2023 년 전기 졸업과제 착수보고서

컴퓨터 비전을 이용한 버스 혼잡도 파악 시스템

팀명 : HKH

201845806 김민욱

201924663 황정호

201904165 황현정

지도교수 : 김종덕

목차

1. 과제 배경 및 목표

1) 과제 배경

서울 교통공사는 2017년부터 시내버스 혼잡도 데이터를 제공하고 있다. 서울특별시의 경우, 98.9%의 승객들이 교통카드로 버스를 타고 하차할 때는 하차태그를 하기 때문에, 교통공사는 이를 기반으로 버스 혼잡도 데이터를 수집하고 있다. 하지만, 일부 승객들은 현금으로 승차하거나 교통카드로 하차를 미태그하는 등의 이유로 데이터의 정확도는 낮아질 수 있다.

해당 문제를 극복하기 위해, 센서 및 비전 기술을 통해 버스 내부 인원의 혼잡도를 측정할 수 있다. 센서로는 레이저 센서 또는 압전 센서 등이 사용될 수 있으며, 비전 기술로는 OpenCV(Open-source Computer Vison)와 YOLO(You-Look Only Once) 등이 사용될 수 있다. 본 프로젝트에서는 센서와 비전 기술을 활용하여 버스 내부 혼잡도 측정 시스템을 제안한다. 이를 통해, 버스 내부에 탑승한 승객들의 승하차 인원수를 측정하고 혼잡도를 모니터링하는 시스템을 구현할 수 있다. 해당 시스템의 도입으로 기존의 버스 인원수 측정 시스템의 한계를 극복하고 승객들의 안전과 편의성을 높이기 위한 추가적인 서비스 제공도 가능할 것으로 기대된다.

2) 과제 세부 목표

비전 기술을 이용하여 버스 내부 영상을 전처리하고 딥러닝 기반 모델을 이용하여 혼잡도를 측정한다. 측정한 혼잡도를 이용하여 유용한 정보를 제공하는 시스템을 개발한다.

① 컴퓨터 비전 라이브러리인 OpenCV를 이용하여 버스 내부 영상을 전처리한다.

② 딥러닝 기반 모델 중 YOLOv5를 사용하여 객체를 탐지하고 혼잡도를 측정한다.

③ 측정한 혼잡도를 이용하여 유용한 정보를 이용자에게 제공한다.

2. 진행 방안

1) 모델 구성

ㄱ) 인원 측정 모델

OpenCV와 YOLOv5를 이용하여 객체 검출을 한다. 각 프레임마다 '사람'으로 인식되는 객체들을 검출 후 인원수를 셀 것이며 트래킹 기술과 접목하여 승객의 위치, 이동 방향 등을 파악한다. 또한 혼잡도와 시간대 데이터를 mariaDB에 저장한 후, 전날 데이터를 토대로 시간대별 인원수 예상 그래프를 제공할 예정이다.

ㄴ) 혼잡도 파악 모델

현재 서울 시내 버스 혼잡도 계산시에 다음과 같은 수식을 사용한다.

$$(\textit{버스 혼잡도}) = \frac{(\textit{인원 수})}{(\textit{좌석 수} + \textit{손잡이 수})} * 100$$

버스 차내 혼잡도는 서울시내 일반버스(간선·지선·순환)의 재차인원을 '여유', '보통', '혼잡' 3단계 수준으로 구분하여 안내된다. '여유'는 좌석에 앉을 수 있는 정도, '보통'은 입석 승객이 손잡이를

하나씩 잡고 서 있을 수 있는 정도, '혼잡'은 입석 승객들 사이 통로에 까지 승객이 서 있고 입석 승객의 몸이 맞닿는 정도 수준이다.[1]

이 연구에서는 이에 추가적으로 온도와 습도를 이용하여 혼잡도를 제공할 계획이다. 혼잡도는 개인마다 체감하는 정도가 달라 혼잡을 느끼는 수준도 다소 차이가 있을 수 있다. 이에 영향을 미치는 변수 중 하나가 온도와 습도라 판단하였고, 기상청에서 제공하는 온도와 습도를 토대로, 불쾌지수를 계산하여 혼잡도 제공시에 반영할 계획이다. 예를 들어, 불쾌지수가 낮을 수록 '여유'에 해당하는 비율을 증가시키고, 불쾌지수가 높을수록 비율을 낮출 것이다. 정확한 수치는 추후 연구를 진행하며 구체화 할 예정이다.

기상청에서 제공하는 온도와 습도의 경우 정각 시간대 별로 API를 호출할 수 있다. 7시, 14시, 19시를 기준으로 온도와 습도를 받아와 불쾌지수를 계산할 것이다. 시간 기준점은 한국인 평균 출퇴근 시간을 기준으로 삼고, 가장 더운 시간대를 기준으로 삼았다.

표 1– 기상청 API

초단기실황	T1H	기온	℃	10
	RN1	1시간 강수량	Mm	8
	UUU	동서바람성분	m/s	12
	VVV	남북바람성분	m/s	12
	REH	습도	%	8
	PTY	강수형태	코드값	4
	VEC	풍향	deg	10
	WSD	풍속	m/s	10

ㄷ) 객체 검출 모델

CrowdHuman[2] 데이터셋에 YOLOv5m을 이용하여 학습한 weight인 crowdhuman_yolov5m를 사용할 예정이다. CrowdHuman 데이터셋은 470,000 사람의 다양한 혼잡한 상황에서 촬영된 이미지와 비디오로 구성되어 있고 각 이미지에 약 23명의 사람이 존재한다. train, validation과 test 이미지 수는 표 2와 같다.

표 1 - train, validation, test 이미지 수

	train	val	test
# of images	15000	4370	5000

기존 yolov5m 모델에서 사람 머리에 대한 weight가 없는데 비해 crowdhuman_yolov5m은 사람 머리에 대한 검출을 제공한다. 혼잡한 상황의 버스는 사람들이 겹쳐 있으면 사람 전체를 탐지하는 것이 어렵기 때문에 사람 머리에 대한 검출을 지원하는 모델을 사용한다.

[1] https://news.seoul.go.kr/traffic/archives/33643, 서울시 국내최초 버스혼잡도 안내 서비스 개시, 2017.06.02

[2] benchmark dataset to better evaluate detectors in crowd scenarios.

사람 머리에 대한 객체 검출 모델의 성능은 그림 1과 같다.

Table 8. Evaluation of Head detection on CrowdHuman benchmark.

	Recall	AP	mMR
FPN [15]	81.10	77.95	**52.06**
RetinaNet [16]	78.43	71.36	60.64

그림 1 - CrowdHuman 모델 사람 머리 객체 검출 벤치마크[3]

FPN이 RetinaNet보다 우수한 성능을 보여준다. FPN의 경우 Recall은 81.10으로 검출해야 하는 객체를 81.10%로 검출한다. 그림 2의 R-CNN 계열과 그림 1 모델의 AP를 비교하면 후자의 정확도가 높음을 알 수 있다.

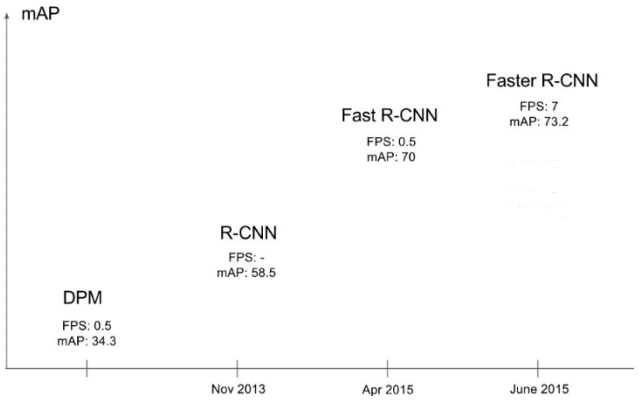

그림 2 - R-CNN계열 모델 성능

2) 전체 시스템 구성도 및 흐름도

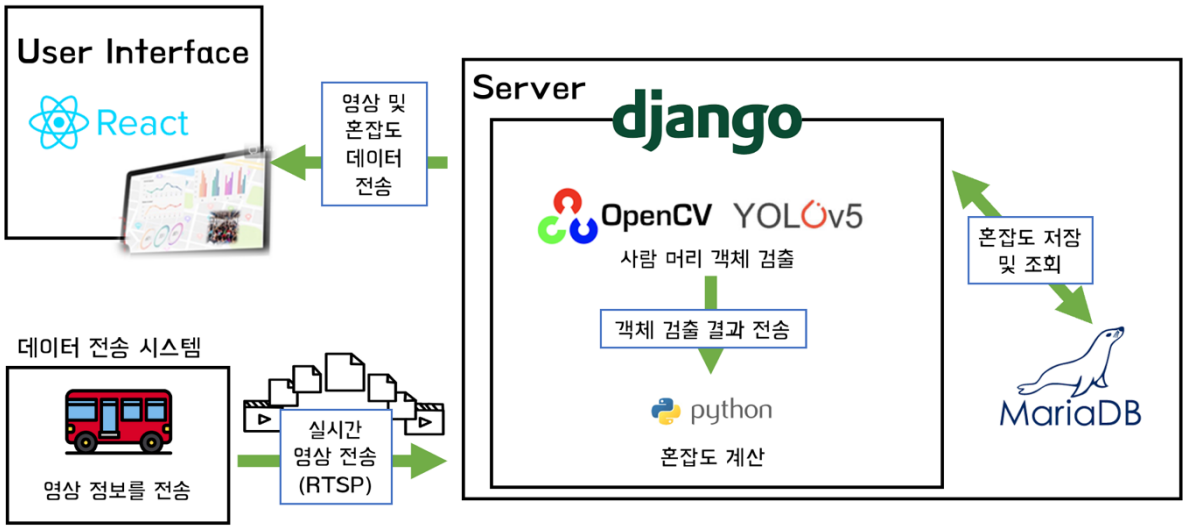

그림 3 - 시스템 구성도

[3] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang & Jian Sun, CrowdHuman: A Benchmark for Detecting Human in a Crowd, 2018, p.7

① 서버에서 RTSP프로토콜을 이용해 CCTV IP에 접속해 실시간으로 영상을 얻는다.
② 서버에서는 openCV와 YOLOYv5를 이용해 사람 머리 객체를 검출한다.
③ 검출 결과를 기반으로 혼잡도를 파악한다.
④ DB에 혼잡도, 시간대를 저장하여 추후 예상 그래프를 그릴 때 사용할 수 있도록 한다.
⑤ 서버에서 영상 및 혼잡도 데이터를 유저 인터페이스에 전송하여 사용자가 편리하게 확인할 수 있도록 한다.

웹을 구현하는 방법으로 Django과 React를 선택하였다. Django의 경우, 파이썬 언어를 기반으로 동작하므로 기존의 OpenCV와 YOLOv5를 웹 기반으로 발전시키기에 용이하다 판단하여 사용한다. React의 경우, 실시간 그래프 및 차트 구현의 편리함에 중점을 두어 혼잡도 예상 그래프를 나타내기 위해 선택하였다. 혼잡도 예상 그래프를 제공하기 위해 현재 날짜, 시각, 혼잡도를 저장해야 한다. 테이블의 변동성이 적어 관계형 데이터베이스 중 하나를 선택하려 하였고, 그 중 가볍고 빠른 mariaDB를 선택하였다.

3. 문제 해결

1) 문제 정의

객체를 검출의 정확도를 높이는 것이 주요한 문제이다. 먼저 컴퓨터 비전 라이브러리인 OpenCV를 이용하여 객체 검출 및 혼잡도를 측정하여 성능을 평가한다. 이 때 기대하는 객체 검출 성능이 나오지 않을 것으로 예상된다.

객체를 검출하는 문제와 한 객체의 움직임을 감지하여 트래킹 하는 것은 별개의 문제이다. 따라서 검출한 객체를 트래킹 해야 하는 문제가 있다.

마지막으로 CCTV 영상을 서버로 전송하고 이것을 이용자에게 전송해주기 위해 실시간 전송 프로토콜이 필요하다.

2) 객체 검출

객체 검출은 영상 또는 비디오에서 객체 인스턴스를 찾기 위한 컴퓨터 비전 기법으로 한 물체가 아닌 여러 물체에 대해 사물이 어떤 것인지 분류하는 Classification과 사물이 어디 있는지 위치 정보를 나타내는 Localization문제로 나눌 수 있다.

객체 검출은 크게 1-stage Detector와 2-stage Detector로 구분할 수 있다. 1-stage detector는 Classification과 Localization을 동시에 수행하는 방법이고, 2-stage detector는 이 두 문제를 순차적으로 행하는 방법이다. 따라서 1-stage detector는 비교적으로 빠르지만 정확도가 낮고, 2-stage detector는 비교적 느리지만 정확도가 높다. 1-stage detector에는 YOLO, SSD계열이 있고, 2-stage detector: R-CNN계열이 있다.

버스 혼잡도 측정은 real-time 객체 검출이 필요하다고 생각하여 1-stage detector중 사용하기 편리하고 빠른 속도 대비 높은 정확도를 보여주는 YOLO를 선택하였다.

ㄱ) YOLOv5

CNN 기반 객체 인식 알고리즘으로 Faster R-CNN 보다 빠른 학습 속도를 가지고 있어 실시간 객체 탐지에 주로 사용된다. 전체적인 구조는 그림 4와 같다.

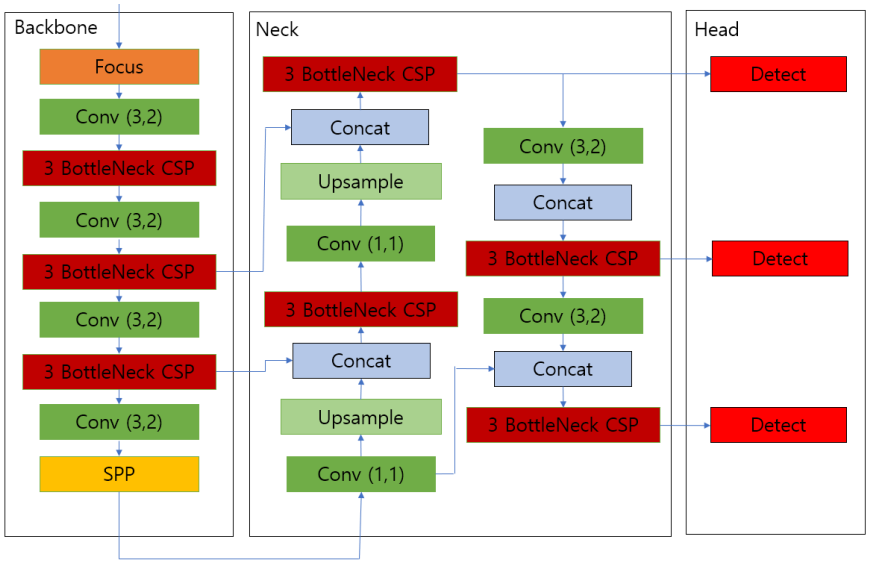

그림 4 - YOLOv5 구조

이미지로부터 feature map을 추출하는 Backbone, 앞에서 추출한 feature를 집계하는 Neck과 추출된 피쳐맵을 바탕으로 bbox를 찾고 classification을 하는 Head 부분으로 나뉜다. YOLOv4의 구조와 유사하나 YOLOv5는 CSPNet backbone 사용하여 더 적은 매개변수를 사용하여 더 빠르고 정확한 객체 검출 가능하다. YOLOv5의 각 모델 성능은 그림 5와 같다.

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

그림 5 - YOLOv5 성능

YOLOv5는 네트워크의 크기에 따라 n(nano), s(small), m(medium), l(large), x(xlarge)로 여러 버전이 제공된다. n모델은 빠르지만 정확도가 가장 낮으며, x모델은 느리지만 정확도가 가장 높다. 그림 3의 mAP_0.5, mAP_0.5:0.95 (여기 캡션)와 params 지표를 비교해보면 확인할 수 있다. mAP가 높을수록 정확도가 높고, params가 많을수록 느리다. 아래 5개의 모델은 위 5개 모델보다 많은 계층과 파라미터를 가지고 있어 복잡하지만 더 정확한 결과를 얻을 수 있다.

3) 객체추적

ㄱ) DeepSORT

SORT(Simple Online and Realtime Tracking)는 간단한 알고리즘이면서 효과적이고 실용적인 다중 대상 추적 알고리즘이다. Occlusion[4]과 Id Switching[5]이라는 문제점을 가지고 있지만 Deep SORT가 등장해 기존 SORT의 단점을 어느정도 보완했다.

여기서 적용된 Kalman filter와 Hungarian algorithm이 존재하는데 Kalman filter는 예상한 결과(Motion)와 실제 측정된 결과물(Measurement)을 기반으로 실제 물체의 위치를 최적으로 판단할 수 있는 알고리즘이고 Hungarian algorithm은 할당 문제의 해결 방법 중 하나로 이분 그래프에서 최적의 매칭을 찾는 알고리즘이다. 즉 이전 프레임에서 발견된 개체와 다음 프레임에서 발견된 개체가 동일하다는 것을 판별할 수 있다.

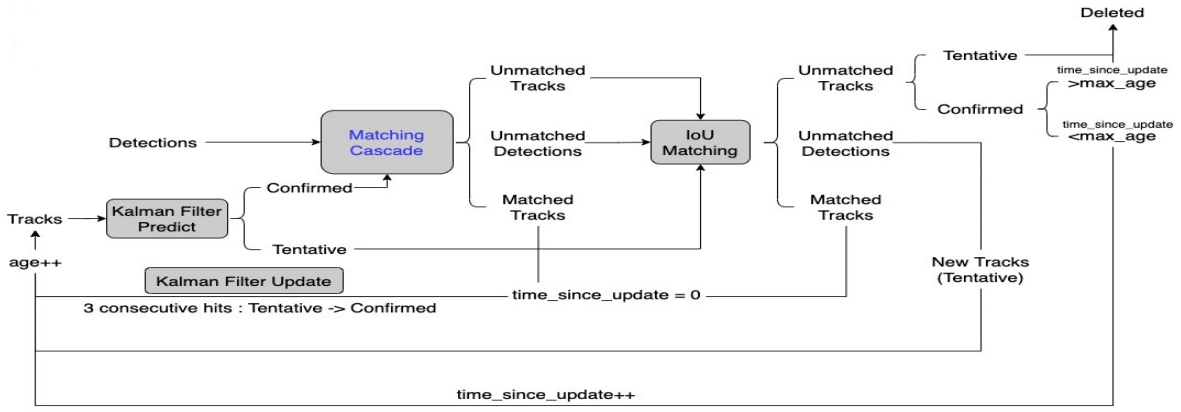

그림 6 – DeepSORT 동작 구조

그림6에서 Tracks는 현재 추적되고 있는 객체의 정보 관리, Detections는 현재 이미지에서 탐지한 물체의 경계박스(bbox)와 물체의 클래스, IOU는 겹치는 영역에 대한 수치 값을 나타낸다. 트래킹은 Motion estimation(Kalman Filter), Data association(IOU, Hungarian algorithm)과 Update과정으로 이루어져 있는데 Motion estimation은 이전 프레임에서 발견된 경계들이 현재 프레임에서 어디에 위치할 지 예측하는 단계이며, Data Association은 연속된 프레임에서 경계가 서로 같은 객체의 경계인지 최적 값을 찾는 과정이다. 최적 값을 찾으면 Motion estimation의 결과를 현재 프레임에서 찾은 경계정보를 이용해 Update한다.

DeepSORT는 SORT의 단점을 보강하기 위해 Data association에 사용하는 특징을 보강하고 추적한 객체와 현재 프레임에서 탐지한 객체를 매칭 하는 방법을 강화한 버전을 사용한다. 트랙과 탐

[4] 물체가 가려질 때 탐지하기 힘든 경우.

[5] 물체가 빠르게 이동할 때 id추적이 어려워 대상을 놓치는 경우.

지된 물체 사이의 매칭관계를 구하는 Matching cascade단계가 추가된 것 이외에는 기존의 SORT의 동작과 비슷하다.

4) 스트리밍 프로토콜

ㄱ) RSTP(Real Time Streaming Protocol)

Real Time Streaming Protocol은 실시간 스트리밍 프로토콜로 기존에 영상을 다운받아서 재생해야 하는 불편한 방식에서 벗어나 실시간으로 음성이나 동영상을 송수신할 수 있는 통신규약이다. 클라이언트에서 다양한 명령어를 요청해 스트리밍 서버 영상을 조작 및 사용할 수 있다. HTTP규약과 비슷한 서버/클라이언트 구조를 가지고 있으며 포트번호는 554를 사용하며 외부 IP카메라 같은 장치에 적용해 촬영된 영상을 TV나 모바일로 볼 수 있다.

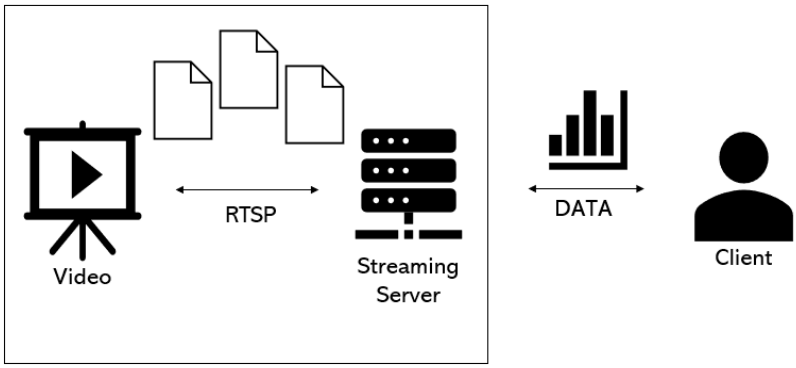

그림 7 – RTSP 동작 구조

4. 개발 일정 및 역할 분담

1) 개발 일정

5월			6월				7월				8월					9월			
3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4
지식 공부																			
							서버환경 구축												
							RTSP 시스템 구축												
							OpenCV 객체 탐지 기술 개발												
							YOLOv5 객체 탐지 기술 개발												
							버스 내부 환경을 고려한 객체 추적 기술 개발												
									중간 보고서										
											객체 탐지 및 추적 성능비교 및 평가									
											혼잡도 계산 모델								
											테스트 및 디버깅								
																UI 디자인 및 실시간 그래프			
																		최종 보고서	

2) 역할 분담

이름	역할
김민욱	- openCV를 이용하여 영상 처리 모듈 개발(객체 탐지) - DeepSort를 이용한 트래킹 모듈 개발(객체 트래킹) - Docker 환경 설정
황정호	- RTSP 프로토콜을 이용한 영상데이터 전송 구현 (실시간 영상 전송) - DeepSORT를 이용한 트래킹 처리(객체 트래킹) - 실시간 영상 처리 서버 구축
황현정	- 객체 탐지 및 트래킹을 통해 생성된 결과값을 사용해 혼잡도 계산 - YOLOv5를 이용하여 객체 검출 모듈 개발 (객체 탐지) - Docker 환경 설정