4족 보행 로봇 기반의 실시간 사람 검출 방법

한성민, 유상중, 이건, 박명숙, 김상훈
한경대학교 전기전자제어공학과
kimsh@hknu.ac.kr

Real-time human detection method based on quadrupedal walking robot

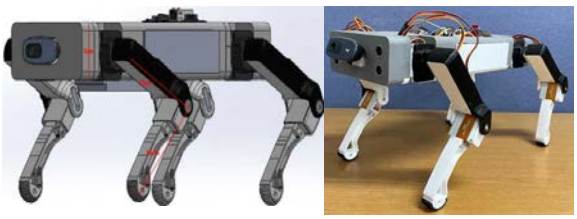
Seong-Min Han, Sang-jung Yu, Geon Lee, Myeong-Suk Pak, Sang-Hoon Kim
Dept. of Electrical, Electronic and Control Engineering, Hankyong National University

요 약

본 논문은 강화학습 POMDP(Partially Observable Markov Decision Process) 알고리즘을 사용하여 자갈밭과 같은 비평탄 지형을 극복하는 4족 보행 지능로봇을 설계하고 딥러닝 기법을 사용하여 사람을 검출한다. 로봇의 임베디드 환경에서 1단계 검출 알고리즘인 YOLO-v7과 SSD의 기본 모델, 경량 또는 네트워크 교체 모델의 성능을 비교하고 선정된 SSD MobileNet-v2의 검출 속도를 개선하기 위해 TensorRT를 사용하여 최적화를 진행하였다

1. 서론

4족 로봇은 인간의 단순하고 반복적인 정찰 업무 혹은 위험지역 탐사를 대체하여 편의와 안정성을 제공한다. 하지만 기존에 존재하는 대표적인 4족 로봇 Boston Dynamics 사의 스폿(Spot) 혹은 MIT 대학의 치타(Cheetah)는 고가이기 때문에 단순 업무를 수행함에 있어 비효율적이고 사용화가 어렵다. 본 연구에서는 저비용의 단순 정찰 업무를 수행하는데 적합한 소형로봇을 설계하고 성능을 검증하기 위해



(그림 1) 본 논문에서 설계한 4족 보행로봇의 외형

이전 연구[1]를 바탕으로 (그림 1)과 같은 로봇을 실제 제작하였다. 로봇이 포장된 도로 외에도 자갈밭과 같은 비평탄 환경에서의 정찰이 가능하도록 강화학습인POMDP 알고리즘[2]을 사용하여 균형 잡힌 보행을 구현한다. 본 논문은 로봇의 설명 형식(URDF) 파일을 만들어 시뮬레이션 환경에서 200 Episode로 학습[3] 시킨 정책(Policy)을 임베디드 시스템에 적용하고 IMU 센서와 PID 기법을 사용해 균형을 제어한다.(그림 2)

(그림 2) 시뮬레이션 POMDP 학습 / 실제 보행

이동 로봇의 정찰시 주된 목적은 사람(Person) 검출이며, 본 논문에서 사용한 Jetson TX2 임베디드 시스템(Embedded System)은 데스크톱에 비해 전력이 낮고 CPU, GPU, RAM 등 연산 자원이 제한되기 때문에 검출 속도가 낮은 문제가 있다. 검출속도의 문제해결을 위해 1단계 검출(1-Stage Detector) 알고리즘 YOLO-v7[4]과 SSD[5]를 사용하며, 두 알고리즘의 기본 모델(Basic), 경량(Light-weight) 혹은 네트워크(Network) 교체 모델을 사용하고 검출속도(FPS)와 사람 예측 확률을 비교함으로써 성능을 검증한다. 정확도와 속도를 고려하여 모델을 선정하고 실시간 사람 검출 속도를 가속화하기 위해 딥러닝 인퍼런스 최적화 라이브러리인 TensorRT를 사용한다.

2. 실시간 사람 검출 학습 및 성능 개선

2-1. MS COCO 데이터 세트 적용

MS COCO 데이터 셋은 Microsoft에서 객체 탐지(Object detection), 세그먼테이션(Segmentation), 키포인트 탐지(Keypoint detection) 등 컴퓨터 비전 및 기계학습 분야의 Task를 목적으로 게시했다. COCO 2017은 학습(Training) 118,287장, 검증(Validation) 5,000장, 테스트(Test) 41,000장으로 구성된다. 그 중 Train/Val로 구성된 123,287장의 이미지는 886,284의 인스턴스로 구성되었으며 사람은 학습 64,135장, 검증 2,673장 테스트 22,217장이다.[6]

2-2. YOLO-v7과 SSD의 성능 비교

MS COCO 데이터 셋을 사용하여 YOLO-v7[7]과 SSD[8]의 기본 모델을 학습하고, Jetson TX2에서 입력 크기 640*480 카메라로 1분 동안 실시간 예측 확률과 검출 속도를 비교한다. 사람과 카메라 사이 거리는 1m로 사람의 해상도를 유지하였다. YOLO-v7은 사람을 93~97 % 확률로 예측하고 검출 속도는 3.9~4.2FPS이다. SSD는 사람을 87~93% 확률로 예측하고 검출 속도는 5~6.5FPS를 유지하였다. 임베디드 환경에서 사용하기 위해 경량 모델인 Tiny YOLO-v7을 사용해 측정한 결과 74~83% 확률과 15.4~17.6FPS가 나왔고, SSD의 기존 VGG 네트워크를 Mobile Device 전용 Network인 Mobile Net-v2[9]로 교체한 결과 89~95% 확률과 8~8.5FPS를 유지하였다.(그림 3) Tiny YOLO-v7의 경우 속도는 빠르지만 예측 확률이 상당히 낮기 때문에 SSD MobileNet-v2를 선정했고, 정찰하기 위해서는 평균 30FPS을 넘어야 하기 때문에 TensorRT로 최적화한다.

(a) (b) (C) (d)

(그림 3) (a) YOLO-v7, (b) SSD, (c) Tiny YOLO-v7, (d) MobileNet-v2 결과

2-3. 모델과 네트워크 최적화

TensorRT는 Tensorflow에 내장된 TF-TRT와 NVIDIA가 개발한 TRT가 있다. TF-TRT는 Tensorflow 훈련 그래프 중 일부분에서 최적화가 수행되지만 TRT는 그래프 전체적으로 최적화가 수행되기 때문에 더 빠르다. 따라서 TRT 라이브러리를 사용하여 기존 딥러닝 모델의 구조를 개선 후 최적 모델로 변환하고 실행시간에서 구동하여 추론 능력을 향상 시킨다.(그림 4)

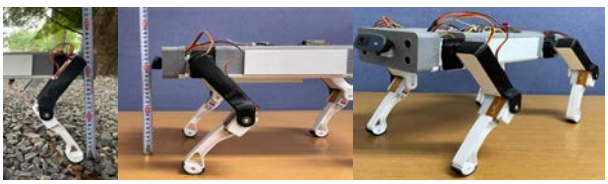
(그림 4) TensorRT engine process[10]

2-4. 학습 환경 및 방법

본 논문은 Tensorflow Object Detection API[11]를 이용하여 SSD MobileNet-v2를 학습한다. 학습에 사용한 사람 이미지는 COCO 데이터 셋을 이용하였고, 학습한 컴퓨터 환경은 Intel Core i7-8700K 3.70GHz, 32GB RAM, Geforce TITAN Xp, Ubuntu 18.04 LTS, OpenCV 4.2이다. 학습 설정은 Momen-tum=0.9, Weight decay=0.0002, Batch Size=32, Learning rate=0.1로 통일하고 Learning rate는 100 Epoch과 150 Epoch일 때 각각 $\frac{1}{10}$씩 감소시켜 총 200 Epoch 진행한다. Iteration은 100k번 당 1회씩 AP(Average Precision)을 측정하고 활성함수(Acti-vation)는 기본 제공된 것을 사용하고 별도로 변경하지 않았다. Best iteration은 3.8M번이고, IoU(Intersection over Union)=0.5 일 때 AP=1이었다. 테스트는 Jetson TX2에서 수행한다.

3. 실험 및 분석

사람을 검출하도록 학습 시킨 TRT SSD Mobile Net-v2[12]은 Jetson TX2에서 45~52FPS를 가진다. 로봇의 카메라 높이는 지상으로부터 21cm이고, 위쪽으로 15도 상향되어 있다.(그림 5)



(그림 5) 로봇의 높이 21cm / 카메라 각도 상향 15도

로봇의 사람을 검출 범위를 알아보기 위해 객체를 1m~5m까지 1m 간격으로 이동시켰다.(그림 6, 7)

(그림 7) 밝을 때 각 위치 별 사람 뒷모습 예측
(a) 1m, (b) 2m, (c) 3m, (d) 4m, (e) 5m

(그림 6) 밝을 때 각 위치 별 사람 앞모습 예측
(a) 1m, (b) 2m, (c) 3m, (d) 4m, (e) 5m

또한 각 위치마다 3,000장씩(=초당 50장씩 1분동안) 관찰하고 검출 확률과 속도의 평균을 측정하였다.

<표 1> 밝을 때 검출 거리에 따른 평균 확률과 속도

	1m	2m	3m	4m	5m
평균확률(%)	47	78	97	98	83
평균속도(FPS)	52.88	49.75	51.63	51.53	49.13

로봇은 3m~4m 거리에 객체가 있을 때 높은 확률로 사람을 예측할 수 있다. 이는 해상도 내에 사람의 전신이 나오면 예측 확률이 높고, 사람의 신체 일부만 나오면 확률이 낮아지는데, 학습에 사용한 COCO 데이터의 사람 비율이 전신이 많은것으로 예상된다. 같은 방법으로 모든 조건은 동일한 상태에서 (그림 8, 9)처럼 채도만 달라졌을 때 확률과 속도를 측정했다.

(그림 8) 어두울 때 각 위치 별 사람 앞모습
(a) 1m, (b) 2m, (c) 3m, (d) 4m, (e) 5m

(그림 9) 어두울 때 각 위치 별 사람 뒷모습
(a) 1m, (b) 2m, (c) 3m, (d) 4m, (e) 5m

<표 2> 어두울 때 검출 거리에 따른 평균 확률과 속도

	1m	2m	3m	4m	5m
평균확률(%)	47	70	86	64	45
평균속도(FPS)	52.99	52.08	52.34	54.32	50.62

어두울 때 사람 검출 시 확률이 현저히 낮아진 것을 확인할 수 있다. 사람의 해상도가 고르게 분포됐던 3m~4m 또한 채도가 낮아서 배경과 사람이 제대로 분리되지 않고 사람의 Pixel 분포가 해상도 내에 선명하지 못하기 때문이다. 그리고 확률이 낮아지면서 평균 속도가 올라갔는데 객체와 배경을 제대로 분리해 내지 못하고 밝은 일부 혹은 사람의 일부 신체만 검출하여 연산하기 때문으로 생각된다. 마지막으로 10m 거리에서 검출한 결과 흰색 배경과 검은색 객체처럼 잘 분리된 경우에는 90%대의 확률을 나타내지만 하얀 배경과 밝은 객체는 40~50%를 검출한다.

4. 결론 및 향후 연구 방향

Jetson TX2 임베디드 시스템에서 SSD의 네트워크를 모바일 디바이스용으로 바꾸고 TensorRT로 최적화 한 결과, 높은 예측 확률과 30 FPS 이상으로 정찰 업무에 사용하기 적합하다. 사람의 전신이 나온 경우 높은 확률로 검출 성공하였으며 사람의 일부만 보이거나 어두운 환경이 되면 예측 정확도가 낮아졌다. 향후 신체의 일부만 나온 데이터 셋을 추가하고 명도, 채도, 회전 등 데이터 증강을 하면 예측 확률이 개선될 것으로 예상된다. 또한 다음 학습 시 활성함수를 SiLU로 교체하여 파인튜닝(Fine-Tuning) 시 가중치(Weight) 값의 손실을 최소화 하는 연구도 시행이 필요하다.

감사의 글

이 논문은 2022년도 정부(교육부)의 재원으로 한국연구재단 기초연구사업의 지원을 받아 수행된 연구임 (No. 2020R1F1A1067496)

참고문헌

[1] 한성민(Seong-min Han), 박명숙(Myeong-suk Pak), 김상훈(Sang-hoon Kim). 2022. 딥러닝을 이용한 비평탄 지형 극복용 4족 보행 지능로봇의 설계에 관한 연구. 한국정보처리학회 학술대회논문집, 29(1): 288-291

[2] M. Rahme, I. Abraham, M. L. Elwin and T. D. Murphey, "Linear Policies are Sufficient to Enable Low-Cost Quadrupedal Robots to Traverse Rough Terrain," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp.8469-8476, doi: 10.1109/IROS51168.2021.9636011.

[3] https://github.com/OpenQuadruped/spot_mini_mini

[4] Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7:Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." arXiv preprint arXiv:2207.02696 (2022).

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg1/UNC Chapel Hill Zoox Inc. Google Inc. University of Michigan, Ann-Arbor - SSD: Single Shot MultiBox

[6] https://cocodataset.org/#explore

[7] https://github.com/wongkinyiu/yolov7

[8] https://github.com/weiliu89/caffe/tree/ssd

[9] https://github.com/tranleanh/mobilenets-ssd-pytorch

[10] "NVIDIA TensorRT Documentation", https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html

[11] https://github.com/tensorflow/models/tree/master/research/object_detection

[12] https://github.com/jkjung-avt/tensorrt_demos
https://github.com/jkjung-avt/tf_trt_models