비디오 Object Detection에서의 연산량 감소를 위한 방법

*김도영 **강인영 ***김연수 ****최진원 *****박구만
서울과학기술대학교
*전기정보공학과
**ITM학과
***컴퓨터공학과
****기계시스템디자인공학과
*****전자미디어 IT 공학과, 교신저자
E-mail: kacel33@naver.com

Method for reducing computational amount in video object detection

*Do-YoungKIM , **In-YeongKang, ***Yeonsu-Kim,****Jin-WonChoi,
*****Goo-manPark
Seoul National University of Science and Technology
*Dept. of Electronic and Information Engineering
**Dept. of Information Technology Management for Business
***Dept. of Computer Science and Engineering
****Dept. of Mechanical System Design Engineering
******Dept. of Electronic and IT Media Engineering

요 약

현재 단일 이미지에서 Object Detection 성능은 매우 좋은 편이다. 하지만 동영상에서는 처리 속도가 너무 느리고 임베디드 시스템에서는 real-time이 힘든 상황이다. 연구 논문에서는 하이엔드 GPU에서 다른 기능 없이 YOLO만 구동했을 때 real-time이 가능하다고 하지만 실제 사용자들은 상대적으로 낮은 사양의 GPU를 사용하거나 CPU를 사용하기 때문에 일반적으로는 자연스러운 real-time을 하기가 힘들다. 본 논문에서는 이러한 제한점을 해결하고자 계산량이 많은 Object Detection model 사용을 줄이는 방안은 제시하였다. 현재 Video영상에서 Object Detection을 수행할 때 매 frame마다 YOLO모델을 구동하는 것에서 YOLO 사용을 줄임으로써 계산 효율을 높였다. 본 논문의 알고리즘은 카메라가 움직이거나 배경이 바뀌는 상황에서도 사용이 가능하다. 속도는 최소2배에서 ~10배이상까지 개선되었다.

키워드: Object Detection, Object Tracking, Video

1. 서론

Video영상을 처리할 때 Image처리에서와 동일하게 모든 frame마다 Object detection을 수행한다.

그래서 비디오를 real-time으로 재생하는 것의 여부는 Object detection 모델이 얼마나 빠른 속도로 동작하느냐에 따라 달렸다. R-CNN[1]의 등장 이후로, 속도를 개선한 Faster R-CNN[2]이 나왔다. 그리고 Yolov1[3]의 등장 이후로 속도와 정확도를 개선한 YOLO Series[4, 5, 6, 7]가 발표되었다. 하이엔드 GPU에서 YOLO를 이용하면 frames per second(FPS)가 30인 real-time을 넘어서 FPS 100이상의 속도로 object detection을 수행할 수 있게 되었다. 하지만 임베디드 시스템이나 cpu만 있는 환경에서는 속도가 느려서 real-time이 힘든 상황이다. Nvidia에서 출시한 임베디드 시스템 보드 중 하나인 jetson nano에서는 Yolov4를 사용했을 때 fps가 2~3, Yolov4-tiny에서는 fps가 10~12 정도가 나온

다.

이러한 제약을 극복하기 위해서 본 논문에서는 frame skip 알고리즘을 제안했다. 본 논문의 알고리즘은 카메라가 움직이거나 배경이 바뀌는 상황에서도 사용이 가능하다.

2. 관련 연구

2.1 Object detection

Object detection은 Multi-labeld Classification 과 BoundingBOX Regression(Localization) 문제라고 할 수 있다. 초기에는 R-CNN[1], Faster R-CNN[2]와 같은 2-stage모델이 제안되었는데, 2-stage모델은 Localization과 Classification을 순차적으로 하는 방식이다. 이후에 Classification과 Localization을 동시에 하는 1-stage방식의 모델이 제안되었는데 그것이 YOLO[3]이다. 이 때 R-CNN모델에 비해서 속도가 크게 향상되었다. 이후에 YOLO의 여러 버전들[4, 5, 6, 7]이 출시되면서 속도와 정확도가 크게 향상되어 비디오 영상에서도 real-time으로 detection을 할 수 있게 되었다.

2.2 Object tracking

비디오 분야에서는 전의 frame과 현재 frame을 비교하여 object를 tracking하는 연구가 진행되고 있다. SORT[8]와 SORT를 deep learning으로 개량한 Deep-SORT[9]는 모든 frame마다 object detection을 수행하고 전의 frame의 boundingBOX에서 칼만 필터를 적용하여 현재 frame에서의 위치를 예측하고 현재 frame에서 detection된 boundingBOX와 비교하여 물체를 tracking한다. 여기에서 tracking을 하는 이유는 물체를 detection하고 난 뒤에 고유 id를 만들고 tracking하여 id를 유지하는 것이다.

3. 알고리즘

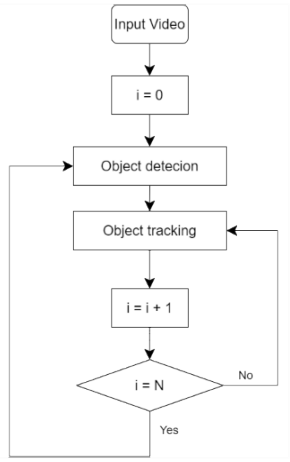

그림1 알고리즘 구현도

본 논문의 알고리즘은 간단하다. 사용자가 YOLO frame을 얼마나 skip할지 지정하고 그것을 N이라고 하면 $kN+1(k=0,1,2\cdots)$번 째 frame에서 YOLO를 구동하고, 나머지 frame에 대해서는 YOLO에서 Detection한 boundingBOX를 ROI로 지정하여 tracker함수로 추적하는 것이다. 사용자가 N를 지정할 수 있으므로 장면이 많이 바뀌는 경우, 새로운 물체가 자주 나타나는 경우에는 N을 작게 하고 화면에 변화가 거의 없는 경우는 N을 크게 하면 된다. 이 알고리즘은 YOLO을 이용하여 처리하던 frame을 상대적으로 가벼운 tracker함수로 처리하기 때문에 임베디드 시스템이나 GPU가 없는 환경에서 큰 효과를 볼 수 있다.

3.1 frame마다 다르게 처리하는 방법

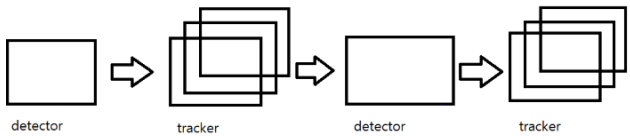

그림 2 frame별 영상 처리 과정

본 논문의 대략적인 구조는 그림2와 같다.

본 논문에서는 계산량이 많은 detector frame을 줄이고 tracker frame으로 경량화 하는 것을 목표로 한다. 처음 생각한 tracker아이디어는 Background Subtraction이었는데 배경이 자주 바뀌거나 카메라가 움직이는 등의 상황에서는 완전히 무력한 알고리즘이라서 다른 알고리즘을 찾아본 결과 opencv tracker함수로 변경하였다. opencv tracker함수는 사용자가 Region if Interest(ROI)를 지정하면 그 영역 안에 있는

물체를 tracking해주는 라이브러리이다. 여기서는 사용자가 직접 ROI를 지정해주는 것이 아닌 YOLO frame에서 얻은 ROI를 바탕으로 tracker기능을 수행한다. 초기에는 detector frame에서 tracker frame으로 넘어간 후에 tracker frame에서 큰 변화가 있을 때에만 detector frame으로 넘어가는 방법으로 진행하였지만, 카메라가 움직이면서 촬영한 영상 같은 경우에는 영상이 shift되면서 변화를 측정하기가 어려운 단점이 있었다.

따라서 좀 더 간단하고 일반적인 알고리즘을 사용하기 위해 $kN+1(k=0,1,2\cdots)$번 째 frame마다 detection을 수행했다.

3.2 구 현

본 논문에서 추가적으로 개발한 알고리즘은 detector frame과 tracker frame을 구분하여 영상처리 하는 것인데 detector frame에서는 YOLO를 tracker frame에서는 OpenCV tracker 함수의 종류인 MOSSE또는 CSRT함수를 사용하는 것이다. MOSSE tracker는 상대적으로 정확도가 낮지만 빠른 속도를 가지고 CSRT tracker는 속도는 느리지만 매우 높은 정확도를 가지고 있다.

여러 개의 물체를 tracking	Cv2.legacy.MultiTracker_create()
물체 tracking	Cv2.legacy.TrackerMOSSE_create() Cv2.legacy.TrackerCSRT_create()
Object Detection	YOLO(cv2.dnn.readNET)

표 1. 활용한 공객 라이브러리 주요 API

4. 실험 결과 및 분석

4.1 실험 환경 사항

실험 환경은 윈도우OS에서 아나콘다로 설치된 Python3.8버전과 Opencv-python 4.5.0버전을 사용하였다. 데이터셋은 일반적인 영상으로 실험하기 위하여 유튜브를 통해 다운받아 실험하였다[10]. CPU는 i9-9900KF를 사용하였다.

4.2 실험 결과

현재 환경에서 일반적으로 YOLO는 frame 당 0.8초가 걸렸고, MOSSE tracker는 frame당 0.01초가, CSRT tracker는 frame 당 0.08~0.2초가 걸렸다.

그림 3 모든 frame에 YOLO를 사용한 기존의 방식

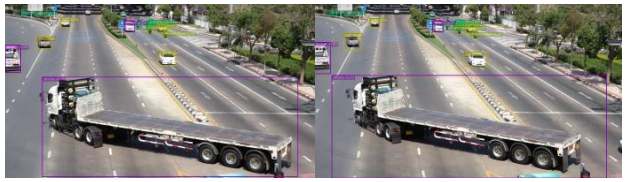

그림 4 yolo를 사용한 뒤에 opencv tracker함수를 이용하여 tracking한 방식

	원래 영상	YOLO	YOLO+MOSSE ()는YOLO/YOLO+MOSSE	YOLO+CSRT ()는 YOLO?YOLO+CSRT
N=4	8초	219초	65.2초(3.36)	86.7초(2.53)
N=10			31.2초(7.02)	51.3초(4.27)
N=50			14.5초(15.1)	36.4초(6.02)
N=100			13.3초(16.5)	38.2초(5.73)

표 2. YOLO, MOSSE, CSRT 각 알고리즘 별 계산 속도

한 동영상에서만 비교한 숫자이지만 다른 영상에서도 detector frame에서는 0.8초(YOLO), tracker frame에서는 일반적으로 0.008~0.02초(MOSSE), 0.08~0.21(CSRT)의 속도가 나왔다.

하지만 물체가 많은 경우(10개 이상) MOSSE에서는 크게 속도 저하가 없었지만, CSRT 알고리즘은 frame당 0.4초로 느려졌다.

실험 환경은 배경이 바뀌는 경우, 카메라가 움직이는 경우, 물체가 많은 경우, 물체가 적은 경우로 나눠서 수행하였다. tracker함수를 이용했기 때문에 배경이 바뀌는 경우와 카메라가 움직이는 경우에도 다른 영상과 차이가 없었다. 그리고 물체가 적은 경우에도 MOSSE는 추적하는데 오차가 있었지만 CSRT의 경우에는 오차가 거의 없었다. MOSSE의 경우에는 n을 낮춰서 오차를 줄일 수 있었다.

하지만 물체가 많은 경우(10개 이상)에는 CSRT를 수행할 때 시간이 너무 많이 걸려서 frame당 처리

시간이 0.4초로 늘어났다.

새로운 물체가 나타났을 경우 다음 detector frame으로 되기 전까지는 물체 감지가 불가능하다는 단점도 있었다.

비디오 영상의 특성상 정확도가 얼마나 떨어졌는지는 정량적으로 판단할 수가 없다고 생각하여 주관적으로 진행하였다. MOSSE는 n=50, 100인 경우에 정확도가 상당히 많이 떨어진 반면 n=4, 10인 경우에는 비교적 정확하게 tracking하였다. CSRT는 n=100인 경우에도 정확도 높게 추정하였다.

4.3 실험 결과 분석

N=4,10,50,100으로 실험하였고, OpenCV Tracker 함수가 여러 개 있었지만 계산 속도가 빠른 MOSSE와 계산 속도는 상대적으로 느리지만 정확한 CSRT를 사용하였다. 실제로 YOLO같은 Object Detection 논문에서도 하이엔드 GPU를 사용하고 높은 fps를 달성하여 real-time이 가능하다고 쓰이는 것을 많이 볼 수 있었다. 단일 이미지에서는 Object Detection이 성능이 좋고 굉장히 효과적이지만 서로 연관성 있는 여러 개의 frame이 있는 비디오에서는 전의 frame의 정보를 사용하는 것이 상당히 괜찮은 아이디어라고 생각한다.

N=10이하에서 MOSSE tracker는 괜찮은 성능을 보여주었지만 N이 커질수록 오차는 커졌다. CSRT tracker는 MOSSE보다 계산 속도가 10배이상 느리지만 결과가 더 좋았다.

딥러닝에서는 계산 처리 속도가 10%만 향상되어도 논문에 쓰이는데 간단한 트릭으로 계산 속도를 2배로 늘릴 수가 있었다. 처음에 그냥 YOLO를 홀수번째 frame에서만 돌린 경우에도 모든 영상에서 오차를 발견하기 힘들었다.

N=10이고 MOSSE를 사용한 경우에 9.8배 정도의 계산 속도 향상이 있었고, CSRT를 사용한 경우 3배 정도의 계산 속도 향상이 있었다.

본 논문에서는 tracker frame에서 Deep Learning 방식이 아닌 OpenCV함수를 사용하였는데 tracker 함수를 사용하지 않고, 이전 frame과 현재 frame을 input으로 하고 Detector에서 Detection된 영역을 deep learning 방식으로 tracking하는 방법으로 개선하면 정확도가 크게 향상될 것이라고 기대한다.

Acknowledgement

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00751, 0.5mm 급 이하 초정밀 가시·비가시 정보 표출을 위한 다차원 시각화 디지털 트윈 프레임워크 기술 개발)

참고문헌

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[2] R. Girshick, "Fast R-CNN," in IEEE International Conference on Computer Vision (ICCV), 2015.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In CVPR, 2016.

[4] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In CVPR, 2017.

[5] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[6] Alexey Bochkovskiy, Chien-Yao Wang, and HongYuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[7] glenn jocher et al. yolov5. https://github.com/ultralytics/yolov5, 2021

[8] A. Bewley, G. Zongyuan, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in ICIP, 2016,pp. 3464- 3468.

[9] Nicolai Wojke, Alex Bewley, Dietrich Paulus. 2017.Simple Online and Realtime Tracking with a Deep Association Metric.

[10] YouTube. (2019,April 26). Raw Run || 70 mph in Switzerland - YouTube