딥러닝과 Optical Flow를 활용한 보행자 사고 방지 모델*

김영민[O1] 장규진[2] 배현재[3] 김영남[2] 김진평[2]
[1]인천대학교 경제학과
[2]차세대융합기술연구원 컴퓨터비젼 및 인공지능 연구실
[3]성균관대학교 소프트웨어학과
winston1214@naver.com gjjang@snu.ac.kr jason0425@snu.ac.kr hwarangmoon@snu.ac.kr jpkim@snu.ac.kr

Pedestrian Accident Prevention Model Using Deep Learning and Optical Flow

YoungMin Kim[O1] GyuJin Jang[2] HyunJai Bae[3] YoungNam Kim[2] JinPyung Kim[2]
[1]Department of Economics, Incheon National University
[2]Computer Vision and Artificial Intelligence laboratory, Advanced Institute of Convergence Technology
[3]Department of Computer Science and Engineering, Sungkyunkwan University

요 약

자율주행 이동체의 운행에서 보행자 충돌 사고 문제는 꾸준하게 대두되고 있다. 이를 해결하기 위해 CNN(Convolution Neural Network) 및 Optical Flow를 활용하여 보행자를 탐지하고 움직임을 추정하려는 다양한 연구들이 진행되고 있다. 하지만 실시간에 근접하게 보행자를 정확하게 탐지하고 이동 속도를 추정해야 하지만 CNN기반의 딥러닝 알고리즘과 Dense Optical Flow은 실행 속도가 느린 문제로 어려움이 있다. 본 논문에서는 이를 해결하기 위해 처리속도가 빠른 YOLOv5 알고리즘을 활용하여 실시간으로 객체 탐지를 수행하고, 탐지된 객체를 기반으로 기존의 Dense Optical Flow에 기반한 Local Dense Optical Flow을 사용하여 객체의 진행 방향과 속력을 빠르게 추정하는 방법을 제안한다. 이를 바탕으로 보행자와 주행체 간의 충돌 시간과 충돌 지점을 추정할 수 있는 시스템을 제시한다.

1. 서 론

스마트시티(Smart City)에서 트램(tram)은 주요 대중교통수단으로 구축될 예정이며[1] 자율주행에 대한 연구도 활발히 진행되고 있다. 또한, 시장규모도 2035년에 743조원으로 성장할 것으로 전망된다[2]. 특히, 주행 안전성을 보장하기 위해서는 보행자와 충돌을 방지하는 것이 핵심기술이다.

본 연구에서는 이러한 자율주행체의 충돌 사고를 방지하기 위한 시스템으로 카메라의 영상 정보를 이용한 방안을 제안한다. 차량에 설치된 카메라의 주행 영상에서 객체 탐지 딥러닝 알고리즘(YOLOv5)를 기반으로 전방 객체를 탐지한다[3]. 탐지한 객체를 기반으로 Optical Flow[4]을 이용하여 객체가 움직일 방향을 추정할 방법을 제시한다. Optical Flow는 객체에 대한 추적을 위해 일반적으로 쓰이는 방법이고 그 중 Dense Optical Flow은 높은 화질의 영상일수록 객체의 움직임에 대해 높은 정확성을 가진다. 그러나 많은 연산으로 인해 실시간으로 적용하기 어렵고 주행 영상(Dynamic Scence)에서는 배경(Background)이 움직이기 때문에 적용하는데 어려움이 있다.

본 연구에선 이러한 한계점을 극복하기 위해 객체가 탐지된 영역 내의 Optical Flow를 이용하는 방법인 LDOF(Local Dense Optical Flow)을 제시한다. 이러한 방법을 활용하여 객체 탐지를 수행하고 객체의 움직임을 예상하여 객체와 주행체 간의 충돌 예상 시간(TTC : Time To Collision)을 추정한다. 또한, 이를 바탕으로 주행체와 보행자가 충돌할 지점을 추정하고 충돌이 예상되는 경우 경고 메시지를 통해 사전에 객체와의 충돌을 방지한다. 전체적인 알고리즘 흐름도는 <그림 1>과 같다.

2-1. LDOF(Local Dense Optical Flow)

본 논문에선 Farnebäck 알고리즘[5]을 기반으로 Optical Flow를 계산한다. 특징점을 추출하여 계산하는 LK(Lucas-Kanade) Optical Flow를 대신하여 모든 이미지 픽셀에 대해 계산하여 객체의 움직임을 정확하게 추정하기 위해 Farnebäck 알고리즘을 사용한다.

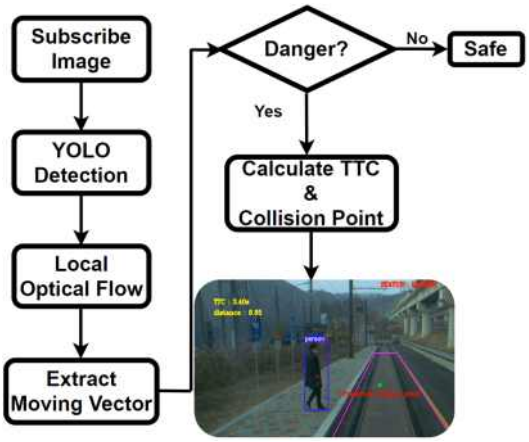

그림 1 Model Flow Chart

Optical Flow를 이용하여 이동 벡터를 추출하는 과정은 <그림 2>와 같다. YOLOv5 알고리즘을 이용하여 사람을 탐지한 후 Farnebäck 알고리즘으로 Optical Flow를 계산한다. 이를 통해 움직이는 물체에 대해 Optical Flow를 이용하면 해당 물체의 이동 벡터로 다음 움직임을 예상할 수 있다.

$$\vec{v_0} = \frac{1}{N_b}\sum_{A_b} I(x,y) | (x,y) \in A_b \quad where\ I(x,y) = (u,v) \quad (1)$$

* 본 연구는 국토교통부/국토교통과학기술진흥원의 지원으로 수행되었음 (과제번호 20LTSM-B156015-01)

식(1)에서 A_b는 물체가 탐지된 영역, N_b는 물체가 탐지된 영역 내 픽셀의 총 개수, I(x,y)는 Optical Flow로 계산된 흐름 벡터를 의미하며 (u,v)는 흐름 벡터의 수평 성분의 크기(Horizontal)와 수직 성분의 크기(Vertical)를 나타낸다.

$$\vec{v_c} = \frac{1}{N_s}\sum_{A_s} I(x,y) \mid (x,y) \in A_s \quad (2)$$

$$\vec{v} = \vec{v_o} - \vec{v_c} \quad (3)$$

A_s는 물체가 탐지된 주변 영역을 의미하고, N_s는 탐지된 주변 영역 내 픽셀 총 개수를 의미한다. 객체의 프레임 간 원시 이

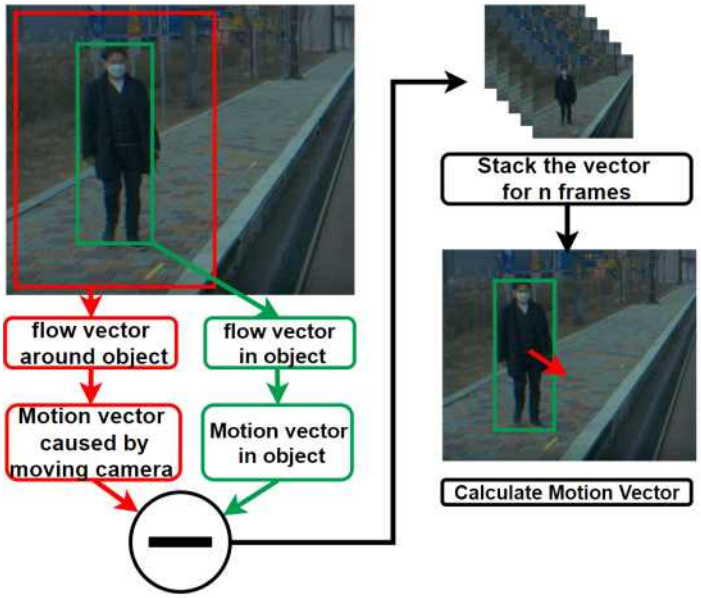

그림 2 Flow Chart of Acquiring Motion Vector by Optical Flow

동 벡터(Primitive Motion Vector per Frame) v_0는 탐지된 영역 내의 모든 흐름 벡터를 평균한 것을 의미하고 동적인 카메라의 흐름 벡터를 v_c로 정의한다. 객체의 프레임 간 원시 이동 벡터에서 카메라에 의해 생긴 객체 이동 벡터를 감산함으로써 동적인 영상에 의한 영향이 제거된 객체의 프레임 간 이동 벡터를 구한다. 계산한 이동 벡터의 민감도를 안정화하기 위해 프레임을 누적한다[6]. 이러한 방법을 통해 이동 벡터를 추출한다.

Dense Optical Flow는 픽셀 수가 많을수록 연산량이 증가하여 처리 속도가 느려진다. 이러한 문제점을 해결하기 위해 탐지된 객체의 영역 주변을 계산영역으로 이용한다[7]. 따라서 이를 LDOF라고 정의한다.

2-2. 충돌 예상 시간 및 충돌 지점 추정

Optical Flow를 이용하여 계산된 객체의 이동 벡터로 객체의 속도를 계산한다. 이를 통해 주행체와 객체 간의 거리, 상대 속도의 크기를 구한 뒤 충돌 시간을 계산하고 속도의 방향을 이용하여 충돌 예상 지점을 예측한다.

충돌 예상 시간은 뉴턴역학 속력-시간 식 (4)를 이용한다. t_c는 충돌 예상 시간, s_0는 카메라와 물체 간의 거리, v_r은 카메라와 물체 사이의 상대 속도를 의미한다.

$$t_c = \frac{s_0}{v_r} \quad (4)$$

이미지에서 물체와의 거리는 식 (5)를 이용한다.

$$s_o = \frac{h_o \times f}{c_s} \times \frac{I_h}{I_o} \quad (5)$$

h_o는 실제 객체의 높이, f는 초점 거리, c_s는 카메라의 CCD 센서의 높이, I_h는 이미지의 총 세로 픽셀 수, I_o는 이미지 내에서의 객체의 세로 픽셀 수를 의미한다.

위 식을 이용하여 객체의 속도의 크기와 방향을 알 수 있으며, 이를 통해 주행체와 충돌 가능성을 예측할 수 있다.

3-1. 실험 환경

주행체 중 하나인 트램의 운행 영상으로 실험을 진행하였다. 트램의 주행 영상은 오송 트램 시험선에서 촬영하였다. 운영체제는 Ubuntu 18.04.5 LTS, GPU는 NVIDIA RTX 2070 super, CPU는 Intel Core i7-9700K @CPU 3.60GHz 환경에서 진행되었다.

객체 인식을 위한 가중치 모델은 COCO 데이터셋으로 사전 학습된 가중치 중 가장 처리속도가 빠른 yolov5s.pt를 이용하였다.

Optical Flow는 식(1)에서 계산된 탐지영역 A_b를 중심으로 탐지된 영역의 2배를 계산하였다. 또한, 식(2)에서 정의한 주변 영역 As는 탐지된 영역을 중심으로 1.5배의 영역에서 탐지 영역을 제외한 부분으로 계산하였다.

객체의 이동 벡터는 5프레임 간 이동 벡터를 더하여서 계산하였고, 주행체 간의 거리를 계산하기 위한 식 (5)에서 각각 h_0는 1.75m, c_s는 6.6mm. f는 8mm로 설정하였다.

3-2. 실험 결과 및 평가

본 실험은 트램의 정지선과 객체와의 거리를 10m에서 30m까지 10m 간격으로 설정하고 객체가 진입하는 각도를 주행체의 진행 방향과 수직인 방향에 대해 -45°, 0°, 45° 로 설정하였다.

객체가 존재하는 프레임의 개수에서 객체가 존재하지만 객체를 탐지하지 못한 프레임의 개수를 뺀 값을 백분위한 값을 객체 탐지율로 정의하고 이를 평가지표로 설정한다. 거리와 진입 각도에 따른 객체 탐지율은 <표 1>과 같다.

표 1. Object Detection Ratio

Distance of the Object from stop line	Direction of the object(°)	Detection ratio
10m	-45	100%
	0	100%
	45	100%
20m	-45	100%
	0	99.63%
	45	100%
30m	-45	100%
	0	91.2%
	45	90.85%

객체 탐지에서 모두 90% 이상의 탐지율이 도출되었다. 이 결과에 기반하여 실제 객체의 이동 방향과 계산된 이동 벡터의 방향이 일치하는지 확인한다. 또한, 사람의 걷는 속도는 평균 1.38m/s임으로 이를 기준으로 객체의 속도를 잘 추정했는지 판단한다. 보행자의 속도는 식 (6)으로 계산하여 추정한다.

$$v_o = \frac{s_o}{f} \times C_{sw} \frac{I_{ow}}{I_w} \times fps \quad (6)$$

$v_o = Real\ Speed\ of\ Object(m/s)$
$s_o = Distance\ between\ Tram\ and\ Object$
$(5m, 10m, 15m, 20m, 25m, 30m)$
$f = Focal\ \leq ngth(8mm)$
$C_{sw} = CCD\ Sensor\ \ Width(8.8mm)$
$I_w = \fallingdotseq\ Width(2040px)$
$I_{ow} = Object\ \ Width(px)$
$fps = 30(s^{-1})$

객체 이동 벡터의 각도 및 크기 추출 결과는 <표 2>와 같다.

표 2. Flow Result

Distance of the Object from stop line	Direction of the object (°)	Average Angle of Motion Vector in Videos(°)	Average Motion Vector Magnitude of Object in Videos(px)	Estimated Speed (m/s)
10m	-45	117.99	1.06	0.69
	0	28.76	0.77	0.50
	45	85.52	0.49	0.32
20m	-45	-2.22	1.06	1.37
	0	1.45	0.94	1.22
	45	25.27	0.49	0.63
30m	-45	-0.15	0.53	1.03
	0	0.80	0.48	0.93
	45	21.21	0.21	0.41

충돌 예상 판단 기준은 트램이 레일 안으로 진입하는 방향이고 충돌 예상 시간이 5초 이내인 경우 경고 알람을 띄우도록 하였다. <그림 3>은 보행자가 20m에서 0도 방향으로 진입하는 경우에 경고 알람이 띄워지는 경우를 나타낸다.

그림 3 Result of Warning Alarm

<표 3>은 본 논문의 제안 방법과 기존의 방식과 비교하였을 때 알고리즘 계산 속도 결과이다. 각 알고리즘의 프레임 당 소요 시간을 ms(Millisecond)로 나타낸다.

표 3. Average Runtime

(ms)

Method	Min	Max	Avg
Farnebäck	526	571	536
SimpleFlow[8]	2902	3104	2973
Previous Method(2016)[9]	380	600	490
Our Method(LDOF)	9	66	20

Farnebäck 알고리즘은 프레임 당 평균 536ms, SimpleFlow는 2973ms, LK 알고리즘을 바탕으로 한 기존의 연구는 평균 490ms가 소요된다. 반면에 본 연구의 제안 방법은 소요 시간이 평균 20ms으로 기존의 연구보다 소요 시간을 크게 단축하였다.

4. 결론 및 향후 연구

제안한 방법을 통해 FHD 영상에 대해 실시간처리를 달성할 수 있었으며 이 방법을 통해 보행자 간의 충돌 가능성을 판단하고 사고를 사전에 방지할 수 있다. 또한, 기존에 제안되었던 방법보다 약 10배 정도 빠른 것을 확인하였다.

하지만 가까운 객체에 대해 광학 흐름을 계산하는 경우 움직이는 장면으로 인해 생긴 흐름 벡터를 완전히 제거하지 못하여 객체의 이동 벡터에 영향을 주게 되었다. 또한, 하나의 카메라를 이용하여 깊이(Depth) 성분을 정확하게 추정할 수가 없어 객체의 이동 벡터 상하성분과 객체의 실제 상하성분의 차이가 존재하였다.

이러한 한계점을 극복하기 위해 딥러닝을 활용한 Optical Flow 방식을 적용할 것이고, 실시간성을 위해 비교적 가벼운 딥러닝 모델을 활용하여 연산 속도를 줄이고 더 정확한 연산이 될 것이다.

참고 문헌

[1] 김도년, et al. “스마트시티 마스터플랜 계획기법에 관한 연구: 바이칼스마트시티 마스터플랜의 공간계획을 중심으로.“ 한국도시설계학회지 도시설계 16.5 (2015): 109-122.
[2] Lee, Byeong-Yun. “국내외 자율주행자동차 기술개발 동향과 전망.“ Information and Communications Magazine 33.4 (2016): 10-16.
[3] https://github.com/ultralytics/yolov5
[4] Horn, Berthold KP, and Brian G. Schunck. “Determining optical flow.“ Artificial intelligence 17.1-3 (1981): 185-203.
[5] Farnebäck, Gunnar. “Two-frame motion estimation based on polynomial expansion.“ Scandinavian conference on Image analysis. Springer, Berlin, Heidelberg, 2003.
[6] Mao, Jue, et al. “Complementary Motion Vector for Motion Prediction in Video Coding with Long-Term Reference.“ 2019 Picture Coding Symposium (PCS). IEEE, 2019.
[7] Muhlhausen, Moritz, et al. “Iterative optical flow refinement for high resolution images.“ 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019.
[8] Tao, Michael, et al. “SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm.“ Computer graphics forum. Vol. 31. No. 2pt1. Oxford, UK: Blackwell Publishing Ltd, 2012.
[9] KUO, Ying-Che; TSAI, Cheng-Tao; CHANG, Chih-Hao. Fast Estimation of Pedestrian Movement. Sensors and Materials, 2017, 29.6: 713-726