
 한국전자기술연구원 <small>Korea Electronics Technology Institute</small>	VDL활용 속도 추정 알고리즘 매뉴얼				
	문서분류	문서관리자	버전	최초작성일	최종수정일
	보고서	장수현	1.1	2021년 2월 15일	2021년 3월 8일

VDL활용 속도추정 알고리즘 매뉴얼

작성자	한국전자기술연구원	한서우
	한국전자기술연구원	장수현

 KETI 한국전자기술연구원 Korea Electronics Technology Institute	VDL활용 속도 추정 알고리즘 매뉴얼				
	문서분류	문서관리자	버전	최초작성일	최종수정일
	보고서	장수현	1.1	2021년 2월 15일	2021년 3월 8일

차 례

1. 속도 추정 알고리즘	3
1.1 전체 모델 설명	3
1.2 속도 추정 알고리즘 설명	3
2. 속도 추정 알고리즘 활용 방법	4
2.1 Code Tree	4
2.2 속도 추정 함수 사용법	5
3. 속도 추정 알고리즘 결과	6

KE-TI 한국전자기술연구원 Korea Electronics Technology Institute	VDL 활용 속도 추정 알고리즘 매뉴얼				
	문서분류	문서관리자	버전	최초작성일	최종수정일
	보고서	장수현	1.1	2021년 2월 15일	2021년 3월 8일

1. 속도 추정 알고리즘

1.1. 전체 모델 설명

속도 추정을 하기 위해서, 객체 검출 및 트래킹이 먼저 이뤄져야 한다. 그림 1과 같이 입력 영상이 들어오면 한 프레임씩 입력받아 전처리(Preprocessing), 객체 검출(Object detection), 객체 트래킹(Object tracking) 과정을 거친 후, 최종적으로 객체 트래킹의 결과물인 Bboxes와 track 클래스를 이용하여 객체별 속도를 추정한다.

▶ Preprocessing

이미지 전처리 과정으로서, 차로를 구분하기 위한 polygon 및 속도 추정을 위한 virtual detection line(VDL)을 생성함

▶ Object Detection

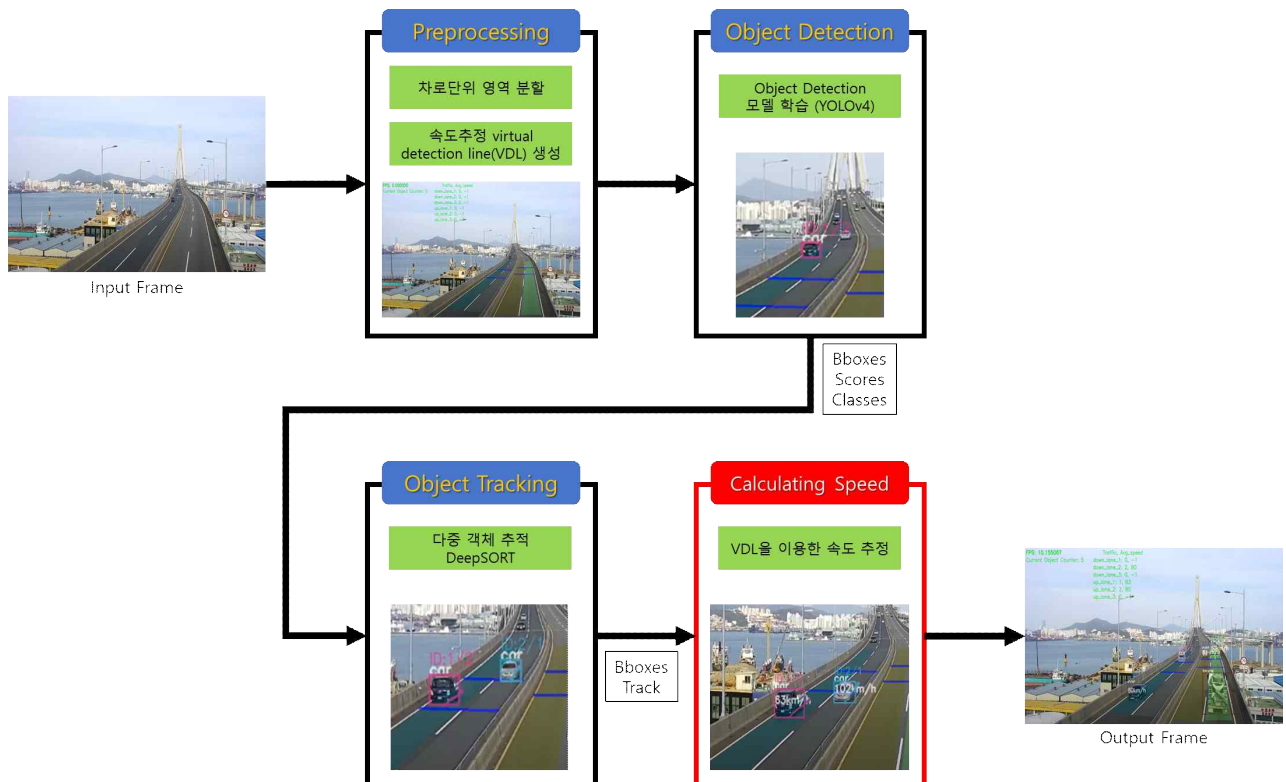
객체 검출 과정으로서, YOLO, Detectron2 등의 다양한 객체 검출 알고리즘을 이용하여 Bboxes, Scores, Classes 정보를 얻음

▶ Object Tracking

Object detection 모듈에서 얻은 Bboxes, Scores, Classes 정보를 Deep SORT, SORT 등의 객체 트래킹 알고리즘의 입력으로 사용하여 Bboxes, Track 정보를 얻음

▶ Calculating Speed

VDL을 이용하여 속도를 추정하는 모듈로서, Object tracking 모듈의 결과를 이용하여 객체별 차량 속도를 VDL을 완전히 통과한 순간 계산함.



[그림 1] 속도 추정 알고리즘 전체 모델

1.2. 속도 추정 알고리즘 설명

▶ Virtual detection line(VDL)을 이용한 속도 추정

- 그림 2와 같이 차로를 가로지르는 VDL 2개를 그려 차량이 각 line을 지나는 시간을 계산하여 차량의 순

KE-TI 한국전자기술연구원 Korea Electronics Technology Institute	VDL 활용 속도 추정 알고리즘 매뉴얼				
	문서분류	문서관리자	버전	최초작성일	최종수정일
	보고서	장수현	1.1	2021년 2월 15일	2021년 3월 8일

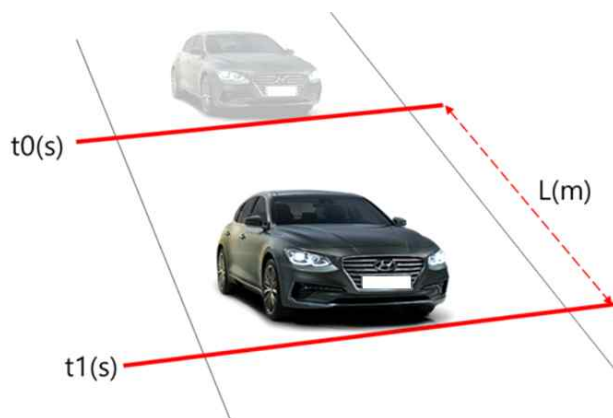
간 속도를 계산함

- 계산이 간단하며 빠르게 속도 추정할 수 있다는 장점이 있음
- 그러나, bounding box(Bbox) 좌표, 카메라 calibration 정확도에 따라 속도 오차가 발생할 수 있음
- VDL 2개를 완벽히 지나야지만 속도를 계산하기 때문에, VDL 내에서 정체가 발생하는 경우 속도 값을 얻기 힘들
- 지속해서 속도를 업데이트하지 못함

▶ VDL을 이용한 속도 추정 공식

- 그림 2의 상황일 때, VDL을 이용하여 속도를 추정할 시 아래와 같은 수식으로 속도를 추정할 수 있음

$$speed(km/h) = \frac{L(m)}{t1(s) - t0(s)} \times 3.6$$




[그림 2] virtual detection line을 이용한 속도 추정의 예시

2. 속도 추정 알고리즘 활용 방법

2.1. Code Tree

- 객체 검출 및 트래킹의 baseline 코드는 GitHub¹⁾를 참고하였으며 객체 검출은 YOLOv4, 객체 트래킹은 Deep SORT 모델을 사용
- 객체 검출 및 트래킹 그리고 속도 추정 알고리즘을 사용하기 위한 전체 모델은 표 1과 같은 폴더 구조를 가짐
- 표 1에 있는 파일/폴더에 대한 자세한 설명 및 코드공개 여부는 표 2에서 확인 가능

1) https://github.com/yehengchen/Object-Detection-and-Tracking/tree/master/OneStage/yolo/deep_sort_yolov4

 한국전자기술연구원 Korea Electronics Technology Institute	VDL 활용 속도 추정 알고리즘 매뉴얼				
	문서분류	문서관리자	버전	최초작성일	최종수정일
	보고서	장수현	1.1	2021년 2월 15일	2021년 3월 8일

[표 1] Code Tree

— ./setting.py
— ./calc_speed.py
— ./decode_np.py
— ./deep_sort
— ./deep_sort/detection.py
— ./deep_sort/detection_yolo.py
— ./deep_sort/_init_.py
— ./deep_sort/iou_matching.py
— ./deep_sort/kalman_filter.py
— ./deep_sort/linear_assignment.py
— ./deep_sort/nn_matching.py
— ./deep_sort/preprocessing.py
— ./deep_sort/tracker.py
— ./deep_sort/track.py
— ./input
— ./main.py
— ./model_data
— ./model_data/coco_classes.txt
— ./model_data/market1501.pb
— ./model_data/mars.pb
— ./model_data/mars-small128.pb
— ./model_data/obj.txt
— ./model_data/yolo_anchors.txt
— ./classes.txt
— ./tools
— ./tools/generate_detections.py
— ./yolo4
— ./yolo4/model.py
— ./yolo4/utils.py
— ./yolov4_anchors.txt
— ./yolov4_best_busan.h5

[표 2] 전체 모델 파일/폴더 세부 설명

이름	파일/폴더 구분	설명	코드 공개 여부
setting.py	파일	입력 영상에 따라 차로 영역 및 VDL 픽셀 좌표 정보 기입	비공개
calc_speed.py	파일	객체별 속도 추정 모듈	공개
decode_np.py	파일	입력 영상 프레임을 입력받아 YOLO 모델을 이용한 객체 검출 실행 파일	비공개
deep_sort	폴더	GitHub에서 clone 해 온 코드로써 track.py만 속도 추정 목적에 맞게 수정	일부 공개 (track.py 공개)

KE-TI 한국전자기술연구원 Korea Electronics Technology Institute	VDL활용 속도 추정 알고리즘 매뉴얼				
	문서분류	문서관리자	버전	최초작성일	최종수정일
	보고서	장수현	1.1	2021년 2월 15일	2021년 3월 8일

input	폴더	입력 영상 저장 폴더	비공개
main.py	파일	객체 추정 및 트래킹, 속도 추정 알고리즘을 사용하는 메인 실행 코드	비공개
model_data	폴더	GitHub에서 clone 해 온 코드로써 YOLO 모델을 사용하기 위한 폴더	비공개
classes.txt	파일	객체 검출 시, Car, Truck, Bus와 같은 클래스를 정의한 텍스트 파일	비공개
tools	폴더	GitHub에서 clone 해 온 코드로써 객체 검출 시, 사용하는 폴더	비공개
yolov4	폴더	GitHub에서 clone 해 온 코드로써 YOLO 모델 및 YOLO에서 사용하는 layer와 기능들을 정의한 폴더	비공개
yolov4_anchors.txt	파일	YOLOv4에서 사용하는 anchors를 정의한 텍스트 파일	비공개
yolov4_best_busan.h5	파일	1분 단위 500시간 06부산 지점 영상 환경에 맞춰 yolov4를 학습한 weight 파일	비공개

2.2. 속도 추정 함수 사용법

1. Baseline으로 사용한, YOLOv4 + Deep SORT 코드를 clone 한다.
2. deep_sort 폴더 내에 있는 track.py 파일을 제공한 track.py로 변경한다.
3. main.py 상단에 아래와 같이 입력하여, calcSpeed 함수를 import 한다.

```
from calc_speed import calcSpeed
```

4. 객체 검출 결과를 이용하여 객체 트래킹 모듈로부터 출력값을 얻은 뒤, 해당 출력값을 이용하여 아래 구문과 같이 속도 추정 함수를 사용한다.

```
track = calcSpeed(track, luLine, ldLine, ruLine, rdLine, bbox, frame_idx)
```

calcSpeed의 입출력 파라미터는 아래 표 3과 같다.

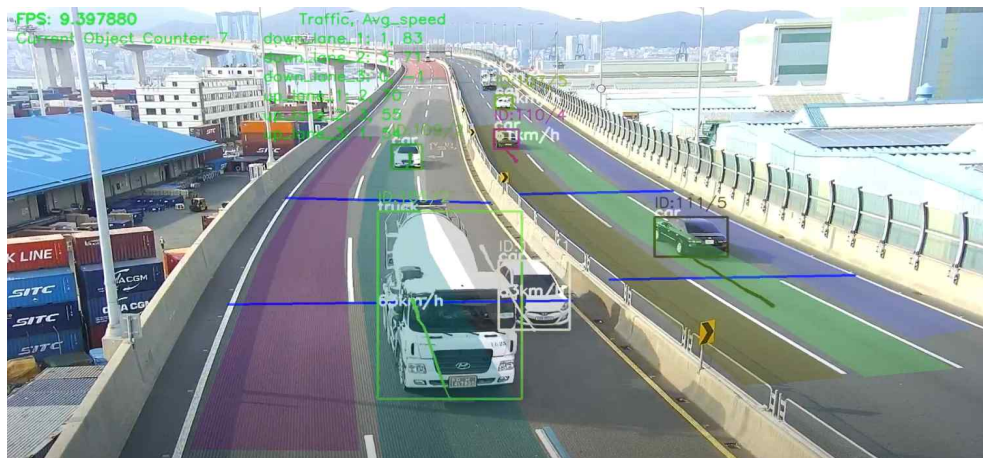
[표 3] calcSpeed 함수의 파라미터 설명

입/출력	파라미터 이름	설명
입력	track	track.py 내의 track 클래스, 트래킹 결과물로 얻음
	luLine	입력 영상으로부터 좌측 상단 VDL 픽셀값
	ldLine	입력 영상으로부터 좌측 하단 VDL 픽셀값
	ruLine	입력 영상으로부터 우측 상단 VDL 픽셀값
	rdLine	입력 영상으로부터 우측 하단 VDL 픽셀값
	bbox	Deep SORT로부터 얻은 bounding box(Bbox), Bbox의 포맷은 (min x, min y, max x, max y)
	frame_idx	입력 영상의 프레임 숫자
출력	track	track 클래스(차량 ID, 속도, Bbox 값 등 포함)

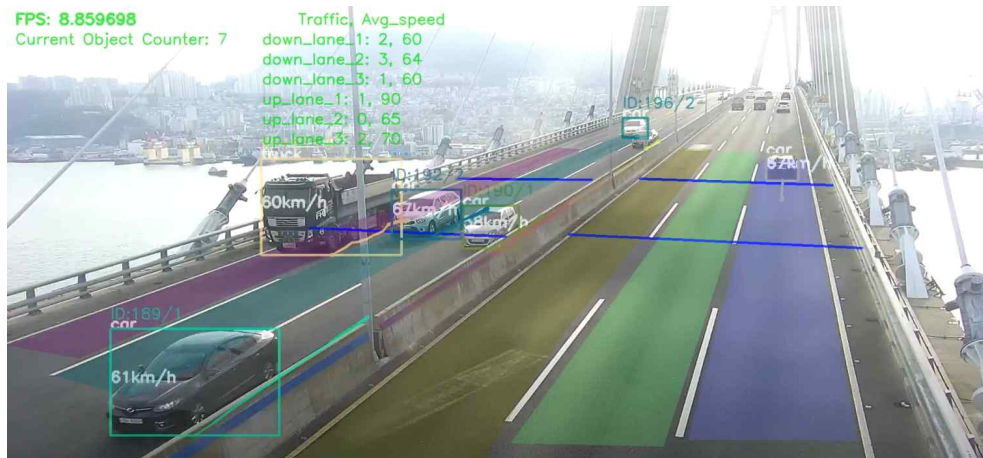
3. 속도 추정 알고리즘 결과

- ▶ 오프라인 영상 기반 객체 검출, 트래킹 및 속도 추정 소요 시간은 약 8~10fps
- ▶ 그림 3, 4는 객체별 검출, 트래킹 및 속도 추정 결과 이미지
 - 파란색 가로선 4개는 속도 추정에 사용한 VDL
 - 인퍼런스 시, 객체별 차종, 속도 및 차로별 차량 수, 평균속도를 얻을 수 있음

KEITI 한국전자기술연구원 Korea Electronics Technology Institute	VDL활용 속도 추정 알고리즘 매뉴얼				
	문서분류	문서관리자	버전	최초작성일	최종수정일
	보고서	장수현	1.1	2021년 2월 15일	2021년 3월 8일



[그림 3] 1분 단위 500시간 06부산 지점 영상 부산항대교6 속도 추정 알고리즘 결과 이미지



[그림 4] 1분 단위 500시간 06부산 지점 영상 부산항대교3 속도 추정 알고리즘 결과 이미지