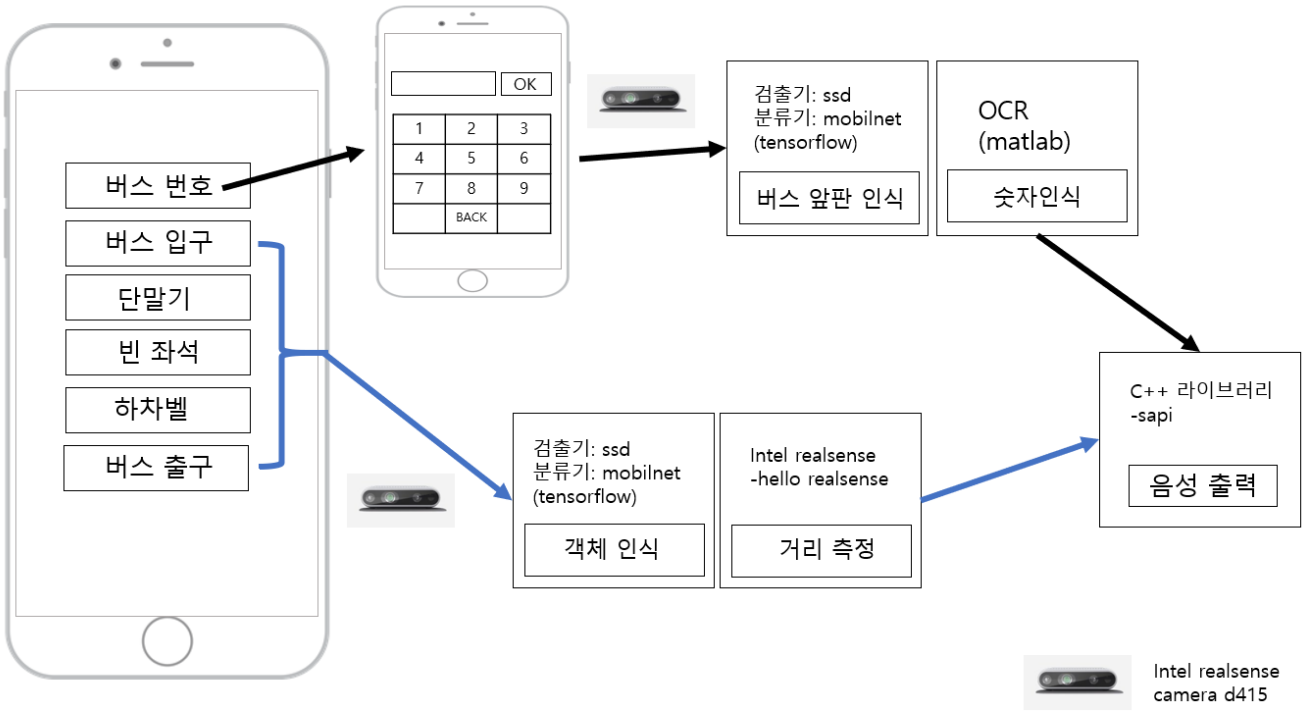
시각장애인을 위한 버스 보조 응용프로그램

김해린, 백소현, 이세미, 이세정
세종대학교 지능기전공학부
https://github.com/lini1634/2020-Capstone-Design

요 약



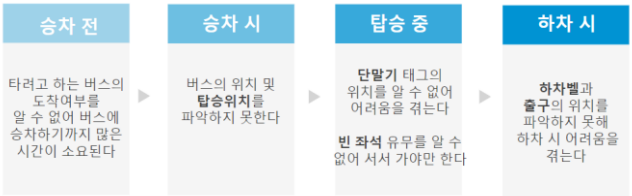
웹앱에서 사용자가 버스 번호, 단말기, 하차 벨, 빈 좌석, 버스 입구의 버튼을 누르면 그 객체를 인식해서 사용자와 객체까지의 거리를 음성으로 출력하여 사용자에게 전달한다. 웹앱은 python Flask 프레임워크를 기반으로 하였고, 버스 숫자인식은 matlab OCR알고리즘을 이용하였다. 각 객체의 데이터를 직접 촬영하여 수집하였고 tensorflow기반의 ssd 검출기와 mobilenet 분류기, faster rcnn검출기와 inception분류기를 비교, 사용하였다. 음성인식은 C++라이브러리인 sapi를 이용하였다.

1. 개발 동기와 목적

1.1 개발 동기

”버스 탄 시각장애인 봤나요… 타고 내리는게 전쟁…” 이라는 주제의 TV 프로그램을 보았다. 앞이 전혀 보이지 않는 시각장애인은 버스가 어디에 섰는지, 버스 안에 빈자리가 있는지 등 대중교통 이용에 있어서 큰 어려움을 겪고 있었다. 시각 장애인은 보조 기구, 수단으로써 안내견, 점자, 지팡이를 이용하고 있지만 우리는 그 이외의 방법으로 시각장애인의 대중교통(버스) 이용을 도울 수 있는 방법에 대해 고찰해 보았다. 사용자로부터 거리를 측정함으로써 사람의 눈을 닮은 depth camera 를 통해 시각장애인의 눈이 되어보고자 했다. 이번 프로젝트에서는 승차 전, 승차 시, 탑승 중, 하차 시로 나누어 문제점을 해결한다. 승차 전에는 사용자가 타고자 하는 버스 번호를 입력하고 버스 정류장에 오는 버스 번호를 인식하여 버스를 확인할 수 있도록 한다. 승차 시에는 버스 입구를 인식하여 사용자와 버스 입구로부터의 거리를 음성으로 알려준다. 탑승 중에는 단말기, 빈 좌석을 인식하여 사용자부터의 거리를 측정하고 음성으로 안내해준다. 하차시에는 하차 벨 위치를 앞선 방법과 동일하게 적용하여 사용자에게 알려준다.

1.2 개발 목적



2. 개발 과정

이미지 인식을 하기 위해 버스, 하차 벨, 단말기, 빈 좌석, 버스 출입구 이미지 데이터를 수집하고 객체인식을 위해 라벨링 하였다. 숫자인식 알고리즘은 MATLAB 개발환경의 OCR 알고리즘을 사용하였고 객체인식의 검출기는 Python 의 tensorflow 기반의 SSD, FASTER-RCNN 과 분류기는 MOBIL-NET, Inception 을 이용하였다. 이후 FLASK 를 기반으로 웹 애플리케이션의 백 엔드를 개발하고 HTML 과 JAVA-SCRIPT 로 프론트 엔드를 개발하였다.

2.1 숫자 인식

숫자를 인식하는 범위를 종이 버스 번호판으로 한정하여 데이터 수집을 하고 이를 labelImg프로그램을 통해 라벨링 하였다. matlab프로그램에서 버스 번호판을 객체로 인식한 결과인 좌표 값과 카메라에서 원본 이미지를 받아와 그에 맞게 이미지를 자르고 사이즈를 통일하였다. 그 후 이미지를 흑백영상으로 바꾸고 노이즈를 제거한 뒤 대비를 통하여 이진영상으로 바꾸어 숫자를 명확하게 하는 데이터 전처리 작업을 거쳤다. 그 다음으로 matlab에 있는 ocr함수를 이용하여 숫자인식을 하여 버스 번호를 인식하였다.

2.2 객체 인식과 거리측정

하차 벨 227개, 버스 입구 237개, 단말기 153개, 빈 좌석 225개, 버스 번호가 포함된 버스 226개 이미지 데이터를 수집한 후 LabelImg프로그램으로 객체의 좌표가 저장되어 있는 xml파일을 생성했다. Python 3.6, tensorflow 1.15.0 버전의 환경에서 ssd 검출기와 mobilenet 분류기가 동시에 실행되는 object_detection_helper tool*을 이용하여 학습했다. 6가지 종류의 이미지를 40000번의 step으로 학습하는데 계산 속도 향상을 위해 GPU사용이 가능한 구글 Colaboratory를 이용하였고 총 열시간정도가 소요되었다. 매 스텝당 약 0.4~0.6초가 걸렸고 동일한 작업을 CPU로 실행했을때는 스텝당 3~4초로 열배정도의 계산속도차이를 보였다. Depth camera로 실시간 촬영되는 영상의 이미지를 사용하기 위해 Intel Realsense의 save-to-disk(.cpp)를 이용하여 캡쳐되는 color이미지 파일을 학습된 모델(.py)로 객체인식을 진행하였고 이를 위해서는 객체인식파일을 cpp파일에서 실행할 수 있도록 연동이 필요했다. 또한 depth이미지 파일은 인식된 객체의 x, y평균 좌표를 추출하여 Intel Realsense의 hello-realsense파일의 코드를 참고하여 거리를 측정하였다.

2.2 웹 애플리케이션 제작

Html로 버스 번호, 버스 입구, 단말기, 하차벨 빈좌석을 버튼 6개로 구성하여 웹의 UI를 개발하였다. javascript를 이용해서 버튼을 눌렀을 때 새로운 브라우저로 이동할 수 있도록 했다. Python의 Flask웹 프레임 워크를 이용하여 새로운 브라우저에서 앞서 제작한 숫자인식과 객체인식 응용프로그램을 실행하도록 개발하였다.

3. 성능향상 노력 및 분석

카테고리별 정확도를 산출했을 때 빈좌석과 단말기의 정확도가 가장 낮았다. 단말기의 경우에는 단말기 기종을 한가지 종류로 한정하여 데이터 수집을 하였고 이후 96%의 정확도를 얻었다. 빈좌석의 경우, 버스 노약자석과 임산부석으로 좌석을 한정하였지만 40%의 AP 로 향상되지 않았다. 데이터셋의 근본적인 수정이 필요하다고 생각하였다. 처음 구글에서 크롤링한 데이터셋은 하차 벨 193 개, 버스 출구 17 개, 버스 입구 39 개, 단말기 47 개, 빈 좌석 50 개, 버스 번호가 포함된 버스 50 개 정도의 이미지 데이터로 카테고리별 난이도를 고려하지 않았고 화질이 좋지 않았다. 그래서 직접 버스에서 하차 벨, 빈 좌석, 입구, 단말기의 이미지를 각각 약 200 장씩 수집하였다. 구글에서 이미지를 크롤링 하는 것과 비교하여 고화질의 이미지 데이터를 얻을 수 있었다.

위 그래프는 faster rcnn, inception 과 ssd, mobile net 을 비교한 결과이다. 두 모델 모두 40000 번 학습했을 때, faster rcnn, inception 이 정확도가 더 높았기 때문에 최종적으로 faster rcnn, inception 모델을 사용했다.

참고자료

[1] https://github.com/5taku/tensorflow_object_detection_helper_tool

[2] https://github.com/tzutalin/labelImg

[3] Intel Realsense SDK 2.0