IoT와 딥러닝을 활용한 대중교통 버스 승·하차 계수 시스템 개발 연구

(Developing Public Transportation Bus Passenger Counting System Based on IoT and Deep Learning)

이 호 성 ††　　이 진 우 ††　　김 성 백 †††

(Ho-Sung Lee)　　(Jin-Woo Lee)　　(Seong-Baeg Kim)

요 약 각 지역에 대중교통 버스정보시스템을 운영하며 편의를 제공하고 있다. 하지만 대부분 버스 재차 인원 정보는 제공되고 있지 않다. 그리고 차내 혼잡도 형태로 재차 인원 정보가 제공되는 일부 지역의 경우에는 정보가 정확하지 못한 문제점이 있었다. 그 원인은 재차 인원 계산을 위한 승객 정보 수집이 카드 태그로 이루어져서 하차 미 태그 승객과 현금 사용 승객을 계수하지 못했기 때문이다. 이에 본 논문에서는 도출한 문제점을 해결하기 위해 버스 승·하차 계수 시스템을 제안하고 이를 설계 및 개발·구현한다. 기존 연구와 다르게 본 연구는 동영상 처리의 컴퓨팅 연산 부담을 줄이기 위해 IoT와 클라우드 환경을 활용하여 빠르고 효율적인 계수를 가능하게 하는 것을 목표로 한다. 제안된 시스템은 IoT 센서 활용 승객 감지와 딥러닝을 통한 승·하차 계수로 나누어진다. 각각 IoT 센서를 통해 승객이 승·하차 계단으로 접근하는 것을 파악하고, 딥러닝 활용 안면 움직임 추적을 통해 승·하차 계수를 판단하였다. 결론적으로, 본 시스템을 활용하면 교통 카드 태그와 상관없이 모든 버스 승객의 승·하차를 계수하고 나아가 이를 데이터화하고 정보를 활용할 수 있다.

키워드 : 딥러닝, 안면 인식, 버스 승·하차 계수 시스템, IoT, 교통 카드

Abstract Each local self-governing body has been operating the public transport bus information system for bus convenience. However, information on the number of bus passengers is not being provided. Additionally, there exists a problem in some regions in that the bus passenger number information is inaccurate. The reason is that the method of determining the number of people on the bus is accomplished based on card tag information collecting. Thus, the system doesn't factor passengers using cash instead of the card. In this paper, we propose a bus boarding and de-boarding counting system to resolve the derived problem and develop it. Unlike the previous studies, the purpose of this study was to enable rapid and efficient counting by using the IoT and cloud environments to reduce the computational burden of video processing. The proposed system is classified into passenger detection using the IoT sensors and boarding/de-boarding counting through deep learning. Each IoT sensor identifies passengers' access to the boarding and de-boarding stairs, and decisively determines the boarding and de-boarding counting through facial motion tracking using deep learning. In conclusion, the use of this system has the advantage of counting all bus passengers boarding/de-boarding, regardless of the transportation card tag, and further converting counts into data and using information.

Keywords: deep learning, face recognition, bus counting system, IoT, transportation card

• 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었습니다.
• 논문 작업에 조언을 준 광주과학기술원(GIST) 전기전자컴퓨터공학부 김예찬 연구원님께도 감사의 말씀을 전합니다.

논문접수 : 2020년 00월 00일
(Received 00 00 2020)
논문수정 : 2020년 00월 00일
(Revised 00 00 2020)
심사완료 : 2020년 00월 00일
(Accepted 00 00 2020)

†† 비 회 원 : 제주대학교 전산통계학과 교수(Jeju Nat'l Univ.)
kimcs@jejunu.ac.kr
††† 종신회원 : 제주대학교 컴퓨터교육과 교수(Jeju Nat'l Univ.)
sbkim@jejunu.ac.kr
(Corresponding Author임.)

정보과학회논문지 제00권 00호(2020.0)

1. 서 론

최근 제주특별자치도에서는 제주버스정보시스템을 운영하며 버스 편의를 제공하고 있다. 해당 시스템은 버스 도착시간, 위치, 배차정보 등을 활용하여 이용시민에게 버스 관련 정보를 제공한다. 하지만 제주 버스에서는 재차인원에 대한 정보가 제공되지 않는다. 제주 이외 일부 타 지역의 경우 버스 정보 시스템에서 차내 혼잡도 형태로 재차인원정보가 제공되었는데, 해당 정보가 정확하지 못했다. 그 원인은 버스 내 승객 정보 수집이 교통 카드 태그로 이루어져 하차 미 태그 승객과 현금 사용 승객을 계수하지 못했기 때문이다.

제주에서도 승객의 승차 및 하차 정보를 얻기 위한 방식으로 카드 태그 방식을 사용한다. 실제 카드 미태그로 인한 데이터 결측이 드러나는지 파악하기 위해 제주 빅데이터 센터로부터 제공받은 2019년 6월 1일자 버스 승객 카드 이용 데이터(tb_bus_usage)를 통해 카드 미 태그 발생 빈도수를 분석했다[1].

버스 승객 이용 데이터 인스턴스의 총 개수 130,980건 중, 약 44,000건의 데이터 인스턴스의 하차 정보가 미결측으로 나타났다. 승객의 하차 정보가 'NULL'인 미결측 데이터들은 하차 미 태그 상황에 발생한다. 해당 데이터는 총 인스턴스의 약 33.5%를 차지하며, 이 수치는 서울의 차내 혼잡도 정보 평가와 마찬가지로, 내부 승객 인원 계산에 이용할 경우 부정확한 결과가 도출될 것으로 판단된다.

버스 승객 카드 이용 데이터로는 현금을 사용하는 승객 이용량 측정이 불가하다는 문제가 존재했다. 제주특별자치도의 제3차 지방대중교통계획 보고서에 따르면, 제주도의 교통카드 이용실적은 지속해서 증가하고 있으나 다른 지역에 비해 이용실적이 부족했다. 해당 보고서에 따르면 교통카드 이용률이 총 버스 이용 건수 36,003,083건 중 63.6%이었다 [2]. 교통카드 이용률이 점진적으로 늘어나겠지만 일정 부분 현금 사용 인원이 포함되어 있다. 따라서 실제 내부 인원 계산에 현금 사용 승객 인원이 포함되어 측정되어야 정확한 재차 인원을 파악할 수 있다.

현재 제주 버스 내부 인원 제공의 어려움은 교통 카드 사용 승객과 현금 사용 승객 모두 정확한 측정이 어렵기 때문이다. 실시간으로 승·하차 인원의 정확한 측정이 가능하다면, 그를 통해 버스 별 내부 인원 정보 제공 또한 가능해진다 [3]. 따라서 실시간 승·하차 인원을 정확히 측정할 수 있는 계수 시스템을 연구하여 개발하고자 한다. 이러한 계수 시스템은 하차 태그와 현금을 사용하는 국내의 모든 지역 버스에 적용할 수 있다.

2. 선행 연구 분석

버스 내부 인원 정보를 정확하게 측정하기 위해 계수 방법 관련 선행 연구를 분석한 결과 대표적으로 영상처리 분야의 천장형 카메라 계수 연구가 많이 진행되었다.

천장형 카메라 계수 연구는 일반적으로 실시간으로 비디오카메라로 촬영된 일련의 영상 프레임을 분석하게 된다. 프레임과 프레임을 비교하며 변화하는 픽셀 색상값들을 보고 움직임을 파악하거나[4], 일관된 픽셀 색상값들을 통해 물체의 형태를 파악하는 승·하차를 감지하는 방식을 채택하였다 [5].

해당 연구들에서 발견된 문제점은 버스 운행 중 생기는 진동에 의해 카메라가 흔들리면 픽셀 색상값 기준으로 계산하여 발생하는 오류가 생길 수 있다는 점이다. 또한 운행 중에 광원의 변화로 밝기나 그림자가 달라지면 픽셀들의 색상값도 바뀌기 때문에 원활한 계산이 어려운 상황이 많이 발생한다. 또한 카메라가 버스 운행 중에 계속 동작 할 경우, 처리해야할 데이터도 많아지며, 그 데이터를 분석하기 위한 컴퓨팅 파워 또한 비례하여 증가하게 된다. 이러한 문제들을 줄일 수 있는 계수 방법과 해결책이 필요하게 되었다.

프레임간의 픽셀 데이터 차이를 고려하지 않고, 밝기 변화에 덜 영향 받는 비디오 프로세싱 방법을 딥러닝 기술을 가지고 접근한다. 인공지능 딥러닝 기술을 이용하여 사람의 안면을 파악, 승객을 감지하고 안면의 움직임을 추적하는 것으로 승·하차 계수를 판단하면 밝기 변화와 진동에 덜 민감한 계수 방법이 될 것이라 판단된다.

선행 연구들에서의 부족한 부분은 실시간 처리를 고려하지 않았다는 점이다. 선행 연구들은 영상을 촬영 후에 나중에 프로그램을 통해 데이터를 처리하고 확인하는 것으로 연구를 진행했다. 본 연구는 데이터를 처리하고 승·하차 인원을 전송하는 IoT 시스템을 적용하여 기존 선행 연구와 차별점을 두었다.

또한 IoT 센서로 승·하차 승객을 감지될 때만 영상을 촬영하여 경제성과 효율성을 확보할 수 있을 것이다. 무선 센서등[6]와 스마트 원격 감시 장치[7]에 대한 사례에서 PIR 센서를 통해 효과적인 인체 감지가 가능하다는 점을 주목했다. 본 연구는 선행 연구와 다르게 PIR센서 뿐 아니라 초음파 센서를 활용하여 승·하차 승객의 거리를 파악하여 더 정확하게 판별한다.

3. 설계

제안 시스템은 동영상 처리의 많은 처리량을 줄이기 위한 IoT와 클라우드 환경을 활용하여 빠르고 효율적인 계수를 목표로 한다. 제안한 버스 승·하차 계수 시스템의 설계는 IoT 모듈과 클라우드 딥러닝 분석으로 구분한다. 이 구분은 IoT 모듈의 감지 역할을 이용해 필요한 구간만 촬영을 하여 효율적인 분석이 가능하도록 한다.

3.1 전체 설계도

그림 1은 승·하차 계수기 시스템의 전체적인 구성요소를 보여주고 있다. 그림에서 보는 것처럼 버스 내에서 데이터

를 수집하는 IoT 모듈과 클라우드 상의 딥러닝 기반 승·하차 계수 프로그램으로 구분할 수 있다.

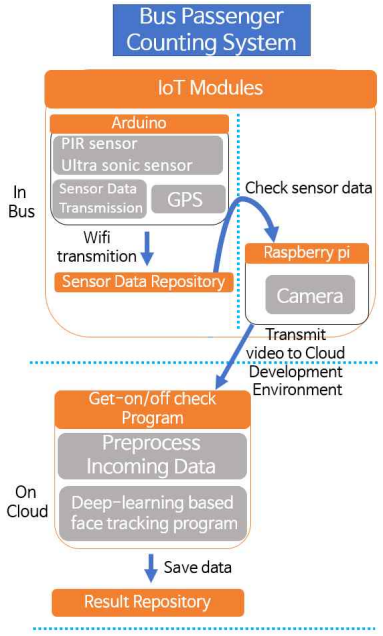

그림 1 전체 모듈별 설계도
Fig. 1. Overall module design diagram.

버스 내에 설치된 IoT 모듈은 크게 아두이노(Arduino)와 라즈베리파이(Raspberry PI)로 구성된다. 아두이노에는 GPS, PIR, 초음파 센서 등이 부착되어 있으며, 센서 값을 Wifi 모듈을 이용한 통신을 통해 데이터베이스에 저장한다. 라즈베리파이에는 카메라가 부착되어 있다. 라즈베리파이는 데이터베이스에서 센서 값을 읽으면서 승객이 있다고 판단될 때 트리거로서 카메라를 작동시킨다. 비디오와 센서 데이터를 클라우드에서 처리하여 데이터 저장소에 저장한다.

3.2 IoT 모듈 설계

IoT 모듈 부분은 PIR 센서 및 초음파 센서, GPS, 센서 데이터 전송, 카메라 4개의 구성 요소로 이루어진다. 설계 내용을 구성요소 단위로 소개하겠다.

3.2.1 PIR 센서 및 초음파 센서

PIR 센서와 초음파 센서는 승객의 감지 및 위치 파악을 위해 사용되는 센서이다. 이 센서들은 출입구 접근 승객 파악 센서이다. 출입구 접근 승객 파악 센서는 IoT 센서들을 활용하여 문에 접근하는 승객을 감지하는 동작을 수행한다. 이 역할을 수행하기 위해 PIR 센서와 초음파 센서를 선정한다. 해당 시스템 구성요소의 센서 컨트롤러는 GPIO(General Purpose Input Output) 사용에 적합한 아두이노를 사용한다. 센서 별로 구체적인 설계를 진행한다.

첫째, PIR 센서는 인체 감지 센서로, 적외선 움직임을 통한 인체 감지에 특화되어 있다. 또한, 넓은 영역을 감지하고 범위 내 사람의 움직임에 민감하게 반응한다. 이러한 특징들 때문에 PIR 센서를 이용해 출입구에 접근하는 승객을 감지한다 [8].

이때 PIR 센서의 감지 범위가 넓기 때문에 PIR 센서가 출입구를 바라보게 해야 하고, 통로를 등지게 설치해야 한다. 이러한 조건을 고려했을 때 센서 설치 위치는 계단 내벽이 적절하다. 제주 버스 계단 너비를 측정했을 때, 최대 너비가 125cm 정도이다. PIR 센서의 감지 범위를 고려하면 센서 1개로 출입구 한 곳을 확인할 수 있다. 대략적인 설치 위치는 그림 2와 같다.

그림 2 PIR 및 초음파 센서 설치 위치
Fig. 2. PIR and ultrasonic sensors installation location.

둘째, 초음파 센서의 설계 내용이다. 초음파 센서는 초음파를 통해 거리를 측정하는 센서이다. 본 시스템에서 하차 계수 시 승객의 하차 위치에 맞는 카메라를 작동시킬 필요가 있는데, 이때 승객의 위치를 파악하는 역할을 초음파 센서가 수행한다.

초음파 센서는 계단 챌판 상단 너비 15cm마다 설치하여 승객이 계단 어느 위치에 발을 딛든 감지할 수 있게 한다.

정리하면 출입구 접근 승객 파악은 승객의 출입문 접근을 감지하는 부분으로 PIR 센서와 초음파센서로 구성된다. PIR 센서는 출입구에 접근하는 승객을 파악하고, 초음파 센서는 승객의 세부 위치 파악을 하게 된다. 결과적으로 PIR 센서와 초음파 센서의 값들을 통해 문으로 접근하는 승객을 파악할 수 있다.

3.2.2 GPS

GPS에서는 GPS 센서를 활용해 GPS 값을 측정한다. GPS 센서는 아두이노를 사용하여 컨트롤한다. GPS를 측정하는 이유는 승·하차 정류소에 대한 정보를 얻기 위함이다. 공공데이터포털 제주특별자치도 버스 정류소 현황에서 정류소별 위도 경도가 제공되는데, 이 데이터와 GPS 센서값과의 데이터 비교를 통해 승·하차가 이루어지는 해당 정류소를 파악한다.

3.2.3 센서 데이터 통신

센서 데이터 통신은 아두이노에서 측정한 값들을 서버에 보내는 역할을 수행한다. 이 역할을 수행하기 위해 아두이노 통신 모듈을 사용한다.

아두이노 무선 통신에는 블루투스 통신과 WIFI 통신이 있는데, WIFI 통신이 블루투스 통신보다 통신 속도가 빠르고 다중 기기 통신이 가능하다. 또한 제주 버스에 갖춰진 WIFI 환경을 이용하기 적합하여 WIFI 통신을 사용한다. 아두이노 WIFI 통신은 WIFI 모듈을 통해 수행한다. WIFI 모듈을 이용해 센서 값을 데이터베이스에 전송하게 된다.

3.2.4 카메라

카메라는 승·하차 승객의 모습을 촬영하는 시스템 구성요소이다. 이 역할을 수행하기 위해 카메라 모듈과 라즈베리파이를 이용한다. 라즈베리파이는 카메라 모듈을 활용하는 데 적합하고 WIFI를 통해 다른 장치와 통신할 수 있다는 장점이 있다.

본 시스템에서 카메라는 승·하차 승객의 안면을 포함할 수 있도록 촬영해야 한다. 따라서 버스 환경에 맞게 카메라 모듈을 선정해야 한다. 버스 내부와 같이 좁은 실내 환경에서는 일반적인 카메라 렌즈를 통해 출입구를 촬영할 때, 카메라가 충분한 안면 움직임을 확보하기 어렵다. 따라서 광각 렌즈를 장착한 카메라 모듈을 사용하여 움직임을 포착하기 위한 적절한 시야를 확보한다.

카메라는 출입구 넓이에 따라서 앞문 계단은 1개 뒷문 계단은 2개를 설치한다. 하차 구간은 안정 봉에 의해서 두 부분으로 나누어져 하차가 이루어지기 때문에, 보다 정밀한 하차 판별을 위해서 개별적으로 카메라를 설치한다. 이때, 출입구 접근 승객 파악에서 승객의 하차 위치를 센서를 통해 확인하고 하차가 이루어지는 부분의 카메라만 작동시킨다.

다음은 카메라 설치 장소에 대한 설계이다. 카메라 설치 장소는 다음과 같은 조건들을 만족해야 한다.

첫째, 승·하차 판별을 원활히 할 수 있도록 출입구를 온전히 촬영할 수 있는 카메라 시야를 확보해야 한다. 시야 확보가 원활해야 보다 더 정확한 계수가 가능하다.

둘째, 하차의 경우 두 대의 카메라의 시야가 겹치지 않도록 서로 촬영구간을 절반씩 나눈다. 이 두 개의 카메라는 출입구 접근 승객 파악에서 전송한 센서 값에 따라서 사용되는 카메라가 결정된다.

셋째, 하차하는 승객이 계단을 내려갈 때 급격히 떨어지는 승객의 안면을 포착할 수 있는 설치 높이 및 카메라 각도를 설정한다.

이러한 조건들을 만족하는 본 시스템의 이상적인 카메라 설치 장소는 그림 3과 같다.

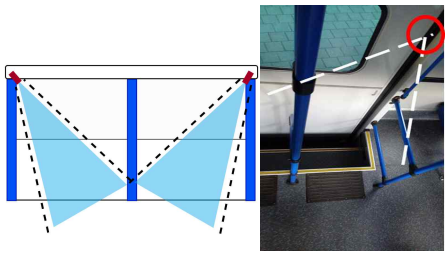

그림 3 카메라 설치 위치
Fig. 3. Camera installation location.

라즈베리파이는 IoT 센서의 데이터를 확인하고 카메라를 동작한다. 카메라 동작 후에도 지속적인 데이터 확인을 하는데, 센서 데이터를 통해서 승객이 없는 것으로 파악되었을 때 촬영을 종료한다. 그 후 촬영한 동영상과 IoT 시스템에서 수집된 데이터들을 클라우드 개발 환경에 전송한다.

3.3 승·하차 계수 프로그램

클라우드 환경의 승·하차 계수 프로그램은 IoT 모듈로부터 여러 데이터를 전송받아 처리하는 작업을 수행한다. IoT 모듈로부터 받은 GPS 데이터, 카메라 영상 데이터를 각기 목적에 맞게 처리한다. 또한 영상의 안면 움직임을 분석하여 승·하차 판단 계산을 수행하고, 그를 통해 버스 내부 인원에 대한 정보도 계산한다. 처리된 모든 데이터들은 클라우드 환경 내 결과 저장소에 저장한다. 해당 환경은 작업에 따라 2개의 구성요소로 이루어진다.

3.3.1 데이터 전처리

데이터 전처리에서는 IoT 모듈에서 측정한 GPS값, 카메라가 촬영한 영상 데이터를 가공하는 역할을 수행한다. 데이터 전처리를 통해 승객 안면 계수 이후 저장소의 저장되는 데이터가 정리되어 저장될 수 있도록 한다. GPS 데이터와 녹화된 영상 데이터는 승·하차 판별 프로그램에 입력한다.

GPS값은 승·하차 판별 프로그램에 넘겨서 후에 승·하차가 이루어진 것으로 확인되면 가까운 위치의 정류장에서 승·하차한 것으로 처리한다.

3.3.2 딥러닝 기반 안면 추적 프로그램

딥러닝 기반 안면 추적 프로그램에서는 승·하차하는 인원을 계수한다. 먼저 카메라를 통해 촬영한 승·하차 모습에서 사람의 안면을 인식하고, 인식한 안면의 이동을 추적하여 승·하차를 판별하게 된다.

라즈베리파이의 연산 능력을 고려하였을 때, 실시간 처리에는 적합하지 않았다. 따라서 클라우드 환경에서 승·하차 계수 프로그램을 작동할 수 있도록 한다. 프로그램 내용은 그림 4와 같다.

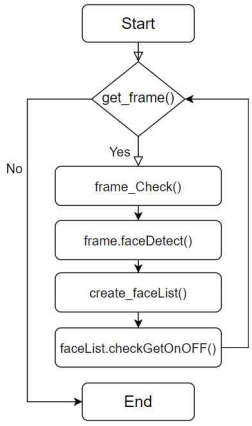

그림 4 프로그램 흐름도
Fig. 4. Program flowchart.

get_frame()을 통해 촬영된 동영상을 한 frame 단위로 읽어, frame_Check()와 fram.faceDetect()에서 frame의 포착된 안면의 좌표를 기록한다. 그 후 create_faceList()가 이전 프레임의 포착된 안면 좌표를 기준으로 가장 근접한 안면의 좌표를 동일 인물로 판단하여 같은 배열에 넣는다.

프레임 단위 연산이기 때문에 프레임 당 좌표 변화폭이 적어 이전 프레임과 유사한 인식된 얼굴 좌표일 경우 동일 얼굴로 판단한다.

한 배열의 처음 위치와 마지막 위치까지의 좌표 변화를 한 인물의 움직임이라 판단한다. faceList.checkGetOnOff()의 동작으로 안면의 위치가 승·하차 기준선을 통과하면 승·하차가 이루어진 것으로 판단한다.

승·하차 기준선은 카메라가 담기는 승·하차 승객의 모습에서 공통으로 이르게 되는 승·하차 위치를 기준으로 설정한다.

4. 구현

이 장에서는 설계의 내용을 바탕으로 제작한 시스템의 전체적인 형태, 모듈별 구현 내용에 대해 기술한다..

4.1 IoT 센서의 결과 값

IoT 모듈 테스트는 버스 뒷문 계단으로 그림 5에서 보는 것처럼 실험 환경을 설정하여 실험을 진행했다. 해당 실험은 초음파 센서와 PIR 센서가 설계의 내용처럼 계단에 발을 딛는 승객 감지가 가능한지 확인하는 데 목적을 둔다. 실험은 실제 버스 계단 폭과 비슷한 122cm 폭의 계단에서 진행하였다. 그림 5와 같이, 계단에 발이 닿기도 전에 바로 PIR 센서가 반응하여 회로에 불이 들어오는 것을 확인할 수 있었다.

그림 5 내딛는 도중 사진
Fig. 5. Snapshot of walking down.

출입구 승객을 계산하기 위해 승객이 어느 영역에 있는지 PIR 센서와 초음파 센서를 통해 파악해야 한다. 사람이 어느 곳에 얼마나 떨어진 곳에 있는지 확인하기 위한 실험을 진행했다. 그림 6은 PIR 센서와 초음파 센서가 WIFI 모듈을 통해 수집된 데이터를 전송하는 실험 구성을 보여준다. 그림에서 보면, 폭이 122cm인 계단에 승객 거리 파악을 위한 초음파 센서와 PIR 센서를 15cm 간격으로 설치하고 WiFi 모듈(ESP-8266)을 통해 센서 값을 전송한다.

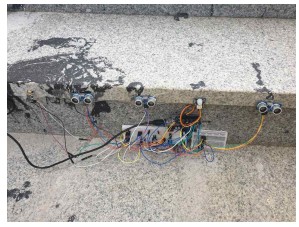

그림 6 계단에 설치된 시스템
Fig. 6. The system installed on stairs.

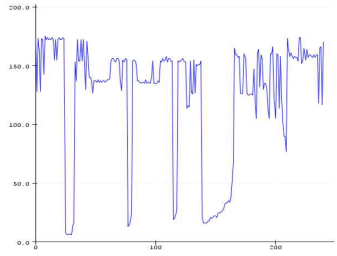

그림 7 초음파 센서 값 그래프
Fig. 7. Graph of ultrasonic sensor values.

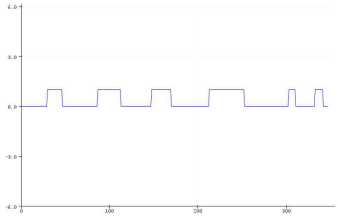

그림 8 PIR 센서 값 그래프
Fig. 8. Graph of PIR sensor values.

해당 센서들로 회로를 구성하여 실험을 진행하고, 그래프의 변화를 확인하였다. 해당 그래프들은 아두이노의 오실로스코프 기능을 통해 나타냈다. 그래프의 X축 값은 시간이며, Y축은 센서가 나타내는 값이다. 그림 7에서 보는 것처럼, 초음파 센서는 사람이 있을 때 급격하게 값이 감소하는 것을 확인할 수 있었다. 또한 그림 8에서 보는 것처럼, PIR 센서 값도 사람의 움직임이 감지될 경우 값이 1로 변하였고, 사람이 사라진 이후에도 일정시간 값이 지속되는 것을 발견했다. 이 센서 값들을 WIFI 모듈을 통하여 웹 데이터베이스로 전송한다.

표 1은 출입구 접근 승객 파악 센서 값이 저장되는 데이터베이스 테이블의 구성 컬럼 명과 데이터 형을 보여준다. 표에 나타난 것처럼, 테이블 안에 수집된 센서 값들이 컬럼 별로 저장된다. 표에서 보는 것처럼, 컬럼에는 3개의 초음파 센서와 센서의 값을 측정한 시간 값이 들어가게 된다.

표 1 출입구 접근 승객 파악 센서 테이블
Table 1. Entrance approach passenger identification sensor table.

Column Name	Data Type	Data Description
Data_PIR	INT	PIR Sensor Value
Data_DIS1	FLOAT	1st Ultrasonic Sensor Value
Data_DIS2	FLOAT	2nd Ultrasonic Sensor Value
Data_DIS3	FLOAT	3rd Ultrasonic Sensor Value
Data_Time	STR	Measured Time of Sensor Value

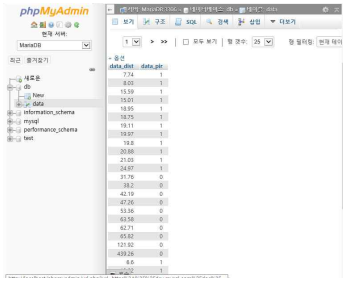

그림 9 MariaDB에 저장된 실험 데이터 일부
Fig. 9. Part of experimental data in MariaDB.

해당 웹 데이터베이스는 PHP 7.3 버전, MariaDB 10.4 버전, APACHE 2.4 버전으로 구성하였다, 그림 9에서 센서 값의 일부인 Data_dist와 Data_pir 값이 Data 테이블에 저장된 형태를 쿼리를 통해 확인했다. WIFI 모듈은 지정된 로컬호스트 내 PHP에 접속한다. 그 후 PHP는 전송된 값에서 센서 데이터를 추출하여 쿼리로 만들어 데이터베이스에 전송한다. 로컬호스트(localhost)의 MariaDB는 쿼리로 전송된 값을 실행하여 해당 데이터를 저장한다. 그 후 지속적인 데이터베이스 확인을 통해 센서 값이 하차인원이 있다고 감지될 경우 카메라를 통해 라즈베리파이에서 촬영을 실행하게 된다.

4.2 GPS 데이터 처리

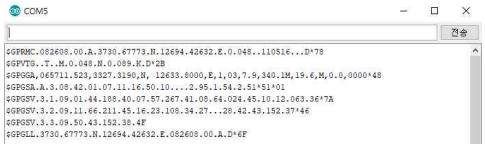

그림 10 GPS 모듈 데이터
Fig. 10. Data from GPS module.

그림 10에서 보는 것처럼, 아두이노 GPS 모듈을 동작시키면 NMEA 코드 형식의 많은 데이터를 얻게 된다. 이 데이터 중 GPGGA 형태의 코드에 현재 내 위치를 알 수 있는 경도 및 위도를 얻을 수 있다. 해당 로그를 TinyGPS 라이브러리를 활용하여 각도 형태로 변환한다. 변환 후에 공공데이터 포털에서 제공되는 제주 버스 정류장 위치의 노선 내 정류장 위치와 비교하여 현재 위치와 가장 가까운 정류장 위치를 알아낼 수 있게 된다.

GPS 센서로 얻은 값을 파싱하여 GPGGA 값에서 현재 위도 및 경도의 도분초 값을 얻어내었다. 해당 값의 오차가 실제 위치와 1~2미터 차이로 크지 않았다. 얻어낸 위도와 경도로 CSV 내의 노선별로 가까운 정류장 위치의 비교가 가능했다. 따라서, 승·하차 위치를 GPS로 파악가능하다.

4.3 클라우드 환경

동영상은 YR-022 모듈을 이용해 촬영되어 클라우드 환경으로 전송된다. 영상은 화각 160도, 해상도 1920 x 1080, 초당 30프레임으로 촬영했다. 승·하차 장면을 녹화한 영상을 Get on/off check algorithm을 통해 처리하는 작업은 많은 컴퓨팅 파워가 필요하다. Get on/off check algorithm 처리는 라즈베리파이의 컴퓨팅 파워로는 부족하다. 승·하차 정보의 실시간 처리를 위해 영상 파일을 다른 컴퓨팅 환경으로 전송한다.

영상과 같이 용량이 큰 파일을 전송할 수 있고 저장할 수 있고, 저장된 영상 파일을 바로 런타임 환경에서 입력으로 받아들일 수 있는 환경인 Google Colab을 선택했다.

카메라가 촬영을 마치면 녹화된 영상 파일을 클라우드 저장소인 구글 드라이브에 전송한다. 전송이 완료되면 웹에서 클라우드 기반 런타임 환경인 Google Colab에 미리 작성된 Get on/off check algorithm이 담긴 프로그램을 동작시킨다.

4.4 안면 인식

Get on/off check algorithm에서 가장 먼저 필요한 것

은 얼굴 인식을 할 수 있는 딥러닝 모델이다. 버스 승·하차하는 사람의 얼굴을 학습시키기 어려워 일반적인 사람의 안면을 인식하는 모델을 Github에서 찾았다. 마스크나 선글라스를 착용한 사람의 얼굴도 인식하며, 사람의 얼굴을 높은 정확도로 인식하였다.

해당 모델의 경우 YOLO 기반 모델에 WIDER FACE Dataset을 학습시킨 모델이다[9][10]. 실제 사람의 사진을 주었을 때 사람이라고 판단하는 정확도는 92%를 넘는다. 선글라스나 마스크를 착용한 사람도 사람이라고 판단하는지 실험을 해보았다. 이 경우, 그림 11과 같이 선글라스, 마스크 착용한 안면을 인식하여 사각형으로 표시하였다.

그림 11 YOLO-face 모델 실험
Fig. 11. YOLO-face model test.

촬영한 영상을 딥러닝 신경망에 입력하여 영상 매 프레임마다 포착한 얼굴의 좌표를 기록한다. 이전 프레임의 인식된 얼굴 좌표를 기준으로 가장 가까운 얼굴 좌표를 동일한 인물의 안면으로 판단한다. 프레임 단위 연산이기 때문에 좌표 변화폭이 적어서 이러한 방식으로 한 인물의 움직임을 추적할 수 있다.

이때 가까운 좌표의 기준은 유사 얼굴 좌표 기준을 x, y 픽셀 좌표 값의 차(차는 절댓값 연산으로 구한다.)의 합이 400보다 작아야 하는 것으로 설정했다. 하지만 이 기준은 카메라 각도, 위치, 촬영 반경에 따라서 달라질 수 있다.

한 인물의 안면 움직임을 기록하게 되면 승·하차 판별을 할 수 있다. 마지막으로 포착된 안면의 좌표가 기준선을 넘으면 승·하차로 판단하는 것이다. 기준선은 승·하차 행위 후의 사람 안면의 위치를 기준으로, 충분히 승·하차 행위로 간주할 수 있는 안면 위치를 고려하여 설정한다.

그림 12에서는 실제 버스에서 하차 승객을 가정하고 촬영한 영상을 승·하차 프로그램에 입력했다. 안면이라고 판단되는 부분을 사각형으로 표시하였고, 마지막 계단에 설정한 기준선을 사각형이 통과하는 순간 카운트가 되는 것을 확인했다.

그림 12 버스 하차 계수 테스트
Fig. 12. bus de-boarding count test.

4.5 승·하차 판별 후 데이터 저장

승·하차 판별이 끝나면 현재 버스 인원의 변화 값을 서버로 전송하고, 관련 데이터들을 JSON 형식으로 클라우드 내에 저장한다. 관련 데이터로는 시간, IoT 센서값들, 승·하차 승객 수가 있다.

5. 실험

개발 시스템의 검증을 위해 실험 환경을 설정하고 실험을 실시하였다. 실시한 실험은 복잡한 상황에서의 안면 움직임 추적에 대한 실험(실험 1)과 버스 유사 환경에서의 승·하차 계수 실험(실험 2)이다.

5.1 실험 1 환경 설정

제안 시스템이 승하차 계수를 제대로 확인하기 위해서는 우선 버스 내 상황이 복잡한 경우에 프로그램이 정상적으로 작동하는지 검증이 필요하다. 이를 위해 실험 환경을 다음 두가지로 설정하였다.

첫 번째는 하차 승객이 너무 많아서 몰리는 경우이다. 특히, 계단 바로 위에 사람들이 몰려 있는 상황가 발생할 가능성이 높기 때문에 이 경웨 제대로 계수할 수 있는 지를 알아보았다. 두 번째는 카메라 관점에서 승객과 다른 승객의 얼굴 위치가 일시적으로 겹치는 경우이다. 즉, 카메라에 한 승객이 포착 된 이후 다른 승객이 지나가며 안면을 가리는 상황이다.

5.2 실험1 결과

첫 번째로 승객이 밀집해 있는 상황이 그림 13 (a)에 나타나 있다. 프로그램의 안면 포착에 장애가 없는지 확인하기 위해서 성인 남성 3명을 줄 세우고 앞 사람 어깨에 손을 올리게 해서 최대한 밀집해 있는 환경에서 얼굴형이 가려져도 문제없이 얼굴을 포착함을 알 수 있었다.

두 번째로 카메라 관점에서 승객과 다른 승객이 일시적으로 겹친 경우가 그림 13(b)에 나타나 있다. 얼굴 겹침으로 인해 안면 움직임 추적이 방해를 받는 경우도 있었다. 그렇지만 그러한 경우에도 화면에서 포착할 수 있는 최대 안면 개수만큼 움직임을 추적 및 갱신하기 때문에 계수가 누락되지는 않았다. 그림에서 보는 것처럼 특정 승객 A의 움직임을 추적한 후 다른 승객 B와 얼굴이 겹친다. 하지만 B의 얼굴은 결국 지나가게 되고, A의 얼굴이 다시 드러난다. 다수의 사람이 등장하고 여러 번의 안면 겹침이 일어났지만 결국 성공적으로 움직임 추적을 완료하였다..

(a) 밀집 (b) 겹침

그림 13 밀집 및 겹침 안면 인식 테스트

Fig. 13. Dense and overlapped face recognition test.

5.3 실험2 환경 설정

구현된 시스템 작동 여부를 확인하고, 계수 정확도 측정을 위한 실험을 실시하였다. 실험 환경은 버스 승·하차 환경과 유사한 건물 내부 계단이다. 실험은 시스템을 계단에 설치하고, 실험 인원이 승·하차 행동을 수행한다. 승·하차 행동은 계단을 오르고 내리는 것이고, 최대 두 명이 승·하차 행동을 진행한다. 실제 지나간 인원과 측정된 인원수를 비교하여 시스템의 정확도를 측정한다. 정확도는 측정 인원 / 실제 인원으로 계산한다. 각 테스트 샘플은 승차와 하차 동영상으로 나누어 각각 5회분으로 구성한다. 동영상마다 승·하차 인원을 최소 1명 최대 5명으로 설정하였다.

또한, 최근 코로나 바이러스로 인해 버스 이용 시 마스크를 필수로 착용해야 하므로 마스크 착용 승객에 대한 실험의 필요성이 대두되었다. 따라서 승객 마스크 착용 여부를 구분하여 실험을 진행하였다..

5.4 실험2 결과

승객이 마스크를 착용하지 않은 상황에서의 실험 결과는 표 2와 같다. 테스트 샘플(Test Sample)은 승차(In) 동영상, 하차(Out) 동영상 각각 5회씩 구성된다. 승차 인원은 버스 재차 인원(Bus Passenger Number)에 더해주고, 하차 인원은 버스 재차 인원에서 감소시켜 계산한다.

표 2 마스크 미착용 실험 결과

Table 2. Test result without mask.

Test Sample	Real Passenger Number	Counted Passenger Number	Bus Passenger Number
In_1	5	5	5
In_2	3	3	8
In_3	3	3	11
In_4	4	4	15
In_5	1	1	16
Out_1	2	2	14
Out_2	3	3	11
Out_3	4	4	7
Out_4	3	3	4
Out_5	4	4	0

표 2를 분석하면, 마스크를 착용하지 않은 승차 하차 인원을 모두 정확히 계수했다. 승차의 경우 1명씩 승차하여 전체적으로 안면 정확도가 높았고, 계수 또한 정확했다. 하차의 경우 최대 2명씩 하차했다. 그리고 하차 시 측면 얼굴이 정면 얼굴에 비해 안면 정확도가 낮았으나, 계수에 영향을 주는 정도는 아니었다.

표 3 마스크 착용 실험 결과

Table 3. Test result with mask.

Test Sample	Real Passenger Number	Counted Passenger Number	Bus Passenger Number
In_1	5	5	5
In_2	3	3	8
In_3	3	3	11
In_4	4	4	15
In_5	1	1	16
Out_1	2	2	14
Out_2	3	3	11
Out_3	4	3	8
Out_4	3	3	5
Out_5	4	4	1

마스크를 착용한 실험 결과는 표 3과 같다. 실험 결과 표 구성은 표 2와 동일하다. 표 3에서 승차 인원은 모두 정확히 계수 했지만, 하차_3 샘플에서 1명의 계수를 실패하였다. 계수 실패 원인은 지속적인 안면 감지가 이루어지지 못했기 때문이다. 즉, 일부 구간에서 안면 감지를 실패했다. 감지 실패 원인은 실험 인원끼리의 안면이 순간적으로 겹쳤기 때문이다. 그리고 마스크로 인한 얼굴 면적 감소 또한 안면 감지 실패 원인이다. 안면 움직임의 대부분을 포착하였으나, 기준선 근처 혹은 이후의 움직임을 파악하지 못해 계수에 실패했다. 이에 대한 상황은 그림 14를 참고할 수 있다.

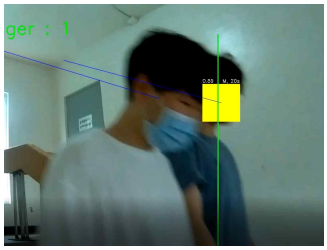

그림 14 계수 실패 예시
Fig. 14. A fail case of counting.

표 4 정확도 결과
Table 4. Accuracy result.

Real Passenger Number	Counted Passenger Number	Accuracy
60	59	0.98

정확도는 실제 승·하차 인원(Real Passenger Number)과 측정된 인원(Counted Passenger Number)을 통해 측정한다. 표 4는 총 승·하차 인원과 총 측정 인원을 통해 표 2와 표 3의 실험 결과를 바탕으로 시스템의 계수 정확도를 계산한 것이다.

표 4의 내용을 살펴보면, 표 2와 표 3에서의 총 승·하차 인원인 60명, 총 측정 인원인 59명을 기준으로 정확도 0.98을 도출했다.

실험 결과를 정리하면, 총 60명의 승·하차 인원 중 59명의 계수를 성공하고, 1명의 계수에 실패했다. 계수 실패는 마스크 착용과 얼굴 겹침으로 인한 안면 감지 정확도 감소 때문이다. 하지만, 전체적으로 0.98의 높은 승·하차 계수 정확도를 가졌다.

6. 결론

본 연구는 IoT 및 딥러닝 모델을 활용하여 실시간 버스 승·하차 자동 계수 시스템을 제안 및 구현하였지만, 실제 운행하고 있는 버스에서의 실험 데이터는 얻지 못했다. 실제 버스 운행에 적용했을 때 실험과 버스의 진동, 통신 지연 등에서 오차가 발생할 수 있다고 예상한다. 버스 운행 시의 데이터 확보를 통해 카메라 위치 및 딥러닝 모델 개선 등 추가 연구가 필요하다.

또한 이 시스템을 활용하면, 교통 카드 태그와 상관없이 모든 버스 승객의 승·하차를 계수하고 이를 정보화 할 수 있다. 추후, 마스크 착용과 같은 안면이 일부 가려진 승·하차 승객의 인식 정확도를 높이기 위해 지속적인 딥러닝 모델의 개선 연구를 진행할 것이다.

References

[1] Y. Kim, C. Kim, and S. Kim, “An Algorithm for Extracting Tourists’ O-D Patterns Using Encrypted Smart Card Data of Public Transportation: Application to Tourist City, Jeju,” *KIISE Transactions on Computing Practices*, Vol. 26, No. 8, pp. 349-361, Aug. 2020.(in Korean)

[2] Transportation Tourism Planning Team of Jeju Special Self-Governing Province, Final Report on the 3rd Public Transportation Plan of Jeju Special Self-Governing Province, Jun. 2017, Jun 1).(in Korean)

[3] H. Hwang, and J. Lee, “Bus Information System Providing Passenger Number Information,” *The Journal of the Korea Contents Association*, Vo1. 9, No. 12, p. 32, Dec. 2009. (in Korean)

[4] J. Perng, T. Wang, Y. Hsu, and B. Wu, "The Design and Implementation of a Vision-based People Counting System in Buses," *Proc. of 2016 International Conference on System Science and Engineering (ICSSE)*, pp. 1-3, 2016.

[5] B. Dan, Y. Kim, Suryanto, J. Jung, and S. Ko, "Robust People Counting System based on Sensor Fusion," *IEEE Transactions on Consumer Electronics*, Vol. 58, No. 3, pp. 1013-1021, Aug. 2012.

[6] C. Tsai, Y. Bai, C. Chu, C. Chung, and M. Lin, "PIR-sensor-based Lighting Device With Ultra-low Standby Power consumption," *IEEE Transactions on Consumer Electronics*, Vol. 57, No. 3, pp. 1157-1164, Aug. 2011.

[7] S. Prasad, P. Mahalakshmi, A. John C. Sunder and R. Swathi "Smart Surveillance Monitoring System Using Raspberry PI And PIR Sensor," *(IJCSIT) International Journal of Computer Science and Information Technologies*, Vol. 5 (6), pp. 7107-7109, 2014.

[8] H. Woo, “Distance Classification Algorithm by Analyzing PIR,” *Kyungpook National University Graduate school of Industry : Industrial engineering department Instrumentation and control engineering academic thesis*, p. 5, Jun. 2016. (in Korean)

[9] J. Redmon, and A. Farhadi. YOLOv3: An Incremental Improvement. Apr. 2018. Available: https://arxiv.org/abs/1804.02767

[10] S. Yang, P. Luo, C. Loy, and X. Tang, “Wider Face: A face Detection Benchmark,” *Proc. of the IEEE conference on computer vision and pattern recognition*, p. 5531, Jun. 2016.

이 호 성 (Ho-Sung Lee)

- 제주대학교 사범대학 컴퓨터교육과 졸업예정 (이학사)
- 관심분야: Big Data Analysis, AI, Vision, IoT

이 진 우 (Jin-Woo Lee)

- 제주대학교 사범대학 컴퓨터교육과·블록체인 보안 연계 전공 졸업예정 (이학사)
- 2020년 ㈜Kakao Track 수료
- 관심분야: AI, Blockchain, Web.

김 성 백 (Seong-Baeg Kim)

- 1996년 ~ 현재: 제주대학교 사범대학 컴퓨터교육과 교수
- 서울대학교 대학원 컴퓨터공학과 졸업 (공학석사, 공학박사)
- 관심분야: Computer Systems, SW Convergence Education