KIPS Trans. Softw. and Data Eng.
Vol.10, No.12 pp.561~568
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2021.10.12.561

A Study of Tram-Pedestrian Collision Prediction Method Using YOLOv5 and Motion Vector

Young-Min Kim† · Hyeon-Uk An†† · Hee-gyun Jeon††† · Jin-Pyeong Kim†††† · Gyu-Jin Jang††††† · Hyeon-Chyeol Hwang†††††

ABSTRACT

In recent years, autonomous driving technologies have become a high-value-added technology that attracts attention in the fields of science and industry. For smooth Self-driving, it is necessary to accurately detect an object and estimate its movement speed in real time. CNN-based deep learning algorithms and conventional dense optical flows have a large consumption time, making it difficult to detect objects and estimate its movement speed in real time. In this paper, using a single camera image, fast object detection was performed using the YOLOv5 algorithm, a deep learning algorithm, and fast estimation of the speed of the object was performed by using a local dense optical flow modified from the existing dense optical flow based on the detected object. Based on this algorithm, we present a system that can predict the collision time and probability, and through this system, we intend to contribute to prevent tram accidents.

Keywords : Tram, Dense Optical Flow, Estimation of Collision point, TTC(Time-To-Collision), YOLOv5

YOLOv5와 모션벡터를 활용한 트램-보행자 충돌 예측 방법 연구

김 영 민† · 안 현 욱†† · 전 희 균††† · 김 진 평†††† · 장 규 진††††† · 황 현 철†††††

요 약

최근 자율주행에 관한 기술은 고부가가치 신기술로서 주목받고 있으며 활발히 연구가 진행되고 있는 분야이다. 상용화 가능한 자율주행을 위해서는 실시간으로 정확하게 진입하는 객체를 탐지하고 이동속도를 추정해야 한다. CNN(Convolutional Neural Network) 기반 딥러닝 알고리즘과 밀집광학흐름(Dense Optical Flow)을 사용하는 기존 방식은 실행 속도가 느려 실시간으로 객체를 탐지하고 이동속도를 추정하기에는 한계가 존재한다. 본 논문에서는 트램에 설치된 카메라를 통해 획득된 주행영상에서 딥러닝 알고리즘인 YOLOv5 알고리즘을 활용하여 실시간으로 객체를 탐지를 수행하고, 탐지된 객체영역에서 기존의 밀집광학흐름(Dense Optical Flow) 대신 연산량을 개선한 부분 밀집광학흐름(Local Dense Optical Flow)을 사용하여 객체의 진행 방향과 속력을 빠르게 추정하는 방식을 제안한다. 이를 바탕으로 충돌 시간과 충돌 지점을 예측할 수 있는 모델을 설계하였으며, 이를 통해 트램(Tram)의 주행 중 전방 충돌사고를 방지할 수 있는 시스템에 적용하고자 한다.

키워드 : 트램, Dense Optical Flow, 충돌지점 추정, 충돌시간 추정, YOLOv5

1. 서 론

트램(Tram)은 일반 도로 위 레일 상에서 운행되며, 유럽에서 흔히 볼 수 있는 대중교통이다. 하지만 일반 도로에서 운행되는 특징으로 인해 트램과 보행자와 간의 사고가 빈번하게 발생한다[1]. 최근 2020년 이탈리아 밀라노에서 트램 사고로 인한 보행자의 사망 사례가 있다. 현재 라이다(LIDAR) 센서와 카메라를 이용한 자율주행 연구가 활발히 진행되고 있다[2]. 트램은 일반 자율주행 차량과는 다르게 대중교통이므로 급정거는 권장되지 않으며, 도로 위 레일 위에서 운행하므로 일반 차량과는 다르게 사고를 피하기 위한 차선 변경이 불가능하다. 즉, 먼 거리에 있는 사람 객체에 대해 탐지가 필요하고, 일반적인 자율주행 차량보다 충돌 예상 시간과 지점을 더 이른 시간에 계산할 필요가 있다. 일반적으로 이용하는 라이다 센서는 정확성이 높으나 탐지 가능 거리 및 해상도가 높아질수록 고비용이 요구된다. 이에 비해

※ 이 논문은 한국철도기술연구원 자율주행 트램 기술고도화 및 시험운행 과제의 지원을 받아 수행된 연구임(PK2103C5).
※ 이 논문은 행정안전부 극한재난대응기반기술개발사업의 지원을 받아 수행된 연구임(2020-MOIS31-014).
† 비 회 원 : 인천대학교 경제학과/컴퓨터공학부 학사과정
†† 비 회 원 : 고려대학교 전기전자공학부 학사과정
††† 비 회 원 : 한국철도기술연구원 스마트트램연구실 선임연구원
†††† 정 회 원 : 차세대융합기술연구원 선임연구원
††††† 정 회 원 : 차세대융합기술연구원 컴퓨터비전 및 인공지능연구실 연구원
†††††† 비 회 원 : 한국철도기술연구원 스마트트램연구실 책임연구원
Manuscript Received : April 7, 2021
Accepted : July 11, 2021
* Corresponding Author : Hyeon-Chyeol Hwang(hchwang@krri.re.kr)

카메라는 경제적이며 사물인식이 가능한 측면이 있다.

광학 흐름(Optical Flow)은 영상에서 객체에 대한 추적(Object Tracking)을 위해 일반적으로 쓰이는 방법이다. 밀집광학흐름 방식은 높은 화질인 영상일수록 객체의 움직임에 대해 높은 정확성을 나타내지만 많은 연산으로 인해 실시간(Real-Time)으로 적용하기 어렵다. 또한, 광학 흐름은 정지된 카메라의 영상(Static Scene)을 가정하였기 때문에 움직이는 카메라에 대해 이 방법을 적용하기 어렵다.

본 연구에서는 라이다 센서 대신 높은 화질의 카메라를 이용한 트램주행 영상에서 딥러닝(YOLOv5)알고리즘 기반으로 전방객체를 탐지한다. 차량에 설치된 카메라의 주행영상(Dynamic Scene)에서 객체가 탐지된 영역 내에 밀집광학흐름을 활용하여(Local Dense Optical Flow) 객체가 움직일 방향을 빠르고 정확하게 예측할 방법을 제시한다.

위 방법을 활용하여 객체를 탐지하여 객체의 움직임을 예상하고, 객체와 주행체 간의 충돌 예상 시간(TTC:Time-To-Collision) 및 충돌 지점을 추정한다. 위 정보를 이용하여 트램과 충돌이 예상되는 경우 경고 메시지를 통해 사전에 충돌을 방지한다.

2장에서는 본 논문에서 사용하고자 하는 딥러닝 기반의 객체 탐지 알고리즘과 모션벡터를 추출하기 위한 기존의 광학 흐름 기술을 살펴본다. 3장에서는 기존의 광학 흐름에서 계산 속도가 향상된 부분 밀집광학흐름 알고리즘을 제안하고 이를 이용하여 객체와 주행체 간의 충돌 시간 및 지점을 추정한다. 4장에서는 객체 탐지의 거리별 정확도의 측정과 제안한 알고리즘의 처리 속도 및 정확성을 측정한다. 마지막으로 5장에선 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

자동차의 자율주행은 최근 딥러닝 모델을 활용한 많은 연구가 진행되고 있다. [3]에서는 하나의 이미지를 이용하여 SSD-MR(Single-Shot MultiBox Detector with Motion Representation) 이라는 딥러닝 모델을 통해 보행자를 탐지하고 보행자와의 충돌에 대한 위험도(Risk-Level)을 판단한다. [4]에서는 Far-Shot 영상을 활용하여 PID(Prediction in Intelligent Driving)라는 딥러닝 모델을 이용하여 보행자의 다음 이동을 예측한다.

2.1 YOLOv5

YOLO(You Only Look Once) 알고리즘[5]은 객체 탐지를 위한 딥러닝 알고리즘 중 하나이다. 특히 YOLO 알고리즘은 높은 FPS(Frames per second) 성능을 내기 때문에 실시간성에 적합한 알고리즘이다. 그 중 가장 최근에 제안된 YOLOv5의 아키텍처(Architecture)는 2021년 2월 기준으로 Fig. 1과 같이 구성된다. 입력 이미지에서 중요한 특징을 추출하는 역할을 하는 모델 백본(Backbone)은 BottleNeck[6]과 CSPNet[7]을 결합하고 활성화 함수로 SiLU[8]를 사용한 C3 모델을 사용하면서 학습 능력을 향상시켰다.

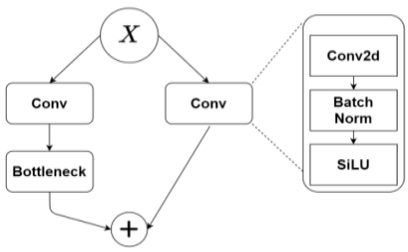


Fig. 1. C3 Architecture

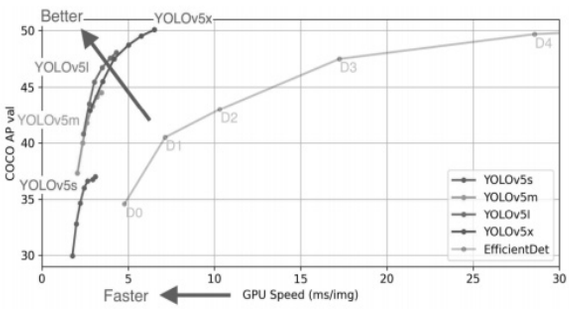


Fig. 2. Performance of YOLOv5 Algorithm[12]

백본에 형성된 기능을 혼합하는 모델의 넥(Neck)은 PA-Net[9] 피쳐 피라미드 방식을 사용한다. 객체 탐지를 수행하는 모델의 헤드(Model Head)는 기존의 YOLOv3[10] 모델의 구조를 따른다.

COCO 데이터셋[11]을 이용하여 AP(Average Precision)와 FPS를 계산하였을 때 s 모델 기준 AP^{test} 50.1, FPS_{V100} 455의 성능이 구현된다. 결과는 Fig. 2와 같다.

2.2 광학 흐름(Optical flow)

광학 흐름[13]은 관찰자와 장면 사이의 상대적 움직임으로 해당 장면에서 물체, 표면, 가장자리의 움직임 패턴을 계산하는 것이다. 최근 연구에서도 전경 탐지(Foreground Segmentation)를 위해 광학 흐름을 이용하였다[14]. 광학 흐름은 다음 식을 이용하여 계산한다.

$$I(x,y,t) = I(x+dx, y+dy, t+dt) \tag{1}$$

등식의 오른쪽에 테일러 근사(Taylor-Approximation)를 이용하면

$$\frac{dI}{dx}\delta x + \frac{dI}{dy}\delta y + \frac{dI}{dt}\delta t = 0 \tag{2}$$

이 식을 δt로 나누면 광학 흐름 방정식(Optical Flow Equation)이 나오게 된다.

$$\frac{dI}{dx}u + \frac{dI}{dy}v + \frac{dI}{dt} = 0 \tag{3}$$

u, v는 각각 x 방향과 y 방향에 대한 광학 흐름 값이며, I는 이미지의 픽셀값을 의미한다.

Equation (3) 광학 흐름 방정식은 하나의 픽셀을 이용하

Fig. 3A. Lucas-Kanade

Fig. 3B. Gunner Farnebäck

여 계산할 수 없는 부정 방정식 형태이다. Equation (3)을 풀기 위해서 주변 픽셀의 정보를 이용하여 광학 흐름을 계산한다. 광학 흐름을 계산하는 방법은 크게 두 가지이다. 한 가지 방법은 일부의 픽셀(Sparse Set)을 이용한 부분 광학 흐름(Sparse Optical Flow) 방식인 LK 알고리즘(Lucas-Kanade Algorithm)이며, 다른 하나는 모든 픽셀(Universe Set)을 이용한 밀집광학흐름(Dense Optical Flow) 방식인 Farnebäck 알고리즘(Gunner Farnebäck's Algorithm)이 있다.

1) LK(Lucas Kanade) 알고리즘

LK 알고리즘[15]은 특징점(Keypoint)을 이용해 해당 특징점의 주변 픽셀 정보를 이용하여 광학 흐름을 계산하는 방법이다. Fig. 3A는 LK 알고리즘을 적용한 예시이다. 특징점의 주변 픽셀에 대해 각각 계산하므로 빠른 계산이 장점이지만, 객체의 움직임을 올바르게 예측하기 위해 특징점 선정이 중요하며, 특정 영역만을 이용하여 계산하기 때문에 실제 객체의 움직임을 잘못 예측할 위험이 있다.

2) Farnebäck 알고리즘

Farnebäck 알고리즘[16]은 전체 이미지를 이용하여 각 픽셀에 대해 주변 픽셀 정보를 활용하여 이차 다항식 근사(Polynomial-Approximation)로 광학 흐름을 계산하는 방법이다. Fig. 3B는 Farnebäck 알고리즘을 그리드 형태로 적용한 예시이다. 이미지의 모든 영역을 이용하여 계산하므로 물체의 움직임 예측에 대해 정확도가 높지만, 계산되는 이미지의 픽셀 수가 많을수록 계산량이 많아지는 단점이 있다.

이외에도 최근 딥러닝을 활용한 광학 흐름을 학습하고 계산하는 FlowNet과 같은 알고리즘이 많이 제안되고 있다[17,18]. 딥러닝을 활용한 알고리즘은 정확성이 높지만, GT 데이터(Ground Truth)을 활용한 학습이 필요하며, 학습된 정보에 대하여 의존성이 높다.

2.3 동적환경(Dynamic scene)에서 모션 추정

광학 흐름은 폐쇄 회로 카메라(CCTV)와 같은 정적인 카메라를 가정하여 제안한 방법이다. 하지만 자율주행 차량과 같이 카메라가 움직이게 되어 문제가 있다. 움직이는 카메라에 대해 객체에 대한 광학 흐름을 계산하기 위해서 많은 방법이 제안되었다. 크게 두 가지 방법이 제안됐으며, 한 가지는 배경을 제거하여 객체를 탐지 및 추적하거나, 다른 하나는 사영변환(Homography Transformation)을 이용하여 프레임 간 정렬(Stabilize)을 통해 위의 알고리즘을 적용한다. 배경을 제거하는 방법으로는 [19]에서 차선의 상대 이동 벡터(Motion Vector)를 이용하여 움직이는 카메라 1대를 이용하여 카메라의 이동(Egomotion)을 측정하여 이를 통해 영상에서 주행체로 인해 생기는 객체의 이동 벡터를 제거하는 방법을 제시한다.

Table 1. Evaluation of Flow (KITTI 2012 Evaluation, Flow)

Method	Out-All	Avg-All	Runtime	Environment
PolyExpand	54.00 %	25.3 px	1 s	1 core @ 2.5 Ghz (C/C++)
Pyramid-LK	40.08%	29.6 px	1.3 s	4 cores @ 3.5 Ghz (C/C++)
FlownetS + ft	44.49 %	9.1 px	0.08 s	GPU @ 1.0 Ghz (C/C++)

[20]에서 특징점 픽셀들(Superpixels)을 이용하여 배경을 제거한 후 움직이는 물체에 대해 세그멘테이션(Segmentation)을 진행한다.

사영변환을 이용한 방법으로는 [21]에서 이전 프레임과 현재 프레임 간의 사영변환을 통해 주행체에 의한 이동 벡터의 영향을 제거한 후(Stabilized image) 밀집광학흐름을 이용하여 움직이는 객체를 탐지한다.

3. 제안 방법

본 논문에서는 빠르게 객체를 탐지할 수 있는 알고리즘과 탐지된 객체의 영역에서 광학 흐름을 계산하고, 이를 통해 객체의 이동정보를 추정한다. 위와 같은 단계를 적용하여 빠른 객체 탐지 및 객체의 움직임 정보를 통해 트램과의 예상충돌시간 및 지점을 추정하는 방법을 제안한다.

제안하는 방법은 먼저 딥러닝 기반 YOLOv5 알고리즘을 이용하여 사람과 차 등의 객체를 학습한 뒤 탐지한다. 탐지된 객체의 주변영역에 밀집광학흐름 알고리즘인 Farnebäck 알고리즘을 적용하여 객체의 이동 벡터를 계산하여 객체를 추적하고 이동을 예측한다. 그리고 객체의 이동 벡터를 이용하여 객체와 주행체 간의 속도를 계산하고 충돌 예상 지점 모델에 활용하여 충돌 예상 시간 및 지점을 추정한다.

3.1 Local Dense Optical Flow

본 연구에서는 Farnebäck 알고리즘 기반의 광학흐름을 계산한다. 카메라의 이동으로 인해 영상이 동적으로 변하기 때문에 특징점의 소실 위험이 있는 LK 알고리즘 대신 모든 이미지 픽셀에 대해 계산하여 객체의 움직임을 비교적 정확하게 알 수 있는 Farnebäck 알고리즘을 활용하였다.

광학 흐름을 이용하여 이동 벡터를 얻는 과정은 Fig. 4와 같다. YOLOv5 알고리즘을 이용하여 사람을 탐지한 후 Farnebäck 알고리즘을 이용하면 전체 영상의 광학 흐름을 알 수 있으며, 이를 통하여 움직이는 물체에 대해 광학 흐름을 이용하면 해당 물체의 이동 벡터로 다음 움직임을 예측할 수 있다.

$$I(x,y) = (u,v) \tag{4}$$

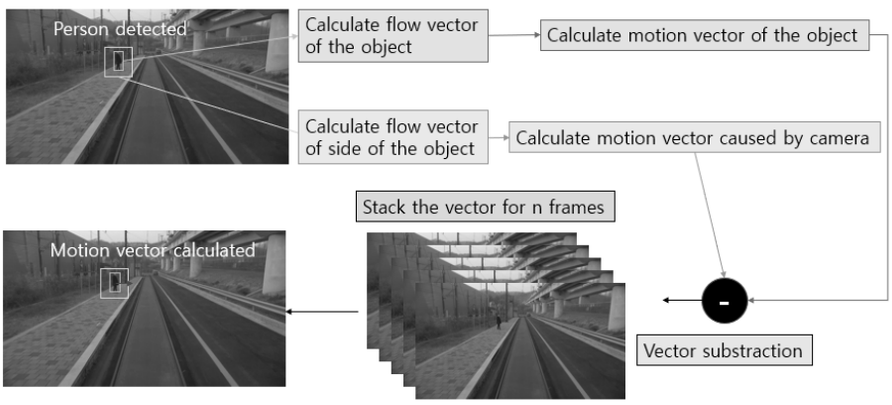


Fig. 4. Flow Chart of Acquiring Motion Vector by Optical Flow

$$\overrightarrow{v_o} = \frac{1}{N_b}\sum_{A_b} I(x,y) \mid (x,y) \in A_b \tag{5}$$

A_b는 물체가 탐지된 영역, N_b는 물체가 탐지된 영역 내 픽셀의 총 개수, I(x,y)는 좌표 (x,y)에서 Farnebäck 알고리즘을 이용하여 계산된 흐름 벡터(Flow Vector)를 의미하며 (u, v)는 흐름 벡터의 수평 성분의 크기(Horizontal)와 수직 성분의 크기(Vertical)를 나타낸다.

객체가 탐지되었을 때, 객체의 프레임 간 원시 이동 벡터(Primitive Motion Vector per Frame) v_o는 탐지된 영역 내의 모든 흐름 벡터를 평균한 것으로 정의한다.

$$\overrightarrow{v_c} = \frac{1}{N_s}\sum_{A_s} I(x,y) \mid (x,y) \in A_s \tag{6}$$

$$\overrightarrow{v} = \overrightarrow{v_o} - \overrightarrow{v_c} \tag{7}$$

A_s는 물체가 탐지된 주변 영역(외곽 경계 부분) , N_s는 물체가 탐지된 주변 영역 내 픽셀의 총 개수를 의미한다.

카메라가 움직이는 상황이므로 탐지된 영역의 주변 흐름 벡터값을 평균하여 이를 카메라에 의한 이동 벡터(Motion Vecter Caused by Camera) v_c로 정의한다. 객체의 프레임 간 원시 이동 벡터에서 카메라에 의해 생긴 객체의 이동 벡터를 감산함으로써 움직이는 카메라에 의한 영향이 제거된 객체의 프레임 간 이동 벡터(Motion Vector per Frame) v를 구한다.

$$\overrightarrow{v_a} = \sum_n \overrightarrow{v} \tag{8}$$

프레임 간 이동 벡터는 짧은 시간으로 인해 민감도(Sensitivity)가 높다. 객체의 이동 벡터를 안정화하기 위해 프레임을 누적한다[22]. 프레임 간 이동 벡터을 n번 누적하여 이를 객체의 이동 벡터 v_a로 계산하였다. 이를 통해 객체의 이동 벡터를 움직이는 카메라에서 얻어낼 수 있다.

밀집광학흐름은 픽셀 수가 많을수록 연산량이 증가하여 많은 시간이 소요된다. 이 문제를 해결하기 위해 탐지된 객체의 영역 주변을 계산영역으로 이용한 논문의 방식을 차용한다[23]. 위 논문은 정확한 광학 흐름을 계산하기 위해 제안된 방법이지만 본 연구에서는 연산할 픽셀 수를 감소시켜 연산 시간을 줄이는 방법으로 활용한다. 탐지된 객체를 포함한 영상의 일부에서의 밀집광학흐름을 부분밀집광학흐름(Local Dense Optical Flow)이라고 정의하고, 이를 이용하여 빠르고 정확한 계산을 할 수 있다.

$$A_c = \{(x,y) \mid (x,y) \in R \subseteq U\} \tag{9}$$

$$(A_{bp} \cup A_{sp}) \cup (A_{bn} \cup A_{sn}) \subseteq A_c \tag{10}$$

A_{bp} : *Detected Area of Previous Frame*
A_{sp} : *Surrounding Area of Previous Frame*
A_{bn} : *Detected Area of Present Frame*
A_{sn} : *Surrounding Area of Present Frame*

A_c는 부분밀집광학흐름을 계산할 영역집합을 의미한다. U는 전체 이미지 픽셀의 집합, R은 전체 이미지 픽셀의 부분 집합이다. 부분밀집광학흐름을 계산하기 위해서 이전 프레임과 같은 크기의 이미지가 필요하므로 계산 영역 A_c는 이전 프레임의 탐지 영역과 탐지 주변 영역, 현재 프레임의 탐지 영역과 탐지 주변 영역 모두를 포함한 영역이여야 한다.

3.2 충돌 예상 시간 및 충돌 지점 추정모델

광학 흐름을 이용하여 계산된 객체의 이동 벡터로 객체의 속도를 구한다. 이를 통해 주행체와 객체 간의 거리, 상대 속도의 크기를 구한 뒤 충돌 시간을 계산하고 속도의 방향을 이용하여 충돌 예상 지점을 예측한다. 본 논문에서는 1차원 충돌 모델(1-Dimension Collision Model)을 사용한다. Fig. 5에서 V_{ox} 와 V_{oy}는 객체의 이동 속도, V_{cy}는 주행체의 이동 속도, D_x는 객체와 충돌 지점간의 거리, D_y는 주행체와 충돌 지점과의 거리를 의미한다.

$$t_c = \frac{s_o}{v_r} \tag{11}$$

충돌 예상 시간은 뉴턴역학 속력-시간 Equation (11) 을 이용한다. t_c는 충돌 예상 시간, s_o는 카메라와 물체 간의 거

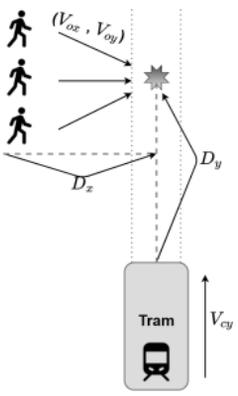


Fig. 5. 1-Dimension Collision Model

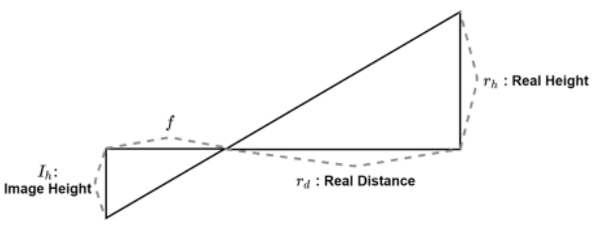


Fig. 6. Estimating Distance between Camera and Object

리, v_r은 카메라와 물체 사이의 상대 속도를 의미한다. 이미지에서 물체와의 거리는 다음 식에서 알 수 있다.

$$s_o = \frac{h_o \times f}{c_s} \times \frac{I_h}{I_o} \tag{12}$$

ho는 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.

4. 실험 결과 및 평가

4.1 실험 환경

오송 트램 시험선에서 촬영된 운행영상을 활용하여 실험을 진행하였다. 운영체제는 Ubuntu 18.04.5 LTS, ROS melodic1, GPU는 NVIDIA RTX 2080 super, CPU는 Intel Core i7-9700K @CPU 3.60GHz 환경에서 진행되었다.

알고리즘의 검증은 편의를 위해 위 실험환경에서 촬영된 영상을 바탕으로 진행되었다. 영상 정보는 Table 2와 같다.

객체 인식은 YOLOv5 알고리즘을 이용하며 COCO 데이터셋으로 사전 학습된 가중치 모델 중 가장 처리속도가 빠른 S모델을 적용한다. 입력 이미지 크기는 640×640으로 설정한다.

광학 흐름을 Equation (9)에서 정의된 계산영역 A_c는 Equation (5)에서 정의된 탐지영역 A_b의 중심을 기준으로

Table 2. Video Information

Category	Information
Size	2040×1086 (≒FHD)
Frame rate(FPS)	8 frame/sec
Filming Date	2020.12.11
Additional Information	5m~30m, step : 5m, 3 different angle except 5m (2 different angle), total 17 videos

탐지된 영역의 2배의 영역을 계산하였다. Equation (6)에서 정의된 주변영역 A_s는 탐지된 영역을 중심으로 1.5배의 영역에서 탐지 영역을 제외한 영역으로 설정하였다.

Farnebäck 알고리즘의 실행은 한 계층마다 0.5 비율로 스케일링 되는 5계층 피라미드 스케일링(Level:5, Pyramid scale ratio : 0.5)을 이용하여 각 블록의 크기는 15×15 픽셀이며 (Window size=15), 재계산은 3회 진행되며(Iterations:3), 다항식 근사에 계산되는 블록의 크기는 5×5픽셀이고 (Polynominal block size=5), 가우시안 1.1의 표준편차 (Polynominal sigma=1.1) 값으로 진행되었다.

Equation (8)에서 정의된 객체의 이동 벡터는 5프레임(n=5) 동안의 프레임 간 이동 벡터를 더하여 계산하였다.

Equation (12)에서 정의된 객체와 주행체 간의 거리를 구하기 위해 객체의 높이(h_o)는 1.75m로 가정하였으며, 실험에 사용된 카메라의 CCD 센서 크기는 2/3"(8.6mm×6.6mm, c_s=6.6mm). 초점 거리(f)는 8mm이다.

4.2 실험 결과 및 평가

1) YOLOv5 기반 객체 탐지결과

본 실험에서는 트램의 정지선과 객체와의 거리를 5m에서 30m 까지 5m 간격으로 설정하고 객체가 진입하는 각도를 트램의 진행 방향과 수직인 방향에 대해 -45°, 0°, 45°로 설정하였다.

모델의 평가방법으로는 객체 인식률로 설정하고 객체 거리 추정 오차를 계산한다. 오차 계산방법은 Equation (13)과 같이 백분율 오차를 이용하였으며, P는 객체가 존재하는 프레임의 개수, P_E는 YOLOv5에 의해 사람이 탐지된 프레임의 개수이다. 가중치 모델에 따른 탐지율 및 속도는 Fig. 7에서 확인할 수 있다. 거리와 각도에 따른 객체의 탐지율은 Table 3에서 확인할 수 있다.

$$Accuracy(\%) = \frac{P - P_E}{P} \times 100 \tag{13}$$

2) 객체의 Optical Flow 실험 결과

본 실험에서는 실제 객체의 이동방향과 계산된 이동 백터의 방향이 일치하는지 확인한다. 사람의 걷는 속도는 평균 1.38m/s라고 알려져 있다. 측정된 속도는 Equation (14)과 같다.

Table 3. Object Detection Ratio

Distance of the Object from Stop Line	Direction of the Object(˚)	Detection Ratio
5m	0	100%
	45	100%
10m	-45	100%
	0	100%
	45	100%
15m	-45	100%
	0	100%
	45	100%
20m	-45	100%
	0	99.63%
	45	100%
25m	-45	100%
	0	95.37%
	45	100%
30m	-45	100%
	0	91.2%
	45	90.85%

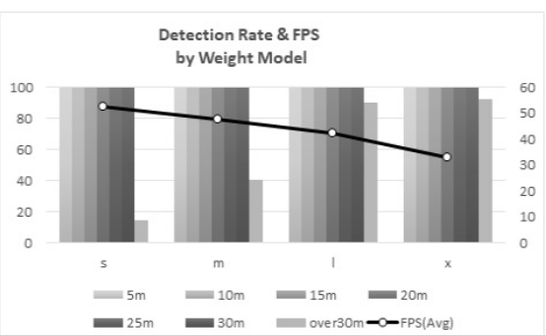


Fig. 7. YOLOv5 Model Comparison

$$v_o = \frac{s_o}{f} \times C_{sw} \frac{I_{ow}}{I_w} \times fps \tag{14}$$

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

영상 내 객체의 이동 벡터의 크기와 각도 평균은 Table 4와 같다.

Fig. 9는 객체가 15m에서 0도로 진입하는 경우에서 흐름 벡터가 계산된 결과를 나타낸다. 그리드(Grid) 형태로 객체 주변의 흐름 벡터만을 계산하여 나타내고 객체의 대표 모션벡터를 추출한 그림이다.

3) 충돌예상 판단 기준

객체의 이동 방향이 트램의 레일 안으로 진입하는 방향

Table 4. 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)
5m	0	146.87	2.23	0.72
	45	117.16	1.36	0.44
10m	-45	117.99	1.06	0.69
	0	28.76	0.77	0.50
	45	85.52	0.49	0.32
15m	-45	16.98	0.72	0.70
	0	15.92	0.77	0.75
	45	91.95	0.58	0.56
20m	-45	-2.22	1.06	1.37
	0	1.45	0.94	1.22
	45	25.27	0.49	0.63
25m	-45	-0.03	0.66	1.07
	0	3.26	0.61	0.99
	45	11.11	0.46	0.74
30m	-45	-0.15	0.53	1.03
	0	0.80	0.48	0.93
	45	21.21	0.21	0.41

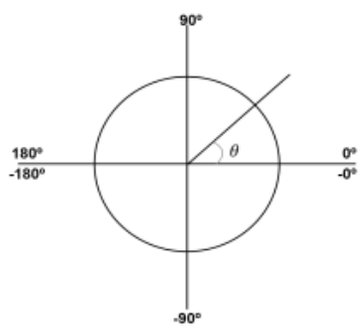


Fig. 8. Angle of Motion

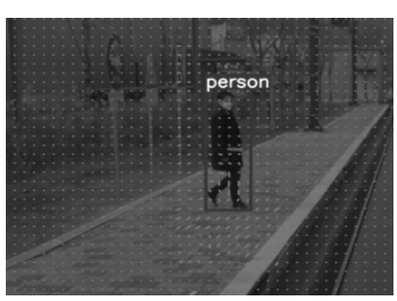


Fig. 9. Object Detection and Motion Vector Extraction, Flow Vector Grid

이고 충돌 예상 시간이 5초 이내인 경우 경고 표시를 하도록 하였다. Fig. 10는 15m에서 0도로 진입하는 경우에서 시스템이 실행된 화면이다. 충돌이 예상되는 이러한 경우에 경고 메시지를 표시한다.

4) 객체 모션정보 알고리즘 계산 속도(Runtime)

Table 5는 각 알고리즘의 프레임 당 소요 시간을 나타낸다. 단위는 ms(Milisecond)이다.

Fig. 10. TTC, Predicted Collision Point & Warning Message

Table 5. Average Runtime (ms)

Method	Min	Max	Avg
Farnebäck	526	571	536
SimpleFlow	2902	3104	2973
Previous Method(2016)[24]	380	600	490
Our Method	9	66	20

Farnebäck은 프레임당 평균 536ms, SimpleFlow는 평균 2.973ms, LK 알고리즘을 활용한 방법[24]은 490ms가 소요된다. 반면에 본 연구의 제안 방법은 평균 20ms가 객체의 이동정보 추출을 위한 연산량과 소요기간을 단축하였다.

5. 결론 및 향후 연구

제안한 방법을 통해 FHD 영상에 대해 실시간(Real-time, 30fps 기준)처리를 달성할 수 있었으며, 위의 방법을 통해 트램과 보행자 간의 충돌 가능성을 판단하고 사고를 사전에 방지할 수 있다. 기존에 제안되었던 LK 알고리즘을 이용한 방법보다 객체의 이동을 더 정확한 예상을 할 수 있으며, 처리속도는 기존의 방법보다 약 10배 빠른 것을 확인할 수 있다.

다만, 가까운 객체에 대해 광학 흐름을 계산하는 경우 카메라로 인해 생긴 흐름 벡터를 완전히 제거하지 못하여 객체의 이동벡터에 영향을 주게 되었다. 하나의 카메라를 이용하여 깊이(Depth) 성분을 정확히 추정할 수 없어 객체의 이동벡터의 상하 성분(Y axis Component)이 객체의 실제 상하성분과 차이가 존재했다.

이러한 문제를 해결하기 위해 객체의 광학흐름 데이터를 학습한 딥러닝 모델을 적용할 계획이다. 그러나 딥러닝 모델을 활용한 광학 흐름의 계산 현재 연산처리시간(Runtime)이 긴 편으로 향후 그래픽카드(GPU)의 구조 및 성능 향상과 빠르고 가볍게 계산 가능한 딥러닝 모델을 이용하여 흐름 벡터를 계산하여 위 방법보다 더 빠르고 정확하게 객체를 탐지하고 이동 벡터를 탐지할 수 있을 것이다.

References

[1] S. Lee, M. Myung, and T. Kim, "A study on tram traffic accidents characteristics and safety measures," Vol.39, No.4, pp.505-512, 2019.

[2] A. Manglik, X. Weng, E. Ohn-Bar, and K. M. Kitanil, "Forecasting time-to-collision from monocular video: Feasibility, dataset, and challenges," In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.8081-8088, 2019.

[3] H. Kataoka, T. Suzuki, K. Nakashima, Y. Satoh, and Y. Aoki, "Joint pedestrian detection and risk-level prediction with motion-representation-by-detection," In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp.1021-1027, 2020.

[4] Y. Cai, L. Dai, H. Wang, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, "Pedestrian motion trajectory prediction in intelligent driving from far shot first-person perspective video," *IEEE Transactions on Intelligent Transportation Systems*, pp.1-16, 2021.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.779-788, 2016.

[6] C. Y. Wang, H. Y. M. Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp.390-391, 2020.

[7] J. Park, S. Woo, J. Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2018.

[8] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, Vol.107, pp.3-11, 2018.

[9] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.8759-8768, 2018.

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," In *Computer Vision and Pattern Recognition*, pp.1804-2767, 2018.

[11] T. Y. Lin, et al., "Microsoft coco: Common objects in context," European Conference on Computer Vision. Springer, Cham, pp.740-755, 2014.

[12] Ultralytics. YOLOv5 [Internet] https://github.com/ultralytics/yolov5

[13] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, Vol.17, No.1-3, pp.185-203, 1981.

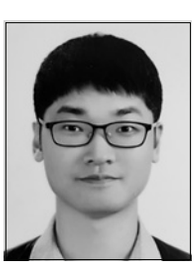
[14] A. B. Khalifa, I. Alouani, M. A. Mahjoub, and N. E. B. Amara, "Pedestrian detection using a moving camera: A novel framework for foreground detection," *Cognitive Systems Research*, Vol.60, pp.77-96, 2020.

[15] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods," *International Journal of Computer Vision*, Vol.61, No.3, pp.211-231, 2005.

[16] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," *Scandinavian Conference on Image Analysis*. Springer, Berlin, Heidelberg, 2003.

[17] A. Dosovitskiy, et al., "Flownet: Learning optical flow with convolutional networks," In *Proceedings of the IEEE International Conference on Computer Vision*, pp.2758-2766, 2015.
[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.2462-2470, 2017.
[19] M. Bevilacqua, A. Tsourdos, and A. Starr, "Egomotion estimation for monocular camera visual odometer," *IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pp.1-6, 2016.
[20] J. Lim and B. Han, "Generalized background subtraction using superpixels with label integrated motion estimation," *European Conference on Computer Vision*, Springer, Cham, pp.173-187, 2014.
[21] L. Kurnianggoro, A. Shahbaz, and K. H. Jo, "Dense optical flow in stabilized scenes for moving object detection from a moving camera," *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, pp.704-708, 2016.
[22] J. Mao, H. Yu, , X. Gao, and L. Yu, "Complementary motion vector for motion prediction in video coding with long-term reference," In *2019 Picture Coding Symposium (PCS)*, pp.1-5, 2019.
[23] M. Mühlhausen, L. Wohler, G. Albuquerque, and M. Magnor, "Iterative optical flow refinement for high resolution images," In *IEEE International Conference on Image Processing (ICIP)*, pp.1282-1286, 2019.
[24] Y. C. Kuo, C. T. Tsai, and C. H. Chang, "Fast estimation of pedestrian movement," *Sensors and Materials*, Vol.29, No.6, pp.713-726, 2017,

김 영 민
https://orcid.org/0000-0002-5948-456X
e-mail : winston1214@naver.com
2020년 차세대융합기술연구원 컴퓨터비전 및 인공지능연구실 연구원
2016년 ~ 현 재 인천대학교 경제학과/컴퓨터공학부 학사과정
관심분야 : Computer Vision & BigData



안 현 욱
https://orcid.org/0000-0002-8127-864X
e-mail : hyeonukahn@korea.ac.kr
2016년 ~ 현 재 고려대학교 전기전자공학부 학사과정
관심분야 : Deep Learning, Computer Vision & Bigdata

전 희 균
https://orcid.org/0000-0002-5407-5193
e-mail : llhgll@krri.re.kr
2020년 대구경북과학기술원(DGIST) 정보통신융합공학과(박사)
2020년 ~ 현 재 한국철도기술연구원 스마트트램연구실 선임연구원
관심분야 : Model Predicted Control for TramSensor Fusion, Resilient Cyber-Physical Systems, Robust Control

김 진 평
https://orcid.org/0000-0003-4840-7216
e-mail : jpkim@snu.ac.kr
2006년 성균관대학교 전자전기컴퓨터공학과(석사)
2014년 성균관대학교 전자전기컴퓨터공학과(박사)
2016년 ~ 2018년 한국철도기술연구원 선임연구원
2018년 ~ 2019년 한국도로공사 도로교통연구원 책임연구원
2019년 ~ 현 재 차세대융합기술연구원 선임연구원
관심분야 : Artificial Intelligence & Computer Vision

장 규 진
https://orcid.org/0000-0002-1575-2796
e-mail : gjjang@snu.ac.kr
2011년 성균관대학교 전자전기컴퓨터공학과(석사)
2013년 성균관대학교 전자전기컴퓨터공학과(박사수료)
2018년 한국철도기술연구원 스마트모빌리티연구팀 주임연구원
2020년 ~ 현 재 차세대융합기술연구원 컴퓨터비전 및 인공지능연구실 연구원
관심분야 : Computer Vision & Artificial Intelligence

황 현 철
https://orcid.org/0000-0002-3886-6956
e-mail : hchwang@krri.re.kr
1997년 인하대학교 전자공학과(학사)
1999년 인하대학교 전자통신(석사)
2006년 인하대학교 전자통신(박사)
2006년 ~ 현 재 한국철도기술연구원 스마트트램연구실 책임연구원
관심분야 : Railway Signalling and Communication