碩士學位 請求論文

指導敎授 朴 炫 珍

3DCNN을 이용한 블랙박스 영상 속 차량 속도 예측

成均館大學校 一般大學院

데이터사이언스융합學科

鄭 安 栽

碩士學位 請求論文

指導敎授 朴 炫 珍

3DCNN을 이용한 블랙박스 영상 속 차량 속도 예측

The Estimation of Car Speed in Black Box using 3DCNN

成均館大學校 一般大學院

데이터사이언스융합學科

鄭 安 栽

碩士學位 請求論文

指導敎授 朴 炫 珍

3DCNN 이용한 블랙박스 영상 속 차량 속도 예측

The Estimation of Car Speed in Black Box using 3DCNN

이 論文을 工學 碩士學位請求論文으로 提出합니다.

2019 年 4 月 日

成均館大學校 一般大學院

데이터사이언스융합學科

鄭 安 栽

이 論文을 鄭 安 栽의 工學

碩士學位 論文으로 認定함

2019 年 6 月 日

審査委員長 _____

審査委員 _____

審査委員 _____

목차

표목차

그림목차

논문요약

3DCNN을 이용한 블랙박스 영상 속 차량 속도 예측

차량과 차량의 교통 사고시, 차량 속도는 사고의 과실 비율 및 보험료를 계산하는데 중요한 지표가 된다. 그리하여 사고가 난 후 보험사는 블랙박스의 영상을 보고 사고가 나기 전의 시간과 거리를 대략 짐작하여 차량의 속도를 예측하지만 이는 정확하지 않다. 대부분의 논문들이 제한된 거리나 3인칭시점에서의 주행 영상 혹은 블랙박스외의 여러 대의 카메라를 이용하여 차량의 속도를 재는 시도는 해왔지만, 블랙박스 영상 하나, 오직 1인칭 시점의 주행영상만으로 차량의 속도를 예측하거나 주행 영상 속 시공간적인 특징에 대한 연구는 없었다.

지난 몇 십년 동안 컴퓨터 비전은 인공신경망과 딥러닝 기술의 발전으로 인하여 비약적으로 발전하였고 영상 데이터, 시공간적인 특징을 잘 학습할 수 있는 3차원 컨볼루션 신경망(3D Convolutional Neural Network)이 나왔다. 3DCNN을 이용한 주된 연구들은 주로 3DCNN을 이용하여 영상 속의 행동을 파악하거나 분류를 하였다. 하지만 본 논문에서는 3DCNN중 C3D 구조를 통해 Regression, 1인칭 시점의 주행 영상을 통하여 영상 속 차량 속도를 예측하고 이를 위해 모델이 어떤 시공간적인 특징에 집중하는지 연구하였다. 결과적으로 3분길이의 21개 검증 데이터, 다양한 속도 및 환경의 동영상들에서 RMSE(Root Mean Squared Error) 평균 제곱근 편차 5.1 km/hr, 정확도 82%의 결과를 얻을 수 있었다.

주제어 : 블랙박스, 차량 속도 예측, 3DCNN, C3D, 비디오 분석

제 1 장 서 론

[11] 에서 나온 통계 중, 2015년 한국의 교통사고에서 과속으로 인한 교통사고 건수는 전체 교통 사고 발생 중 0.3%에 불과하다. 이는 2015년 한해 동안 과속단속카메라에 단속된 적발 건수 56만여건에 비해 과속이 원인이 되는 사고율은 0.1%이고 이 의미는 교통사고 발생시 차량 속도에 대한 추정 및 연구를 하지 않았기 때문이다. 또한 보험회사에서 교통사고시의 차량 속도는 보험료를 계산하거나 보험사기를 방지하기 위한 중요한 지표가 된다. 요즘에 나온 좋고 비싼 블랙박스들은 속도도 1초마다 나오지만 2000년 초반이나 GPS 호환이 안되는 블랙박스들은 속도가 표시되지 않아서 아직도 많은 차량에 설치되어있는 블랙박스에는 속도가 나오지 않는다. 대부분의 차량 속도를 구하는 시도를 한 논문들은 고정된 거리에서 지나가는 차들의 속도를 여러 대의 다른 카메라를 이용하는 방법들이었다. 특히 여러 대의 카메라를 사용하는 방법, 이는 [12], 스테레오 비전과 관련이 있다. 하나의 카메라로 찍은 사진은 3차원의 물체를 찍었다 하더라도 물체의 반사광이 2차원 센서에 투영 시키기 때문에 카메라에서부터 물체까지의 거리 정보는 사라지게 된다는 의미이다. 그리하여 사라진 거리 정보를 복원하기 위해서는 다른 위치에서 찍은 다른 사진이 필요하게 된다. 스테레오 비전이란 여러 대의 카메라를 이용하여 같은 물체에 대해 각각 다른 장소에서 촬영한 이미지들을 이용하여 촬영한 물체의 3차원 정보를 계산하는 학문분야이다. 하지만 차에 블랙박스를 2대를 다는 사람은 아무도 없다. 실제로

하나의 카메라로만 속도를 예측 하려하면 직접 눈으로 보고 본 거리와 시간을 어림짐작하여 기본 물리, '속력 = 거리 / 시간' 을 이용한 시도로 재야 하지만, 이러한 시도와 결과는 하나하나의 영상 마다 거리를 직접 계산해야 하고 많은 한계가 있기 때문에 정확한 속도를 구하기가 어려웠다.

처음엔 버클리 대학에서 오픈소스로 내놓은 자율주행 데이터로 사용하려 했지만 영상에 속도를 입히고 눈으로 확인해본 결과 오류가 너무 많아 학습 데이터로 쓸 수가 없었다. 그리하여 직접 속도가 기록되는 좋은 기능의 블랙박스를 설치하고 낮과 밤을 가리지 않고 고속도로, 일반도로 등 다양한 도로환경을 오가며 데이터를 모았다. 이렇게 모은 데이터들은 블랙박스 K900 특성상 3 분으로 나눠져 수집되었고 속도 데이터 1 초마다 기록이 되어 csv 파일로 나왔다. 후진하는 영상을 제외하고, 영상은 30 프레임, 1 초에 30 장의 이미지를 추출할 수 있었고 데이터 증식과 더 자세한 예측을 위해 1 초사이의 속도를 30 개로 선형적으로 나누어 주었고, 478 개의 3 분짜리 영상, 25 시간 가량의 주행 영상을 수집할 수 있었다.

	KITTI	Cityscapes	ApolloScape	Mapillary	BDD100K
# Sequences	22	~50	4	N/A	100,000
# Images	14,999	5000 (+2000)	143,906	25,000	120,000,000
Multiple Cities	No	Yes	No	Yes	Yes
Multiple Weathers	No	No	No	Yes	Yes
Multiple Times of Day	No	No	No	Yes	Yes
Multiple Scene types	Yes	No	No	Yes	Yes

그림 1-1 BDD100K 데이터

순번	날짜	시간	요일	위도	경도	값	주행거리	GPS속도
962	2019.4.4	9:14:13	THU	3731.1734	12701.6849	A	2.19	38
963	2019.4.4	9:14:14	THU	3731.1673	12701.6873	A	2.2	43
964	2019.4.4	9:14:15	THU	3731.1607	12701.6901	A	2.21	48
965	2019.4.4	9:14:16	THU	3731.1536	12701.6937	A	2.22	51
966	2019.4.4	9:14:17	THU	3731.1464	12701.6982	A	2.24	54
967	2019.4.4	9:14:18	THU	3731.1387	12701.7027	A	2.26	57
968	2019.4.4	9:14:19	THU	3731.1309	12701.7067	A	2.27	56
969	2019.4.4	9:14:20	THU	3731.1231	12701.7105	A	2.29	53
970	2019.4.4	9:14:21	THU	3731.1154	12701.714	A	2.3	53
971	2019.4.4	9:14:22	THU	3731.1077	12701.7177	A	2.32	54
972	2019.4.4	9:14:23	THU	3731.1	12701.7216	A	2.33	56
973	2019.4.4	9:14:24	THU	3731.092	12701.7257	A	2.35	57
974	2019.4.4	9:14:25	THU	3731.0836	12701.7296	A	2.36	59
975	2019.4.4	9:14:26	THU	3731.075	12701.7336	A	2.38	61
976	2019.4.4	9:14:27	THU	3731.0581	12701.7416	A	2.41	57
977	2019.4.4	9:14:28	THU	3731.0504	12701.7453	A	2.43	54
978	2019.4.4	9:14:29	THU	3731.0433	12701.7487	A	2.44	47
979	2019.4.4	9:14:30	THU	3731.0372	12701.7515	A	2.46	40

그림 1-2 K900 블랙박스 데이터

본 논문은 다음과 같이 구성되어 있다. 2 장 에서는 3D 컨볼루션 신경망에 대해서 설명하고 3 장에서는 속도에 의해 영향 받는 특징을 잡아내기 위한 이미지 전처리 방법들에 대해 설명한다. 그리고 마지막 4 장에서는 본 논문에서 제안하는 방법과 결과들에 대하여 정리하고 결론을 맺는다.

제 2 장 3D 컨볼루션 신경망

2.1 CNN 설명

Convolutional Neural Network(CNN)은 딥러닝에서 많이 사용되는 알고리즘 중 하나로 주로 이미지를 다루는 컴퓨터 비전 분야에 많이 쓰이고 있다. CNN은 뉴럴 네트워크 앞쪽에 여러 컨볼루션 레이어를 통해서 입력 받은 이미지에 대한 특징을 추출하게 되고 이를 뉴럴 네트워크를 통해 원하는 output을 만들어 낼 수 있다. 컨볼루션 레이어는 입력데이터에서 특징을 추출하는 일을 하는 필터와 이 필터값을 비선형값으로 바꾸어 주는 활성 함수로 이루어진다. 필터는 말그대로 데이터에 그 특징이 있는지 걸러주는 함수이고 활성함수는 필터함수를 통해 특징이 추출된 후, 추출한 값을 비선형 값으로 바꿔준다. 예를 들어 만약 시그모이드 함수를 쓴다면 필터에 들어간 이미지의 특징이 있다면 1에 가까운 값이나 0에 가까운 값으로 리턴해준다. 컨볼루션 레이어를 통해 추출된 특징들은 필요에 의해 풀링이라는 단계를 거치는데 모든 특징을 사용하는 대신 특징의 값이 큰값이나 평균을 내어 사용한다. 그리하여 이렇게 컨볼루션 레이어는 필터와 활성함수 그리고 풀링레이어를 반복적으로 조합되어 특징을 추출하고 마지막에 전결합층(Fully Connected Layer), 뉴럴 네트워크에 넣어서 원하는 결과값을 얻는다. 마지막 출력층에서 이미지 분류에서는 주로 마지막에 소프트맥스함수를 써서 이미지를 분류하는데, 시그모이드 함수와 달리 소프트맥스함수는 여러 개의 분류를 할 수 있다. 클래스

분류를 할 때 클래스 별 확률로 변환하여 나타내어 출력이 0에서 1사이, 그리고 합이 1이 되도록 한다. 하지만 본 논문의 연구에서는 이미지 input을 통해 숫자를 예측하는 것 이였기 때문에 소프트맥스함수 대신 값이 그대로 나오도록 출력 층 output layer를 1로 만들어 주어 출력 값이 숫자하나로만 나오게 만들었다.

2.2 2DCNN과 3DCNN 비교 및 설명

2D 와 3D 는 구조적으로 큰 차이가 있는데 2D 는 이름 그대로 2 차원 배열 구조와 간단한 격자 합성 곱 필터를 적용할 수 있다. 하지만 3D 에서는 단순 이미지를 넘어 시공간적 특징도 학습에 표현할 수 있고 합성 곱 또한 3 차원의 필터가 적용된다. 2D CNN 와 비교하면, 3D CNN 는 3D 컨볼루션과 3D 풀링으로 인해 시간적 정보를 모델링할 수 있다. CNN 모델에서 깊이 부분이 결정적으로 다르다, 2DCNN 에서는 깊이가 하나의 이미지의 채널이지만 3DCNN 에서의 깊이는 여러 겹의 이미지가 쌓인 형태이다. 2D CNN 은 여러 이미지에 적용되더라도 채널로 인식하기 때문에 하나의 이미지만을 생성시키지만, 3D 컨볼루션은 입력 이미지의 시간 정보를 보존하고 생성 결과물이 volume 형태가 된다.

2D convolution :

$$v_{ij}^{xy} = \tanh\left(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pq} \, v_{(i-1)m}^{(x+p)(y+q)} \right)$$

3D convolution :

$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} \, v_{(i-1)m}^{(x+p)(y+q)(z+r)} \right)$$

위 공식에서 (x,y,z)는 특징 볼륨의 좌표이고, p 와 q 는 커널의 시공간을 나타내고 r 은 시간적 차원을 나타낸다. jm 은 특징 볼륨을 의미하고, i 는 컨볼루션 레이어를 의미한다. b_{ij}는 특징 볼륨의 bias 를 의미한다. tanh()는 활성함수가 hyperbolic tangent 라는 것이다. 두 공식을 비교 하였을 때 위에서 설명하였던 시공간적 특징을 학습하는 또 하나의 Sum 함수, R 이 2DCNN 과 달리 3DCNN 에 존재한다.

2.3 3DCNN 관련 연구

행동 인식은 지난 수십 년 동안 컴퓨터 비전에서 깊이 연구되었고 이 분야에는 3DCNN 모델이 가장 좋은 성능을 보여주었다. 행동 인식은 주변 배경을 무시하고 큰 이미지의 움직임에만 집중하지만 본문의 주제는 모든 이미지에서 움직이는 것들에 하나하나 반응해야한다. 그 뜻은, 본문의 주제에서 움직이는 모든

것이 다 중요한 특징이 되는 것이 아닌, 속도에 의해 일정하게 움직이는 것들만 잡아내야 정확한 속도 예측이 가능하다. 예를 들어, 도로 선은 속도 추정을 위한 훌륭한 지표가 될 수 있지만, 반대 방향에서 오는 다른 차량은 혼동 요인이 될 수 있다. 그럼에도 불구하고 같은 3DCNN 모델을 사용했고 영상 속 시공간적인 특징에 대해 연구했기 때문에 관련 연구를 공부하는 것은 본 논문 연구에 도움을 주었다.

3차원 컨볼루션 신경망을 이용한 연구에는 [1, 2, 4] 등이 있다. 그 중에서도 [1]에서 제안한 C3D라는 모델 형태가 본 논문의 실험에서 가장 좋은 결과를 나타내었다. 이 논문에서는 3DCNN은 2DCNN보다 시공간적인 특징을 학습하는데 더 효과적인것을 강조하면서 3x3x3의 kernel과 2x2x2의 stride인 통일된 형태를 쓰는 것이 3DCNN에서 가장 성능이 좋다고 하였다. 하지만 맨 처음 풀링레이어의 kernel과 stride는 2x3x3과 1x2x2였는데 이는 풀링 사이즈를 조금 줄임으로서 처음 레이어에서 시간적 특성이 너무 빨리 없어지는 것을 방지하기 위함이라고 하였다. [2]의 연구는 3DCNN 뿐만 아니라 Convolutional LSTM 을 붙여 제스쳐 인식률을 높였다. Convolutional LSTM은 보통 CNN와 LSTM 내부 오퍼레이션 자체에 컨볼루션을 넣은 형태로 output또한 이미지로 나온다. 본 연구에도 적용해보았지만 결과가 좋게 나오지 않았다. 연구 [4] 또한 3DCNN으로 학습했지만 형태는 달랐다. Residual의 형태를 가진 3DCNN으로 영상 분류를 시도하였다. LSTM처럼 처음 3DCNN 레이어 값이 끝에 쪽으로 유지되어 가는 형태이고 Batch Normalization과 relu가 합쳐지었다. 본 논문에도 Residual 3DCNN을 실험해보았지만 값이 Nan으로 나왔다. 이유는 처음 값이 끝 쪽의 레이어로 유지되어 가는 방식은 시간적 특징을 학습하는 본 논문의 주제와는 맞지 않기 때문이었다.

2.4 C3D Regression Model 설명

3DCNN에서 모델의 input size는 Depth * Height * Width * channel를 가지며, 이는 각각 frame의 길이, frame의 높이, 폭 그리고 channel을 나타낸다. Convolution kernel size는 3 × 3 × 3이며, pooling kernel size는 Depth * Kernel * Kernel로 나타내고, Depth 와 Kernel은 각각 temporal depth 및 spatial size를 나타낸다.

이 논문에서 사용한 모델에 대하여 자세히 기술하고자 한다. Input dimension 은 3 × 16 × 112 × 112 이며, 5 그룹의 convolutional layers 와 5 개의 pooling layers, 그리고 2 개의 fully-connected layers 및 마지막 출력층 fully-connected layers 하나로 구성되어 있다. Convolution layer 의 filter 수는 각각 64, 128, 256, 512, 512 개이고 모든 convolution kernel 의 size 는 2x2x2 이다. Stride 과 zero padding 을 사용하여 convolution layers 간의 크기 변화는 없지만 모든 pooling layer 의 kernel size 가 2 × 2 × 2 의 max pooling 을 사용하기 때문에 input 에 비해 output 의 size 가 8 배수로 감소하게 된다. 처음 input 크기는 (16, 112, 112, 64)이지만 fully-connected layers 들어가기전의 사이즈는 (1, 4, 4, 512)로 바뀌고 두개의 fully-connected layer, 4,096 개와 연결되고 마지막으로 출력층 하나의 fully-connected layer, linear 함수를 가진 output 을 가진다. 4096 개의 FC 사이에는 50%의 dropout 을 가진다.

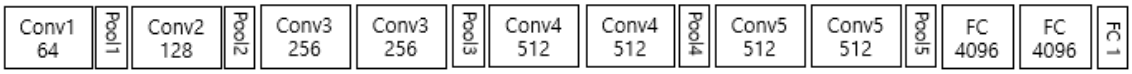


그림 2-1 C3D for Regression Model Structure

C3D Regression Model 은 차량 속도 예측을 위해서 주행영상에서 feature extractor 로서의 역할을 수행할 수 있다. 속도 예측에 중요한 feature 을 추출하기 위해서, 영상은 16 frame 으로 나뉘고 이중 8 개의 frame 은 다음 8 개의 frame 과 중첩 시킨다. 이 중첩된 clip 은 C3D network 로 전달되고 relu activation 을 거쳐 4096-dimension video descriptor 가 생성되고 마지막으로 L2 normalization 을 거치게 된다. C3D 초반의 convolution layer 는 초기 몇개의 frame 을 통해 전체적인 모습을 구상하고, 이후 frame 에서는 두드러진 action 을 추적한다.

3D 컨볼루션을 이용하여 영상으로부터 시공간적인 특징을 학습하는 것이 가능하며, video 분석을 위한 여러 2D 컨볼루션에 비해 좋은 성능을 보여주고 있다. 성능의 우수성을 제외하고도, 이 모델은 영상 분석이라는 많은 computing power 가 요구됨에도 불구하고, compact 하면서 simple 한 구조는 computation 부하를 줄이다는 관점에서 상당한 장점을 내포하고 있다.

제 3 장 다양한 이미지 전처리 시도

3.1 optical flow 설명

2D이미지의 경우, 최초 시간 t에서 이미지의 위치를 (x,y)라고 하고 intensity를 I(x,y,t)라고 하고 짧은 시간 동안(dt) 픽셀의 intensity는 변하지 않고 위치의 변화만 일어났다 하자. 각각의 x와 y축에서 조그마한 움직임 dx와 dy가 일어났다면, 그때의 intensity는 I(x + dx, y + dy, t + dt)라고 할 수 있다. 그러면, 아래와 같은 식이 성립하게 된다.

$$I(x, y, t) = I(x + \ dx, y + dy, t + dt) \ - (1)$$

(1)의 식을 테일러 전개를 하면 (2)와 같은 식이 된다.

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{dI}{dx}dx + \ \frac{dI}{dy}dy + \frac{dI}{dt}dt \ - (2)$$

이때 (1)과 (2)의 방정식이 성립하려면, (2)의 오른쪽에 있는 미분식의 합이 0이 되어야 한다.

$$\frac{dI}{dx}dx+ \frac{dI}{dy}dy + \frac{dI}{dt}dt = 0$$

그리고 dx, dy, dt 를 다시 쓰면,

$$\frac{dI}{dx}\frac{dx}{dt}+ \frac{dI}{dy}\frac{dy}{dt} + \frac{dI}{dt}\frac{dt}{dt} = 0$$

dx/dt, dy/dt 는 Vx, Vy, x 방향의 속도이다. 또한, dI/dx, dI/dy, dI/dt 는 각각 Ix, Iy, It, 이미지 I에 대한 x와 y의 변화량 이다. 마지막으로 다음식은 아래와 같이 쓸 수 있다.

$$IxVx + IyVy = - It$$

매트릭스 식으로는 아래처럼 같이 쓸 수 있다.

$$\nabla I^T \cdot \vec{V} = -I_t$$

요약하자면, 이미지의 X와 Y의 Moving Vector를 구하는데 있어서, 각각의 축으로 미분하고, 시간축으로 미분하여 식에 대입해서 Matrix를 풀면 구할 수 있다.

3.2 Gunnar farneback’s algorithm & Optical Pyramid를 이용한 다양한 시도

Comma ai라는 미국 블랙박스 회사에서 2018년 7월 12일에 speed challenge라는 블랙박스 속 1인칭 주행 영상 데이터를 이용한 속도 예측 대회를 열었다. 데이터로는, 20400 프레임 약 17분가량의 주행데이터를 오픈소스로 내놓았고 각 프레임, 이미지 마다 y라벨, 속도가 있었다. 좋은 성적을 낸 참가자중 한명이 쓴 알고리즘은, 앞서 설명한 Optical flow 종류 중 Gunnar farneback's algorithm, dense optical flow 알고리즘은 인접한 두 프레임간의 움직임을 계산하는 알고리즘이다. 장점은 모든 픽셀들의 displacement를 계산하기 때문에 이미지의 움직임을 포착하는 정확도가 높지만 모든 픽셀의 움직임을 계산하고 moving vector를 그림으로 표현하기 때문에 계산 시간이 오래 걸린다. 이 방법을 이용한 [8]이 moving vector를 그림으로 생성한 것을 Alexnet에 input으로 넣고 y를 근접한 두 이미지의 속도 평균을 넣어 모델링을 하였고 이는 좋은 결과를 보였다.

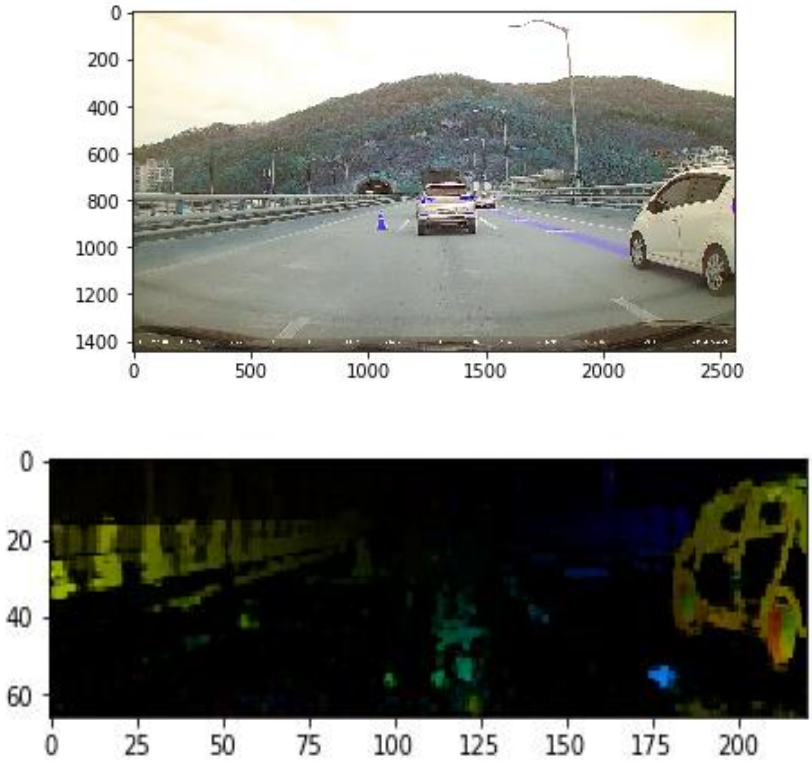


그림 3-1 Speed Challenge Dense Optical Flow

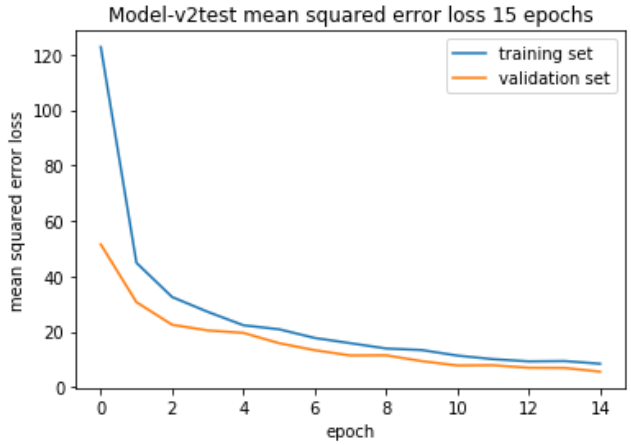


그림 3-2 Speed Challenge Dense Optical Flow loss graph

그래서 이 방법을 직접 수집한 K900 에 적용하였지만 그리 좋은 결과가 나오지 않았다.

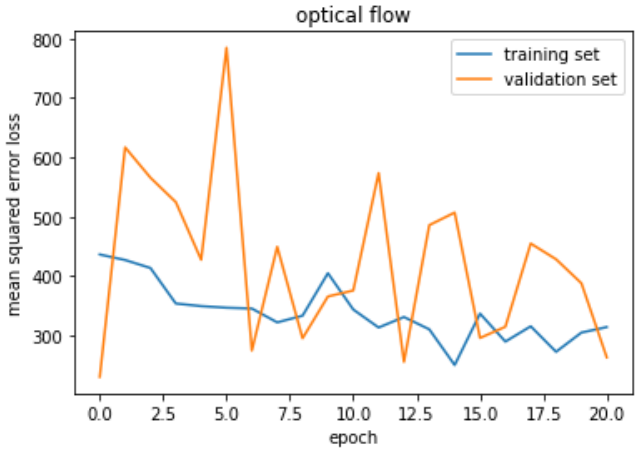


그림 3-3 K900 Optical Flow Loss Graph

Loss 가 학습할수록 loss 의 그래프가 위아래로 이동하는 것을 보아하니 K900 데이터에서 dense optical flow 를 썼을 때 속도를 예측하기 위한 좋지 않는 특징이 잡히기 때문이라고 생각된다. 예를 들어 굉장히 천천이 운전하거나 감속 및 가속이 많을 경우 인접한 두 프레임의 dense optical flow 를 실시한다 하더라도 이것이 일정한 속도로 운전하는것과 비슷한 형태로 나올 수 있기 때문이다.

또 다른 방법으로는 optical pyramid 란 방식으로 dense optical flow 와 다르게 전체 픽셀의 moving vector 를 포착하는 대신 특정한 점의 움직임만 파악하고 이의 움직임만 계산하는 방식이다. 이 방법을 이용해 지정된 영역안에 도로선을 잡는 방법은 [9, 10], speed challenge 에서 좋은 예측률을 보인 참가자의 아이디어였고 speed challenge 데이터로 구현하였다.

그림 3-4 Optical Pyramid 를 이용한 차선 추출

Optical pyramid 옵션 중 지정한 영역에서만 특징점을 찾게 하는 옵션이 있어서 도로선이 들어올만한 영역을 지정해준 후 특징점의 위치값을 리턴하게 하였다. 그리고 그 값을 linear regression 을 통해 coeffient 값을 구한 후 나머지 test 를 예측하였다.

Train & Test Graph

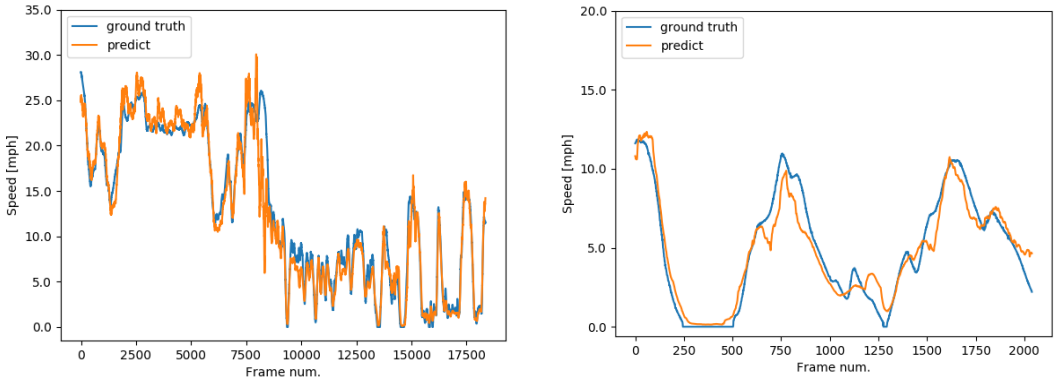


그림 3-5 차선 추출 Speed Challenge Train & Test Graph

각각 좋은 결과가 나와서 실제 블랙박스로 수집한 K900 데이터에 적용해 보았지만 결과가 좋게 나오지않았다. 그 이유는 speed challenge 영상의 대부분은 앞차와 일정거리를 두고 주행하기 때문에 지정된 영역에 차선 이외의 것들은 잘들어오지 않았지만 직접 수집한 데이터에서는 많은 도로와 상황들이 나오기 때문에 저 지정된영역에 도로선이외의 많은 것들이 들어왔기에 예측이 잘 안되었다.

Train graph

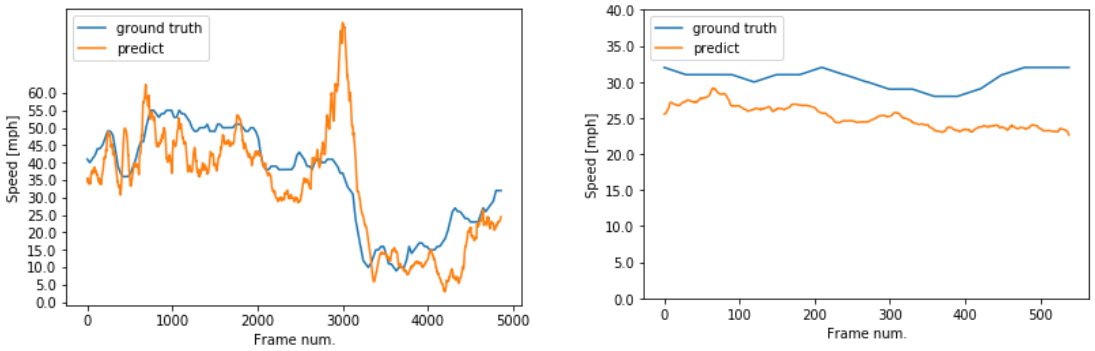


그림 3-6 차선 추출 K900 Train & Test Graph

결론적으로 다양한 환경 및 상황에서 특정한 알고리즘만으로 속도를 예측한다는 것은 불가능 했다.

3.3 Hough Transform

허프 변환의 기본아이디어는 하나의 점이 지나는 모든 직선을 방정식, b = -mx + y 라 하였을 때 b 와 m 에 대한 평면에서 하나의 직선으로 표현할 수 있다는 것이다. 하지만 m-b Parameter Space 에서 표현할 수 없는 문제가 있는데 이경우는 기울기가 무한대일 경우이다. 따라서 이 문제를 해결하기 위해 P-θ parameter Space 가 생기었고 이를 Hough Space 라고 한다.

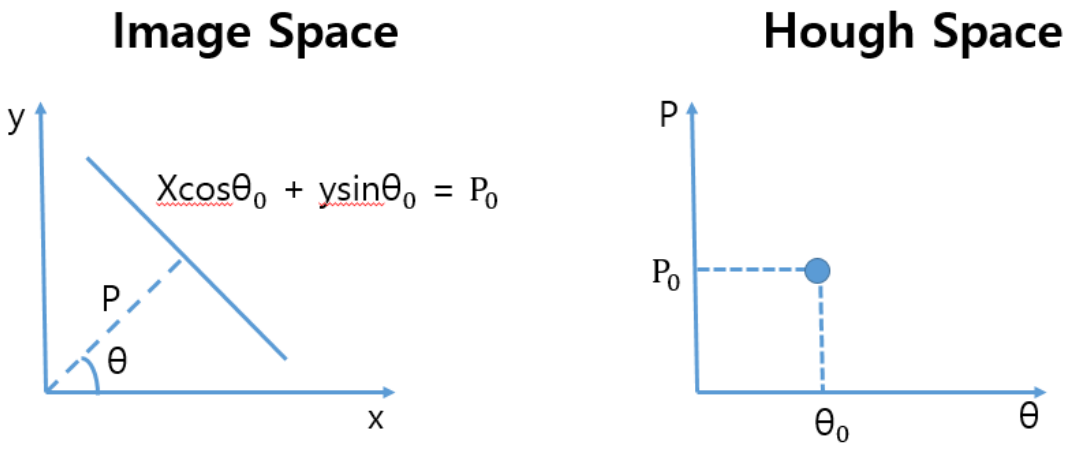


그림 3-7 Image Space & Hough Space

그리하여, m 의 기울기가 무한이 되는 문제를 해결하였고 똑같이 선들의 교차점이 Image Space 에서 각 점들을 잇는 직선이 된다. 하나의 점을 기준으로 점의 직선을 지나가는 모든 직선의 b 와 m 을 저장해놓고, 같은 직선에 속하는 점들이 몇 개나 있는지 여러 개의 점들에 대해 반복하여 검사하는 방법이다. 같은 직선에 속하는 점들이 몇 개나 있는지 검사하는 방법이다. 그리하여 많은 점들이 같은 직선상에 존재한다면, 저장된 파라미터를 이용하여 이를 직선으로 표현 후 직선을 검출 하는 방법이다. [15]를 참조하여 K900 데이터에 이를 이용하여 차선 검출을 실시하였다.

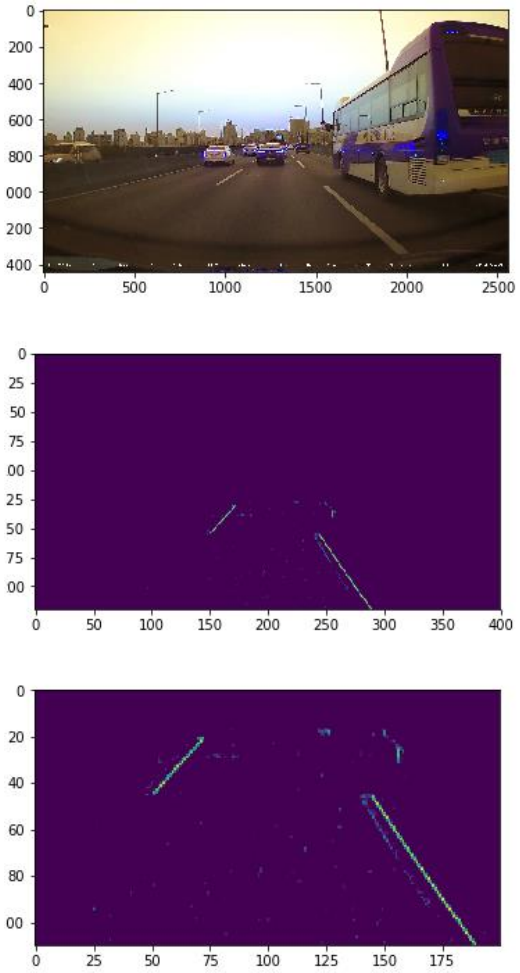


그림 3-8 Hough Transform 을 이용한 차선 검출

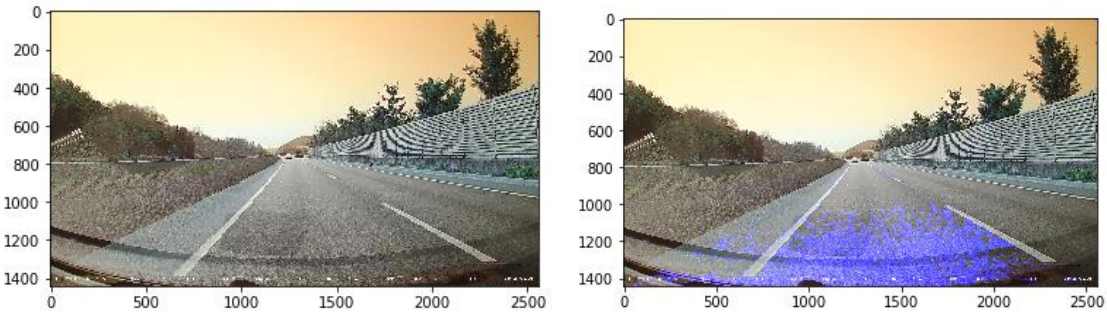


그림 3-9 Hough Transform 차선 검출 실패

그림 3-7 처럼 햇빛이 강력하지 않고 차선이 뚜렷이 잘 보일 때는 직선 검출을 이용한 차선 검출이 잘되었지만 그림 3-8 처럼 햇빛이 강렬할 때에는 차선 검출이 잘되지 않고 다른 것들이 검출 되었다. 앞서 설명한 Optical Flow 를 이용한 방법들과 같이 실제의 여러가지 상황에서 유용한 알고리즘은 아니었다. 그리하여 많은 상황에서도 예측이 잘되는 그런 모델링이 필요하였다.

제 4 장 실험 결과 및 분석

4.1 학습 방법 및 실험결과

설치한 k900 블랙박스 데이터 특성상 영상이 3 분마다 잘라져 나왔고 속도는 csv 에 한꺼번에 붙여 나왔다. 영상 제목과 csv 와 key 값이 있어서 매칭이 가능하였고 데이터 전처리를 통하여 30 프레임, 이미지한장마다 속도 라벨을 붙여줄 수 있었다. csv 에 나오는 데이터는 1 초마다 속도가 있었지만 1 초사이를 영상의 frame 수, 30 만큼 선형으로 나누어 0.03 초마다의 속도와 프레임 마다 이미지를 확보하였다. 1 초안에 속도가 바뀌지않는다는 전제하에 속도 데이터 전처리를 하였다. 총 478 개 3 분영상, 25 시간정도의 데이터를 확보했고 이미지로 환산을 하면은 478 개 동영상 * 180 초 * 30 프레임 = 2,581,200 개의 이미지를 확보할 수 있었다. Train 과 테스트의 비율을 95%와 5%로 나누어 457 개의 3 분영상을 학습, 21 개의 영상을 검증으로 쓰었고 train generator 를 이용하여 457 개의 학습 영상들 중 하나를 랜덤으로 뽑고 3 분길이안에서 어느 한 구간을 또한 랜덤으로 찍은 후 거기서 연속적인 16 프레임, 대략 0.5 초 길이의 이미지를 추출하였다. 이는 대용량의 동영상을 한번에 올리기에는 메모리가 과다하기 때문에 효율적인 학습방법 이었다. 각각의 연속된 이미지 전처리는 BGR 에서 RGB 로 변환 후 학습에 도움이 안되는 도로 위 부분, 하늘 부분을 자르고, 사이즈를 112x112 로 줄인 후 위아래에 padding 으로 채워주었다.

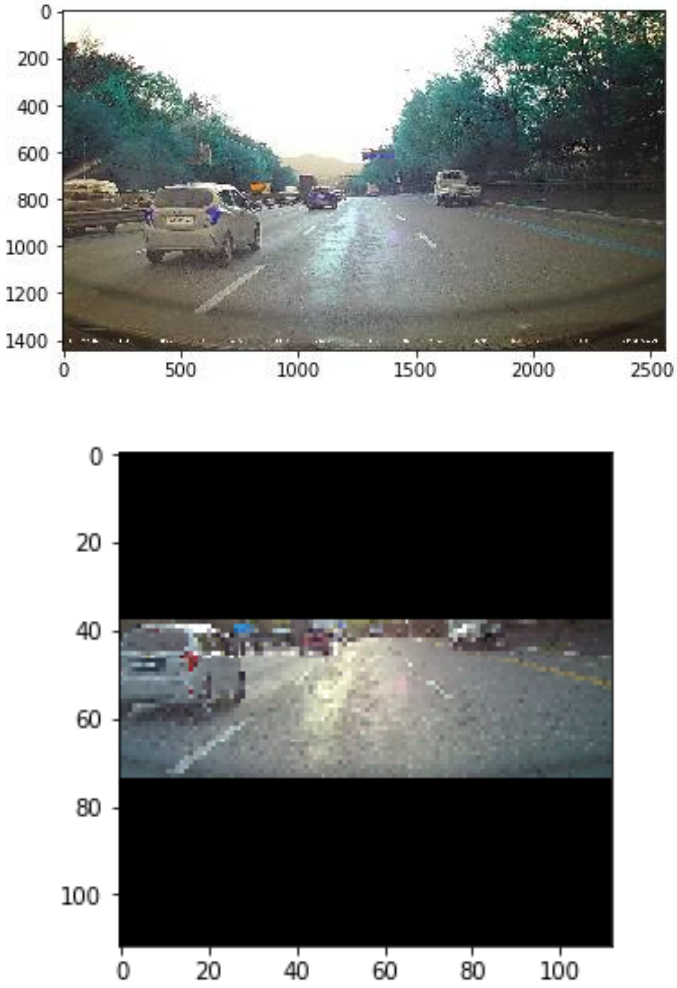


그림 4-1 이미지 전처리

이외에도 C3D 형태뿐만 아니라 C3D 형태에서 마지막 블락과 풀링을 제거해가며 기존의 input 길이, 16 개의 이미지를 8 개 그리고 4 개로 실험해보았다. 그뿐만 아니라 3DCNN 에서 비디오 분류 모델 중 좋은 성능을 보였던 Inception 3DCNN 을 실험해보았고, 앞서 설명한 Optical Flow 나 Optical Pyramid 를 적용한 이미지를 넣는 시도도 하였다.

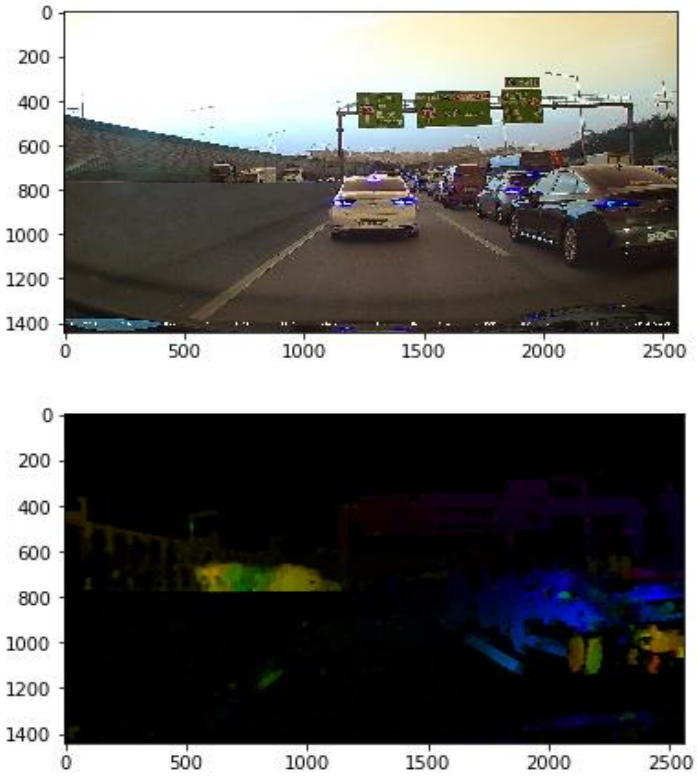


그림 4-2 K900 Optical Flow Input Image

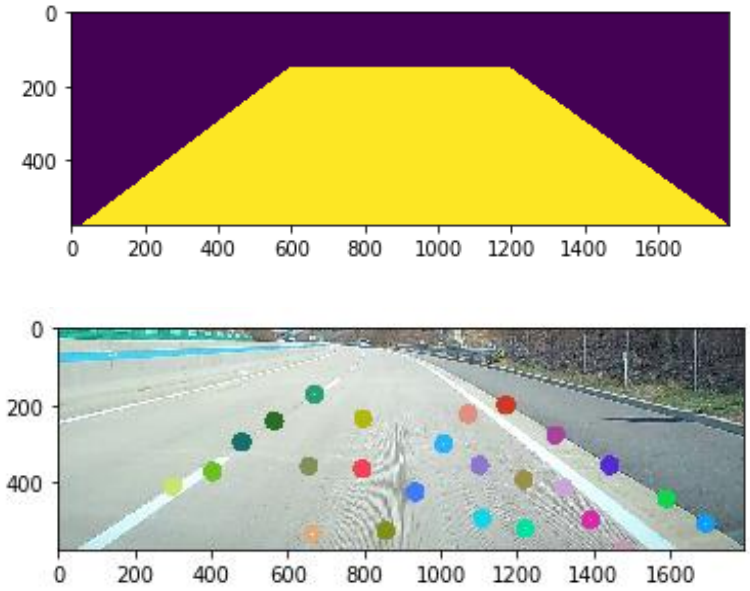


그림 4-3 Optical Pyramid Input Image

단위 km/h(kph)

Loss 단위 MSE Sqrt

method/loss	Train_loss	Validation_loss
C3D(Length=16, Optical Flow Pyramid input)	13.442	11.691
C3D(Length=16, Dense Optical Flow input)	20.248	18.275
Inception 3D CNN	13.427	15.141
C3D(Length=8)	11.952	10.562
C3D(Length=4)	13.500	12.326
C3D(Length=16)	**4.513**	**5.538**

표 4-1. 방법 및 모델링에 따른 MSE Square Root Top 5

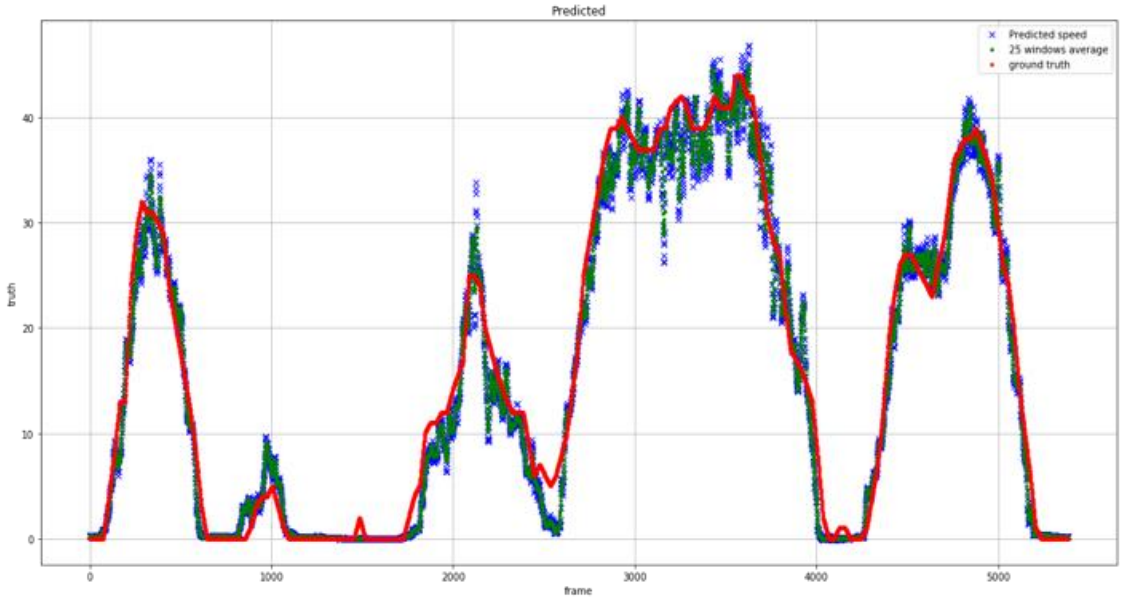
표 4-1 에 나온 방법 외에도 다른 모델링이나 C3D 형태에서 batch normalization 이나 pooling 및 kernel 사이즈를 바꾸고 여러가지 방법을 하였지만 위 표에 나온 방법들이 많은 시도 중에서 가장 좋은 결과를 보였다. 그중 단연 C3D 형태, 연속적인 input image 가 16 개, 0.48 초가 들어갔을때 가장 좋은 결과가 나왔다. 이는 연속되는 이미지가 16 개일때 속도를 예측할때 가장 좋은 특징이 잡힌다고 생각이 된다. Optical flow 나 Pyramid 는 앞서 말한 움직이는 특징들이 C3D 형태에서 시공간특징을 학습할 때 전처리를 안한 이미지보다 오히려 안좋은 노이즈를 가져와 주었기 때문에 오히려 결과가 안좋게 나왔다. Inception 3DCNN 또한 C3D 보다 파라미터 수가 적었지만 시공간적인 특징을 학습 할 때에는 C3D 보다 학습률이 저조하였다.

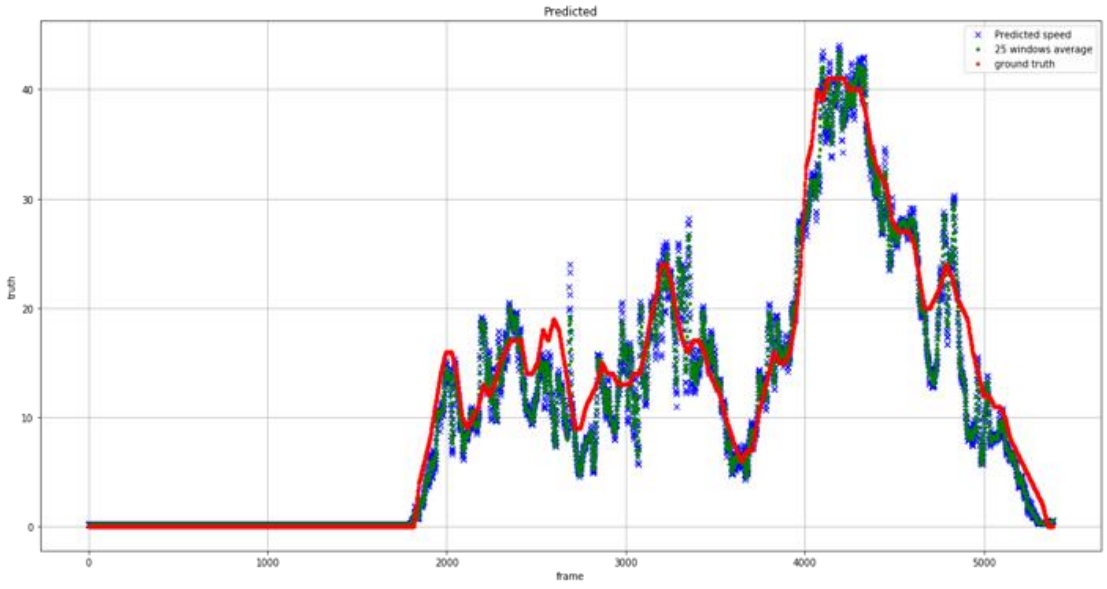
※ 21 개의 테스트 데이터 C3D result
※ error_rate = MSE_sqrt / speed_mean
※ error_rate_smooth = MSE_smooth_sqrt / speed_mean

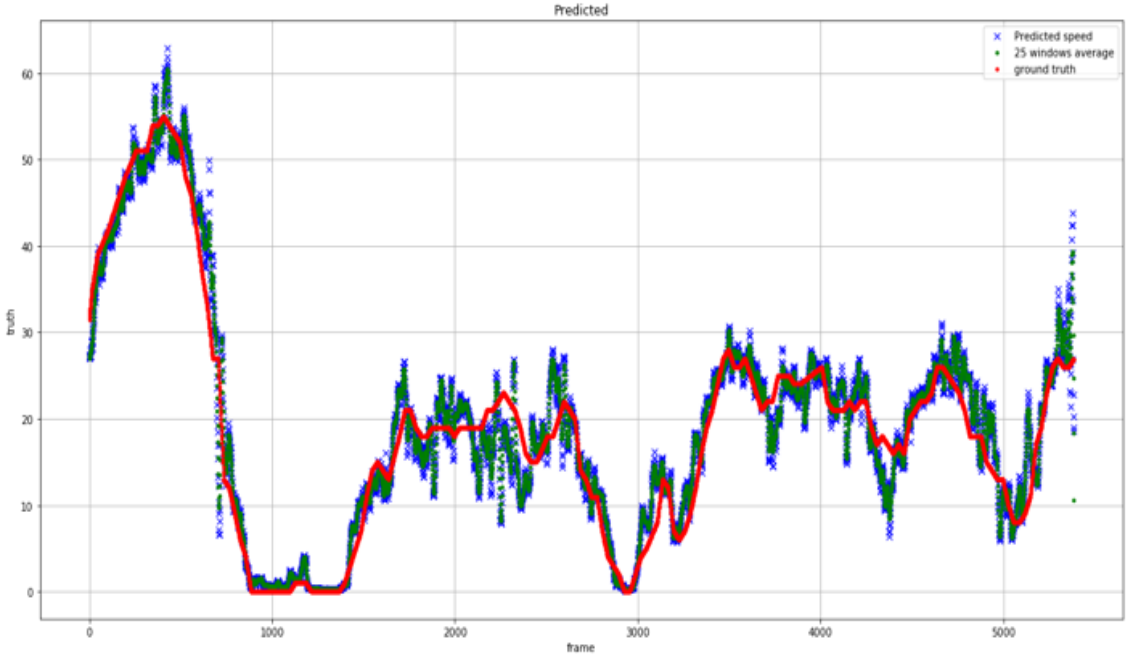
index	speed_mean	speed_max	speed_std	MSE_sqrt	MSE smooth_sqrt	error_rate	error_rate smooth
0	11	41	11	2.738	1.253	0.249	0.205
1	16	44	14	2.745	2.242	0.172	0.140
2	14	29	7	3.215	3.028	0.230	0.216
3	19	55	13	3.228	2.546	0.170	0.134
4	13	45	13	3.685	3.325	0.283	0.256
5	14	32	8	3.804	3.462	0.272	0.247
6	15	52	16	3.891	3.494	0.259	0.233
7	28	81	25	4.320	3.853	0.154	0.138
8	18	40	11	4.418	4.097	0.245	0.228
9	100	108	3	4.810	4.439	0.048	0.044
10	55	74	13	4.853	4.192	0.088	0.076
11	35	75	21	5.069	4.243	0.145	0.121
12	14	33	5	5.161	5.010	0.369	0.358
13	38	55	9	5.395	4.370	0.142	0.115
14	27	66	20	6.014	4.796	0.223	0.178
15	19	35	6	6.483	6.311	0.341	0.332
16	67	89	10	6.762	6.091	0.101	0.091
17	36	57	9	7.914	7.386	0.220	0.205
18	82	93	7	10.325	9.820	0.126	0.120
19	54	97	15	10.524	9.564	0.195	0.177
20	99	108	5	14.135	13.639	0.143	0.138
전체 평균	36.857	108	11.476	5.690	**5.150**	0.199	**0.179**

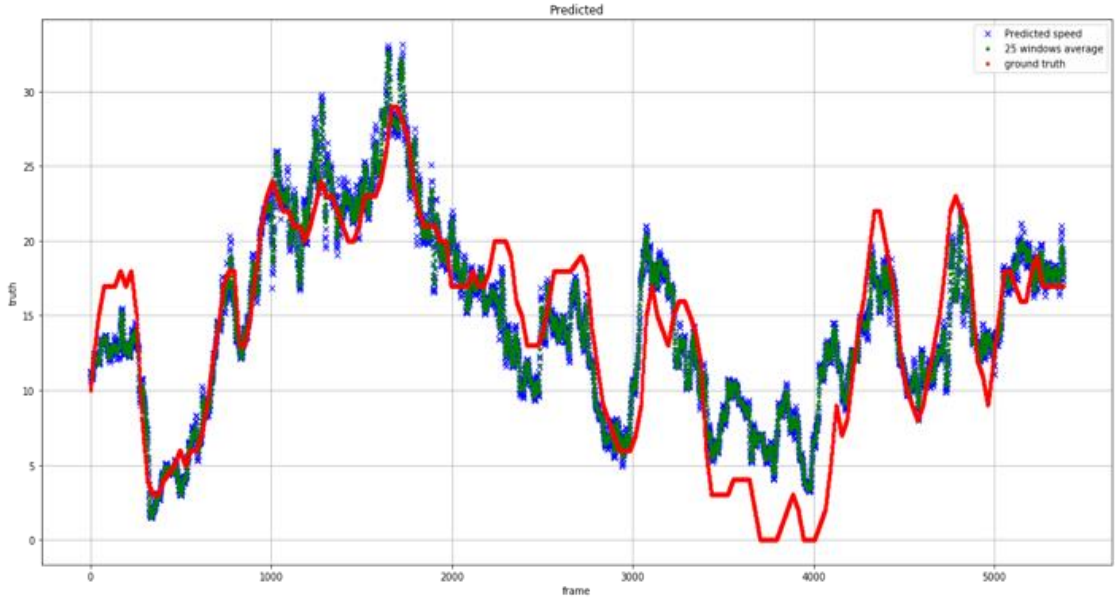
표 4-2 Test 결과표

그림 4-4 검증 결과 Top10 그래프(index0~9)

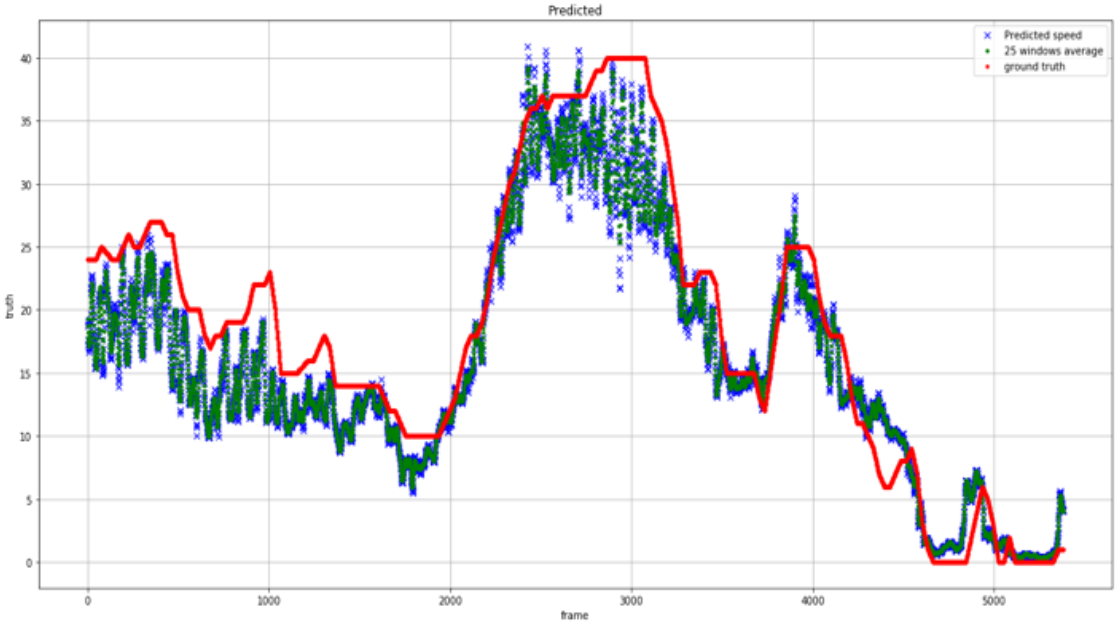




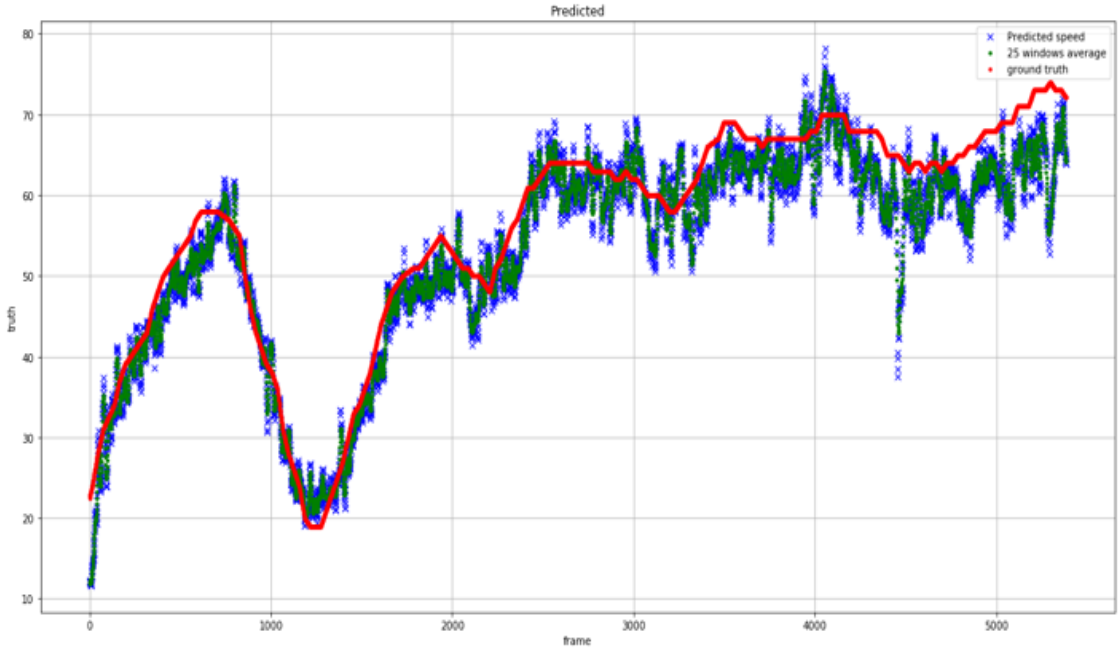

Predicted
Predicted speed
25 windows average
ground truth
truth
frame
0
10
20
30
40
50
60
0
1000
2000
3000
4000
5000

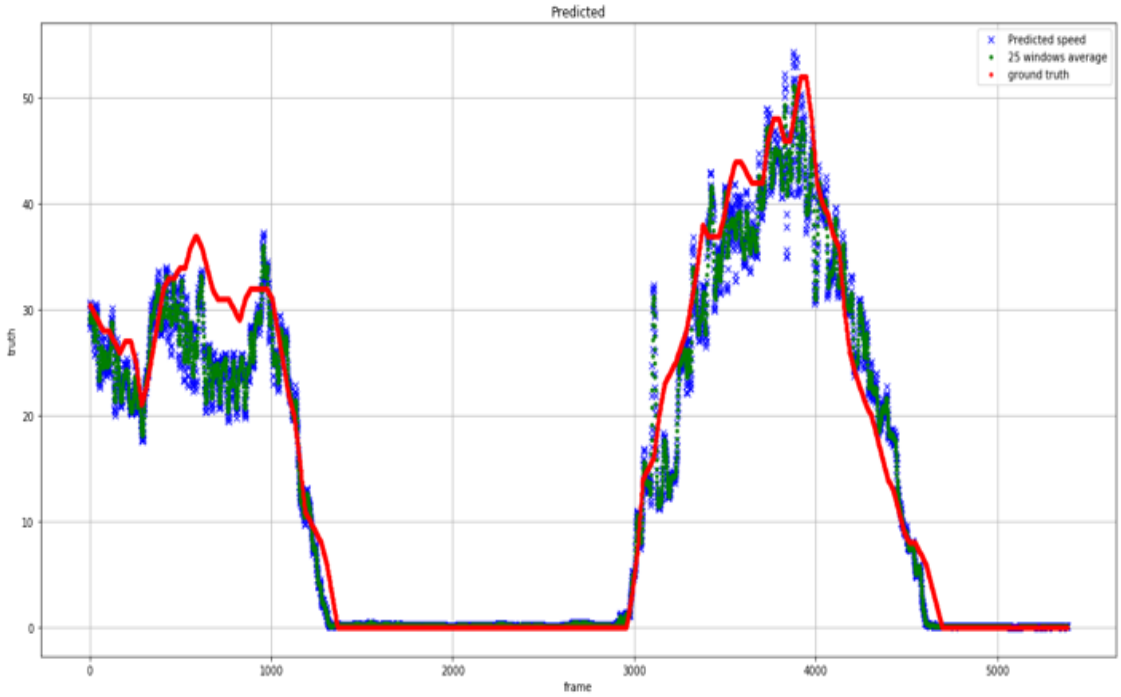

Predicted
Predicted speed
25 windows average
ground truth
truth
frame
0
5
10
15
20
25
30
0
1000
2000
3000
4000
5000


Predicted
Predicted speed
25 windows average
ground truth
truth
frame
0
10
20
30
40
50
0
1000
2000
3000
4000
5000


Predicted
Predicted speed
25 windows average
ground truth
truth
frame
0
5
10
15
20
25
30
35
40
0
1000
2000
3000
4000
5000


Predicted
Predicted speed
25 windows average
ground truth
truth
frame
0
20
40
60
80
100
0
1000
2000
3000
4000
5000


Predicted
Predicted speed
25 windows average
ground truth
truth
frame
10
20
30
40
50
60
70
80
0
1000
2000
3000
4000
5000


Predicted
Predicted speed
25 windows average
ground truth
truth
frame
0
10
20
30
40
50
0
1000
2000
3000
4000
5000

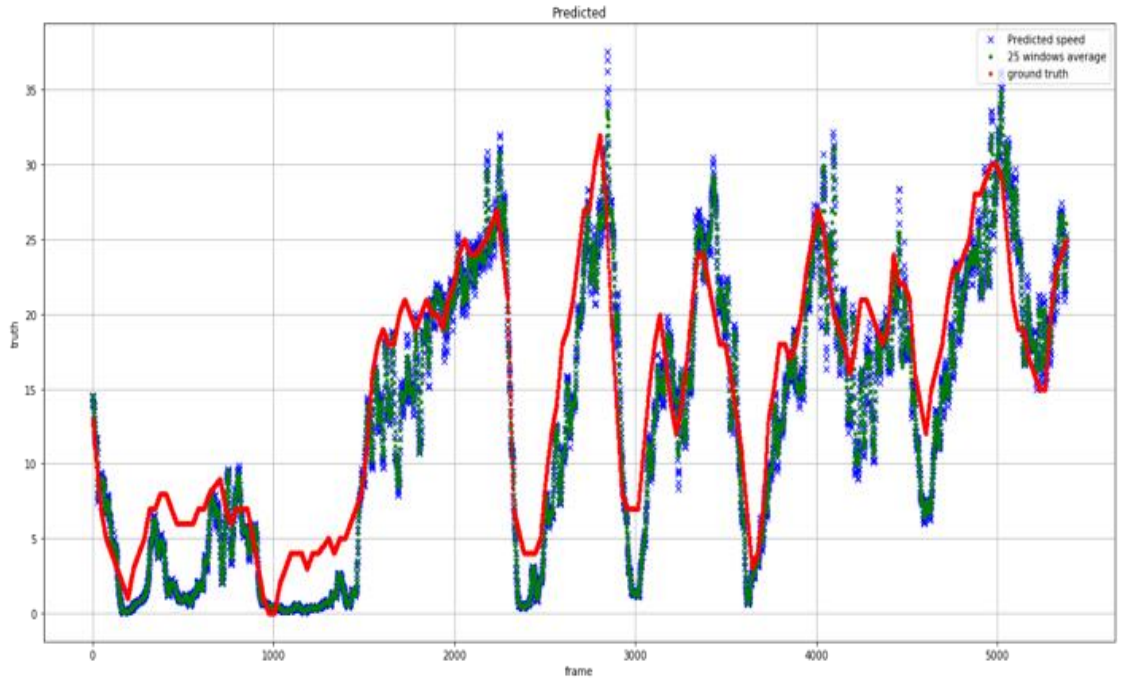
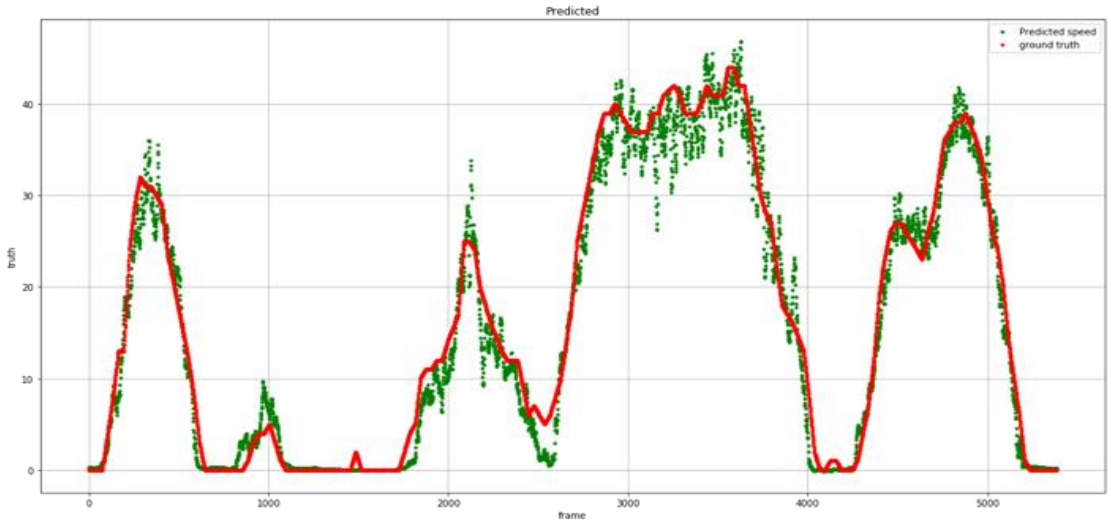
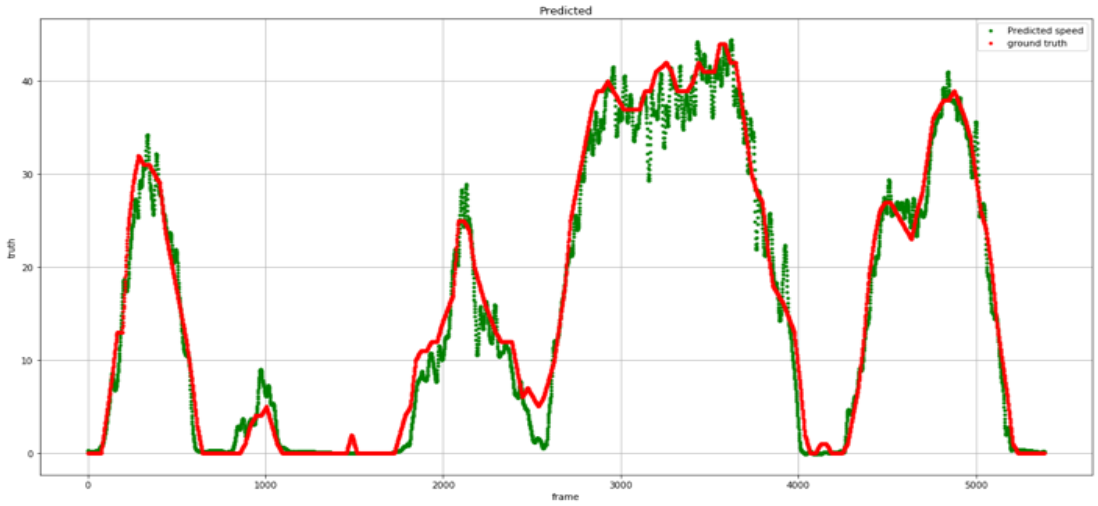

Predicted
Predicted speed
25 windows average
ground truth
truth
frame
0
5
10
15
20
25
30
35
0
1000
2000
3000
4000
5000

그림 4-5 검증 결과 Smoothing 기법 적용

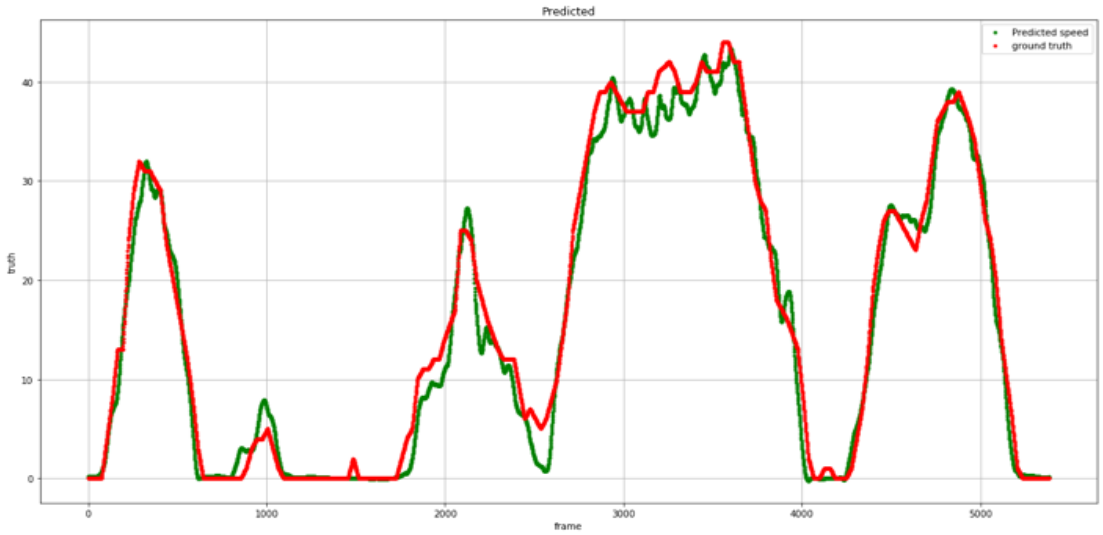
None smoothing(index 0)



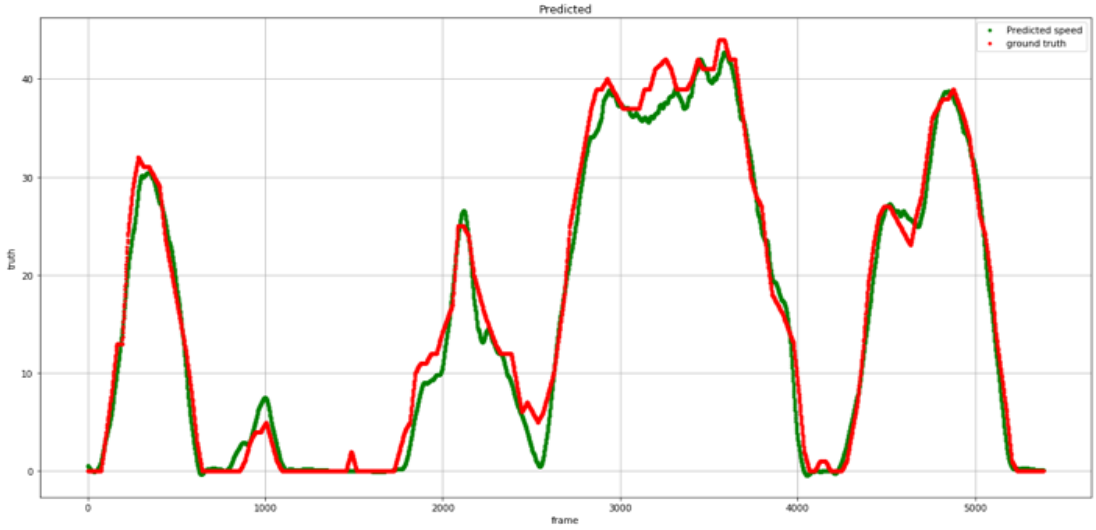
30 frame(=1sec) smoothing



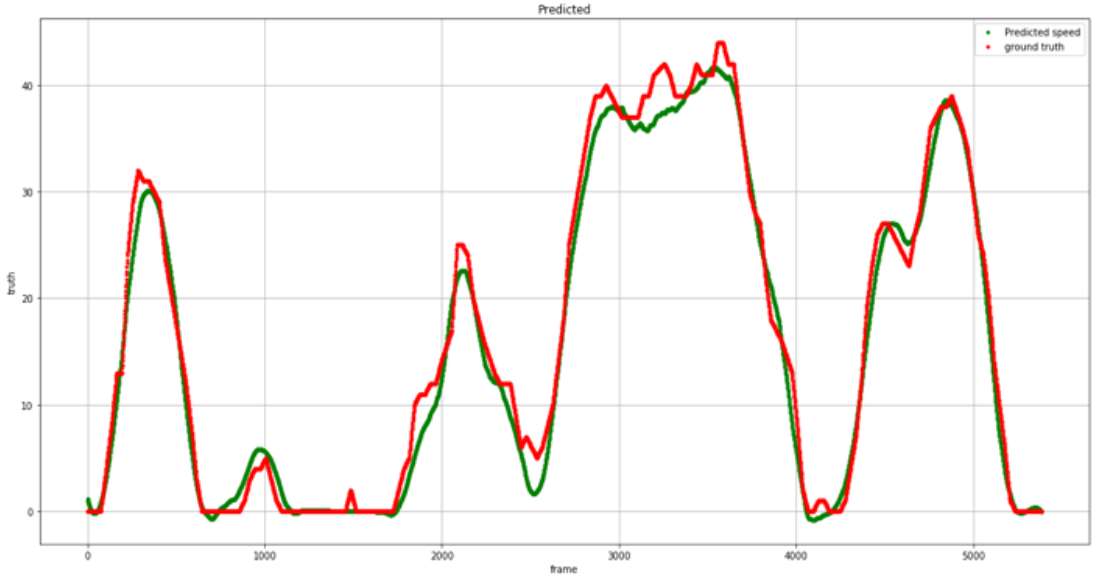
90 frame(=3sec) smoothing



150 frame(=5sec) smoothing



300 frame(=10sec) smoothing



MSE_sqrt: 2.738 -> MSE_sqrt: 1.253 (km/hr)

위의 표와 그림들은 검증결과 표와 그래프이다. 3 분길이의 영상을 프레임 하나하나 예측을 하였고 그 뜻은 180sec * 30frame = 54,000 개의 첫 16 프레임을 뺀 53,984 개 각각의 이미지에 대해 예측한 것과 같다. 그리하여 숫자상이나 표로 봤을때 조금 어긋나는 부분이 있지만 smoothing 기법을 통해 예측률이 조금 더 올라가게 되었다.

4.2 검증 결과 분석

그림 4-6 결과 분석(Worst case vs Best case)

Worst case one

Worst case two

Best case

오차가 가장 큰 비디오와 오차가 가장 적은 검증 비디오를 각 프레임 마다 원래 속도, 예측한 속도 그리고 그 차이를 비디오에 붙인 후 각각 보았다. 의아하게도, 노이즈라 생각되는 순간, 즉 반대편에서 차량이 빨리 오는 경우가 각각의 검증 비디오에서 발견되었음에도 불구, 어떠한 검증데이터는 오차가 적고 어떠한 검증데이터에서는 오차가 컸다. 이 둘의 결정적인 차이는 바로 속도에 의해 일정한 패턴을 보이는 특징, 즉 차선의 종류였다. 오차가 적을 경우, 반대편에서 차량이 오더라도 차선이 차선 변경선, 속도에 의해 지나가는 점선이 존재하였고 그 반대의 경우, 오차가 큰 경우에는 실선, 속도에 의해 모양이 달라지지 않는 선이었다. 이 의미는 모델이 속도를 예측할 때 노이즈가 있더라도 속도를 예측하는데 중요한 특징들, 차선 등 속도에 의해 지나가는 것들이 뚜렷하면 예측을 잘한다는 것이다. 만약 모든 영상에 통일된 길이와 형태의 차선 변경선, 점선이 있었다면 예측 결과는 더욱 뛰어났을 것으로 판단된다. 하지만 도로 종류에 따라 점선의 길이와 점선 사이의 길이는 다르다. 예를 들어 고속도로의 점선 길이는

8m이고 다음 점선까지는 12m이다. 그러므로 한 점선에서 다음 점선까지 거리는 20m가 된다. 만약 어떤 차가 점선 5개를 지나가는 동안 5초가 지났다면 차 속도는 ''20m/s(초)", (72km/hr)가 되는 것이다. 그리고 서울을 비롯한 대부분 지역의 시가지 도로는 점선 길이가 3m, 그 사이 간격이 5m이고 고속도로와는 다르다. 그래서 만약 범위를 나누고 데이터도 이를 기준으로 나누어 훈련하고 검증한다면 더욱 정확한 예측이 될 것이라고 생각된다.

제 5 장 결론

보험사 및 교통사고 분석과(국립과학기술원)에서는 차량 사고가 났을 때 블랙박스를 보고 거리, 시간 등을 어림짐작하여 기본 물리(속력 = 거리/시간)으로 속도를 예측한다. 하지만 이런 어림짐작으로 파악한 거리 및 시간으로는 정확한 차량의 속도를 구하기 어렵다. 그리하여 본 논문에서는 3D Convolutional Neural Network(CNN)중 C3D 모델을 이용하여 블랙박스 속 주행 영상의 차량 속도를 예측하고 시공간적 특징을 파악하였다.

이전에 딥러닝을 이용하여 1인칭 주행 영상만으로 속도를 예측하는 논문은 없었지만 comma ai라는 블랙박스를 만드는 회사에서 speed challenge라는 대회가 있었고 거기에서 주어진 블랙박스 데이터를 이용해 챌랜지에 참가한 사람들이 여러 시도를 하였다. 그 중에서 가장 높은 예측률을 보였던 방법은 Dense Optical Flow를 이용하여 moving vector표현된 이미지를 Alexnet에 넣은 방법이 있었고 Optical pyramid를 이용하여 차선을 검출하여 이를 linear regression으로 예측하는 방법이 있었다. 하지만 이 방법들은 17분의 같은 고속도로를 주행하는 제한된 데이터에서만 좋은 결과를 보였고 실제로 K900, 국내에서 수집한 낮과 밤 그리고 다양한 도로 환경, 25시간 가량의 주행영상에는 예측이 잘되지 않았다. 속도에 의해 어떠한 패턴을 보이는 것을 특징을 잘 잡고 이를 모델이 속도를 예측 하는 데 쓰는게 중요했고 이에 적합한 모델을 찾았어야 하였다.

'Learning Spatiotemporal Features with 3D Convolutional Networks' 라는 논문을

보았고, 이 논문에서 제시한 C3D라는 kernel과 stride가 3x3x3과 1x1x1으로 통일된 3DCNN구조를 알게 되었다. 이 구조는 영상에서 시공간적인 특징을 뽑아내는데 탁월한 성능을 지닌 모델 구조였고 16개의 연속된 프레임(이미지)을 input으로 넣고 예측했을 때 다양한 환경 및 속도의 테스트 데이터에서 평균 RMSE = 5km/hr의 성능을 보였다. 또한 결과 분석에서 마주오는 차량이나 교량 등의 구조물 아래를 지나 갈 때 오류가 크게 발생하지만, 차선변경선이나 속도에 의해 일정한 변화를 보이는 확실한 특징이 있을 때는 이러한 오류에도 영향을 받지 않고 좋은 예측률을 보여주었다. 본 논문이 교통사고 분석 연구 및 자율 주행 연구에 기여가 됬으면 한다.

참고문헌

[1] Bourdev, et al. “Learning Spatiotemporal Features with 3D Convolutional Networks.” *ArXiv.org*, 7 Oct. 2015, arxiv.org/abs/1412.0767.

[2] Liang, Zhang, and Shah Syed Afaq. “*Learning Spatiotemporal Features Using 3DCNN and Convolutional LSTM for Gesture Recogition*” openaccess.thecvf.com/content_ICCV_2017_workshops/papers/w44/Zhang_Learning_Spatiotemporal_Features_ICCV_2017_paper.pdf.

[3] Inhwan, Han. “Car Speed Estimation Based on Cross-Ratio Using Video Data of Car-Mounted Camera (Black Box).” *Forensic Science International*, 30 Nov. 2016, trid.trb.org/view/1436763.

[4] Kensho, Hara, and Satoh Hirokatsu. *Learning Spatio-Temporal Features With 3D Residual Networks* ...openaccess.thecvf.com/content_ICCV_2017_workshops/papers/w44/Hara_Learning_Spatio-Temporal_Features_ICCV_2017_paper.pdf.

[5] Vikram, Mohanty, et al. *DeepVO: A Deep Learning Approach for Monocular Visual Odometry*. arxiv.org/pdf/1611.06069.pdf.

[6] Kun, Liu, et al. *T-C3D: Temporal Convolutional 3D Network for Real-Time* ...haokun-li.github.io/files/publications/T-C3D.pdf.

[7] Jong-Hyuk, Kim, et al. “Reliability Verification of Vehicle Speed Estimate Method in Forensic Videos.” *Forensic Science International*, Elsevier, 9 Apr. 2018, www.sciencedirect.com/science/article/pii/S0379073818301579.

[8] https://github.com/JonathanCMitchell/speedChallenge

[9] https://nicolovaligi.com/car-speed-estimation-windshield-camera.html

[11] 류종익. 영상기록을 통한 차량주행 속도 산출, 응용미약에너지학회지, 2017, 제 15 권, 제 1 호, pp. 13-22 Journal of Applied Subtle Energy, 2017. Vol. 15, No. 1, pp. 13-22

[12] https://en.wikipedia.org/wiki/Computer_stereo_vision

[13] https://en.wikipedia.org/wiki/Hough_transform

[14]https://docs.opencv.org/3.0beta/doc/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html

[15] https://github.com/windowsub0406/SelfDrivingCarND/tree/master/SDC_project_1

ABSTRACT

The Estimation of Car Speed in Black Box using 3DCNN

Ahn Jae Jung

Department of Applied Data Science

Sungkyunkwan University

Car speed is an important indicator for insurance providers after car accidents happen. Following accidents, insurers try to determine car speeds by examining the black boxes of the cars involved in the accidents. However, human eye-based estimates of time and distance are not sufficiently accurate. Most other research papers have attempted to measure car speed by setting distance limits or using several cameras. However, researchers have not yet attempted to measure car speed using only a single windshield camera to analyze spatiotemporal features.

In the last few decades, the computer vision field has evolved dramatically due to the development of artificial neural networks and deep learning techniques. 3D convolutional neural networks have developed to the point that they can extract spatial and temporal characteristics from image sequences. Most studies have focused on identifying or classifying video-captured behavior using 3DCNN. However, this

paper examined temporal and spatial features to predict car speed on video using C3D structure. This approach enabled us to obtain meaningful RMSE (Root Mean Squared Error) results with an average square root deviation of 5.1 km/hr and 82% accuracy from 3 minutes of 21 video test data depicting various speeds and environments.

Keywords : Black Box, Car speed estimation, 3DCNN, C3D, video analysis

碩學位請求論文

3DCNN을 이용한 블랙박스 영상 속 차량 속도 예측

2019

경안재