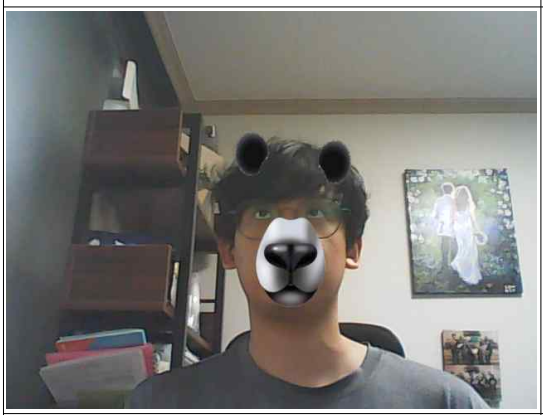
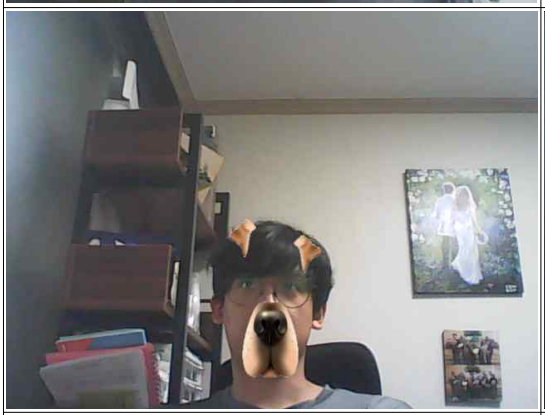
- 제가 사용한 이 기술은 증강현실(Augmented Reality, AR)과 관련이 많습니다. 실제 환경에 이미지, 동영상, 3D 모델과 같은 가상의 요소를 중첩시켜 사용자의 인식 및 주변 환경과의 상호 작용을 향상시키는 기술을 말합니다. 일반적으로 스마트폰, 태블릿 또는 스마트 안경을 통해 컴퓨터에서 생성된 콘텐츠를 사용자의 실시간 보기와 결합합니다. AR은 이미 다양한 산업에서 상당한 관심과 채택을 얻었으며 향후 전망은 매우 유망합니다.

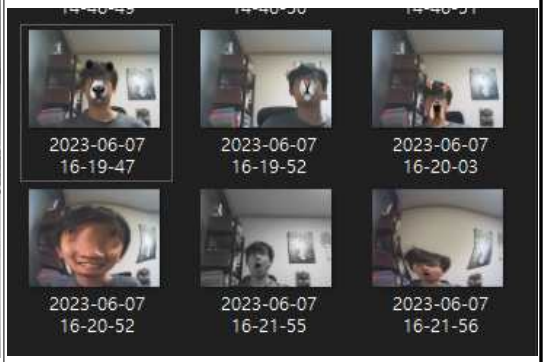
게임 (포켓몬 고)	IT 분야 (증강현실을 이용한 가상 모니터)
	

- SNS에서 제가 만든 필터 카메라처럼 해주는 많은 프로그램들이 존재하는데 필터 카메라는 사용자가 변환된 사진과 비디오를 캡처하고 공유할 수 있도록 하여 소셜 공유를 용이하게 합니다. 독특하고 재미있는 필터 변환이 소셜 미디어 플랫폼에서 관심과 참여를 끌 가능성이 높기 때문에 사용자 생성 콘텐츠 및 입소문 공유를 장려합니다.

TIKTOK	SNOW
	

- 이 결과물 자체를 활용하지 않더라도 수많은 분야에 이 기술을 접목시켜 새로운 것을 만들어 낼 수 있을 것입니다.

	구분	성명	과제 참여 내용(역할)
수행 방법	팀장	오세훈	python, opencv, AI, image learning
	팀원		
	팀원		
	팀원		
	팀원		
	팀원		
결과물			
			
			

		 2023-06-07 16-19-47 2023-06-07 16-19-52 2023-06-07 16-20-03 2023-06-07 16-20-52 2023-06-07 16-21-55 2023-06-07 16-21-56