딥러닝 프로젝트

황용구, 김태환, 박찬민

주제

- 배송 로봇이 택배 상자에 적힌 qr 코드를 인식하여 주소 정보를 받아 sql에 저장 후 YOLO를 이용한 엘리베이터 버튼 인식

1. qr코드 인식

- qr 코드를 감지하고, 감지된 qr 코드의 내용을 파싱하여 sql 데이터베이스에 저장하고. 음성 메시지를 출력한다.

1. qr 코드 인식 - 코드

```python
import qrcode
import cv2
import pyzbar.pyzbar as pyzbar
from playsound import playsound
import mysql.connector
import re

def extract_floor_from_address(address):
    match = re.search(r'(\d{3,4})호', address)
    if match:
        ho = match.group(1)
        return int(ho[:2]) if len(ho) == 4 else int(ho[0]) if len(ho) == 3 else None
    else:
        return None

def invoice_save():
    data_list = []
    used_codes = []
    split_code = []
    window_closed = False  # 창을 닫을지 여부를 나타내는 변수

    try:
        f = open("qrbarcode_data.txt", 'r', encoding='utf8')
        data_list = f.readlines()
    except FileNotFoundError: # try를 통해 코드를 실행했을 때, 만약 오류가 나면 except를 실행
        pass
    else:
        f.close()

    cap = cv2.VideoCapture(0)

    for i in data_list:
        used_codes.append(i.rsplit('\n'))
```

1. qr코드 인식 - 코드

```
while True:
    success, frame = cap.read()

    if success:
        cv2.imshow('cam', frame)

        for code in pyzbar.decode(frame):
            cv2.imwrite('my_qr_code.png', frame)
            my_code = code.data.decode('utf-8')
            if my_code not in used_codes:
                print('인식 성공 : ' , my_code)
                print(my_code.split(','))
                split_code = my_code.split(',')
                playsound("/home/pcm/qr&barcode/data/qrbarcode_beep.mp3")
                used_codes.append(my_code)

                f2 = open('qrbarcode_data.txt', 'a', encoding='utf8')
                f2.write(my_code+'\n')
                f2.close()

                window_closed = True  # 창을 닫을 준비
                break
            else:
                print('이미 인식된 코드입니다.')
                playsound("/home/pcm/qr&barcode/data/qrbarcode_beep.mp3")

    key = cv2.waitKey(1)
    if key == 27:  # 'Esc' 키를 누르면 루프를 종료
        break

    if window_closed:  # 창을 닫을 준비가 되면
        cv2.destroyAllWindows()  # 창을 닫음
```

1. qr코드 인식 - 코드

```
if split_code:
    address = split_code[0]
    name = split_code[1]
    phone_number = split_code[2]
    floor = extract_floor_from_address(address)

    remote = mysql.connector.connect(
        host="final-database.chobs2hlpex2.ap-northeast-2.rds.amazonaws.com",
        port=3306,
        user="admin",
        password="12345678",
        database="vision"
    )

    cur = remote.cursor()

    cur.execute(f"INSERT INTO invoice (address, floor, name, phonenumber) VALUES ('{address}', '{floor}', '{name}', '{phone_number}')")

    cur = remote.cursor(buffered=True)
    cur.execute("SELECT * from invoice")

    result = cur.fetchall()
    for result_iterator in result:
        print(result_iterator)

    remote.commit()
    remote.close()
    return floor
```

2. 목소리 출력

- TTS(Test-to-Speech) 모듈을 사용하여 텍스트를 음성으로 변환하여 말을 재생

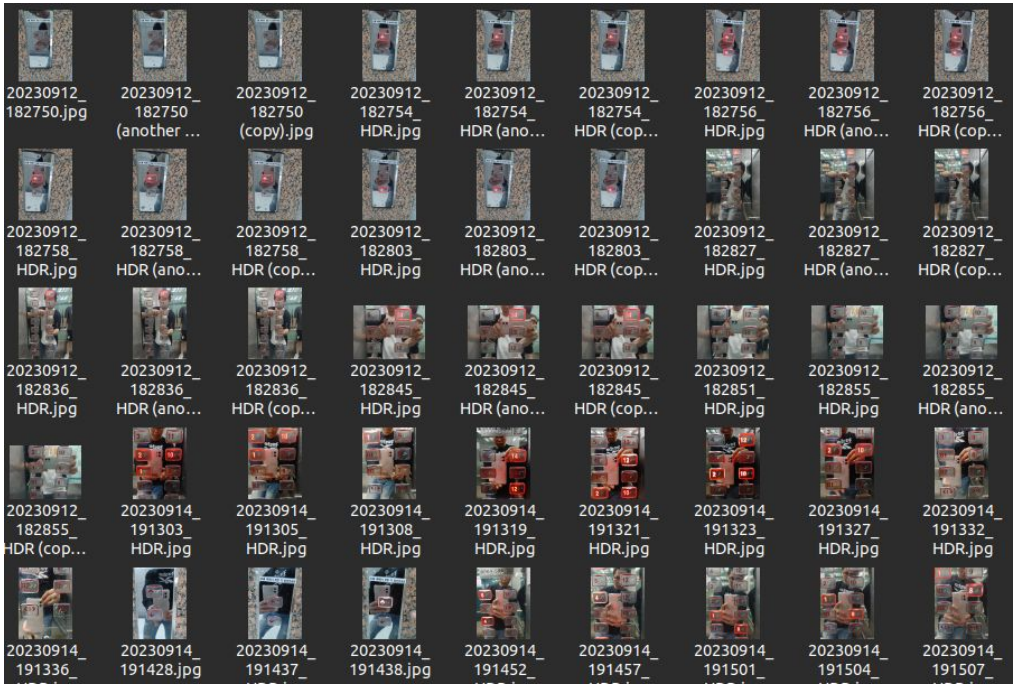
2. 목소리 출력 - 코드

```python
#pip inatall gtts playsound==1.2.2
from gtts import gTTS
from playsound import playsound
# # 영어
# audio = '/home/addinedu/dev_ws/gtts/speech_en.mp3'
# language = 'ko'
# speech = gTTS(
#     lang=language,
#     text='Hello, My name is add in edu . Nice to see you. Have a good day!',
#     slow=False
# )
# speach.save(audio)
# playsound(audio)
#한글
def voice(data):
    audio = '/home/pcm/dev_ws/urdf_study/build/urdf_study/CMakeFiles/3.27.1/CompilerIdC/tmp/speech_ko.mp3'
    text = '/home/pcm/dev_ws/urdf_study/build/urdf_study/CMakeFiles/3.27.1/CompilerIdC/tmp/test.txt'
    language = 'ko'
    with open(text, 'w') as f:
        # data = "태환씨 취업 축하해요 잘 가요"
        f.write(data)
    with open(text, 'r') as f:
        data = f.read()
    speech = gTTS(
        lang=language,
        text=data,
        slow=False
    )
    speech.save(audio)
    playsound(audio)


if __name__ == "__main__":
    print("파일이 직접 실행됨.")
```

3. 버튼 인식 - 사진촬영

- train 588장
- valid 257장
- 총 845장으로
- 라벨링후 훈련 실시



가중치및 각종 매개변수 조정

```
def get_args_parser(add_help=True): #add_help인자를 기본값으로 받는 get_args_parser 함수를 정의합니다.
    parser = argparse.ArgumentParser(description='YOLOv6 PyTorch Inference.', add_help=add_help) #argparse를 사용하여 명령행 인자 파서를 생성하고 설명과 도움말 옵션을 추가
    parser.add_argument('--weights', type=str, default='/home/taen/dev_ws/YOLOv6/weights/best_ckpt.pt', help='model path(s) for inference.')#모델 가중치 경로를 지
    parser.add_argument('--source', type=str, default='data/images', help='the source path, e.g. image-file/dir.')#입력 소스 경로를 지정하는 명령행 인자를 추가합니다. 사
    parser.add_argument('--webcam', action='store_true', default=True, help='whether to use webcam.') #웹캠 사용 여부를 나타내는 명령행 인자를 추가 합니다. action= 'sto
    parser.add_argument('--webcam-addr', type=str, default='/dev/video1', help='the web camera address, local camera or rtsp address.')#웹 캠 주소를 지정하는 명령
    parser.add_argument('--yaml', type=str, default='/home/taen/dev_ws/YOLOv6/data/dataset.yaml', help='data yaml file.') #데이터 yaml파일 경로를 지정하는 명령행 인자
    parser.add_argument('--img-size', nargs='+', type=int, default=[640, 640], help='the image-size(h,w) in inference size.') #이미지 크기를 지정하는 명령행 인자를 추
    parser.add_argument('--conf-thres', type=float, default=0.5, help='confidence threshold for inference.')#신뢰도 임계값을 지정하는 명령행 인자를 추가합니다.
    parser.add_argument('--iou-thres', type=float, default=0.6, help='NMS IoU threshold for inference.')# nms iou임계값을 지정하는 명령행 인자를 추가합니다. 임계값보다 높
    parser.add_argument('--max-det', type=int, default=1000, help='maximal inferences per image.')#이미지당 최대 추론 개수를 지정하는 명령행 인자를 추가합니다.
    parser.add_argument('--device', default='0', help='device to run our model i.e. 0 or 0,1,2,3 or cpu.') #모델을 실행할 디바이스를 지정하는 명령행 인자를 추가합니다. 0은
    parser.add_argument('--save-txt', action='store_true', default=False, help='save results to *.txt.') #결과를 텍스트 파일에 저장할지 여부를 나타내는 명령인자
    parser.add_argument('--save-img', action='store_true', default=True, help='do not save visuallized inference results.')#시각화된 결과를 저장할지 여부를 나타내는 명
    parser.add_argument('--save-dir', type=str, default='/home/taen/dev_ws/YOLOv6/', help='directory to save predictions in. See --save-txt.') #예측 결과를 저장할
    parser.add_argument('--view-img', action='store_true', default=True, help='show inference results')#추론 결과를 화면에 표시할지 여부를 설정합니다.
    parser.add_argument('--classes', nargs='+', type=int, default=None, help='filter by classes, e.g. --classes 0, or --classes 0 2 3.') #특정 클래스 또는 클래스를
    parser.add_argument('--agnostic-nms', action='store_true', default=False, help='class-agnostic NMS.') #클래스에 관계없이 nms(비최대 억제)를 수행할지 여부를 설정합니다
    parser.add_argument('--project', default='runs/inference', help='save inference results to project/name.')# 추론 결과를 저장할 프로젝트 경로를 설정합니다.
    parser.add_argument('--name', default='exp', help='save inference results to project/name.') #추론 결과를 저장할 프로젝트 경로에 추가적으로 이름을 지정합니다.
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide labels.')#시각화된 추론 결과 이미지에서 라벨을 숨길지 여부를 설정 합니다. 만약 이 옵
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide confidences.')#시각화된 추론 결과 이미지에서 신뢰도(확률)를 숨길지 여부를 설정합니다.
    parser.add_argument('--half', action='store_true', default=False, help='whether to use FP16 half-precision inference.')#fp반정밀도(inference)를 사용할지 여부를
```

결과 값의 좌표를 구하고

```
or *xyxy, conf, cls in reversed(det):
    if save_txt:  # Write to file
        x = torch.tensor(xyxy).view(1, 4)
        y = x.clone() if isinstance(x, torch.Tensor) else np.copy(x)
        y[:, 0] = (x[:, 0] + x[:, 2]) / 2  # x center
        y[:, 1] = (x[:, 1] + x[:, 3]) / 2  # y center
        y[:, 2] = x[:, 2] - x[:, 0]  # width
        y[:, 3] = x[:, 3] - x[:, 1]  # height

        xywh = (y / gn).view(-1).tolist()  # normalized xywh
        line = (cls, *xywh, conf)
        print()
        with open(txt_path + '.txt', 'a') as f:
            f.write(('%g ' * len(line)).rstrip() % line + '\n')


    if save_img:
        class_num = int(cls)  # integer class
        label = None if hide_labels else (class_names[class_num] if hide_conf else f'{class_names[class_num]} {conf:.2f}')

        # Add one xyxy box to image with label
        lw = max(round(sum(img_ori.shape) / 2 * 0.003), 2)
        p1, p2 = (int(xyxy[0]), int(xyxy[1])), (int(xyxy[2]), int(xyxy[3]))
        center_point = round((p1[0] + p2[0])/2), round((p1[1]+p2[1])/2)
        cv2.circle(img_ori, center_point, 5, (0,255,0), 2)
        cv2.putText(img_ori,str(center_point),center_point,cv2.FONT_HERSHEY_PLAIN,2,(0,0,255))
        cv2.rectangle(img_ori, p1, p2, color, thickness=lw, lineType=cv2.LINE_AA)
```

중심값의 수치를 화면에 띄웁니다.

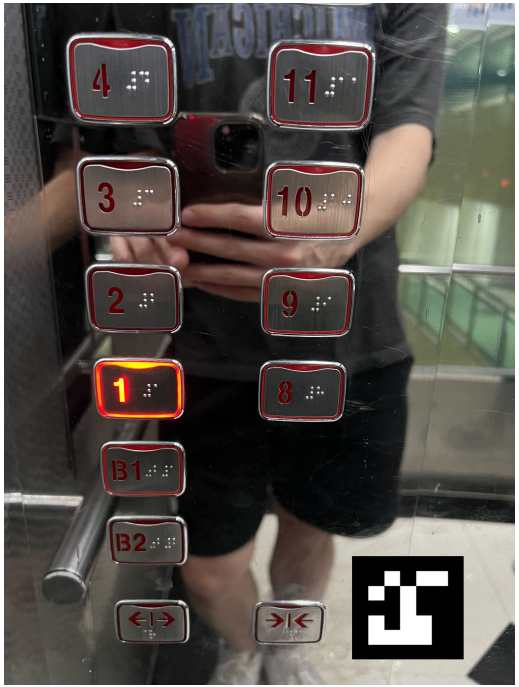
```
# Add one xyxy box to image with label
lw = max(round(sum(img_ori.shape) / 2 * 0.003), 2)
color=generate_colors(class_num, True)
p1, p2 = (int(xyxy[0]), int(xyxy[1])), (int(xyxy[2]), int(xyxy[3]))
center_point = round((p1[0] + p2[0])/2), round((p1[1]+p2[1])/2)
cv2.circle(img_ori, center_point, 5,(0,255,0),2)
cv2.putText(img_ori,str(center_point),center_point,cv2.FONT_HERSHEY_PLAIN,2,(0,0,255))
cv2.rectangle(img_ori, p1, p2, color, thickness=lw, lineType=cv2.LINE_AA)
if label:
    tf = max(lw - 1, 1)  # font thickness
    w, h = cv2.getTextSize(label, 0, fontScale=lw / 3, thickness=tf)[0]  # text width, height
    outside = p1[1] - h - 3 >= 0  # label fits outside box
    p2 = p1[0] + w, p1[1] - h - 3 if outside else p1[1] + h + 3
    cv2.rectangle(img_ori, p1, p2, color, -1, cv2.LINE_AA)  # filled
    cv2.putText(img_ori, label, (p1[0], p1[1] - 2 if outside else p1[1] + h + 2), cv2.FONT_HERSHEY_COMPLEX, lw / 3, (
                thickness=tf, lineType=cv2.LINE_AA)
    print(p1,p2)
```

시연 영상

시연 영상

시연 영상

4. 아루코 마커를 통한 거리 확인



- 버튼 인식과 아루코마커 인식을 동시에 진행
- 사진에 아루코 마커를 삽입하여 버튼과 카메라 사이의 거리 측정

거리 값과 코너값 산출

```python
def pose_esitmation(frame, aruco_dict_type, matrix_coefficients, distortion_coefficients):#함수를 입력된 프레임에서 아루코 마커를 검출하고 해당 마커의 포즈(위치와 자세)를 추

    value = None
    corners = None
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)#입력 프레임을 흑백 이미지로 변환 합니다.
    cv2.aruco_dict = cv2.aruco.Dictionary_get(aruco_dict_type)#사용할 아루코마커 사전을 설정합니다. 별도의 지정이 없으면 DICT_ARUCO_ORIGINAL이 기본값으로 설정되어 있으므로 별도
    #만약 특별하게 다른 딕트를 이용하려면 aruco_dict_type = cv2.aruco.DICT_6X6_250 이런식으로 넣어주면 된다.
    parameters = cv2.aruco.DetectorParameters_create() #아루코마커 검출을 위한 파라미터 설정을 생성합니다.

    corners, ids, rejected_img_points = cv2.aruco.detectMarkers(gray, cv2.aruco_dict, parameters=parameters) #아루코마커를 검출하고 마커의 코너 좌표(corners),id및 거부
    print(f"ids : {ids}")
        # If markers are detected
    if len(corners) > 0: #마커가 검출되었는지 확인합니다.
        for i in range(0, len(ids)):# 검출된 각 마커에 대한 반복문 입니다. 모든 마커에 대해 아래 동작을 수행합니다.
            # Estimate pose of each marker and return the values rvec and tvec---(different from those of camera coefficients)
            rvec, tvec, markerPoints = cv2.aruco.estimatePoseSingleMarkers(corners[i], 0.03, matrix_coefficients,#여기서 0.03은 한변의 길이를 나타냅니다.
                                                                           distortion_coefficients) #이함수를 사용하여 각 마커의 포즈(회전 벡터 rvec 및 변환 벡터 tvec)
            #이 함수는 검출된 마커의 코너좌표를 corners에 마커의 id를 ids에 거부된 이미지 포인트를 rejected_img_points에 반환합니다. 마커가 검출되지 않으면 corners, ids rejected_i
            # 회전 벡터를 회전 행렬로 변환
            r_matrix, _ = cv2.Rodrigues(rvec)
            # 마커의 실제 세계 좌표 계산
            real_world_coordinates3d = -np.matmul(np.linalg.inv(r_matrix), tvec[0, 0].reshape(-1, 1))
            image_coordinates2d, _ = cv2.projectPoints(real_world_coordinates3d, rvec, tvec, matrix_coefficients, distortion_coefficients)
            # print(f"image_coordinates2d{image_coordinates2d}")
            # print(f"real world coordinates3d: {real_world_coordinates3d}")
            # print(f"rotation vector : {rvec}")
            # print(f"translation vector : {tvec}\n")
            # print(markerPoints)
            value = round(tvec[0, 0, 2] * 100, 1) + 6
            print(value)
            # Draw a square around the markers
            # cv2.aruco.drawDetectedMarkers(frame, corners) #함수를 사용하여 입력 프레임에 검출된 아루코 마커 주위에 사각형을 그립니다. 이렇게 그려진 프레임은 마커가 감지된 것을 시각

    return value, corners
```

아루코 마커와 버튼인식을 동시에 실시

```
''' Model Inference and results visualization '''
windows = [] #비어있는 리스트 windows를 생성합니다. 이 리스트는 이후 결과 이미지를 저장 할 때 사용됩니다.
for img_src, img_path, vid_cap in tqdm(files):#files 에서 이미지 소스 이미지경로 비디오 캡처 객체를 하나씩 반복하면서 이미지에 대한 추론을 수행합니다.

    value, corners = pose_esitmation(img_src, aruco_dict_type, k, d)

    img = letterbox(img_src, img_size, stride=stride)[0]#letterbox함수를 사용하여 이미지를 크기 조정합ㄴ디ㅏ. 이함수는 이미지를 원하는 크기와 스트라이드에 맞게 조
    # Convert
    img = img.transpose((2, 0, 1))[::-1]  # HWC to CHW, BGR to RGB 이미지의 차원 순서를 변경하여 HWV(높이, 너비, 채널)에서 CHW(채널, 높이, 너비)변경하고, BGR
    img = torch.from_numpy(np.ascontiguousarray(img))#넘파이 배열에서 파이토치 텐서로 이미지를 변환합니다.
    img = img.half() if half else img.float()  # uint8 to fp16/32  half변수가 true이면 이미지를 16비트 부동 소수점 형식으로 변환하고, 그렇지 않으면 32비트 부동
    img /= 255  # 0 - 255 to 0.0 - 1.0 이미지의 픽셀 값 정규화
    img = img.to(device)# 이미지를 목표 디바이스로 이동 시킨다. 이로써 모델을 사용하여 추론을 수행할 디바이스로 이미지가 전송됨
    if len(img.shape) == 3: #이미지의 차원을 확인하고 3차원 이미지인 경우에는 배치 차원을 추가하여 4차원 텐서로 만듭니다. 이것은 모델에 입력으로 공급하기 위해 필요한 작업
        img = img[None]
        # expand for batch dim
    pred_results = model(img)#모델을 사용하여 이미지에 대한 추론을 수행하고 추론 결과를 pred_results 변수에 저장합니다.
    det = non_max_suppression(pred_results, conf_thres, iou_thres, classes, agnostic_nms, max_det=max_det)[0]#non_max_suppression 함수를 사용하여
```

화면에 거리 수치 띄우기

```
            print(pipe)

        img_src = np.asarray(img_ori)

        cv2.putText(img_src,'distance: '+str(value),(460,474), cv2.FONT_HERSHEY_COMPLEX, lw / 3, (255, 255, 255),
                    thickness=tf, lineType=cv2.LINE_AA)

        cv2.aruco.drawDetectedMarkers(img_src, corners)
    if view_img:
        if img_path not in windows:
            windows.append(img_path)
            cv2.namedWindow(str(img_path), cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
            cv2.resizeWindow(str(img_path), img_src.shape[1], img_src.shape[0])
        cv2.imshow(str(img_path), img_src)
        key = cv2.waitKey(1)  # 1 millisecond
        if key == ord('q'):
            cv2.destroyAllWindows()
            break  # Exit the loop when 'q' is pressed
```

4. 시연 영상

Q & A

감사합니다.