Pix2Pix와 MobilenetSSD를 활용한 라즈베리파이 기반 보행자 검출 방안

손푸름*, 옥재은*, 류수현*, 강수명**, 이준재*
*계명대학교 컴퓨터공학부 게임모바일공학전공
**계명대학교 컴퓨터공학과
e-mail : sonpulum@naver.com

Raspberry-pie Based Pedestrian Detection Method Using Pix2Pix and MobileNetSSD

Pu-Lum Son*, Jae-Eun Ock*, Su-Hyeon Ryu*, Su-Myung Gang**, Joon-Jae Lee*
*Dept. of Game Mobile, Faculty of Computer Engineering, Keimyung Univ.
**Dept. of Computer Engineering, Keimyung Univ.

요 약

기존의 보행자검출 관련 연구에서 데이터베이스 구축에 어려움을 겪는 것과는 달리, 본 연구에서는 딥러닝을 통한 보행자 검출 방안의 자동화 파이프라인을 제안한다. 학습된 Pix2Pix 모델을 통해 원 영상에서 보행자를 검출한 마스크를 자동으로 생성가능하다. 또한, 보행자 영상과 마스크 영상을 객체 검출 데이터 셋 형태로 변환한 뒤 이를 MobilenetSSD 모델에 학습시켜 라즈베리파이에 적용하였다.

1. 서론

최근 스마트폰의 보급 및 활용 증가로 인해 횡단보도 인근에서의 보행자 사고가 증가하고 있다. 이는 교통안전공단에서 수행한 「스마트폰 사용이 보행안전에 미치는 위험성 연구」에서 발표되었다. 그 외에도 다양한 이유로 보행자의 사고가 빈번히 발생하고 있다. 따라서 이러한 사고를 컴퓨터 비전을 활용한 보행자 검출을 통해서 줄이려는 시도가 늘고 있다[1-3].

과거 특징기술자 추출에 의존한 방법에서, 현재 가장 검지 정확도가 높은 딥러닝 모델을 이용하는 기법들이 연구되면서 성능이 꾸준히 향상하고 있다. 하지만 딥러닝의 경우 많은 데이터베이스 확보가 필수적이며 수많은 데이터베이스를 통해 보편적으로 적용 가능 한 모델이 학습 가능하다. 또한 많은 데이터베이스를 확보하였더라도 횡단보도 인근에 정지한 보행자의 경우 일반적인 사람형태가 아닌 여러 가지 형태를 띄고 있어 데이터베이스 구축에 세밀함에 따라 결과가 좌우된다. 즉, 사람은 잘 구분하는 모델이더라도 모자쓴 사람, 우산 쓴 사람, 휠체어에 탄 사람, 자전거에 탄 사람, 유모차에 탄 사람, 여러 사람이 겹쳐있는 경우 등 여러 가지 다른 형태의 사람이 있을 수 있다. 따라서 데이터베이스를 한 번 구축하는 것으로 학습이 완료 되는 것이 아니라, 지속적인 데이터 업데이트가 필요하다. 특히 본 연구와 같이 물체인식(object detection) 분야에서는 원 영상과 마스크영상을 레이블링(lableing) 하는 경향이 많아 데이터베이스 구축이 어렵다[1-5].

본 논문에서는 데이터셋 구축과 보행자 검출을 자동화하는 파이프라인을 제시하고, 직접 데이터베이스를 만들어 학습시킨 MobileNetSSD 모델을 통해 라즈베리파이를 이용하여 보행자 검지를 하는 방안에 대해 연구한다.

2. 관련연구

빈운택 등[1]의 연구에서는 COCO(Common Objects in Context) 데이터셋에 의해 학습 MobileNetSSD 모델과 라즈베리파이를 이용하여 보행자 검출 시스템을 구현 하였다. 즉, 해당 연구에서는 직접 데이터베이스를 훈련시킨 것이 아니라, 기존 21개 COCO 데이터 셋으로 학습된 모델에서 간단한 필터링으로 사람 등을 포함하는 3개만을 검출하는 모델로 바꾸어 적용하였다.

송수호 등[3]의 연구에서는 Faster R-CNN 기법에 새로운 학습 모델을 적용하여 보행자 검출 오류를 줄이는 기법을 제안하였다. 기존 계층적 분류기 방법에 비해 ConvNet이 좋은 결과를 보이지만 야간에서의 검출 성능이 크게 떨어지는 한계를 보였다.

안명수와 강대성[4]은 컨볼루션 신경망 모델을 이용하여 객체 분류기 생성 기술로 보행자를 검출하는 방법을 제안하였다. 이를 통해 영상분석 봇 시스템을 구현하였다. 기존 특징분류 방법을 사용한 결과에 비해 CNN의 결과가 96.23%으로 매우 좋음을 보여주었다.

라즈베리파이는 저사양 임베디드 환경으로서 빠른 연산이나 효과적인 퍼포먼스를 보여주기에는 한계가 있으나, 보행자 검출 외에도 라즈베리파이에 딥러닝이 활용될 수 있음을 강현준과 김녹원[5]의 연구에서 알 수 있다. 해당

연구에서는 기존의 바코드가 훼손되거나 인식이 잘 되지 않는 상품의 경우에는 추가적인 조치가 필요한 시스템의 문제들을 해결하기 위해 라즈베리파이에 영상 인식 딥러닝 기반의 자동화 계산 시스템을 결합한 방법을 제안하였다. 이상훈 등[6]은 보행자 검출과 보행자 재인식에 관한 연구를 진행했다. 이에 성능을 높이기 위한 다양한 방법들이 존재하는데 연구결과 딥러닝을 이용한 방법이 기존의 방법들보다 우수한 성능을 보이고 있다.

기존 연구를 살펴볼 때, 라즈베리파이 환경에서 충분히 딥러닝을 활용 가능 할 것으로 보이나 속도 측면에서 한계가 있는 것을 알 수 있다. 또한 데이터베이스 구축에 한계를 보이는 연구도 있으며, 온라인상에 공개된 사전 학습 모델을 활용하는 등의 연구가 주류를 이루고 있다.

3. 실험 방법 및 구현

3.1 Pix2Pix 기반 마스크 생성

Pix2Pix는 영상을 입력으로 받아서 다른 형식의의 영상을 출력하는 GAN 구조의 알고리즘이기 때문에 지도학습 방법이다. 즉, 이를 학습시키기 위해서는 보행자가 있는 원 영상과 보행자 영역이 마스크로 구성된 영상이 필요하다. 초기 학습 데이터를 구축하기 위한 Pix2Pix 역시 딥러닝 모델 중 하나이므로 최초 학습데이터는 수작업이 필요하다는 한계가 있으나, 많은 초기 학습 데이터를 구축하여 해당 모델을 구성한 뒤에는 추가 데이터에 대해서는 자동화 된다는 장점이 있다. 보행자 오검출의 경우 앞서 기술한 것과 같이 다양한 사례가 있으므로 마스크 영상 생성 자동화는 매우 편리할 것으로 기대할 수 있다.

Pix2Pix는 생성자와 구별자 모델로 이루어져 있으며, 생성자는 U-Net 구조로 구성되어있다. 그림 1[8]은 Pix2Pix에서 사용하는 생성자 구조이다.

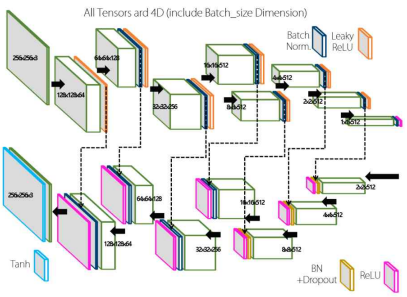

그림 1. Pix2Pix의 생성자 구조

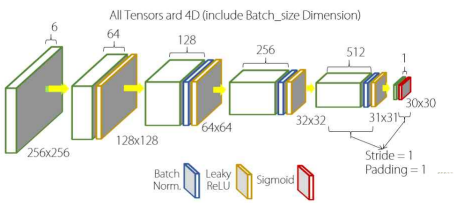

그림 2. Pix2Pix의 구별자 구조

해당 구조는 스킵연결(skip connection)로 각 인코더에서 업샘플링 되는 과정에서 생기는 손실을 디코더에서 활용할 수 있도록 대응되어 있다[7-10]. 구별자 구조에서는 그림 2[9]와 같고, 최종적으로 30×30의 벡터를 출력하여 영상 전체를 900개의 구역으로 비교할 수 있다[7-10].

3.2 데이터베이스

본 연구에서 활용하고자 하는 데이터베이스는 [11]에서 제공하는 원본 보행자 영상과 보행자 컬러 마스크 영상을 사용하였다. Pix2Pix에서 활용할 초기 데이터베이스는 손수 구축해야하는 한계가 있으므로 공개적으로 제공된 데이터베이스를 사용하였다.

[11]에서 제공하는 영상데이터의 양이 170장으로 딥러닝 학습에 활용하기에는 매우 부족했기 때문에 제공하는 영상을 좌우 반전시킨 뒤 추가하여 원본 영상 340장, 컬러 마스크 영상 340장으로 데이터의 양을 두 배 늘려주었다.

일반적으로 물체인식 연구에 활용하는 데이터 셋 형식은 COCO 형식과 PascalVOC등 두 가지를 주로 활용한다. 따라서 raw data로부터 두 가지 형식을 모두 활용할 수 있도록 하는 것이 효율적이다[12-15].

COCO 데이터셋을 만들기 위해서는 원본 영상과 이진화 마스크 영상이 필요하고, 한 영상에 둘 이상의 객체가 존재 할 경우 하나 씩 따로 마스킹 한 영상이 필요하기 때문에 [11]에서 제공되는 보행자 컬러 마스크 영상을 OpenCV의 HSV색 공간을 활용하여 색깔별로 한 사람씩 따로 분류하고 이진화를 통해 그림 3와 같이 흑백 마스크 영상으로 생성한다.

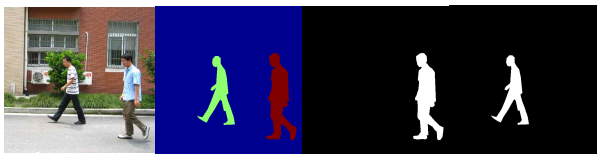

그림 3. 만들어낸 흑백 마스크 영상
(왼쪽부터 원영상, 컬러 마스크, 분리된 마스크1과 2)

해당 결과를 통해 Pycocotools[12]를 이용하여 원본 보행자 영상과 생성한 흑백 마스크 영상으로 COCO 데이터셋을 생성하고, 온라인에 공개된 소스코드를 통해 PascalVOC로도 변환 할 수 있다[13]. 이때 원시 데이터를 COCO로 먼저 변환하는 것은, PascalVOC 형태로 바로 변환하는 것에 비해 COCO로 변환하는 것은 원 영상에 개별 마스킹 영상만 준비하는 전처리가 훨씬 편리하기 때문이다[12-13]. 변환된 데이터는 최종적으로 텐서플로우에서 용량은 적게 차지하기 때문에 연산 속도는 빠르지만 데이터 손실이 적은 TFrecord로 변경한다[14-15].

3.3. MobileNetSSD 기반 보행자 검출 방법

본 논문에서는 3.1절에서 언급한 Pix2Pix 방법으로 데이터베이스가 확보되었음을 가정하고, 3.2절에서 확보한 데이터베이스를 통해 라즈베리파이에 텐서플로우를 설치하고 직접 만든 데이터베이스를 학습시킨 MobileNetSSD 모

델을 사용하여 보행자 검출을 실시한다[14-15].

그림 5는 앞서 기술한 데이터베이스 구축 자동화를 통한 보행자 검출 방법의 파이프라인이다. 특히 보행자 검출 시에 발생하는 새로운 타입의 에러 영상마다 저장하여 신규 영상을 확보하고 이를 통해 마스크 영상을 생성하여 신규 모델을 트레이닝할 경우 점차 인식 정확도가 증가하는 모델이 될 것을 기대할 수 있다.

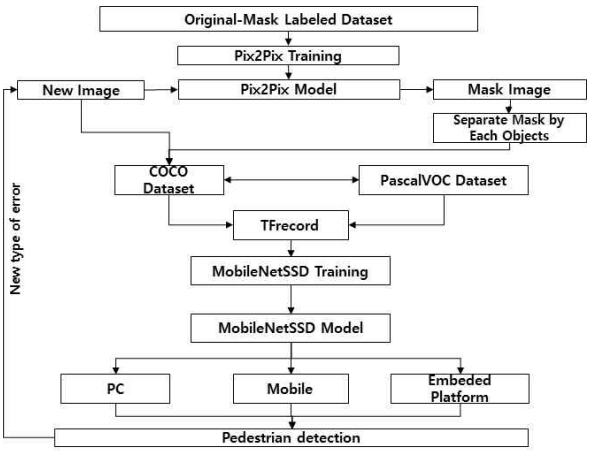

그림 4. Pix2Pix와 MobileNetSSD를 활용한 데이터베이스 구축 자동화 보행자 검출 파이프라인

MobileNetSSD는 임베디드 환경에서 연산 단축을 위해 제시된 MobileNet의 구조와 물체 인식에서 가장 빠른 결과를 보이는 SSD(Single Shot Multi Detector)의 구조를 합한 알고리즘으로 convolution 연산 대부분이 separable -depthwise convolution을 통해 수행되므로 적은 파라미터로 효과적인 결과를 제시하면서 빠른 물체 인식이 가능한 구조이다[1]. 특히 해당 소스코드는 구글에서 만든 오픈소스 머신러닝 라이브러리인 텐서플로우에서 공개 했을 뿐 아니라 여러 클래스를 가진 학습데이터를 통해 사전 학습을 해놓은 pretrained model도 공개해 두었다. 따라서 본 연구에서도 기반되는 소스코드는 공개된 소스코드를 일부 수정하여 응용한다[15].

또한, 기존 연구에서는 간단한 필터링을 통해서 인식하는 객체의 수를 21개에서 3개로 줄인데 반해, 본 논문은 직접 데이터베이스를 만들어 모델에 학습 시킨 뒤 보행자 검출에 필요한 '사람'의 단일 클래스만 분류하도록 모델을 변경한다.

4. 실험 결과 및 고찰

4.1 실험 결과

데이터베이스를 직접 만들어 MobileNetSSD 모델에 학습 및 Pix2Pix 학습의 환경과, 보행자 검출을 위해 사용한 라즈베리파이의 환경은 표 1 과 같다.

본 연구를 위한 보행자 데이터가 충분히 확보되지 않았다는 한계가 있어, 앞서 기술한 것과 같이 본 연구에서 pix2pix 모델은 가능성을 검증하는 것으로 실험을 수행하였다. 총 320개 데이터를 7.5:2.5로 분리하여 240개의 데이터로 학습, 나머지 80개에 대해 테스트를 수행하였다. 그림 3은 해당 모델을 통해 마스크를 예측한 테스트 결과다.

표 1. An environment of research

Category	Contents	Details
Training	Processor	Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
	RAM	32GB
	Graphic Card	NVIDIA GeForce GTX 1080 8GB
	Operating System	Windows 10 Pro
	Platform	Anaconda3 5.2.0 Tensorflow 1.9.0 / Python 3.5.6
Inference	Processor	Pi 3 Model B+ ARMv7 Processor rev 4 (v7l)
	RAM	1GB
	Operating System	Raspbian GNU/Linux 9 (stretch)
	Platform	Tensorflow 1.13.1 / Python 3.5.3

그림 3. Pix2Pix를 활용한 마스크 예측결과 (왼쪽부터) 원영상, 예측결과, 실제마스크

보행자 검출부의 경우 [11]로부터 구축된 TFRecord 데이터로 학습된 모델은 라즈베리파이에 저장소에 삽입후 보행자 검출 모델로 활용하였다. 이때 입력영상의 크기는 부착된 카메라의 한계로 인해 640×480을 사용하였다. 본 연구에서 학습시킨 모델은 클래스가 '사람' 단일 분류로서, 기존 사전 학습 모델을 적용한 결과인 0.65 FPS 보다 0.86FPS 로서 속도부분에서 좀 더 개선되었다(그림5). 이는 MobileNetSSD의 최종 객체 인식 시에 사용하는 연산 구조가 Class의 개수가 증가되면, 연산 파라미터도 약간 증가되기 때문에 연산 시간도 비례한다.

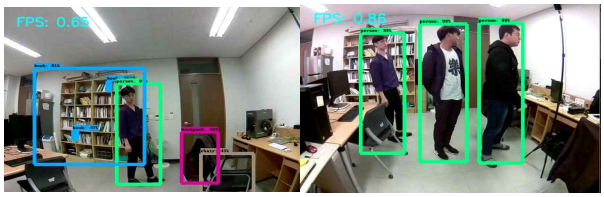

그림 5. 기존 결과(오른쪽)와 본 연구의 결과(왼쪽)

4.2 고찰

본 연구에서는 딥러닝을 활용하여 데이터베이스 구축 자동화 및 라즈베리파이 기반 보행자 검출 파이프라인 및 방안을 제시하고자 하였다. 아직까지 초기 데이터베이스

구축의 한계가 있기 때문에 자동화 수행에는 미흡함이 있었다. 하지만 각 절의 실험에서 도출된 결과에 의해서 제안하는 파이프라인이 수정 보완을 통해 지속 가능한 구조임을 알 수 있었다.

각 절의 실험에 대해 상세한 고찰을 해보면, 우선 Pix2Pix의 결과가 예상 이하의 결과를 보였으나 이는 데이터베이스의 부족에 의한 결과로 볼 수 있다. 기존 마스크 데이터베이스 구축의 경우 수작업이 거의 필수적이나, 이러한 방법으로 수작업의 대부분을 지양하고 일부 미세한 부분을 수정 하는 등 충분한 활용 가능성이 있다.

또한, 라즈베리파이에 MobilenetSSD 라는 가벼운 딥러닝 모델을 사용했지만 초당 1장 이상의 결과는 나오지 않았다. 이는 라즈베리파이의 성능적 한계가 있기 때문으로, 라즈베리파이보다 성능이 더 좋은 임베디드 플랫폼 에서는 속도가 훨씬 개선될 것이다.

하지만 저가형 보드를 그대로 활용할 시에는 반드시 모델을 현존하는 모델보다 더 가벼운 모델로 수정할 필요가 있다. 최근 많은 연구에서 딥러닝의 연산 속도 문제는 컨볼루션에서 기하급수적으로 늘어나는 파라미터 문제때문이라고 하였으며, 이는 적절한 depthwise separable 컨볼루션의 사용으로 해결할 수 있을 것으로 판단된다.

마지막으로, 본 논문의 연구 결과는 명확하게 사람의 형체를 띄고 있는 경우에는 정확히 검출해낸다. 즉, 그림 6과 같은 문제를 야기한다. 하지만, 사람의 신체 일부분이 가려져 있을 때 사람으로 분류하지 못하는 경우가 발생한다. 이를 보완하기 위해서는 앞서 기술한 것처럼 체계적인 사람 형태의 분류체계를 만들 필요가 있다.

그림 6. 신체 일부분이 가려져 검출하지 못하는 경우

5. 결론

본 논문에서는 딥러닝을 활용하여 데이터베이스구축과 보행자 검출을 가능하게 하는 파이프라인을 제시하고, 직접 학습시킨 MobileNetSSD 모델과 라즈베리파이를 이용하여 실시간 보행자 검출을 수행하였다. 해당 결과로 약간의 연산 속도의 이점은 확인 하였으나, 실제 보행자 안전을 우해 신호등 보조기 등에 적용할 정도의 결과는 보이지 못한다. 향후 임베디드 플랫폼의 발전과 함께 딥러닝 연산 모델의 구조적 변경을 통해 좀 더 가볍고 보행자 검출에 최적화된 모델을 구현 하여 문제를 개선 가능하다.

참고문헌

[1] 빈운택, 김지영, 채병록, 최성광, 박순영, 최경호, “라즈베리파이 기반의 보행자 검출 시스템 구현에 관한 연구,” *한국통신학회 학술대회논문집*, pp. 880-880, 2018.

[2] 이주형, 한문석, “비콘 서비스를 사용한 보행자 안전 신호감지시스템의 설계,“ *정보과학회 컴퓨팅의 실제 논문지*, Vol. 22, No. 11, pp. 576-582, 2016.

[3] 송수호, 현훈범, 이현, “심층 신경망을 이용한 보행자 검출 방법,“ *정보과학회논문지*, Vol. 44, No.1, pp. 44-50, 2017.

[4] 안명수, 강대성, “심층 컨볼루션 신경망 모델 기반 객체 분류기를 이용한 영상분석 시스템 개발,“ *한국정보기술학회논문지*, Vol. 14, No. 5, pp. 67-73, 2016.

[5] 강현준, 김녹원, “라즈베리파이, 아두이노 임베디드 플랫폼에 이미지 인식 딥러닝을 적용한 제품 인식 및 자동화 계산 시스템,“ *한국정보과학회 프로시딩*, pp. 1853-1855, 2018.

[6] 이상훈, 추성권, 권기범, 조남익, “지능형 CCTV 시스템을 위한 보행자 검출과 재식별 기술,“ *한국통신학회지(정보와통신)*, Vol. 34, No. 7, pp. 40-47, 2017.

[7] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., *et al.*, “Generative adversarial nets,” *In Advances in neural inormation processing systems*, pp. 2672-2680, 2014.

[8] Isola P., Zhu J. Y., Zhou T., and Efros A. A, “Image-to-image translation with conditional adversarial networks,” *arXiv preprint*, 2017.

[9] Easy Pix2Pix Implementation in Pytorch, https://github.com/taeoh-kim/Pytorch_Pix2Pix, (Accessed Oct. 20. 2018)

[10] 강수명, 이준재, “GAN을 활용한 단일 영상의 깊이맵 추출,” *2018 한국멀티미디어 추계학술대회 프로시딩*, pp. 102-105.

[11] Penn-Fudan Database for Pedestrian Detection and Segmentation, https://www.cis.upenn.edu/~jshi/ped_html (accessed Apr., 17, 2019)

[12] Helper functions to create COCO datasets (2018), https://github.com/asyncbridge/pycococreator (accessed Apr., 17, 2019)

[13] Convert MS COCO Annotation to Pascal VOC format (2018) https://gist.github.com/AlexeyGy/5e9c5a177db31569c20c76ad4dc39284 (accessed Apr., 17, 2019)

[14] Train ssd_mobilenet of the Tensorflow Object Detection API with your own data (2018), https://github.com/naisy/train_ssd_mobilenet (accessed Apr., 17, 2019)

[15] Tutorial to set up TensorFlow Object Detection API on the Raspberry Pi (2019), https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi (accessed Apr., 17, 2019)