creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사 학위논문

다중 카메라 환경에서의 CNN 기반 물체 검출 알고리즘의 속도 향상을 위한 ROI 추출 및 처리 방법

An ROI extraction and processing method to speed up a CNN based object detection algorithm in a multi-camera environment

2019 년 2 월

서울대학교 대학원

전기 정보 공학부

이 유 학

초 록

심층학습이론의 발전과 함께 많은 분야에서 심층 학습을 응용하기 위한 사례들이 생겨났다. 그 중 CNN기반 물체 검출 알고리즘을 이용한 대표적인 예시는 Amazon社의 무인 마트인 Amazon Go가 있다. 여러 대의 카메라의 입력을 실시간으로 처리해주는 기술은 무인 마트 설립을 위한 초기 비용 감소를 위해 필수적이다.

최근 CNN기반 물체 검출 알고리즘이 기존 물체 검출 알고리즘에 비해 좋은 물체 검출 성능을 보여주었다. 초기 CNN 기반 물체 검출 알고리즘은 먼저 관심 영역(Region of Interest, ROI)을 찾고 이 ROI들을 입력 받아 물체를 분류하고 물체의 위치를 추정하였다. 하지만 두 단계로 나뉘어져 있는 점에서 실시간 처리에 한계를 보였다. 이후에 단일 단계 검출기인 SSD, YOLO등 단일 이미지를 실시간 처리할 수 있는 CNN기반 물체 검출 알고리즘이 등장했다. 이들은 속도 대비 물체 검출 성능에서 우수함을 보였다. 하지만 그 속도를 개선한 CNN 기반 물체 검출 알고리즘도 단일 이미지만을 실시간으로 처리 가능했다. 이는 저전력 장치를 사용해야하는 상황 또는 여러 대의 카메라로부터 전달되는 영상을 실시간으로 처리하기에는 무리가 따른다.

본 논문에서는 CNN기반 물체 검출 알고리즘의 속도를 향상시키기 위한 새로운 방법을 제안한다. 카메라에서 전달되는 영상에서 ROI를 추출한 뒤 이를 기존 CNN 기반 물체 알고리즘의 입력으로 전달하여 처리하는 방법이다. 더 나아가서 단일 영상에서 ROI를 추출하여 처리하는 방식은 속도 향상이 제한 됨을 밝힌다. 이는 다중 카메라를 사용해야하는 무인 편의점과 같은 상황에서 큰 문제를 야기한다. 본 논문은 다중 카메라 환경에서 ROI를 병합하는 것을 통하여 해결책을 제안한다. 결론적으로 기존 CNN 기반 물체 검출 알고리즘(YOLOv3)

대비 평균 1.75배의 속도 향상이 있었으며 정확도는 비슷한 수준을 유지하였다.

주요어 : 물체 검출, 합성곱 심층 신경망, 관심 영역, 다중 카메라, 무인 편의점, 루프 최적화
학 번 : 2017-22343

목 차

표 목차

그림 목차

제 1 장 서 론

최근 무인 편의점의 등장은 큰 관심을 불러 일으켰다[1,2]. 대표적으로 Amazon社에서 개장한 Amazon Go가 그 예시이다[2] [그림 1]. 무인 편의점에 필요한 핵심 기술 중 하나는 카메라에서 전달된 영상에서 물체를 정확하고 빠르게 검출해 내는 기술이다. 이러한 수요에 기인하여 이전부터 컴퓨터 비전 분야에서는 물체 검출 알고리즘에 관한 연구를 활발히 진행해 왔다. 본 장에서는 관련 연구들을 소개한 뒤, 한계점을 분석하고 본 논문에서 새로운 접근을 통해 기존 연구 대비 속도향상을 이룬 점을 간략히 서술한다.

[그림 1] Amazon go 매장 모습

1.1 관련 연구

1.1.1 지역 기반 CNN활용 물체 검출 알고리즘

CNN(Convolutional Neural Network)기반 물체 검출 알고리즘의 정확도는 그것이 등장하기 이전 물체 검출 알고리즘의 정확도를 뛰어넘었다. 초기 CNN기반 물체 검출 알고리즘은 지역 기반 CNN이라 불리며 두 단계로 나뉘어져 있었다[3]. Selective Search[4], CNN등을 이용하여 ROI(Region of Interest)를 추출해 내는 앞부분과, 이 ROI를 입력으로 받아 물체를 분류하고 물체의 위치를 박스로 예측하는 뒷부분으로 구성되어 있었다. 대표적인 알고리즘으로는 Faster R-CNN이 있다. Faster R-CNN 이후에도 이러한 지역 기반 CNN을 활용한 물체 검출 알고리즘의 여러 변형된 형태가 나왔다[5,6]. 그 중 R-FCN[5]은 지역 기반 CNN을 활용한 물체 검출 알고리즘 중에서 가장 빠른 속도를 보여주었다[7,8]. 하지만, 이러한 지역 기반 CNN은 단일

이미지에서 ROI를 추출하기 힘들기 때문에 앞부분에서 연산량을 많이 필요로 했고, 추출된 ROI를 입력으로 받아 물체를 검출해야하는 구조적 한계로 인하여서 실시간으로 물체를 검출하기 위해 사용하기에는 어려움이 있었다[7,8].

1. 1. 2 단일 단계 CNN기반 물체 검출 알고리즘

지역 기반 CNN의 속도 측면에서의 한계를 극복하기 위하여 SSD[9]나 YOLO[10]같이 단일 단계 CNN기반 물체 검출 알고리즘이 등장했다. 이는 이미지를 한 알고리즘에서 end-to-end로 처리하는 방식이다[9,10]. 단일 단계 CNN기반 물체 검출 알고리즘의 특징은 지역 기반 CNN 대비 처리 속도는 빠르지만 정확도는 떨어진다는 것이다. 예를 들어 SSD의 처리 속도는 R-FCN보다 빠르지만 정확도는 R-FCN보다 떨어진다[5]. RetinaNet은 SSD에 중심 손실(focal loss)과 특징 피라미드(feature pyramid)를 적용한 단일 단계 CNN기반 물체 검출 알고리즘이다[6]. 이 알고리즘의 정확도는 상당히 개선되었으며, 속도는 SSD와 비슷하다[7,8]. 단일 단계 CNN기반 물체 검출 알고리즘 중 YOLO는 속도와 정확도 측면에서 가장 성능이 좋은 알고리즘 중에 하나이기 때문에 이 YOLO의 세번째

버전인 YOLOv3를 본 연구의 기본 알고리즘으로 선정하였다[8].

CNN기반 물체 검출 알고리즘의 연산 속도는 고성능 GPU를 이용하여 측정되었다[7,8]. 한 예로 Nvidia社 Titan X GPU를 사용하는 환경에서 YOLOv3의 동작 속도는 22ms이며 정확도로는 mAP-50 51.5를 보여주었다[8]. 그러나 무인 편의점은 다수의 카메라들로부터 전달되는 영상에서 실시간으로 물체를 검출하는 것이 요구된다. 만일 YOLOv3를 이용하여 3개의 카메라로부터 전달되는 영상을 처리하기 위해서는 위와 같은 GPU를 사용하는 환경에서 66ms가 소요된다. 이는 기존 YOLOv3를 이용하여서는 다중 카메라 환경에서 실시간으로 물체를 검출하지 못한다는 것을 의미한다. 또한 고성능 GPU를 쓸 수 없는 제한된 상황에서 물체 검출을 실시간으로 하기 위해서는 지금의 단일 단계 CNN기반 물체 검출 알고리즘의 개선이 필요하다.

1. 1. 3 저해상도 ROI를 이용한 단일 단계 CNN기반 물체 검출 알고리즘의 속도 향상

동영상에서 단일 단계 CNN기반 네트워크의 속도향상을 위한 노력 중 하나는 PaD[11]로 소개된 방식이다. 이 방식은 동영상의 시간적 공간적 정보를 이용하여 간단하게 ROI 추출을 한다. 이 추출된 ROI를 특정 크기의 이미지로 압축하여 SSD[9]의 입력으로 전달한다. 이를 통해 연산량을 줄여 기존 SSD의 속도향상을 얻을 수 있었고 인식률 하락은 미미한 수준임을 보여준다[11].

1.2 연구 내용

본 논문에서는 우선 다중 카메라를 이용한 무인 편의점 환경을 구성하고 그 환경에서 무인 편의점 데이터 셋을 제작하고 사용하였다[그림 2]. 그 후, ROI를 동영상에서 간단하게 추출하고 추출된 ROI를 이용하여 YOLOv3에 적합한 입력으로 전달하여 YOLOv3 내에서 이 ROI입력을 적절히 처리하는 방식을 구현하였다. COCO dataset testset[12]에서 확인한 결과 인식률은 mAP-50 기준 2.81%의 하락을 보여주어 기존 YOLOv3와 유사한 수준임을 확인하였다. 이때 속도는 ROI 입력 크기에 따라 선형적으로 변화하는 것을 확인하였다. 더 나아가 초기 YOLOv3기반 단일 영상에서 ROI를 추출하여 처리하는 모델에서 중요한 사실을 발견하였다. 단일 카메라로부터 전달된 영상에서 추출된 ROI를 이용하여 YOLOv3의 입력을 구성할 때 속도 향상의 한계가 있음을 확인했다. 이를 통해 무인 편의점과

같이 다중 카메라로부터 영상을 전달 받는 상황에서 실시간으로 물체를 검출하기 위해서는 다중 카메라로부터 전달된 영상의 ROI를 병합하여 한번에 처리하는 것이 더 유리함을 수식적으로 제시하고 이를 실험적으로 검증하였다. 결론적으로 기존 YOLOv3 네트워크 대비 처리 속도는 평균 1.75배 향상되었으며 무인 편의점 데이터 셋에서 정확도는 비슷한 수준임을 확인하였다.

논문은 다음과 같이 진행된다. 제 2장에서는 기존 연구들의 한계점을 분석하고 해결 방안의 효용성을 수식적으로 검증한다. 제 3장 본론에서는 ROI를 추출하고 처리하는 방식에 대해 자세히 소개한다. 제 4장에서는 관련된 실험결과를 제시한다. 마지막으로 제 5장에서는 결론을 내며 논문을 마무리한다.

[그림 2] 연구에서 사용한 무인 편의점 환경

제 2 장 문제 정의 및 해결 방안

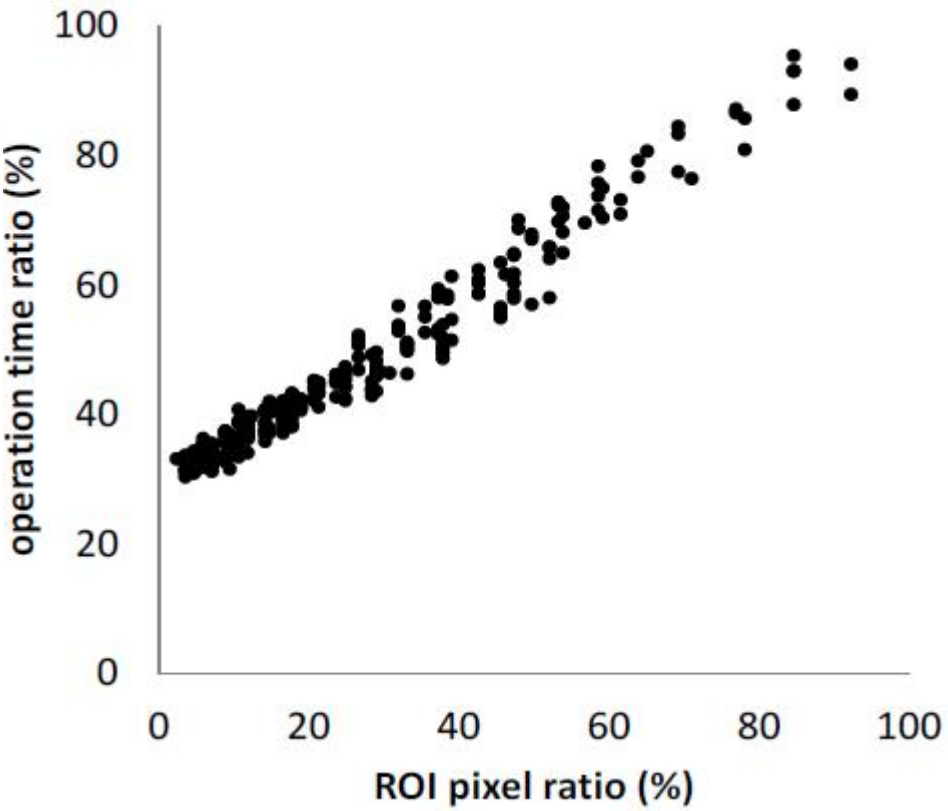
본 장에서는 제 1장에서 살펴본 지금까지 연구에서의 문제점을 검토하고 그 대안을 제시한다. 먼저 2.1절에서는 단일 이미지에서 ROI를 추출하여 처리하는 경우 ROI입력의 크기가 기존 영상의 크기 보다 충분히 작을 경우에도 한계가 있음을 밝히고 그 원인을 분석한다. 2.2절에서는 2.1절에서 밝힌 한계를 극복할 수 있는 방법을 제안하고 수식적으로 그 효과를 예측한다.

2.1 단일 ROI 처리 방식의 한계 및 원인 분석

최근 단일 단계 CNN 기반 물체 검출 알고리즘의 속도를 더욱 향상시키기 위한 시도가 있었다[11]. 동영상 환경에서 시간적, 공간적 정보를 활용하여 간단하게 저해상도 ROI를 추출한 뒤, 이를 SSD의 입력으로 전달하여 처리하는 방식이다. 본 논문에서도 이와 유사한 방식으로 저해상도 ROI를 추출한 뒤, 이를 YOLOv3의 입력으로 전달하여 처리하는 방식을 구현하였다. 그 결과 [그림3]과 같은 중간 실험 결과를 얻을 수 있었다. 기존 이미지 대비 추출된 ROI를 이용하여 구성한 입력 픽셀의 크기가 아무리 작아도 해당 입력이 처리되는 시간에는 한계가 있다는 것을 발견하였다. [11]에서는 고정된 크기의 저해상도 이미지로만 압축했기 때문에 발견하지 못한 결과이다.

이 현상의 원인은 CNN 기반 물체 검출 알고리즘이 동작하는 방식에서 찾을 수 있다. CNN 기반 물체 검출 알고리즘을 처리할

때 CPU는 전체적인 알고리즘의 진행 또는 ROI 추출 등의 작업을 수행한다. GPU는 CPU 에서 특정 layer의 연산을 수행하도록 요청이 오면 이 요청을 수행한 뒤 완료된 결과를 다시 CPU에 전달해준다. 이때 ROI 추출을 통해 구성된 입력의 픽셀(pixel) 수가 너무 작을 경우 두가지 문제가 발생한다. 첫째 문제는 처리할 픽셀이 GPU의 쓰레드(thread) 수보다 적으면 픽셀에 할당 되지 못한 쓰레드들이 발생하고 이는 한 쓰레드의 연산 latency를 다른 쓰레드의 연산 시간으로 충분히 숨겨주지 못하는 결과를 발생시키는 데에 있다. 다른 한 가지 문제는 적은 수의 픽셀을 GPU에서 처리하는 시간이 CPU가 다른 작업을 처리하는 데에 소요되는 시간보다 짧기 때문에 GPU와 CPU가 PCI 통신을 하게 될 때 GPU가 기다리게 되는 현상이 발생하는 데에 있다.



[그림3] 픽셀 비율 대비 YOLOv3 동작 시간 비율 그래프

2. 2 개선 방향 및 기대효과

2.1절에 소개된 것과 같이 단일 이미지에서 ROI를 추출하여 처리하는 방식에는 한계가 있음을 알게 되었다. 이는 다중 카메라 환경에서 전달되는 영상을 GPU가 연속적으로 처리할 경우 속도 향상의 한계가 있음을 보여준다. [그림3]을 다음과 같은 수식으로 선형 근사할 수 있다.

$$t = ax + b$$

(t는 소요 시간, x는 ROI 입력 크기), (a, b 는 상수)

이때, N개의 카메라에서 영상을 전달하는 주기 별로 매번 N개의 영상이 들어온다고 하고, 그 N개의 영상에서 추출된 ROI를 이용한 입력의 크기를 각각 x1, x2 … xN이라고 하자. 이때 이 입력들을 순차적으로 처리했을 때 총 소요되는 시간은 다음과

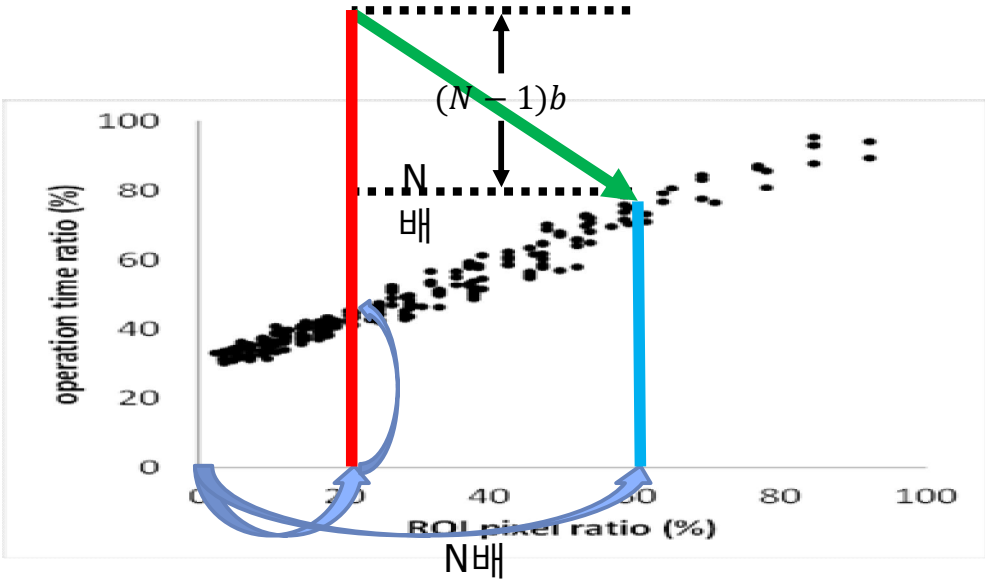
같다.

$$a(x1+x2+\cdots+xN) + N*b$$

만일 각 카메라로부터 오는 ROI를 병합하여 처리할 경우 총 소요되는 시간은 다음과 같다.

$$a(x1+x2+\cdots+xN) + b$$

즉, 기존 방법 대비 $(N-1)*b$ 만큼의 시간을 단축할 수 있다. 이는 [그림 4]을 통해서도 확인할 수 있다.



[그림4] ROI병합을 통한 성능향상 개념도

제 3 장 본 론

본 장에서는 ROI를 추출하는 방법, 이 추출된 ROI를 YOLOv3의 입력으로 구성하는 방법, 그리고 이 ROI입력을 YOLOv3에서 적절하게 처리하는 방식에 대해서 소개한다. 마지막으로 여러 대의 카메라로부터 오는 영상에서 추출된 ROI를 병합하여 처리하는 전체적인 흐름에 대해서 서술한다. 본 장의 적용 결과는 제 4장에서 자세하게 기술한다.

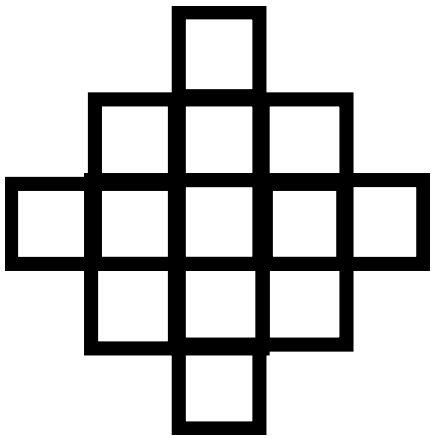
3.1 제안 방법의 ROI 추출 방식

제안 방법에서는 무인 편의점 환경을 구동하고 카메라에서 전달되는 첫 영상을 참조 프레임을 선정하여 4x4 픽셀 단위로 최대값 행렬과 최소값 행렬을 저장해 둔다. 이는 계산을 빠르게 해주며 빛의 간섭이나 카메라 노이즈로 인해 ROI가 잡히는 것을 방지해준다.

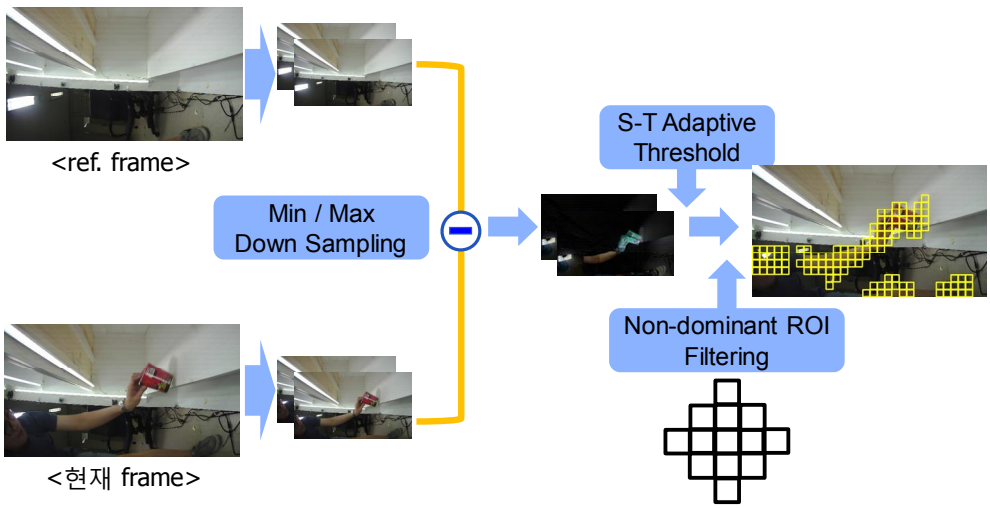
이후 카메라에서 전달되는 현재 프레임도 4x4 픽셀 단위로 최대값 행렬과 최소값행렬을 저장해 둔다. 그 뒤 최대값 행렬 간의 차와 최소값 행렬 간의 차이 영상을 구한다.

차이 영상의 절대값이 주어진 임계 값을 넘는 여부에 따라서 해당하는 영역이 ROI인지 아닌지를 판단하게 된다. 이때 두가지 추가적인 알고리즘이 사용이 된다. 첫째는 공간적, 시간적 정보를 이용하여 적응적으로 임계 값을 결정하는 알고리즘이다. 만일 차이 영상 내 어떤 영역에서 이전 주기에도 해당 영역이 ROI로

선정되거나 해당 영역의 주변 영역이 ROI로 선정된 경우 임계값은 낮아진다. 이때, 차이 영상에서 4x4픽셀 단위로 ROI로 추출된 영역이 있는지 살펴보고 있다면 그 영역을 “ROI블록”으로 표시를 해둔다. (즉, ROI블록은 원본 영상에서 16x16 크기이다.) 두번째는 [그림5]와 같이 제작한 필터를 이용하여 특정 ROI블록에서 주변에 ROI 블록으로 선정된 영역이 있는지 살펴본다. 제작한 필터는 작은 물체를 놓치지 않으면서 중요하지 않은 ROI를 제외하기 위해 만든 필터이다. 만일 필터 내에 ROI블록으로 선정된 영역이 2개 이하라면 해당 ROI블록을 중요하지 않은 ROI블록이라고 판단하여 최종 ROI 추출 결과에서 제외한다. 전체적인 흐름은 [그림 6]에서 확인할 수 있다.



[그림5] 중요 ROI블록 판단을 위한 필터



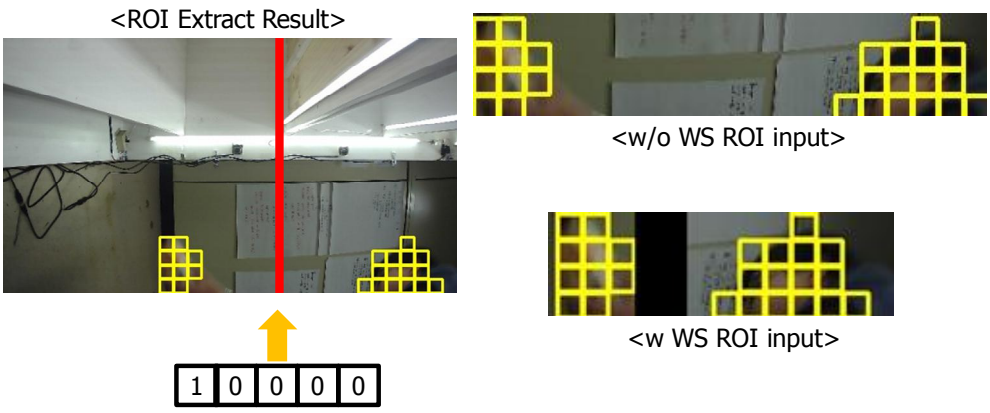
[그림6] ROI 추출 과정의 전체적인 흐름도

3. 2 제안 방법의 ROI 입력 구성 방식

본 절에서는 3.1절에서 추출한 ROI를 YOLOv3의 입력으로 구성하는 방법에 대하여 소개한다. 가장 간단한 방법은 한 영상에서 추출된 ROI블록을 전부 포함하는 최소 크기의 직사각형으로 YOLOv3의 입력을 구성하는 방법이다. 하지만 이 방법은 추출된 ROI 분포가 여러 개로 나뉘어져 있을 경우, ROI로 추출되지 않은 부분이 직사각형 ROI입력에 많이 포함되는 현상이 발생한다.

이를 개선하고자 너비 분할(width slice)이라는 방법을 본 논문에서 제안한다. 이는 ROI 분포가 너비 축으로 나뉘어져 있다고 판단 될 경우, 이미지를 나눈다. ROI 분포가 너비를 축으로 나뉘어 져 있는지 여부는 너비 축 길이의 1차원 배열을 통해서 쉽게 파악할 수 있다. 해당 너비 축에 ROI가 선정이 되어 있는 것이 있다면 1로 그렇지 않다면 0으로 표시한다. 그 후

[그림 7]에 있는 필터를 이용하여 1, 0, 0, 0, 0과 같은 패턴이 있는지를 살펴본다. 만일 그런 패턴이 있으면 너비 축으로 봤을 때 이전까지는 존재해오던 ROI가 더 이상 분포하지 않는다는 의미이므로 이미지를 해당 위치에서 잘라내게 된다. 그 뒤 각 나뉘어진 이미지에서 추출된 ROI블록을 전부 포함하는 직사각형들을 찾고 다시 이 직사각형들을 이어서 YOLOv3의 입력으로 구성하는 방식이다. 이 너비 분할 방식을 통해 YOLOv3 입력으로 들어오는 직사각형에서 ROI로 추출되지 않은 부분의 비율을 줄일 수 있다[그림 7].



[그림 7] ROI입력 구성 방식 비교: 너비 분할 방식의 적용 유무

3. 3 제안 방법의 ROI 입력 처리 방식

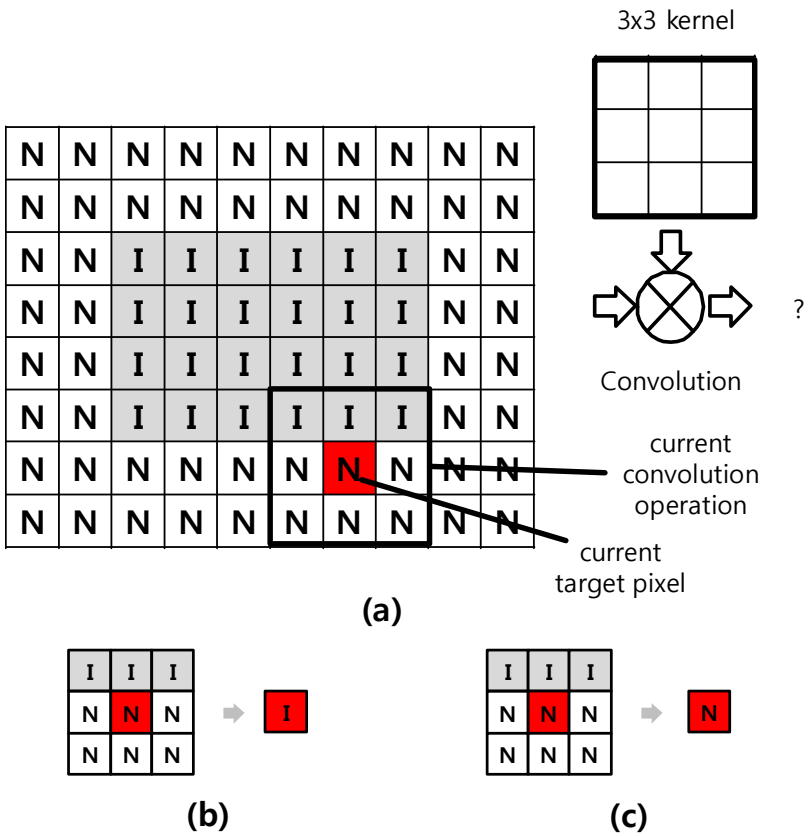
본 절에서는 YOLOv3 의 연산 속도를 높이기 위하여서 3.2절에서 구성한 ROI입력을 처리하는 방식을 소개한다. 직사각형 형태의 ROI입력을 YOLOv3가 전달받게 되면, 이를 처리하는 데에는 여러가지 선택사항이 존재하게 된다. 또한 이 ROI를 처리하는 방법에 따라 정확도에 영향을 줄 수 있다. CNN을 각 layer별로 처리하게 되면서 나오는 결과는 layer를 거듭할수록 수용 영역(receptive field)이 늘어나기 때문이다. 특히 ROI입력의 가장자리 처리 방식에 대한 고려가 필요하다. 아래는 YOLOv3를 구성하는 layer별로 어떻게 ROI를 처리하였는지 소개한다.

3. 3. 1 합성곱 layer(convolutional layer)에서의 ROI 처리

속도 향상을 이루는 주된 방법은 합성곱 layer의 연산량을 줄이는 것이다. YOLOv3 네트워크를 구성하는 107개의 layer중에서 합성곱 layer가 차지하는 수가 단일 종류 layer로는 가장 많은 75개이고 또한 합성곱 layer 자체에서 많은 연산이 이루어지기 때문이다.

ROI입력이 YOLOv3로 전달되면 합성곱 layer은 이 ROI입력을 다양한 방법으로 처리할 수 있게 된다. 예를 들어 합성곱 layer의 필터 사이즈가 3x3 일 경우 필터 내에 ROI와 RONI가 혼재 되어 있을 경우 그 결과 값을 ROI로 취급을 해야하는지 RONI로 취급을 해야하는 지 결정을 해주어야한다([그림8](a)). 만일 ROI 가 한 개라도 존재하면 그 합성곱 연산의 결과를 ROI로 판별하는 경우를 TH0로 정의하였다([그림 8](b)). 또한 필터 내에 ROI가 3개 보다 많이 존재하면 결과값을 ROI로 처리하고 3개 이하로 ROI가 존재하면

RONI로 처리하는 방법을 TH3으로 정의했다([그림 8](c)). [표 3-1]을 보면 TH3에서 false positive 인 경우와 false negative인 경우가 적게 발생하는 것을 확인하였고 연산량 또한 TH3이 적기 때문에 TH3을 제안 방법의 처리 방식으로 채택하였다. 이때 RONI가 필터 내에 있을 때 합성곱 연산 시 그 값을 정의해야 한다. 이는 0으로 처리를 했다 이는 기존 YOLOv3에서 경계 부분에서 합성곱 연산을 진행할 때 사용하는 방식과 동일하다.



[그림 8] 합성곱 연산 결과값에 대한 ROI/RONI 처리 과정: (a) ROI 예시와 합성곱 연산, I는 ROI를 표시하고 N은 RONI로 표시한다. (b) TH0 일 때 ROI처리 결과값 (c) TH0 일 때 ROI처리 결과값

Threshold Type		TH0	TH3
Detection Result		**Number of Objects**	
True positive		62	66
Degradation	False positive	2	0
	False negative	7	2
Improvement	False positive	0	1
	False negative	2	3

[표 3-1] TH0, TH3 방식 비교

3. 3. 2 다른 layer에서의 ROI 처리

아래는 합성곱 layer 이외의 다른 layer에서 어떤 방식으로 ROI를 처리 했는지 짧게 소개한다.

Residual layer(shortcut layer): residual layer에서는 입력들의 해상도가 같기 동일하기 때문에 ROI만 처리해 주는 데에 문제가 발생하지 않는다.

Upsampling layer: 기존 네트워크에서와 동일한 방식으로 ROI 입력을 처리하면 된다.

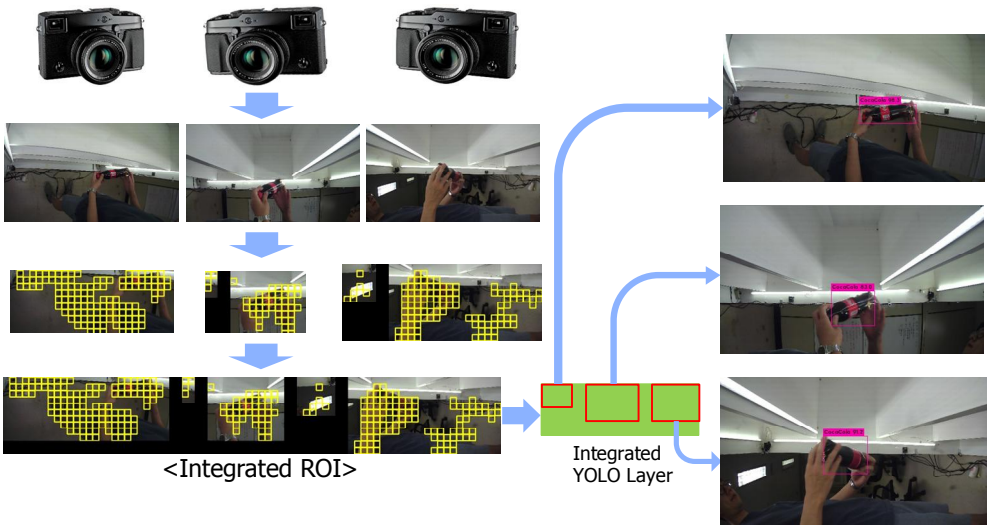
Concatenation layer(route layer): 해상도가 동일한 입력을 채널 축으로 쌓는 연산이 진행되므로 기존 네트워크와 동일한 동작을 하면 된다.

YOLO layer: YOLO 네트워크에서 원하는 형태로 결과값을 변형시켜주는 역할을 한다. 기존 layer와 동일하게 동작시킨다.

3.4 병합 ROI 처리 방식

제 2장에 소개된 것과 같이 단일 이미지에서 ROI를 추출하여 처리하는 방식에는 속도 향상의 한계가 있다. 이를 통해 N개의 카메라에서 추출된 ROI를 연속적으로 처리할 때보다 N개의 카메라에서 추출된 ROI를 병합해서 처리할 때 더 속도 향상이 있을 것임을 예측했다. 이를 검증하기 위해서 3개의 카메라에서 영상을 전달받아 하나의 ROI 입력으로 병합한 뒤 YOLOv3에 전달하여 처리하도록 전체적인 알고리즘을 구성하였다. [그림 9]에서 병합 ROI를 구성하여 YOLOv3에 전달하고 이를 처리하는 전체적인 흐름을 소개한다. 먼저 3대의 카메라에서 영상을 전달받는다. 이후 각각의 카메라로부터 전달된 영상에서 ROI를 추출한다. 이 때 각각의 추출된 ROI를 이용하여 YOLOv3의 입력으로 알맞은 입력을 구성한다. 그 뒤 각각의 ROI입력으로부터 높이는 최대값을 가지며 너비는 모두 합한 크기의 병합된

ROI입력을 구성한다. 이 때 각 ROI 입력 간에는 최소의 공간을 두어 각 ROI 입력 간의 간섭을 줄여준다. 이 병합된 ROI를 YOLOv3의 입력으로 전달하여 YOLOv3는 이 병합 ROI입력을 처리하게 된다. 이후 YOLOv3에서 병합 ROI 입력을 연산한 결과 부분을 추적하여 처음 각 카메라에서 전달 받은 영상에 분류된 물체들과 그 위치를 표시해주는 방식이다. 이는 기존 YOLOv3에서의 입력과 출력은 동일하게 유지 시킴으로써 다른 작업 간의 연결에도 아무 문제가 없게 하기 위함이다.



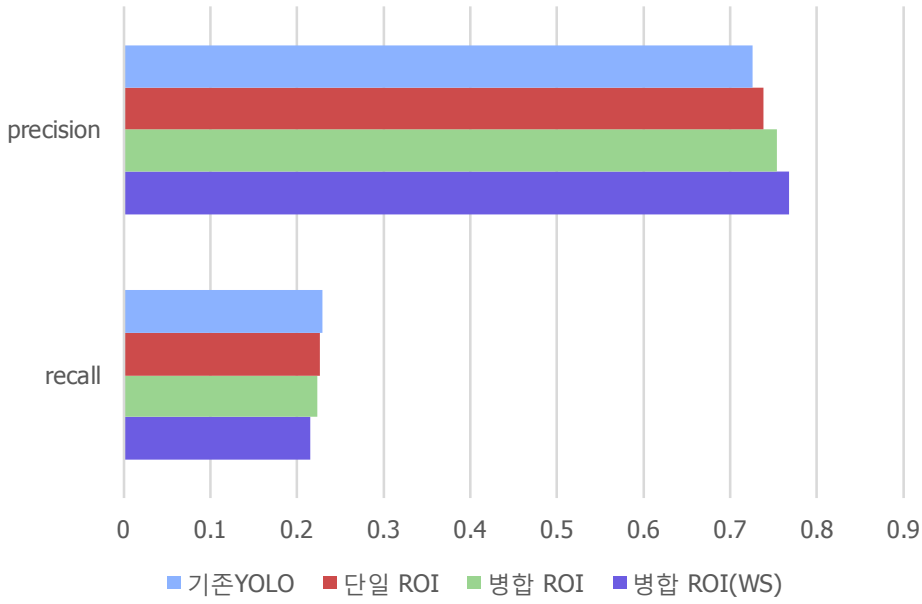
[그림 9] 병합 ROI 구성 및 처리 방식 전체 개념도

제 4 장 실험결과

본 장에서는 제 3장에서 소개된 제안 방법을 구현함으로써 도출된 실험결과를 제시한다. 먼저 소개된 제안 방법들을 적용했을 때의 인식률 변화를 4.1절에서 소개한다. 이후 4.2절에서는 각 방법 별 처리 시간을 보여준다. 마지막으로 4.3절에서는 이전 연구와의 비교를 진행하여 기존 연구 대비 속도 향상을 했으며 이때 인식률 변화는 크지 않음을 보여준다.

4.1 제안 방법의 인식률 변화

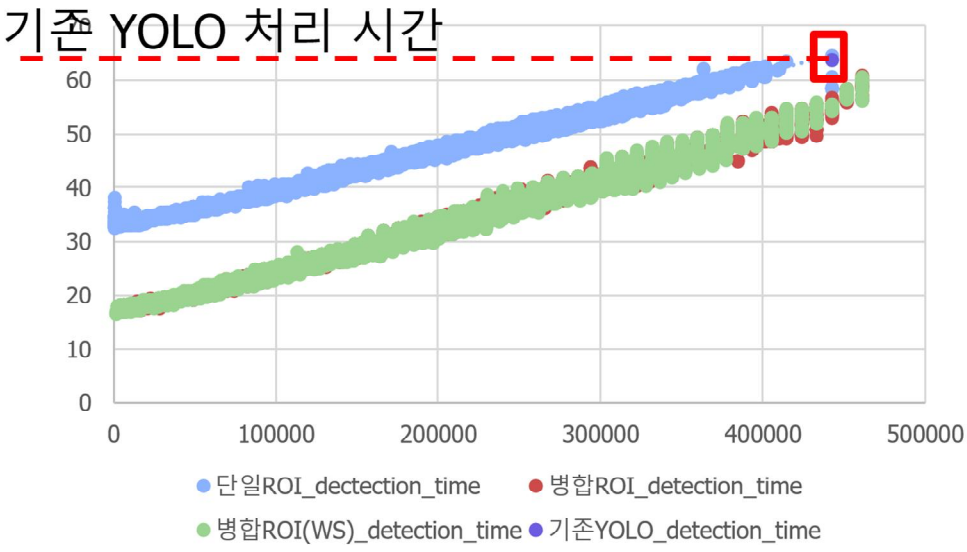
인식률은 COCO dataset testset[12] 초반 300 장에 대하여 단일 영상에서 ROI를 추출하고 ROI 입력을 구성한 뒤 TH3의 방식을 적용한 결과 기존 YOLOv3 네트워크 대비 제안 방법의 mAP-50이 2.81% 하락했다. 무인 편의점 데이터 셋을 구성하고 33636장의 테스트 셋에서 또한 인식률을 비교했다. 이 테스트 셋은 트레이닝 시기에는 사용되지 않았다. 기존 YOLOv3, 단일 영상에서 ROI입력을 연속적으로 처리한 모델, 병합 ROI 입력을 처리한 모델, 그리고 병합 ROI 입력을 너비 분할 방식을 적용하여 구성한 후 처리한 모델의 정확도는 [그림 10]을 통하여 비교할 수 있다. [그림 10] 에서와 같이 인식률에서는 큰 변화가 없음을 관측할 수 있다.



[그림 10] 무인 편의점 테스트 셋에서의 인식률 변화

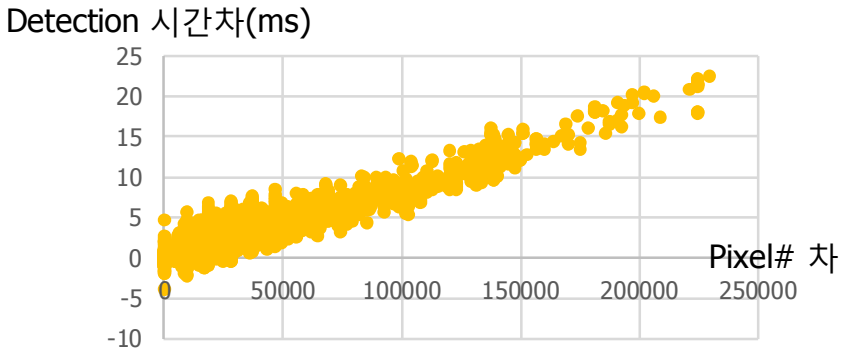
4.2 제안 방법의 처리 시간 비교

3대의 카메라로부터 이미지를 전달 받아 기존 YOLOv3에서의 처리 시간과 3장 각각의 이미지에서 추출한 ROI를 이용하여 구성한 ROI입력 3개를 연속해서 처리했을 때, 추출한 ROI를 병합하여 처리했을 때, 마지막으로 추출한 ROI를 너비 분할 방식을 적용한 뒤 병합했을 때 처리 시간을 비교한 결과가 [그림 11]에 소개되어 있다. 테스트 셋은 무인 편의점 테스트 셋에서 진행하였다. 이는 제2장에서 예측된 결과와 일치하는 결과를 나타낸다. 즉 단일 이미지에서 추출한 ROI를 이용하여 구성한 ROI입력을 연속하여 처리하는 것보다 ROI를 병합하여 하나의 입력으로 구성한 뒤 처리하는 것이 더 속도 향상을 할 수 있음을 보여준다.



[그림 11] 무인 편의점 테스트 셋에서의 처리 시간 비교

병합 ROI를 구성할 때 너비 분할 방식을 적용할 때와 하지 않을 때의 비교 결과는 [그림 12]에서 확인할 수 있다. 그 결과 너비 분할 방식을 적용했을 경우 적용 전 대비 총 병합 ROI 입력의 면적은 11.84% 감소했으며, 총 처리 시간은 7.11% 감소했다. 또한 픽셀이 감소된 프레임 비율은 94.5% 이었으며, 시간이 감소된 프레임 비율은 93.3% 였다. 너비 분할 방식을 적용함으로써 증가한 시간의 최대값은 3.86ms 였다. 이를 통해 너비 분할을 적용하는 것이 약간의 overhead는 존재하나 대부분의 경우 ROI 입력의 면적을 감소시켜주고 결과적으로 연산량을 줄여 속도를 향상 시켜주는 것을 확인하였다.



[그림 12] 너비 분할 방식을 적용 전후 ROI 입력 픽셀 수 차이 대비 처리 시간 차이 도표

4.3 이전 연구와의 성능 비교

이전 연구 결과[11]와 비교한 결과는 [표 4-1]와 같다. 3대의 카메라로부터 영상이 전달 될 때, 각각 단일 영상에서 ROI를 구성하여 이를 연속적으로 처리할 경우는 기존 YOLOv3 대비 COCO dataset testset에서 mAP-50 기준 2.81%의 정확도 하락이 있었다. 이후 무인 편의점 데이터 셋 테스트 셋에서 실험한 결과 3대의 카메라로부터 영상이 전달 될 때, 각각 단일 영상에서 ROI를 구성하여 이를 연속적으로 처리할 경우는 평균 1.33배의 속도향상을 이루었으며 단순한 방법으로 ROI 입력을 구성하여 병합한 결과 평균 1.62배의 속도 향상이 있었다. 추가로 너비 분할 방식을 적용하여 ROI 입력을 구성하여 병합한 결과 평균 1.75배의 속도 향상이 있었다. 모든 경우 인식률의 변화는 미미하였다.

	Pack and detect (SSD 대비 성능) [11]	제안 방법 (YOLO 대비 성능)		
	-	단일 ROI	병합 ROI	병합 ROI (WS)
속도 향상 (times)	1.3x	1.33x	1.62x	**1.75x**
정확도 변화	2.5% drop	2.8% drop (COCO, mAP-50)	-	-

[표 4-1] 기존 알고리즘 대비 속도 향상 비교

제 5 장 결 론

본 연구에서는 우선 무인 편의점 상황을 가정하고 환경을 구성하고 관련된 데이터 셋을 제작하여 실험하였다. 무인 편의점에서는 다중 카메라로부터 전달되는 영상에서 실시간으로 물체를 검출하는 것이 중요하다. 이는 기존 YOLOv3에서 단일 이미지는 실시간으로 처리 가능했지만 다중카메라로부터 전달되는 여러 이미지를 연속해서 처리하기에는 역부족이었다. 이를 개선하고자 본 논문에서는 단일 이미지에서 ROI를 추출하여 YOLOv3의 입력으로 구성하여 처리하는 방법을 우선 제안하였다. 이 방법은 COCO dataset에서 mAP-50 기준 2.81% 하락을 가져온 반면 입력 크기에 따라 선형적으로 속도 향상을 시킬 수 있는 것을 확인하였다. 더 나아가 위 방식의 실험결과의 분석을 통해 다중 카메라를 사용해야하는 무인 편의점과 같은 상황에서 단일 이미지에서 ROI를 추출하여 연속하여 처리하는 경우 속도

향상의 한계가 있음을 보여주었다. 본 논문은 이 문제에 대한 해결책으로 다수의 카메라로부터 전달된 이미지에서 추출된 ROI를 하나로 병합하여 처리하는 방식을 제안했다. 이에 대한 효과를 제 2장에서 수식적으로 예측하고 이를 제 3장에서 어떻게 구현했는지 자세히 서술하였다. 제 4장에서는 구현된 결과와 이전연구와의 성능 비교를 통하여 본 연구의 효용성을 소개했다. 결론적으로 무인 편의점 데이터 셋에서 기존 CNN기반 물체 검출 알고리즘(YOLOv3) 대비 평균 1.75배의 속도 향상을 이루었으며 이때 인식률의 변화는 미미했다.

감사의 글

먼저 여기까지 나를 인도해 오시는 하나님께 감사하다. 가장 옆에서 지도해 주신 이혁재 교수님과 김진성 교수님께도 감사드린다. 곁에서 함께 견뎌와 준 가족과 여자친구 지혜가 없었으면 이 논문은 나올 수 없었을 것이다. 함께 마지막 과제를 같이 진행하고 지도해준 철희형께도 감사를 드리며 같이 힘든 순간을 견뎌온 인터파크 팀에게도 고마움을 표한다. 지금까지 답답한 저를 지도해준 승환형, 태성이형에게도 감사를 드린다. 함께 연구실에 들어와서 지금까지 함께한 보열, 세훈, 지예, 샤오펑, 콘에게도 고맙고 즐거운 시간을 많이 갖지 못한 아쉬움이 있다. 또한 언급하지는 못했지만 즐거운 연구실 생활을 하게 해준 모든 구성원들에게 감사하다. 안암에서까지 나의 투정을 받아준 많은 친구들, 특히 환에게도 고마움을 전한다. 산티아고의 여정에 같이 오르며 여행 중에 논문 쓰는 것을 묵묵히 참아주는

든솔이에게도 고맙다. 마지막으로 이 논문이 완성되도록 끝까지 도와주는 보열이에게도 다시 한 번 고마움의 말을 전한다.

참고 문헌

[1] D. Grewal, A. L.Roggeveen, and J. Nordfalt, “The future of retailing,” Journal of Retailing, vol. 93, pp. 1−6, March 2017

[2] M. McFarland (2018, Oct 03). "I spent 53 minutes in Amazon Go and saw the future of retail." CNN Business Retrieved Dec 6, 2018, https://edition.cnn.com/2018/10/03/tech/amazon−go/index.html

[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R−CNN: Towards real−time object detection with region proposal networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, issue 6, pp. 1137?1149, June 2016.

[4] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," in International Journal of Computer Vision 2013.

[5] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in International Conference on Neural Information Processing Systems, 2016

[6] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2017

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” IEEE International Conference on Computer Vision, October 2017

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv, 2018

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed, "SSD: Single shot multibox detector," in The European

Conference on Computer Vision, 2016

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2016.

[11] A. R. Kumar, B Ravindran, and A. Raghunathan, “Pack and Detect: Fast Object Detection in Videos Using Region-of-Interest Packing,” arXiv, 2018

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in The European Conference on Computer Vision, 2014

Abstract

An ROI extraction and processing method to speed up a CNN based object detection algorithm in a multi-camera environment

성 명 YOU HAK LEE
학과 및 전공 Electrical and Computer Engineering
The Graduate School
Seoul National University

Along with development of deep learning, deep learning or CNN-based object detection algorithm is applied in many areas. One of them is Amazon Go which is unmanned store launched by Amazon Corp. Treating many input images from several camera is important for cost efficient aspect.

Recent research shows that CNN-based object detection algorithm has shown better detection performance compared to previous object detection algorithms. Early CNN-based object detection algorithms found a region of interest (ROI) and used these ROIs to classify objects and estimate the position of objects. However, they showed limitations in real-time processing because it is divided into two stages. Later on, single-stage CNN-based object detection algorithms that can process single images in real time such as SSD and YOLO came out. They showed excellent speed on object detection performance. However, The CNN-based object detection algorithm that has improved speed was only able to

process single images in real time. This cannot process images from multiple cameras in real-time or situations in which low-power devices must be used.

This paper proposes new methods to speed up CNN-based object detection algorithms. This is the method that extracts ROIs from images that are transmitted from cameras and sends them to the input of existing CNN-based object algorithms for processing. Furthermore, this paper proved that the method of extracting and processing the ROI from a single image shows the speed improvement is limited. This poses a big problem in situations such as unmanned convenience stores that need to use multiple cameras. This paper proposes a solution through the merging ROIs in a multi-camera environment. Consequently, there was an average speed improvement of 1.75 times over the existing CNN-based object detection algorithm (YOLOv3) with similar accuracy

Keywords : Object Detection, Convolutional Neural Network, Region of Interest, Multi-camera, Unmanned Store, Loop optimization
Student Number : 2017-22343