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

공학박사 학위논문

다중 카메라 물체 인식을 위한 Dominant Feature Pooling 및 Retinex 알고리즘 최적화

Dominant Feature Pooling for

Multi Camera Object Detection and

Optimization of Retinex Algorithm

2021 년 8 월

서울대학교 대학원

전기 정보 공학부

박 진 우

다중 카메라 물체 인식을 위한 Dominant Feature Pooling 및 Retinex 알고리즘 최적화

지도 교수 이 혁 재

이 논문을 공학박사 학위논문으로 제출함
2021 년 8 월

서울대학교 대학원
전기 정보 공학부
박 진 우

박진우의 공학박사 학위논문을 인준함
2021 년 8 월

위 원 장 김 태 환

부위원장 이 혁 재

위 원 조 남 익

위 원 권 석 규

위 원 김 진 성

초 록

본 논문은 멀티 카메라 object detection CNN을 위한 detection 단계에서 활용하는 새로운 dominant feature pooling 방법을 제안한다. 멀티 카메라 시스템은 다양한 관점에서 물체의 이미지를 캡처하고, 물체의 더 많은 주요 feature를 detection에 활용할 수 있다. 따라서 여러 카메라에서 feature를 pooling하면 detection 정확도를 향상시킬 수 있다. 제안된 방법은 객체의 다양한 뷰포인트에서 얻은 feature vector 중에서 더 많은 정보를 제공하는 주요 feature을 선택하고 선택한 feature vector를 pooling하여 새로운 feature map을 구성한다. 제안된 방법은 단일 카메라에 대한 YOLOv3 네트워크를 기반으로 하며, 멀티 카메라 시스템에 대한 추가 학습 과정이 필요하지 않다. Dominant feature pooling의 효과를 주장하기 위해, 이 연구에서는 feature vector를 시각화하는 새로운 방법도 제안된다.

또한 object detection CNN은 저조도 환경에 대응이 취약하므로 이를 개선할 수 있는 Retinex 알고리즘의 활용 방법을 제안한다. 저조도 영상을 그대로 학습하여 개선을 할 수 있지만, 실사용 환경에서 조도 정도를 예측할 수 없기 때문에 Retinex 개선이 필수적임을 실험을 통해 나타내었다. 또한 개선 효과가 뚜렷하지만 복잡도가 높은 Retinex 알고리즘을 HW 설계를 통해 최적화 하는 방법을 제안한다. Retinex 알고리즘 연산에 필수적인 exponentiation과 Gaussian filtering을 효율적으로 구현하는 방법을 제안하여 높은 해상도에서도 실시간으로 동작이 가능한

HW를 구현하였다.

주요어 : Multi-camera, object detection CNN, CNN features, Retinex algorithm, FPGA
학 번 : 2014-21658

목 차

제 1 장 서 론 1
1.1 연구 배경 1
1.2 연구 내용 2
1.3 논문 구성 4

제 2 장 배경 이론 및 관련 연구 5
2.1 Object Detection CNN 5
2.2 Multi View CNN 6
2.3 Retinex 알고리즘 7
2.3.1 Retinex Algorithm using Gaussian Filter 8
2.3.2 Multiscale Retinex Algorithm 9
2.3.3 Efficient Naturalness Restoration 10

제 3 장 무인 판매대 시스템 12
3.1 무인 판매대 시스템 개요 12
3.2 Object Detection CNN을 활용한 상품 인식 16
3.3 Multi-Object Tracking을 활용한 상품 구매 판단 18
3.4 무인 판매대의 실시간 동작을 위한 최적화 방안 20
3.4.1 카메라 선택 알고리즘 20
3.4.2 Multithreading 24
3.4.3 Pruning 25
3.5 무인 판매대 시스템 성능 평가 27
3.5.1 Object Detection 성능 평가 27
3.5.2 무인 판매대 시스템 전체 결과 29

제 4 장 멀티 카메라 Dominant Feature Pooling 32
4.1 Object Detection CNN과 멀티 카메라 Object Clustering 33
4.1.1 Object Detection CNN 33
4.1.2 멀티 카메라 Object Clustring 35
4.2 Dominant Feature Pooling 방법 37
4.2.1 Dominant Feature Scoring 40
4.2.2 Dominant Feature Pooling 47
4.2.3 YOLOv3의 Detection Layer 재사용 50

4.3 Feature 시각화를 통한 제안 방법 분석 52
4.3.1 제안하는 Feature 시각화 방법 52
4.3.2 기존 단일 카메라 YOLOv3의 Feature 시각화 55
4.3.3 제안하는 방법의 멀티카메라 Feature 시각화 57
4.4 Dominant Feature Pooling 결과 및 분석 59
4.4.1 COCO Dataset에서의 결과 60
4.4.2 Custom Dataset에서의 결과 62
4.4.3 Scoring Method 별 결과 63
4.4.3 Dominant Feature Pooling의 수행시간 결과 64

제 5 장 Retinex Applied Object Detection 및 하드웨서 가속 시스템 65
5.1 기존 Retinex 적용 연구 66
5.2 Retinex Applied Object Detection 68
5.2.1 Retinex Applied Object Detection 학습 68
5.2.2 Retinex Applied Object Detection 결과 72
5.3 Object Detection을 위한 Retinex 최적화 76
5.3.1 Gaussian Filter 크기에 따른 Retinex 효과 분석 ... 76
5.3.2 Gaussain Filter 크기에 따른 Object Detection 결과 80
5.4 Retinex 하드웨어 시스템의 필요성 및 기존 연구 82
5.5 제안 하드웨어 시스템 구현 개요 85
5.6 제안 하드웨어 시스템 구현 특장점 89
5.6.1 Gaussian filter의 구현 89
5.6.2 Exponentiation의 구현 96
5.6.3 HDMI/DVI 지원 및 영상 latency 최소화 103
5.7 제안 하드웨어 시스템 구현 결과 및 분석 106
5.7.1 실시간 동작 및 낮은 latency에 대한 분석 106
5.7.2 제안한 시스템의 영상 처리 성능 결과 분석 109
5.7.3 제안한 시스템의 FPGA Resource Utilization 112
5.7.4 다른 시스템과의 Resource Utilization 비교 114
5.7.5 제안한 시스템의 영상 처리 성능 결과 분석 119

제 6 장 결론 120

참고문헌121

Abstract 131

표 목차

[표 3.1] 각 scenario에 대한 무인 판매대 시스템 정확도.. 31
[표 4.1] COCO test set에서의 dominant feature pooling 결과 및 다른 방법과의 비교 61
[표 4.2] Custom dataset에서의 dominant feature pooling 결과 및 다른 방법과의 비교 62
[표 4.3] COCO test set에서의 score 별 dominant feature pooling 결과 63
[표 4.4] Dominant feature pooling의 timing overhead ... 64
[표 5.1] 조도 변화 test set의 YOLO detection 결과 73
[표 5.2] 조도 변화 test set의 dominant feature pooling 결과 75
[표 5.3] Gaussian kernel 크기에 따른 YOLO와 dominant feature pooling의 결과 81
[표 5.4] 각 module의 출력에 대한 fixed point precision 87
[표 5.5] Gaussian filter approximation 95
[표 5.6] Frame buffer와 line buffer, 제안하는 시스템의 latency 비교 108
[표 5.7] Full size filter와 approximated filter 간의 absolute difference의 평균값 110
[표 5.8] 제안하는 시스템의 Gaussian filter approximation으로 감소한 resource utilization 및 비교 113
[표 5.9] 제안하는 시스템의 주요 module의 resource utilization..... 113
[표 5.10] 제안하는 시스템과 다른 시스템과의 resource 비교 116
[표 5.11] 제안하는 시스템과 비슷한 알고리즘을 구현한 연구와의 resource 비교..... 118

그림 목차

[그림 3.1] 무인 판매대 시스템 구성도 13
[그림 3.2] 무인 판매대의 각 카메라에서 획득한 입력 영상 예시 14
[그림 3.3] 무인 판매대 시스템 알고리즘 순서도 15
[그림 3.4] 무인 판매대에서 판매하는 상품 48종 17
[그림 3.5] 무인 판매대의 선반 영역과 카메라 번호 21
[그림 3.6] 무인 판매대의 선반 영역 23
[그림 3.7] 학습된 YOLOv3의 PR curve 28
[그림 3.8] 무인 판매대 시스템 성능 평가를 위한 구매 scenario 30
[그림 4.1] YOLOv3 network architecture 34
[그림 4.2] 유효한 bounding box 선정을 통한 object 클러스터 방법 37
[그림 4.3] 제안하는 알고리즘의 순서도 39
[그림 4.4] Dominant feature pooling outline 49
[그림 4.5] 제안하는 feature 시각화 방법 54
[그림 4.6] YOLOv3의 layer 별 feature vector 시각화 ... 56
[그림 4.7] Dominant feature pooling의 시각화 56
[그림 5.1] 서로 다른 scale factor로 변환한 image 69
[그림 5.2] 서로 다른 scale factor로 변환한 image의 Retinex 적용 결과 70
[그림 5.3] 저조도 영상 획득을 위한 무인 판매대 환경 71
[그림 5.4] 저조도 testset 예시 71
[그림 5.5] Dark test set에서의 precision과 recall graph 75
[그림 5.6] Gaussian kernel size에 따른 Retinex의 차이 77
[그림 5.7] 실제 촬영 영상에 대해 Gaussian kernel size에 따른 Retinex의 차이 79
[그림 5.8] 제안하는 하드웨어 시스템의 block diagram 88
[그림 5.9] Block diagram상의 각 module의 출력 결과 ... 88
[그림 5.10] Gaussian filter의 σ 값에 따른 Retinex 결과 91
[그림 5.11] Gaussian filter size에 따른 normalize factor 93
[그림 5.12] Full size와 approximated size의 Gaussian

kernel shape 93
[그림 5.13] 범위에 따른 $\log_2 x$의 그래프 99
[그림 5.14] 2^5 크기의 LUT를 활용한 부분선형 근사의 오차 101
[그림 5.15] Exponentiation module의 block diagram..... 102
[그림 5.16] 60fps 영상에서 연속된 frame의 α 값 변화 .. 105
[그림 5.17] 강한 햇빛과 그늘이 있는 영상 110
[그림 5.18] 창문에 어둡게 반사된 영역이 있는 영상..... 110
[그림 5.19] 그림자로 인한 저조도 영역이 있는 영상..... 110
[그림 5.20] 적은 광량으로 인해 저조도 영역이 있는 영상 110

제 1 장 서 론

1.1 연구 배경

Object detection convolutional neural network (CNN)는 많은 학술 및 산업 분야에서 상당한 관심을 받고 있다. 이미 자율 주행 자동차, 현금 없는 매장 및 스마트 공장에서 특정 작업을 수행하기 위해 object detection CNN을 활용하는 애플리케이션이 많이 있다. 그러나 object detection CNN의 최신 architecture들도 높은 detection accuracy를 보장하지 않는다 [1], [2], [3]. 높은 정확도와 빠른 속도로 인기있는 object detection CNN 중 하나 인 YOLOv3 [4]는 COCO dataset [5]에서 입력 이미지 해상도가 416×416 인 경우에 mAP50 기준 55.3%를 보여준다. 실제 상황에서 물체 감지 CNN을 사용하려면 감지 정확도를 개선할 필요가 있다.

감지 정확도를 높이기 위해 멀티 카메라 시스템을 기반으로 일부 연구가 진행되었다. Su는 3D 모양 객체의 분류를 개선하기 위해 multi-view CNN을 제안했다 [6]. CNN에서 얻은 feature들을 view pooling process를 통해 종합하고 추가적인 convolution layer들로 class를 예측하는 구조로 되어있다. 3D 데이터를 분류하기 위해 3D-CNN도 제안되었지만 [7], [8] 훈련에는 3D 형상 모델이 필요하고 입력 데이터에는 point cloud 데이터와 같은 3D 정보가 필요하다. Multi-view CNN의 feature pooling process는 주로 max pooling 또는 average pooling을 기반으로 한다. 따라서 값이 작은 feature은 무시되어 사용되지

않을 수 있다. Multi-view CNN의 사용에는 몇 가지 제한이 있다. 분류 할 대상 물체는 배경 영역이 매우 작아야 하며 이미지 중앙에 위치해야 한다. 따라서 실제 응용에서 multi-view CNN을 이미지 전체에서 탐지해야 할 물체가 차지하는 비율이 적은 object detection task에 직접 적용하는 것은 어렵다.

멀티 카메라 및 multi view를 활용한 object detection network에서의 연구는 같은 물체를 다른 시점을 가진 서로 다른 카메라에서 찾아 동일한 label을 씌우는 re-identification task를 위주로 이루어지고 있다 [9], [10]. 그러나 멀티 카메라에서 얻은 정보를 detection accuracy의 개선을 위해 효과적으로 사용하는 방법에 대한 연구는 아직 부족하다.

1.2 연구 내용

Object detection CNN은 탐지하고자 하는 물체의 상태에 최대한 robust하게 동작하게 하기 위해 다양한 시점에서 물체를 촬영하여 얻은 학습 데이터 세트로 학습한다. 따라서 다양한 시점에서 보여지는 물체의 주요 feature들을 학습할 수 있다. 단일 카메라로 이미지를 detection 할 때는 이러한 주요 feature 중 일부만 detection에 사용된다. 멀티 카메라 시스템에서는 여러 시점의 더 많은 feature들을 detection에 사용할 수 있으므로 효과적으로 feature들을 종합할 수 있다면 detection accuracy를 높일 수 있다.

또한 object detection은 다양한 환경에 robust하게 대응할 수 있도록 학습되어 있지만 조도의 변화, 특히 저조도 상황의 대응에

어렵다는 한계를 지닌다. 조도가 낮은 경우 획득한 image의 dynamic range가 적어 object의 중요한 feature를 얻을 수 있을 만한 detail 정보가 많이 줄어들기 때문에 object detection CNN으로 정확하게 detection을 하기 어렵다. 낮은 조도의 상황에서 object detection이 어렵다는 점을 인지하여 기존에 이를 해결하기 위한 연구가 있고, 그 중 Retinex [11]를 활용하여 개선을 시도한 연구가 있다 [12]. Retinex 알고리즘은 영상에서 어두운 부분만 효과적으로 개선하여 밝은 부분의 saturation을 일으킬 염려 없이 양질의 결과를 얻을 수 있는 알고리즘이다. 기존 연구에서는 Retinex를 활용한 object detection이 효과가 있지만 저조도 영상을 그대로 학습하여 detection에 활용하는 방법도 효과가 있다고 주장하였다 [13]. 그러나 이러한 결과는 학습 dataset과 테스트 dataset의 저조도 환경이 유사하여 dynamic range가 비슷할 경우에는 유효하나, 학습한 dataset과 조도 환경이 차이가 나는 실제 환경에서는 그 효과가 떨어진다. 따라서 본 연구에서는 다양한 조도 환경에 대응할 수 있는 Retinex를 이용한 object detection을 구현하고 이를 dominant feature pooling에도 적용시켜 실사용 환경에서의 유효성을 제안한다.

어두운 영상을 개선하는 Retinex 알고리즘은 그 효과가 매우 뛰어나지만 복잡도가 높고 연산량이 많다. 따라서 실사용 환경에서 사용하기에는 수행 시간이 커서 활용하기 어렵다. 본 연구에서는 복잡도가 높은 Retinex 알고리즘의 실시간 가속화를 위해 전용 HW를 설계하는 방법을 제안하고 FPGA에서 구현 내용을 검증하였다. 제안한 HW에는 복잡도가 높은 연산인 exponentiation과 Gaussian filtering을 HW 환경에서 효율적으로

구현할 수 있는 approximation 방법을 제안하였으며, latency 및 memory 활용을 최소한으로 만들기 위해 frame buffer 대신 line buffer로 구현하였다는 장점이 있다. 제안하는 Retinex HW는 기존의 Retinex 알고리즘 기반의 HW 연구에 비해 매우 적은 resource로 높은 throughput의 출력을 내는 결과를 보인다.

1.3 논문 구성

본 논문의 구성은 다음과 같다. 2장에서는 배경이 되는 기존 연구들에 대해서 설명하고 3장에서는 무인 판매대 시스템의 구현 내용을 설명한다. 4장에서는 멀티 카메라 환경에서 feature들을 종합하여 detection accuracy를 높일 수 있는 새로운 pooling 방법을 제안한다. 5장에서는 저조도 환경에서 object detection의 성능을 높일 수 있는 Retinex 알고리즘과의 상관 관계 및 결합 방법에 대해서 설명하고 복잡도가 높은 Retinex 알고리즘의 최적화 및 HW 설계에 대해 설명한다.

제 2 장 배경 이론 및 관련 연구

2.1 Object Detection CNN

딥 러닝의 등장으로 CNN 기반 모델은 object detection에서 널리 사용된다. R-CNN은 선택적 검색에 의해 생성된 지역 제안을 CNN 기능과 결합하여 기존의 객체 감지 방법과 비교하여 인상적인 결과를 달성한다 [14]. 그러나 작동 속도가 느리고 입력 크기는 R-CNN에서 고정된다. SPPnet은 이러한 두 가지 단점을 해결하였다 [15]. 전체 입력 이미지에 대한 convolution feature map 계산을 통해 결과를 공유함으로써 작동 속도가 향상된다. Spatial pyramid pooling을 사용하여 변수 크기를 변경할 수 있다. Fast R-CNN은 RoI (Region of Interest) pooling을 사용하여 계산을 공유하고 다중 작업 손실을 사용하여 단일 단계 훈련을 가능하게 한다 [16]. Faster R-CNN은 선택적 검색 알고리즘 대신 지역 제안을 생성하기 위해 RPN (Region Proposal Network)을 도입하여 빠른 R-CNN을 개선하여 더 높은 정확도와 더 빠른 속도를 달성한다 [17]. Faster R-CNN은 뛰어난 탐지 정확도를 달성하지만, 속도는 여전히 실시간 요구 사항을 충족할 수 없다. 앞서 언급한 two stage object detection 방법과 다른 YOLO (You Only Look Once)는 object detection을 regression 방법으로 추론한다 [18]. 단일 convolutional network를 사용하여 전체 이미지에서 bounding box와 class probability를 직접 예측하여 약간의 정확도 손실로 실시간 성능을 달성한다. SSD (Single Shot

MultiBox Detector)는 다양한 가로 세로 비율을 가진 다중 스케일 feature map과 bounding box를 사용하여 서로 다른 규모로 객체를 감지하는 또 다른 전형적인 single stage object detection 방법이다 [19].

2.2 Multi View CNN

검출 정확도를 높이기 위해 멀티 카메라 시스템을 기반으로 일부 연구가 수행되었다. Su는 3D object의 classification를 개선하기 위해 multi view CNN을 제안했다 [6]. CNN이 획득한 feature들은 view pooling 프로세스를 통해 pooling되고 class는 나머지 layer들에 의해 예측된다. 3D-CNN은 3D 데이터를 분류하기 위해 제안되었지만 학습을 위해서는 3D shape model이 필요하며, 입력 데이터는 point cloud 데이터와 같은 3D 정보가 필요하여 실제 상황에서 카메라를 통해 입력된 영상에 적용하기 어렵다. Multi view CNN의 feature pooling 프로세스는 주로 max pooling 또는 average pooling을 기반으로 이루어져 값이 작은 feature의 정보는 사용되지 않을 수 있다. Multi view CNN을 사용하는 데는 몇 가지 제한이 있다. 분류할 대상 object는 배경 영역이 작아야 하며 object가 이미지의 중앙에 최대한 꽉 차게 위치해야 한다. 따라서 multi view CNN을 실제 애플리케이션에서 object detection network에 직접 적용하는 것은 어렵다.

2.3 Retinex 알고리즘

검출 정확도를 높이기 위해 멀티 카메라 시스템을 기반으로 일부 연구가 수행되었다 E. H. Land [11] 가 제안한 Retinex Theory는 눈이 빛을 인식할 때 반사 성분과 조명 성분이 함께 눈으로 들어오지만, 뇌가 색에 대한 정보를 인지할 때는 반사 성분을 위주로 인식하게 된다는 것을 수학적으로 보였다. 따라서 눈에서 인지한 색과 밝기 정도가 일치하더라도, 반사 성분과 조명 성분의 구성에 따라 뇌에서는 영상을 다르게 인지한다. 눈에서 인지한 영상은 조명 성분과 반사 성분의 곱으로 표현되고, 반사 성분은 빛의 세기와 상관 없이 물체의 색 정보만 담게 된다. 따라서 반사 성분을 입력 영상에서 추출할 수 있다면 어두워서 잘 식별할 수 없는 부분이 담고 있는 정보를 색 복원을 통해 확실하게 나타낼 수 있다.

Retinex theory를 바탕으로 D. J. Jobson [20], D. Terzopoulos [21], R. Kimmel [22] 외 다른 연구자들은 computer image에서도 인간의 시신경에 대한 색 인지 과정을 algorithm으로 구현하였다. 특정 convolution filter를 통하여 input image의 illumination channel을 얻은 후 기존 pixel value에서 illumination channel을 나눠주어(division) reflectance channel을 분리할 수 있다. Reflectance channel은 input image에 포함된 각 부분의 색 정보를 보다 정확히 표현한다. Illumination channel을 estimate 하는 방법은 E. Land가 제안한 inverse square function [23]과 Moore가 제안한 exponential absolute value [24], Gaussian filter 등 다양한 방법이 있다. Jobson은 Gaussian filter를 사용하는 것이 다른 방법들에 비해 image의 다양한 spatial region과 dynamic

range에 대해서 더욱 flexible하게 good performance를 얻을 수 있기 때문에 가장 효율적인 방법은 Gaussian filter를 통한 방법임을 제시하였다 [25].

2.3.1 Retinex Algorithm using Gaussian Filter

Gaussian filter를 사용한 Retinex algorithm은 다음과 같다. 먼저 입력 영상 $I(x,y)$ 은 식 (2.1) 과 같이 illumination channel $L(x,y)$과 reflectance channel $R(x,y)$의 곱으로 표현된다.

$$I(x,y) = L(x,y) \cdot R(x,y) \tag{2.1}$$

Illumination channel을 얻기 위해 Gaussian filter kernel $F(x,y)$를 식 (2.2)와 같이 구한 후 식 (2.3)과 같이 입력 영상과 kernel을 convolution을 해주면 illumination channel인 $L(x,y)$가 나오게 된다.

$$F(x,y) = Ke^{-\frac{x^2+y^2}{\sigma^2}} \tag{2.2}$$

$$L(x,y) = F(x,y) \otimes I(x,y) \tag{2.3}$$

식에서 σ 는 filter의 standard deviation을 나타내며 Gaussian kernel의 scale을 결정하게 된다. K 는 normalization factor로 $F(x,y)$의 전체 합이 1이 되도록 정해지는 scalar 값이다.

입력 영상을 나타내는 식에서 reflectance channel을 얻기 위해서는 input image에서 illumination channel을 나누어 주면

되는데, 이 과정을 logarithmic form으로 나타내면 식 (2.4)와 같이 나타난다.

$$R(x,y) = log\, I(x,y) - log(F(x,y) \otimes I(x,y)) \tag{2.4}$$

위와 같이 하나의 scale의 Gaussian filter를 사용하여 illumination channel을 estimation하는 경우를 single scale Retinex algorithm 이라고 한다 (SSR).

2.3.2 Multiscale Retinex Algorithm

Gaussian filter는 illumination channel을 효율적으로 estimate 하지만 사람 눈과 같이 정교하게 생성하지는 못한다. 특히 image에 high frequency region과 low frequency region이 함께 있는 경우 하나의 scale의 Gaussian filter로는 Retinex algorithm이 good quality의 결과를 내지 못한다. Jobson은 다양한 scale의 Gaussian filter를 사용하는 Multiscale Retinex [26]를 제안하였다.

Multiscale Retinex algorithm은 식 (2.2)의 Gaussian filter 생성 식에서 standard deviation인 σ값을 조정하여 다양한 scale의 Gaussian filter 결과값을 식 (2.5)와 같이 이용한다. 각각 다른 값의 scale filter를 생성하는 σ_n에 대해 weight인 w_n을 설정하여 식 (2.6)과 같이 R_n을 생성한 후, 식 (2.7)과 같이 모든 scale에 대한 R_n을 더해준다.

$$F_n(x,y) = Ke^{-\frac{x^2+y^2}{\sigma_n^2}} \tag{2.5}$$

$$R_n(x,y) = \log I(x,y) - \log\big(F_n(x,y) * I(x,y)\big) \tag{2.6}$$

$$R_{MSR} = \sum_{n=1}^{N} w_n R_n \quad (2.7)$$

각 scale의 weight 값인 w_n은 w_1부터 w_n까지 더했을 때 총 합이 1이 되도록 설정한다. Jobson은 세 가지 다른 σ를 사용하였고, 각 σ에 대해 동일하게 1/3의 w_n을 설정하여 MSR을 구현했을 때 가장 결과가 좋다고 제안하였다.

MSR을 사용하면 다양한 frequency domain의 영역에 대해 Retinex algorithm을 적용하여 보다 정확한 illumination channel을 생성할 수 있지만 결과물로 얻게 되는 reflectance channel이 매우 부자연스러운 단점이 있다. 뿐만 아니라 여러 개의 Retinex 모듈을 사용해야 하기 때문에 상당한 수준의 연산량을 필요로 한다는 문제점을 갖는다.

2.3.3 Efficient Naturalness Restoration

MSR의 이런 단점을 보완하기 위하여 Y. Shin [27]는 efficient naturalness restoration 방법을 제시하였다. RGB channel의 max 값을 통해 얻은 channel을 이용하여 하나의 scale에 대해 Gaussian filtering을 하여 illumination channel을 얻고, 이 channel을 modify한 후 reflectance channel과 다시 composition을 하여 어두운 부분을 restoration하는 동시에 전체적인 image에서 자연스러운 색감을 나타내도록 하였다. Naturalness 알고리즘의 가장 큰 장점은 자연스러운 색감과 뛰어난 색 복원 능력, 그리고 MSR 대비 낮은 연산량에 있다. 자연스러운 색감과 저연산을 기반으로 한 real-time 구현은 의료용 영상의 색

복원을 위해 필수적인 특징이다. 본 연구에서는 Naturalness algorithm의 장점을 그대로 활용하면서 HW 구현에 필요한 최적화를 적용하여 Naturalness algorithm을 FPGA에 구현하였다.

제 3 장 무인 판매대 시스템

3.1 무인 판매대 시스템 개요

본 연구에서 구현한 무인 판매대는 3개의 shelf로 구성되었으며, 6 대의 카메라를 설치하여 어떤 shelf에서 물체를 구매하여도 detection을 할 수 있도록 설치하였다. 그림 3.1은 구현한 무인 판매대의 구성을 나타낸다. 90도의 화각을 갖는 상단 2 대의 카메라는 45도 기울여 장착되었고, 중간과 하단 카메라는 각각 중간과 하단 shelf를 좌우에서 촬영할 수 있도록 설치하였다. 상단 2 대의 카메라는 무인 판매대 전체를 내려다 보는 시점으로 촬영을 하기 때문에, 중간과 하단 shelf에서 꺼내지는 물체도 촬영할 수 있다. 그림 3.2는 각 카메라에서 capture된 image를 나타낸다.

멀티 카메라 시스템은 서로 다른 각도에서 scene을 바라보기 때문에, 물체의 다양한 면을 인식할 수 있고, occlusion등의 방해 요소가 있어도 다른 시점에서 상품의 scene을 확보할 수 있다. 따라서, 물체에 대한 다양한 시점이 확보되면 detection에 성공할 확률이 높아진다. 또한 서로 다른 위치의 카메라로 인식한 결과를 이용하여 상품의 3차원 위치를 추정할 수 있는데, 이는 상품 구매 판단에 활용될 수 있다. 본 연구에서는 YOLOv3 CNN [4]을 통해 각 카메라에서 판별된 상품의 class 및 위치 좌표를 얻는다.

그림 3.1. 무인 판매대 시스템 구성도

그림 3.2. 무인 판매대의 각 카메라에서 획득한 입력 영상 예시

구매자가 잡은 상품에 대한 detection 이후에는, 잡혀진 상품의 반납 또는 최종 구매를 판별해야 한다. 이를 위해 인식된 상품의 이동 경로를 추적하여, 사용자가 상품을 판매대로 반환하거나, 구매하여 가져가는 행위를 판단한다. 본 연구에서는 optical flow 기반의 tracking 알고리즘으로 구매자의 구매 및 반환 행위 판단을 하는 algorithm을 개발하였다. 제안 알고리즘에서는 일부 카메라와 일부 frame에 대해서 YOLOv3 CNN의 detection error가 발생해도 정확한 구매와 반환 판단을 할 수 있다.

무인 판매대 시스템의 사용자 구매 경험이 만족스럽고 신뢰성을 갖추기 위해서는 시스템이 실시간으로 동작하여 사용자와 시스템간의 반응속도가 좋아야 한다. 따라서 본 연구에서는 시스템이 실시간으로 동작할 수 있도록 카메라 선택 알고리즘, multithreading, pruning 등의 기법을 적용하였다. 제안하는 무인 판매대의 알고리즘 순서도는 그림 3,3과 같이 나타난다.

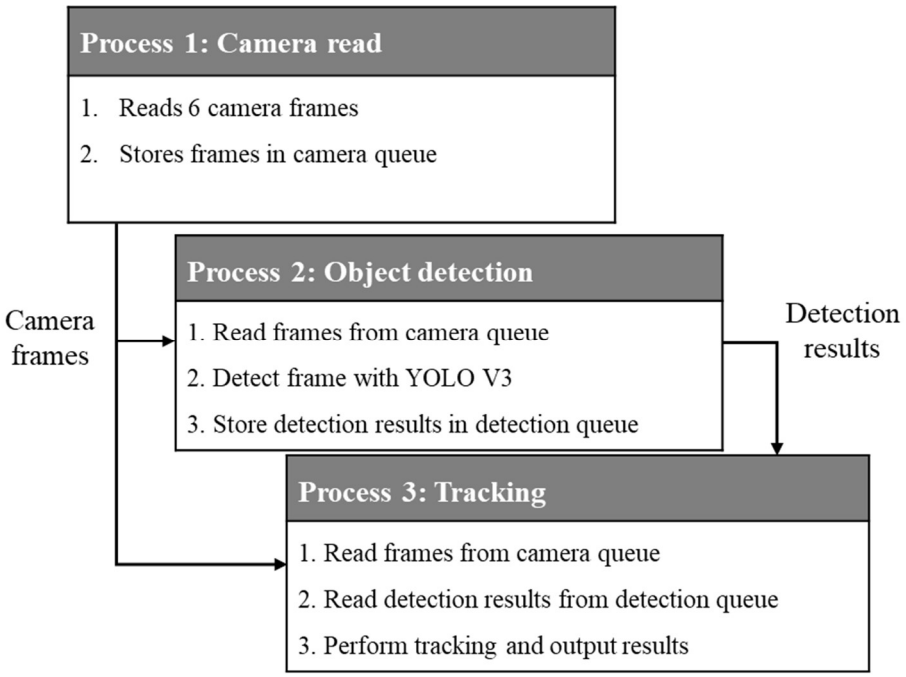


그림 3.3. 무인 판매대 시스템 알고리즘 순서도

3.2 YOLOv3 Object Detection CNN을 활용한 상품 인식

무인 판매대에 진열된 상품을 손으로 집어 드는 행위로 구매를 판별하려면 먼저 구매자가 집은

상품의 class를 인식해야 한다. 판매되는 상품들의 특성 상 외부 포장에 디자인 요소가 포함되어 특징적인 패턴들이 존재하고, 그 크기가 다양하다. SIFT [28], SURF [29]와 같은 local feature detector를 이용한 인식 방법도 있지만, viewpoint의 다양한 변화로 affine 등의 형태 변화가 많아 인식률이 높지 않으며, GPU로 가속한 CNN보다 수행 시간도 느리다. 본 연구에서는 CNN을 활용하여 입력 영상으로부터 object detection을 수행하였다.

본 연구에서는 object detection CNN중 YOLOv3 object detection network를 사용하였다. 비슷한 기능을 하는 object detection network로는 Faster R-CNN, SSD 등이 있다. YOLOv3는 Faster R-CNN과 비교했을 때, 정확도는 조금 떨어지지만, 100배 이상 빠른 수행 시간을 보이므로 실시간 동작에 더 적합하다. 또한 SSD보다 accuracy는 조금 떨어지지만 수행시간이 더 빠름을 알 수 있다. 그리고, 각 network를 활용하여 직접 생성한 dataset으로 학습하여 평가한 다른 연구들을 살펴보면, YOLOv3가 Faster R-CNN과 SSD보다 성능 및 수행시간이 모두 뛰어나다는 결과가 나타난다 [30], [31].

YOLOv3 object detection network를 사용하여 구매하는 상품 48 종을 인식하기 위해 과자, 음료, 그리고 생활 용품 등을 촬영하고 labeling하여 dataset을 구성하였다. 무인 판매대에서 판매하려는 상품 48 개의 class를 학습 대상으로 하였으며, 각

class 당 약 6,500 장, 총 310,117 장의 사진을 촬영하여 labeling을 하였다. 판매대에서 사용될 48종 상품은 그림 3.4에 나타나있다. 상품들은 kiosk에서 꺼내게 될 때 다양한 각도 및 위치에서 등장하므로 training image 촬영 시 상품의 다양한 각도와 크기로 나타나도록 하였다.

그림 3.4. 무인 판매대에서 판매하는 상품 48종

3.3 Multi-Object Tracking을 활용한 상품 구매 판단

무인 판매대에 집은 상품에 대하여 구매와 반환을 판단하기 위해 two stage multi-object tracking processing을 적용하였다. 첫 stage에서는 Markov decision process multi-object tracking (MDP-MOT) [32] 알고리즘을 적용하여 각 카메라의 입력 영상을 tracking을 한다. 두 번째 stage에서는 3차원의 object association을 적용하여 서로 다른 카메라에서의 tracking 결과를 공유하여 정확한 구매/반환 동작을 판단하도록 하였다.

MDP-MOT 알고리즘은 각 카메라의 입력 영상에 독립적으로 적용된다. 각 카메라에서 YOLOv3의 인식 결과를 바탕으로, 인식된 bounding box를 기준으로 tracking 결과를 얻는다. YOLOv3의 인식률이 완벽하지 않기 때문에, MDP-MOT를 활용하여 “Tracked” state의 물체의 인식 결과를 종합할 수 있도록 하여 인식률을 높이는 효과도 같이 얻을 수 있도록 하였다.

멀티 카메라 환경에서는 서로 다른 카메라의 tracking 결과를 종합하기 위해 3차원 공간에서의 object association이 필요하다. 같은 물체를 서로 다른 카메라에서 동일하게 tracking하기 위해서 본 연구에서는 MDP-MOT의 2차원 tracking 결과를 3차원으로 종합하는 방법을 적용하였다. 각 카메라에서 얻은 좌표를 삼각측량법을[33] 통해 3차원 좌표를 얻고, 이를 재투영하여 오차를 계산한다. 인식된 bounding box의 중앙을 기준으로 서로 다른 두 카메라의 카메라 매트릭스와의 연산을 통해 3차원 좌표를 얻고, 이를 다시 재투영 시키면 기존 중앙 좌표와 재투영된 좌표 간의 오차를 구할 수 있다. 구한 오차와 미리 정의된

threshold보다 낮은 값을 가진다면 두 bounding box가 같은 물체를 나타낸다고 판단하여 object association을 진행하게 된다. 3차원 object association 방법을 통해 6 개의 멀티 카메라에서 얻은 여러 인식 결과를 하나의 정확한 tracking 결과로 종합할 수 있다.

3차원 object association으로 얻은 tracking 결과의 birth와 death 좌표를 활용하여 상품의 구매와 반환에 대한 판단을 할 수 있다. Tracking의 birth 좌표가 판매대 안쪽이고, death 좌표가 판매대의 바깥쪽이라면 해당 tracking은 구매 동작을 한 것으로 판단하게 되며, 좌표가 반대로 되어있으면 반환 동작으로 판단하게 된다. 이러한 tracking 기반의 알고리즘 및 YOLOv3의 인식 결과를 바탕으로 무인 판매대에서 사용자의 동작을 판단하여 어떤 상품을 구매/반환 했는지를 인식 할 수 있다.

3.4 무인 판매대의 실시간 동작을 위한 최적화 방안

본 연구에서 제안하는 무인 판매대 시스템이 정확하고 효율적으로 동작하려면 카메라 입력부터 구매 동작 판단까지의 일련의 과정이 실시간으로 이루어져야 한다. 본 연구에서는 무인 판매대의 실시간 동작을 위해 카메라 선택 알고리즘, multithreading, 그리고 pruning을 적용하였다.

3.4.1 카메라 선택 알고리즘

멀티 카메라를 활용한 system에는 다양한 시점에서의 detection을 통해 인식률이 개선되지만, 처리해야 하는 image 수의 증가로 수행 시간이 늘어난다. 6개의 카메라로부터 얻어진 image들에 대해 모두 CNN 연산을 이용하는 object detection을 수행하면 GPU로 가속하더라도 실시간으로 동작할 수 없다. 본 논문에서는 동작 속도 향상을 위해 각 frame마다 3 대의 카메라를 선택하여 이들 카메라에서 얻은 영상만 처리하는 카메라 선택 알고리즘을 적용하였다. 6개 카메라의 viewpoint로부터 얻은 image 중에는, 상품이 없거나, 상품이 매우 작아 detection에 불리한 image가 있다. 따라서 상품 detection에 유리한 카메라를 선택하게 되면, 연산량을 줄이면서 인식 성능을 유지할 수 있다. 제안하는 방법에서는 카메라에서 얻은 image에서의 물체의 크기와 물체의 위치를 분석하여 최적의 카메라를 선택한다.

움직임이 있는 영역에 구매 또는 반환되는 물체가 존재한다고 가정한다. 움직임은 기준 image와 현재 image의

difference를 이용하여 판단한다. Difference가 기준 threshold보다 큰 픽셀을 P_{diff} 라 한다. P_{diff} 가 많으면 해당 카메라에 보이는 물체가 크다고 할 수 있다. 그러나, P_{diff} 에는 물체가 아닌 손이 포함되어 있어 P_{diff}의 수로 물체의 크기를 추정하는 것에는 오류가 있다. 제안 방법에서는 손의 색상 정보를 이용하여 손에 해당하는 P_{diff} 를 제외한다. 처음 선반으로 접근하는 손에 물체가 포함되어 있지 않으므로, 각 camera에서 처음 접근하는 경우의 P_{diff} 에서 손의 HSV 색상 정보를 검출한다. n 번째 카메라에서 얻은 image에서 손의 영역을 제외한 P_{diff} 의 수를 $VarC_n$ 이라 한다. $VarC_n$이 크면 n번째 카메라의 image에서 물체가 크다고 판단한다.

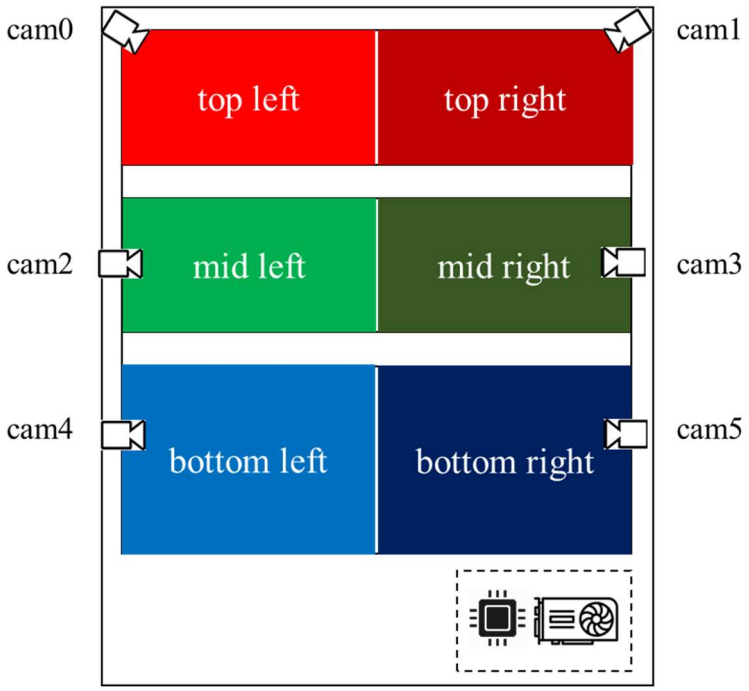
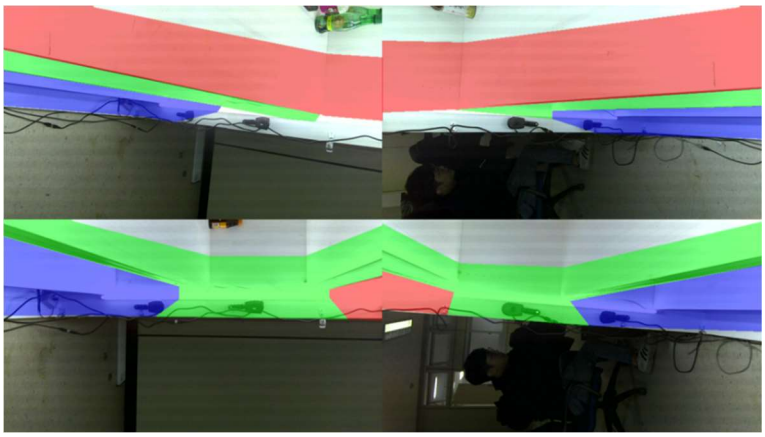


그림 3.5. 무인 판매대의 선반 영역과 카메라 번호

Grabbed object의 선반에서의 위치를 검출하고, 이 위치에 대한 카메라 시점을 고려하여, $VarC_n$ 에 가중치를 곱한다. 그림 3.5는 은 6개로 나눈 선반의 영역 및 cam_n 으로 표기한 각각의 카메라를 나타낸다. 제안 방법은 3 개 선반의 영역에 대한 P_{diff} 의 분포를 이용하여 상품이 꺼내진 선반을 검출한다. 그림 3.6 (a)는 top, mid, bottom 선반 영역을 각각 red, green, blue로 나타내며 그림 3.6 (b)는 left와 right 선반 영역을 각각 red와 green으로 나타낸다. 만약 mid에서 물체를 꺼내는 경우, cam_0 시점에서 볼 때 mid와 bottom 영역은 P_{diff} 를 갖지만, top은 P_{diff} 를 갖지 않는다. 이와 같은 방법으로, 상품이 꺼내지는 위치를 그림 3.5의 6 개 영역 중 하나로 판단한다. 물체의 위치가 결정되면, cam_n 에 대하여 detection에 유리하면 높은 가중치를 설정한다. n번째 카메라에 대한 가중치를 $Weight_n$ 으로 나타낸다.



(a)

(b)

그림 3.6. 무인 판매대의 선반 영역

(a) 카메라 0~3번의 상, 중, 하 영역, (b) 카메라 0, 1번의 좌, 우 영역

제안 방법은 물체의 크기와 위치를 고려하여 적합한 카메라를 선택한다. 각 카메라의 $VarC_n * Weight_n$ 을 계산하고, 이 값이 큰 3 개의 카메라를 선택한다. 제안된 카메라 선택 방법은 1/5 크기로 축소한 image에서 간단한 computer vision 연산들을 사용하기 때문에, 연산량이 적다.

3 개의 frame이 selection 된 후에는 3개의 frame에서 물체가 존재할 것으로 추정되는 ROI들을 선정하고, 이를 1 장의 image로 통합한다. 이 통합 image를 YOLOv3에 입력하여 1 회만 연산을 수행한다 [34]. 이 방법을 통해 매 frame단위마다 6 장의 image를 입력 받아 최적의 image를 선별하여 YOLOv3 network를 사용하여 실시간으로 인식할 수 있다.

3.4.2 Multithreading

제안하는 판매대 system의 연산은 크게 카메라 입력, object detection, 그리고 tracking의 3가지 부분으로 나뉘어져 있다. 각 부분을 순차적으로 수행하게 되면 시간이 오래 걸리게 되며 실시간을 달성할 수 없다. 본 연구에서는 3가지 연산을 각각 독립적은 thread에서 병렬로 수행하도록 구현하였고, 각 thread 사이에서 공유되는 연산 결과의 data양을 최소화하였다. 카메라 입력을 처리하는 thread는 6대의 카메라에서 매 frame마다 6 장의 image를 읽어온 후 이를 다른 두 thread가 읽을 수 있도록 buffer에 저장해 둔다. OpenMP[35]를 활용하여 6대의 카메라 data를 병렬로 read하여 실시간으로 동작할 수 있도록 하였다. Object detection thread는 buffer에서 6장의 image를 불러온 후, 카메라 선택을 통해 선택된 3개의 view에 대하여 YOLOv3 network 의 detection 결과를 저장한다. 마지막 thread인 tracking thread는 image 6장과 detection thread에서 저장된 정보를 읽는다. 읽어온 detection 결과와 image가 동일한 frame의 data가 되도록 guarantee한다. 읽은 data를 이용하여 tracking 연산을 수행하고,

상품의 구매행위를 판단하여 최종 결과를 출력한다. 그림 3.4는 Kiosk system에서 동작하는 알고리즘들의 multithreading 동작을 보인다.

3단계의 연산을 각각 독립적인 thread로 구성하여 얻을 수 있는 이점은 다음과 같다. 제안 시스템은 GPU를 활용하는 CNN 연산과 CPU를 활용하는 computer vision 알고리즘 연산을 필요로 한다. 이러한 시스템에서 CPU 연산 및 GPU 연산을 multithreading을 통해 병렬로 수행하여 효율적으로 resource를 사용할 수 있다. 즉, data-level parallelism (GPU)와 task-level parallelism (CPU)를 통하여 latency hiding을 최대로 하였다.

3.4.3 Pruning

본 연구에서는 YOLOv3 연산 시간 단축 및 convolution weight parameter를 위한 memory space를 줄이기 위해 pruning 기법을 적용하였다. Network slimming [36] 기법을 활용한 sparse training을 적용하여 weight parameter의 channel-level sparsity를 얻었다. Sparse training은 YOLOv3의 loss function을 식 (3.1)과 같이 재구성한 후 학습하여 얻을 수 있다.

$$L_{sparse} = L_{yolo} + \lambda \sum_{\gamma \in \Gamma} |\gamma| \qquad (3.1)$$

L_{yolo} 는 기존 YOLOv3에서 사용되는 loss function 을 의미하며, γ 는 batch normalization (BN) layer에서 활용하는 scale factor이고, λ 는 sparsity-induced penalty를 제어하는

상수이다.

기존 network slimming 방법에서는 classification CNN에 대한 automatic pruning을 적용하였다 [37]. 그러나 object detection CNN의 경우, automatic pruning을 적용할 경우 특히 feature pyramid network (FPN)에서 over-pruning이 일어나 정보가 많이 손실된다. 따라서 weight parameter의 효율적인 pruning을 위해 predefined pruning [38]와 automatic pruning을 결합하여 사용하였다. 각 BN layer는 global redundancy를 바탕으로 한 fixed predefined ratio로 pruning 하였고, pruning 이후 fine-tuning을 통해 정확도를 다시 올리는 과정을 적용하였다. 기존 YOLOv3의 leaky ReLU activation 대신, fine tuning 과정에서는 ELU activation function[39]을 사용하였다.

3.5 무인 판매대 시스템 성능 평가

본 연구에서 구현한 무인 판매대 시스템의 성능을 평가하기 위해 training한 YOLOv3 network의 인식 성능을 측정하였고, 판매대 사용 scenario를 구성하여 구매 및 반환 행위 판단의 정확도를 평가하였다. 마지막으로 전체 system의 실시간 동작을 평가하였다. 전체 무인 판매대 시스템의 알고리즘은 1 대의 pc로 구동되며 Intel i5-8500 CPU와 Nvidia GTX 1080Ti GPU 1 개, 그리고 16GB RAM을 사용하였다.

3.5.1 Object Detection 성능 결과

본 연구에서 사용한 YOLOv3 network는 batch 48, momentum 0.9, learning rate 0.001, decay 0.0005로 120,000 번 iteration을 하여 training을 하였다. YOLOv3의 feature extraction은 Darknet-53을 사용하였다 [40]. Pruning을 적용한 sparse training은 같은 조건으로 40,000 번 iteration을 하여 training 하였다.

본 연구에서 사용한 YOLOv3의 성능을 평가하기 위해, 48개 class에 대해 각 class 마다 약 220 번 등장하는 91,368 장의 image로 test set을 구성하였다. 해당 test set 중에서는 물체가 전혀 등장하지 않는 image도 다수 포함되어 있다. 모든 image는 512×288 크기로 resize 되었다. 본 연구에서 사용한 training set으로 training 한 결과를 test set으로 평가한 결과는 다음과 같다. 그림 3.7은 학습된 YOLOv3 network의 precision

recall curve를 나타낸다. 학습된 YOLOv3는 confidence score는 0.6 에서 0.7553 의 precision과 0.5117 의 recall을 갖는다. Pruning을 적용하였을 때도 precision과 recall은 기존 학습과 큰 차이를 보이지 않는다.

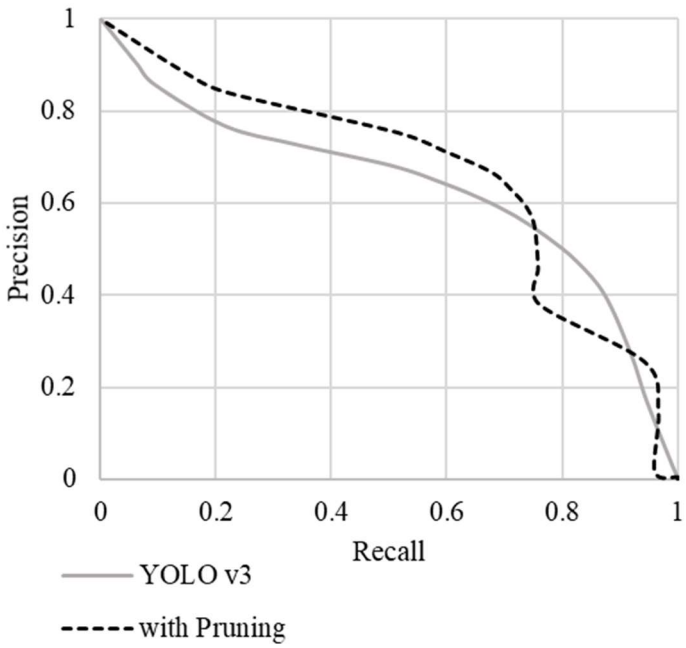


그림 3.7. 학습된 YOLOv3의 PR curve

3.5.2 무인 판매대 시스템 전체 결과

본 연구에서 구현한 무인 판매대의 성능 검증을 위해 다양한 scenario로 물건을 구매하는 video sequence를 활용하여 물체의 구매/반환 인식 성능을 평가하였다. 물건을 판매하고 반환하는 5 개의 scenario에서 구매와 반환에 대한 판단의 정확성을 평가하였다. 사용한 scenario의 일부 sequence는 그림 3.8에 나타나있다. 구매 scenario는 세 가지 상황을 가정하여 만들어졌다 각 scenario는 48 개의 상품에 대한 구매/반환 action으로 이루어져 있다. Scenario 1은 상품 한 개를 판매대에서 꺼내서 카메라 밖으로 가져가 구매를 하고, scenario 2는 상품을 판매대 안으로 집어넣어 반환을 하는 것으로 구성되어있다. Scenario 3는 물체를 다양한 각도로 기울여서 구매하고, scenario 4는 물체를 꺼내는 쪽 선반의 카메라가 가려진 상태로 구매하며 scenario 5는 물체를 최대한 손으로 감싸서 보이지 않도록 구매를 하였다. 각 Scenario에 대한 구매 상황 인식률 결과는 표 3.1에 나타나있다.

다양한 Scenario에서 평균 인식률 90% 의 높은 성능으로 다른 센서의 도움 없이 카메라와 computer vision algorithm 만으로 구매 행위를 정확히 판별하는 것을 볼 수 있다. 이는 object detection CNN만으로는 달성하기 어려운 높은 accuracy를 MDP-MOT tracking 및 3차원 object association을 통해 보정하여 달성한 것으로 볼 수 있다. 특히 기울어져서 물체를 구매한 scenario 2의 결과는 object detection network가 상당히 robust하게 detection을 수행하여 상품의 class와 bounding box를

전달함을 알 수 있다. 또한 scenario 3의 결과를 통해 물체를 주로 비추는 각도의 camera가 가려져도 MDP-MOT tracking algorithm을 통해 검출할 수 있는 detection 결과만으로도 구매 행위를 높은 인식률로 검출할 수 있음을 볼 수 있다.

그림 3.8. 무인 판매대 시스템 성능 평가를 위한 구매 scenario,
(a) scenario 1, (b) scenario 2, (c) scenario 3,
(d) scenario 4, (e) scenario 5

표 3.1. 각 scenario에 대한 무인 판매대 시스템 정확도

Scenario	1	2	3	4	5
No. of Frames	7,512	7,716	12,401	14,937	17,973
총 action 수	48	48	48	48	48
정답	46	46	42	43	39
오답	2	1	3	2	5
미검출	0	1	3	3	4
검출 정확도	95.83 %	95.83 %	87.50 %	89.58 %	81.25
평균 정확도	90.00%				

제 4 장 멀티 카메라 Dominant Feature Pooling

본 연구에서는 멀티 카메라 object detection network를 위한 dominant feature pooling 방법을 제안한다. 다양한 시점에서 얻은 feature들 중에서 dominant한 feature를 선택한다. Dominant feature들만 포함하는 새로운 patch를 생성하고, 새로운 patch는 기존 object detection layer의 중간에 있는 detection layer의 입력으로 사용된다. Pooling된 feature들은 다양한 시점에서 획득한 정보가 포함되어 있기 때문에 단일 카메라를 사용하여 획득 한 feature들에 비해 object의 주요 feature들을 classification에 더 많이 활용하여 detection accuracy를 향상시킨다. 제안하는 방법에서는 먼저 위치 및 class에 대한 정보를 제공하는 YOLOv3 network를 이용하여 object를 감지한다. 여러 카메라의 object bounding box가 클러스터되고 개체에 대한 클러스터의 feature vector가 scoring되고 pooling되어 새로운 feature patch를 생성한다. 단일 시점을 detection 하기 위한 기존의 YOLOv3는 이미 여러 시점에서 얻은 학습 dataset로 학습되기 때문에 제안하는 방법에서는 멀티 카메라에 대응하기 위한 추가적인 학습이 필요하지 않다. Pooling된 feature은 YOLOv3 네트워크의 선택된 layer들을 사용하여 classification되며, 선택된 layer들은 입력 및 출력 구성만 약간 변경되고 weight와 같은 기타 parameter들은 동일하게 유지되어 사용한다.

Dominant feature pooling 방법이 새롭게 제안하는 contribution은 다음과 같다.

1) 추가적인 학습이 필요 없는 멀티 카메라 object detection 방법을 제안한다.
2) 멀티 카메라 시스템에서 object detection accuracy를 효과적으로 개선하는 dominant feature pooling 방법을 제안한다.
3) 수 많은 channel을 직관적으로 파악할 수 있는 새로운 feature 시각화 방법을 제안한다.

4.1 Object Detection CNN과 멀티 카메라 Object Clustering

4.1.1 Object Detection CNN

본 연구에서는 제안 된 알고리즘의 기본 아키텍처로 YOLOv3 network [4]를 사용한다. YOLOv3 network의 아키텍처는 그림 4.1에 나타나 있다. YOLOv3 네트워크는 darknet53 classification network를 backbone으로 사용하며 최종 object detection 결과를 출력하기 위해 각각 7 개의 convolution layer와 YOLO layer가 있는 3 개의 detection layer set가 있다. 각 detection layer set은 서로 다른 크기의 bounding box를 detect하도록 설계되었다. 크기가 더 작은 bounding box를 detect하기 위해 각 detection layer set 사이에는 하나의 upsample layer가 포함된다. 본 논문에서 YOLOv3의 각 layer은 l_n 로 표기한다. 여기서 n 는

layer의 번호이고 detection layer set은 $D_{\{n_s,n_f\}}$로 표시한다. 여기서 n_s는 detect layer set가 시작되는 layer의 번호이며 그리고 n_f는 detection layer set의 최종 layer의 번호이다. 탐지 계층 집합 $D_{\{75,82\}}$, $D_{\{87,94\}}$ 및 $D_{\{99,106\}}$의 각 feature vector 하나는 입력 이미지의 32×32, 16×16 및 8×8 픽셀을 나타낸다. 각 detection layer set의 입력 feature vector의 채널은 1024, 768 및 364이다.

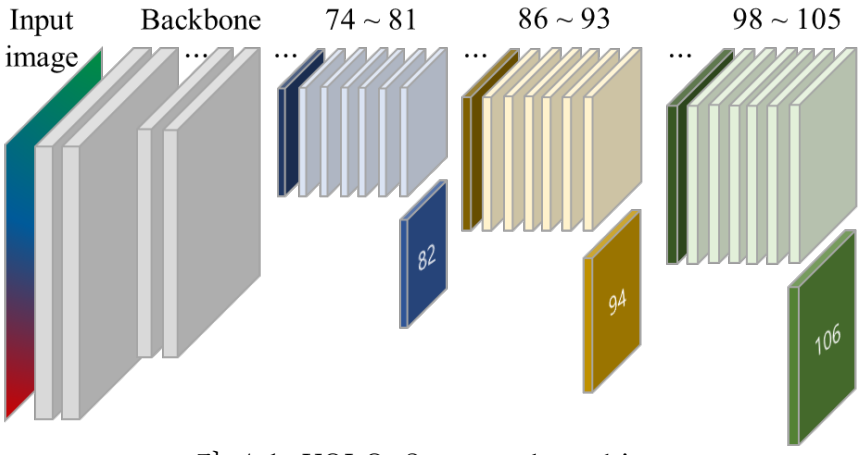


그림 4.1. YOLOv3 network architecture

4.1.2 멀티 카메라 Object Clustering

서로 다른 카메라에서 얻은 동일한 object의 feature들을 pooling하기 위해 먼저 YOLOv3로 detect한 object의 bounding box를 클러스터 한다. 먼저 유효한 bounding box를 결정해야 한다. YOLOv3 네트워크는 사전 정의된 probability threshold를 가진 bounding box를 예측한다. 최대한 많은 수의 target bounding box를 얻기 위해 본 연구에서 활용한 초기 detection threshold는 0.20으로 설정하였다. Objectness score가 0.20을 초과하는 bounding box 중에서 foreground 영역의 부분이 20% 이상인 bounding box는 유효한 box로 선택된다. 경험적인 관찰에 기초하여, 무인 판매대 환경에서 foreground ratio의 20% 임계값은 잘못된 위치에 있는 bounding box를 성공적으로 필터링했다. Foreground 마스크는 MOG2 background subtraction [41]에 의해 획득되며, 동작 시간을 줄이기 위해 입력 영상 크기의 1/4로 수행된다.

각 카메라의 유효한 bounding box들은 3.3장에서 사용된 것과 같은 방법으로 3차원 삼각법을 사용하여 클러스터된다. 3차원 삼각법을 사용하여 reprojection error를 계산하면 여러 카메라에서 촬영된 동일한 object를 하나로 묶어 클러스터로 분류 할 수 있다.

세 가지 다른 detection layer set $D_{\{75,82\}}$, $D_{\{87,94\}}$ 및 $D_{\{99,106\}}$는 feature의 채널의 수가 다르다. Feature을 단일 patch로 pooling하려면 모든 카메라에 대해 동일한 채널 수를 갖는 동일한 layer에서 feature를 추출해야 한다. 대상 layer은 클러스터의 유효한 bounding box를 감지하는 데 가장 많이 사용된 layer에

따라 선택된다. 각 유효한 bounding box는 세 가지 detection layer set 중 하나에서 탐지된다. 클러스터에서 가장 많은 수의 유효한 bounding box를 감지하는 layer가 target layer인 l_t 로 사용된다.

멀티 카메라 시스템의 일부 시점에 대한 YOLOv3는 output layer에서 낮은 detection score로 인해 대상 object의 유효한 bounding box를 출력하지 못했을 수 있다. 그러나 해당 시점에서 얻은 image에도 object class를 분류하는 데 사용할 수 있는 중요한 feature vector가 있을 수 있다. 이러한 누락된 feature vector를 얻기 위해 detection에 실패한 영상에서는 다른 시점의 detection 결과와 함께 3차원 reprojection을 사용하여 bounding box를 예측한다. 그림 4.2에 나타난 것처럼 파란색 점은 탐지된 유효한 bounding box의 중앙점을 나타낸다. 빨간색 점은 YOLOv3에 의해 올바른 box로 감지되지 않은 물체의 예측된 중심점을 나타낸다. 예측 중앙점은 이미지에서 올바른 inbound 2차원 좌표인 경우에만 사용된다. 유효한 bounding box를 선택하고 중앙점을 예측하면 한 프레임에 6개의 카메라에서 얻은 image들에 포함된 object의 클러스터가 생성된다.

본 연구에서 제안된 알고리즘은 멀티 카메라의 feature pooling을 통한 detection accuracy의 개선에 초점을 맞추었기 때문에 한 프레임당 하나의 object만 감지하도록 환경을 설계하고 간단한 클러스터 분류 방법을 적용하였다. 추후 다른 re-identification과 같은 알고리즘과 결합하여 한 프레임에 여러 object를 클러스터 할 수 있도록 개선하여 활용할 수 있으나 본 연구에서는 고려하지 않았다.

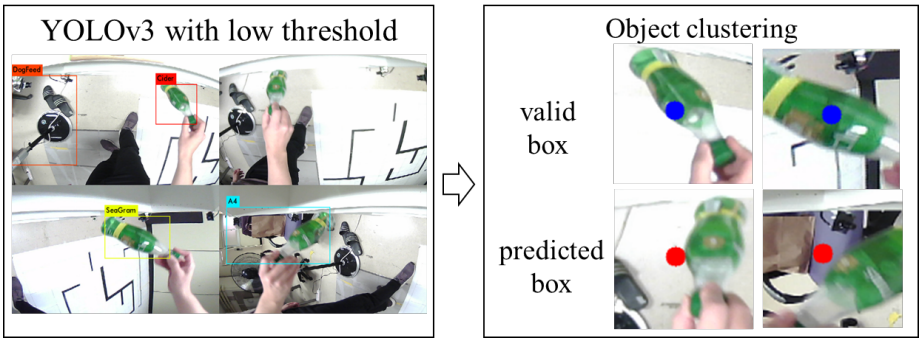


그림 4.2. 유효한 bounding box 선정을 통한 object 클러스터 방법

4.2 Dominant Feature Pooling 방법

멀티 카메라를 활용하여 detection accuracy를 높이기 위해 본 논문에서는 dominant feature pooling 방법을 제안한다. 그림 4.3은 제안하는 알고리즘의 순서도를 보여준다. Target object의 클러스터에서는 각 feature vector의 feature score가 계산되고 feature score가 낮은 feature vector의 순서대로 dominant feature vector로 선택된다. Feature vector는 클러스터에 대해 선택된 대상 계층에 따라 l_{74} , l_{86} 및 l_{98} 에서 추출된다. 이후, 선택된 YOLOv3 layer에서 선택된 dominant feature vector를 포함한 새로운 feature patch를 생성한다. 마지막으로, 생성된 feature patch는 YOLOv3의 detection layer set $D_{\{n_s,n_f\}}$ 의 입력으로 제공되며, l_{n_f} 의 출력 최대 class probability는 대상 object 클러스터의 class를 분류하는 최종 결과로 사용된다.

여러 카메라에서 feature vector를 pooling하는 이 방법의 아이디어는 YOLOv3 네트워크 분석에 기초한다. Object의 각 feature vector는 object class의 다양한 모양과 detail의 일부를 나타낸다. YOLOv3는 다양한 뷰포인트에서 얻은 대상 object의 이미지로 학습되기 때문에 object classification은 다양한 뷰포인트에서 얻은 다양한 feature vector로 학습된다. 단일 카메라의 이미지가 입력으로 제공되면 object detection 네트워크는 class probability를 활성화하는 일부 feature vector만 사용한다. 따라서, 다양한 뷰포인트에서 얻은 feature vector가 YOLOv3의 $D_{\{n_s,n_f\}}$ 에 대한 입력으로 제공된다면, object의 정확한 class를 활성화할 수 있는 feature vector가 더 많이 활용되고, classification accuracy가 향상될 수 있다.

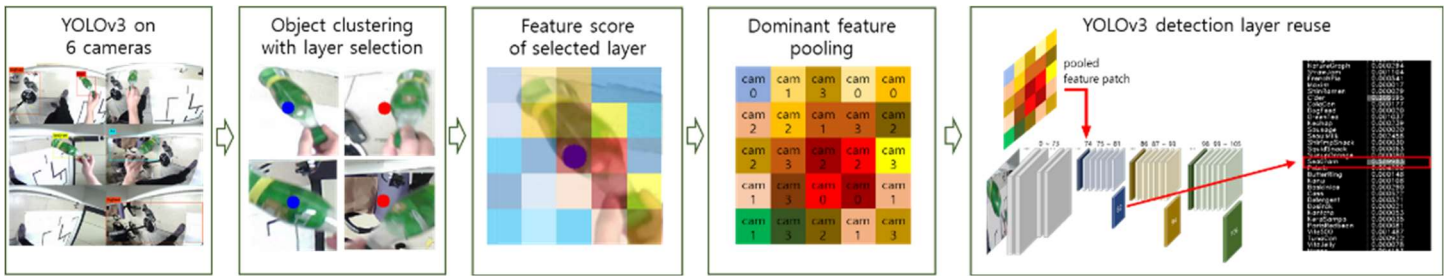


그림 4.3. 제안하는 알고리즘의 순서도

4.2.1 Dominant Feature Scoring

여러 카메라에서 feature vector를 pooling하여 유용한 feature patch를 얻기 위해, 본 연구에서는 클러스터에서 각 feature vector의 효과를 평가하는 dominant feature scoring 방법을 제안한다. 각 클러스터에서 여러 영상에서 대상 object의 중앙점은 YOLOv3 또는 3D reprojection으로 예측된다. 중앙점 주변의 각 feature vector에 대한 score가 제공된다. Scoring을 하는 feature vector의 boundary의 크기는 5×5의 bounding box 크기에서 선택한다. Feature score는 각 feature vector에 대한 정보를 표현하는 다양한 방법을 통해 얻은 score들을 종합하여 linear regression을 통해 얻게 된다. Linear regression을 통해 근사하는 target 값은 해당 feature vector 위치의 object class probability에 대한 L2 loss를 기준으로 하였다. 따라서 종합된 score가 낮을수록 feature vector가 정답에 대한 정보를 많이 포함하는 유용한 vector이며 $D_{\{n_s,n_f\}}$의 convolution을 통해 올바른 class probability를 활성화할 가능성이 높다는 것을 의미한다.

본 연구에서 feature scoring을 종합하기 위해 다섯 가지의 scoring method를 활용하여 linear regression을 통해 한 feature vector 당 한 개의 score 값을 부여하였다. 본 연구에서 활용한 다섯 가지의 scoring method는 간단하게 feature의 대푯값을 구하는 feature vector의 절대값의 합계, 해당 feature vector의 위치에서 prediction한 class probability의 entropy 두 가지와, 기존 active learning 분야에서 feature의 uncertainty를 구하기 위한 값인 mutual information score [42], view divergence [43],

그리고 view divergence를 배경 image와 연산하여 구한 background divergence 세 가지, 총 다섯 가지의 scoring method를 종합하였다.

Feature vector의 절대값을 나타내는 score인 S_{abs} 는 식 (4.1)과 같이 구한다.

$$S_{abs} = \sum_{k=0}^{c} |l_t[k]| \tag{4.1}$$

이때 $l_t[k]$는 target layer l_t의 k 번째 channel의 값을 의미하며 c는 대상 layer인 l_t의 channel depth 값을 의미한다. S_{abs}는 해당 feature vector가 얼마나 정보성이 있는 값을 많이 포함하고 있는지를 의미한다. 한 feature vector의 각 channel의 값이 다음 layer의 convolution에 많은 영향을 미칠 수 있는 정보성을 지니려면 해당 channel의 값이 커야 한다. 따라서 feature vector의 합을 구하면 얼마나 convolution에 영향을 많이 주는 channel 값이 많이 존재하는지 알 수 있다. 본 연구에서 절대값의 합을 이용하는 이유는 YOLOv3 네트워크의 convolution의 activation function은 leaky ReLU를 사용하기 때문이다. Leaky ReLU activation function은 음의 값의 output도 존재하므로 전체의 정보성을 판단하기 위해 절대값을 씌워서 합을 구한다.

Feature vector의 위치에서 prediction한 class probability의 entropy score인 S_{ent}는 식 (4.2)와 같이 구하게 된다.

$$S_{ent} = \sum_{k=0}^{Classes} -prob_{t+8}[k] * \log(prob_{t+8}[k]) \tag{4.2}$$

이때 l_t에서부터 시작되는 detection layer set의 마지막 layer는 $t+8$ 의 layer이며, $prob_{t+8}[k]$ 은 l_{t+8}에서 k번째 class의 class probability를 나타내며 $Classes$는 전체 class의 숫자를 나타낸다. S_{ent}는 해당 feature vector가 detection layer set을 통해 예측한 class probability가 얼마만큼의 확실성이 있는지를 나타난다. S_{ent}가 낮으면 해당 feature vector의 예측 probability의 분포는 정답이라고 여긴 class는 1에 가깝고, 나머지는 0에 가깝게 나타나 확실성이 높다고 볼 수 있고, S_{ent}가 높으면 여러 class에서 적당한 확률로 이 class 일 확률이 높다고 판단하는 probability의 분포로 나타나 확실성이 비교적 낮다고 볼 수 있다. 염두 해 둘 것은 확실성이 높다고 해서 해당 prediction이 정답일 확률이 높다는 점이 아니라 해당 vector의 convolution 결과로 예측된 값을 현재 network는 확신하고 있다는 의미에 가깝다는 점이다.

Mutual information을 활용한 score인 S_{mi} 는 Aghdam이 제안한 보행자 인식 binary network를 위한 active learning에서 uncertainty를 구하기 위한 방법을 사용하였다 [42]. Aghdam은 기존에 Bayesian network [44]에서 사용하던 uncertainty를 구하는 방법 [45], [46]을 CNN에서 활용할 수 있도록 변형한 후 각 pixel의 score를 매기는 방법을 제안하였다. 각 pixel에 매겨진 score는 각 pixel의 prediction probability가 서로 다른 조건에 반복된 prediction에서 얼마나 예측 결과가 일관되는지에 대한 uncertainty를 나타낸다. Pixel score가 낮다는 것은 해당 pixel의 근처 pixel들도 비슷하게 network가 prediction 결과를 냈다는 것을 의미하고, pixel score가 높다는 것은 주변의 prediction 결과에 차이가 많이 존재한다는 것을 의미한다. Aghdam은 해당

pixel score가 높은 pixel들을 active learning을 위해 추가로 label을 해야 할 신뢰도 낮은, 그러나 중요한 정보를 많이 포함하고 있는 위치의 pixel로 판단하여 추가로 해당 pixel들의 region을 구하여 해당 region을 labeling 할 대상으로 분류하는데 사용하였다. 본 연구에서는 pixel score의 개념을 가져와 해당 score가 높은 pixel들은 현재 convolution으로는 예측 결과가 헷갈리는 경향성은 있으나 해당 위치는 target object의 중요한 부분을 포함하고 있다는 것을 나타내는 것을 의미한다는 것으로 활용하였다. Aghdam의 연구에서 pixel score를 구하기 위해 먼저 pixel 위치 m,n 가 보행자일 확률을 나타내는 p_{mn} 를 여러 prediction block K 개에서 구한 후 probability matrix p_{mn}^{k} 로 나타낸다. p_{mn}^{k} 를 사용하여 pixel 위치 m,n 에서의 spatial probability distribution을 구하는 과정은 식 (4.3)에 나타나있다.

$$\hat{p}_{mn}^{k} = \frac{1}{(2r+1)^2}\sum_{i=m-r}^{m+r}\sum_{j=n-r}^{n+r} p_{ij}^{k} \tag{4.3}$$

식 (4.3)에서 r 은 spatial distribution을 구하기 위한 반지름을 나타낸다. 이후, probability matrix를 통해 score matrix s_{mn}^{k} 를 구하는 과정은 식 (4.4)에 나타나있다.

$$s_{mn}^{k} = \mathbb{H}\left(\hat{p}_{mn}^{k}\right) - \frac{1}{(2r+1)^2}\sum_{i=m-r}^{m+r}\sum_{j=n-r}^{n+r} \mathbb{H}(p_{ij}^{k}) \tag{4.4}$$

이때 binary classification에서 entropy를 구하는 함수 $\mathbb{H}(z)$는 식 (4.5)와 같이 나타난다.

$$\mathbb{H}(z) = -z\log z - (1-z)\log(1-z) \tag{4.5}$$

최종적으로 pixel score는 각 matrix의 score 값을 식 (4.6)과 같이 더하여 얻게 된다.

$$s_{mn} = \sum_{k=1..K} s_{mn}^{k} \tag{4.6}$$

본 연구에서 활용한 object detection network는 Aghdam이 활용한 network와 차이가 있기 때문에 몇 가지 변형을 하여 mutual information score로 사용하였다. 먼저 prediction block은 YOLOv3의 3 개의 각 detection layer set 마다 3 개의 anchor의 결과를 사용하므로 총 9개의 anchor를 사용하였고, 각 class, c 의 probability 를 binary classification처럼 사용하여 식 (4.5)와 같이 entropy를 구한 후 score $s_{mn,c}$ 를 구한 후 전체 class에 대해서 합을 식 (4.6)과 같이 구하여 최종 score를 나타내었다.

$$S_{mi}(m,n) = \sum_{c=0..Classes} s_{mn,c} \tag{4.6}$$

이렇게 구한 mutual information score인 S_{mi}는 feature vector scoring을 위한 score 값 중 하나로 사용된다.

다른 feature scoring 방법인 view divergence score S_{vd}는 Siddiqui의 연구에서 제안한 방법을 사용하였다 [43]. Siddiqui는 view point entropy를 통해 semantic segmentation을 위한 active learning 방법을 제안하였으며, label하기 위한 영역을 선택하기 위해 view entropy score와 view divergence score를 제안하였다.

View entropy score는 서로 다른 카메라에서 해당 영역을 얼마나 일관되게 판단하는 지를 평가하는 score이다. 판단이 view에 따라 inconsistent할수록 entropy score가 높게 나온다. View divergence score는 어떤 view가 가장 높은 uncertainty를 가지고 있는지를 판단하는 score이다. 본 연구에서는 view divergence score를 다양한 view에서 얻은 object를 어떤 view의 feature vector가 더 pooling시 효과적인 feature를 제공하는지 판단하기 위해 활용하였다. View divergence score를 얻기 위해 Siddiqui는 우선 image i의 m,n 위치의 각 pixel이 class c에 속할 확률을 segmentation network의 D 번 MC dropout 횟수를 통해 식 (4.7)과 같이 구한다.

$$P_i^{(m,n)}(c) = \frac{1}{D}\sum_{d=1}^{D} P_{i,d}^{(m,n)}(c) \tag{4.7}$$

이후 다른 image j에 해당 pixel 위치를 3차원 cross projection을 통해 얻은 새로운 위치 x,y에 해당하는 probability들의 집합인 probability distribution $\Omega_i^{(m,n)}$을 식 (4.8)과 같이 구한다.

$$\Omega_i^{(m,n)} = \left\{ P_j^{(x,y)}, j \;\middle|\; I_j(x,y)\ cross-projects\ to\ I_i(m,n) \right\} \tag{4.8}$$

Probability distribution을 구한 후 view divergence는 식 (4.9)를 통해 구한다.

$$VD_i^{(m,n)} = \frac{1}{\left|\Omega_i^{(m,n)}\right|}\sum_{P_j \in \Omega_i^{(m,n)}} D_{KL}(P_i^{(m,n)} || P_j^{(x,y)}) \tag{4.9}$$

View divergence를 통해 서로 다른 image에서 얻은 동일한 물체의 feature vector들 중 어떤 feature vector가 더 disagreeing 하고, 다양한 정보를 줄 수 있는지를 판단하는데 활용된다. 본 연구에서는 MC dropout 대신 mutual information에서 사용한 것처럼 서로 다른 anchor의 결과를 평균 내어 활용하였다.

마지막 scoring 방법인 background divergence score인 S_{bd}는 background image와 target image를 view divergence를 구하는 방법과 동일하게 연산하여 얻게 된다. 본 연구에서 활용한 무인 판매대 시스템은 고정된 카메라를 사용하므로 background image를 획득할 수 있다. Background image를 view divergence를 구하는 I_j로 여기고, I_j를 YOLOv3를 통과시켜 얻은 결과값을 통해 feature vector를 얻은 image I_i와의 divergence를 구하여 사용한다.

총 다섯 가지의 score를 바탕으로 본 연구에서는 feature vector마다 하나의 score를 부여하기 위해 linear regression을 활용하여 종합된 최종 score S_r을 구하였다. S_r을 regression하기 위해 dataset의 일부를 가져와 각 feature vector에 해당하는 S_{abs}, S_{ent}, S_{mi}, S_{vd}, S_{bd}를 모두 구한 후 정답 class 분포와 feature vector 위치의 YOLOv3 class prediction result 간의 L2 loss를 정답으로 두어 linear regression을 수행하였다. Linear regression을 통해 얻은 score S_r은 식 (4.10)의 형태로 나타난다.

$$S_r = W_0 S_{abs} + W_1 S_{ent} + W_2 S_{mi} + W_3 S_{vd} + W_4 S_{bd} + W_5 \qquad (4.10)$$

Regression 상수인 W_n은 dataset 마다 다르게 나타나며 각 score

값은 0~1사이 범위로 normalize 하여 사용하였다.

이렇게 얻은 최종 score는 L2 loss를 정답으로 regression 하였으므로 낮을수록 양질의 feature를 나타낸다는 의미의 feature 값으로 활용하였다. 카메라 n 에서 얻은 이미지에 pixel 위치 (u, v)에 존재하는 layer t의 feature vector를 $\hat{y}_{u,v}^{t,cam_n}$라고 표현한다. 그리고 해당 위치 vector의 score는 $S_r(\hat{y}_{u,v}^{t,cam_n})$라고 표현한다.

4.2.2 Dominant Feature Pooling

제안하는 dominant feature pooling은 각 feature vector에 점수를 할당한 후, 여러 카메라에서 얻은 모든 이미지의 feature vector들을 pooling하여 새로운 feature patch를 생성한다. Dominant feature pooling링의 개요는 그림 4.4에 나타난다. 그림 4.4 (a)에 표시된 것처럼 feature score $S_r(\hat{y}_{u,v}^{t,cam_n})$를 가진 모든 feature vector $\hat{y}_{u,v}^{t,cam_n}$는 그림 4.4 (b)에 표시된 것처럼 가장 작은 값부터 정렬된다. 정렬된 feature vector들은 새로운 feature patch로 pooling 된다. 각 detection layer set $D_{\{n_s,n_f\}}$는 네 개의 1×1 convolution과 세 개의 3×3 convolution으로 구성된 일곱 개의 convolution layer들로 구성된다. 세 개의 3×3 convolution을 거치는 detection layer set의 구조의 영향으로 pooling된 패치에서 최종 결과값을 출력하는 중앙 vector인 (u_c, v_c) 는 $\{u_{c-3:c+3}, v_{c-3:c+3}\}$ 범위의 feature의 영향을 받는다. 이 중 중앙 근처의 feature vector들은 일곱 번의 convolution 동안 3×3 convolution을 더 많이 수행하게 되므로 중앙 feature vector (u_c, v_c)에 더 많은 영향을 미친다. 따라서 feature vector는 그림

4.4 (c)와 같이 나선형으로 pooling되어 새 feature patch를 생성한다. 그림 4.4 (d)에는 해당 그리드 영역의 크기로 crop된 이미지를 사용하여 시각화된 새 feature pooled patch가 나타나있다. 각 그리드는 feature vector $\hat{y}_{u,v}^{t,cam_n}$를 나타낸다. 이 patch는 사람의 눈으로 보기에는 deformed된 형태의 이미지처럼 보이지만 pooled feature vector들은 다양한 관점에서 얻은 target object에 대한 풍부한 정보를 포함하고 있다. 중앙 그리드가 모든 feature vector들에 대한 convolution을 활용하기 위해 새로운 patch는 7×7 크기로 생성된다. Score가 부여된 feature vector의 총 수가 7×7 patch를 채우는 데 충분하지 않으면 patch의 나머지 부분은 0으로 padding된다. 단순화를 위해 그림 4.4에서는 5×5 크기로 나타내었다.

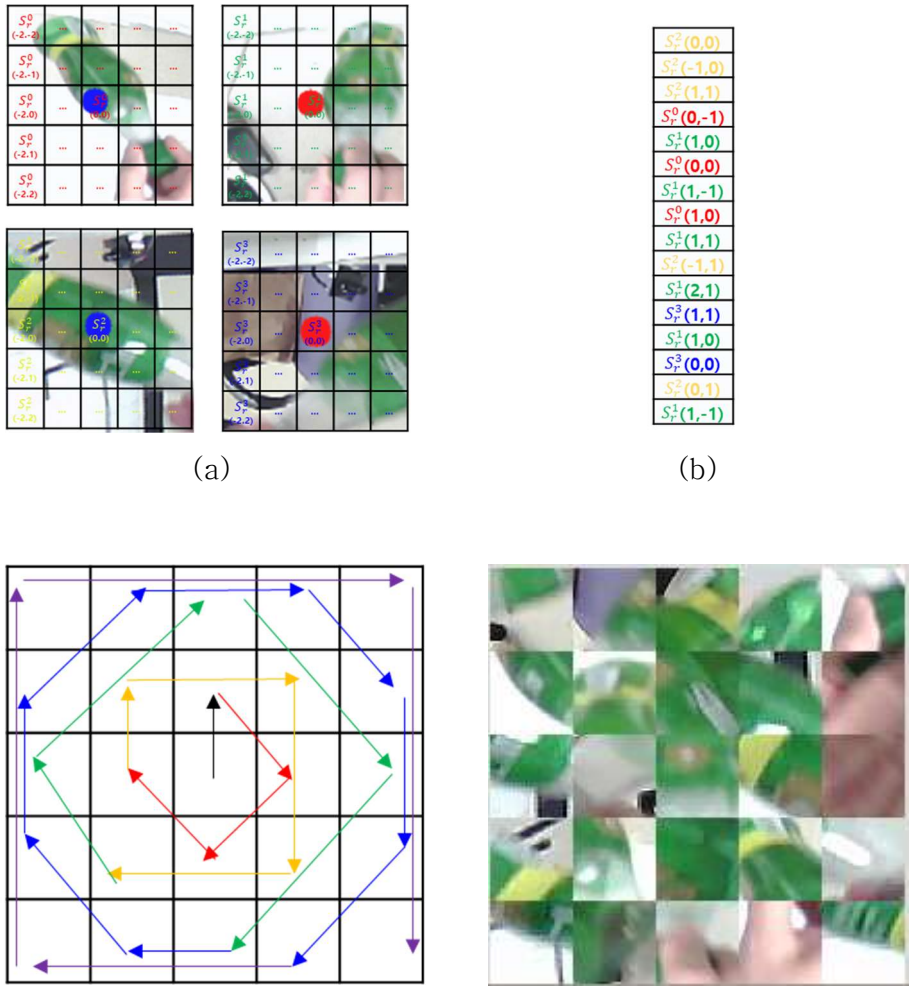


그림 4.4. Dominant feature pooling outline

4.2.3 YOLOv3의 Detection Layer 재사용

Dominant feature pooling 의해 얻어진 새로운 멀티 카메라 feature patch는 $D_{\{n_s,n_f\}}$에 대한 입력으로 제공되며, 각 object 클러스터의 class probability는 l_{n_f}의 출력에서 예측된다. 제안된 방법에서 YOLOv3의 $D_{\{n_s,n_f\}}$는 제안하는 dominant feature pooling 방법에 의해 구성된 새로운 feature patch의 classification을 위해 기존 YOLOv3의 일부를 재사용된다. 따라서 추가로 classification을 하기 위한 네트워크를 설계하고 및 학습하는 과정이 필요하지 않다. 새로운 네트워크를 설계하고, 이를 위한 dataset을 만들고 labeling 하는 힘든 추가 과정 없이 기존에 단일 카메라 dataset로 학습된 YOLOv3 네트워크를 사용하여 해당 dataset과 같은 class를 갖는 object를 detection 하기 위한 멀티 카메라 환경에서 classification 결과를 개선할 수 있다.

새로운 feature patch는 l_t의 feature vector $\hat{y}_{u,v}^{t,cam_n}$들을 모아 dominant feature pooling을 하여 생성되며, 이는 l_t의 다음 계층인 l_{n_s}에서 시작된다. $D_{\{n_s,n_f\}}$의 일곱 개의 convolution 레이어와 한 개의 YOLO 레이어는 bounding box 좌표, objectness 및 class probability의 prediction 결과를 포함하는 l_{n_f}에 output을 생성한다. 이 output 결과 중 l_{n_f}의 class probability는 새로운 feature patch의 class를 분류하는 데 사용된다. 본 연구에서는 새로운 feature patch의 중심 feature vector의 첫 번째 anchor의 최대 class probability를 최종 classification결과로 취한다. 각 $D_{\{n_s,n_f\}}$의 세 개의 anchor의 결과를 모두 취합하는 연구를 추가적으로

수행하면 더 좋은 결과를 얻을 수 있지만, 첫 번째 anchor만 사용하여도 classification accuracy를 향상시키기에 충분하다.

4.3 Feature 시각화를 통한 제안 방법 분석

Dominant feature pooling은 다른 시점을 가진 멀티 카메라에서 object의 다양한 모습을 나타내는 feature vector들을 아이디어를 기반으로 한다. Pooling된 feature vector가 classification accuracy 향상과 관련하여 효과적이라는 것을 입증하기 위해 feature vector를 시각화하는 새로운 방법을 제안한다. 제안하는 시각화 방법은 다수의 채널을 가진 feature vector를 이미지로 표현하여 직관적인 정보를 제공한다.

4.3.1 제안하는 Feature 시각화 방법

Feature 시각화 이미지는 다음 convolution layer에 대한 feature vector 영향을 표현하는 것이 중요하다. YOLOv3의 convolution layer는 activation으로 leaky ReLU를 사용한다. 따라서 절대값이 더 큰 feature 값은 convolution의 다음 layer에 더 많은 영향을 미친다. Feature vector의 모든 값이 그래프로 시각화 되면 feature vector의 분포 정보는 볼 수 있으나 한 눈에 직관적인 정보를 파악하기에는 너무 복잡하다. Feature vector의 모든 channel의 값을 모두 더하여 하나의 pixel로 시각화 하면 전체 feature map의 직관성은 증가하지만 channel 전체의 feature 값의 분포에 대한 정보는 손실된다. 제안하는 시각화 방법에서는 feature vector의 분포 정보 및 convolution에 끼치는 영향을 모두 직관적으로 표현하기 위해 RGB 채널이 있는 4×4 이미지로 하나의 feature vector를 시각화한다. 시각화 이미지는 I_v로 표시되고, I_v의

픽셀은 $p_{v(i,j)}$ 로 표시된다. N 크기의 channel을 가진 feature vector의 경우, 첫 번째 $N/16$개의 channel 값의 절대 합계는 픽셀 $p_{v(0,0)}$ 에 할당된다. 그림 4.5 (a)와 같이 1024개의 채널을 가진 feature vector의 첫 64개의 채널이 그림 4.5 (b)와 같이 RGB 채널로 표현되고, 그 결과는 그림 4.5 (c)와 같이 $p_{v(0,0)}$에 나타난다. 다음 $N/16$ 개 channel의 절대값은 $p_{v(1,0)}$ 에 나타난다. Feature vector의 나머지 값들은 같은 방법으로 $p_{v(i,j)}$에 나타난다. 그림 4.5 (d)는 5×5 feature patch의 시각화의 예를 나타낸다. I_v 의 전반적인 밝기는 feature vector의 전체적인 크기를 보여준다. 즉 I_v 의 전반적인 밝기가 밝을수록 해당 feature vector가 다음 convolution layer에 더 큰 영향을 미칠 가능성이 크다는 것을 의미하고, 전반적인 밝기가 어둡다면 다음 convolution에 미치는 영향이 비교적 작을 것이다 라는 것을 의미한다. 시각화 된 I_v 의 색상 패턴을 통해 여러 feature vector 간의 분포의 유사성을 쉽게 비교할 수 있다. 어떤 channel에서 feature vector가 피크 값을 가지는지, 전체 channel에 어떤 분포로 feature 값들이 존재하는지를 쉽게 파악할 수 있다. 이는 제안된 feature 시각화가 detail과 직관 모두를 활용해 feature vector를 분석하는 데 매우 유용함을 나타낸다.

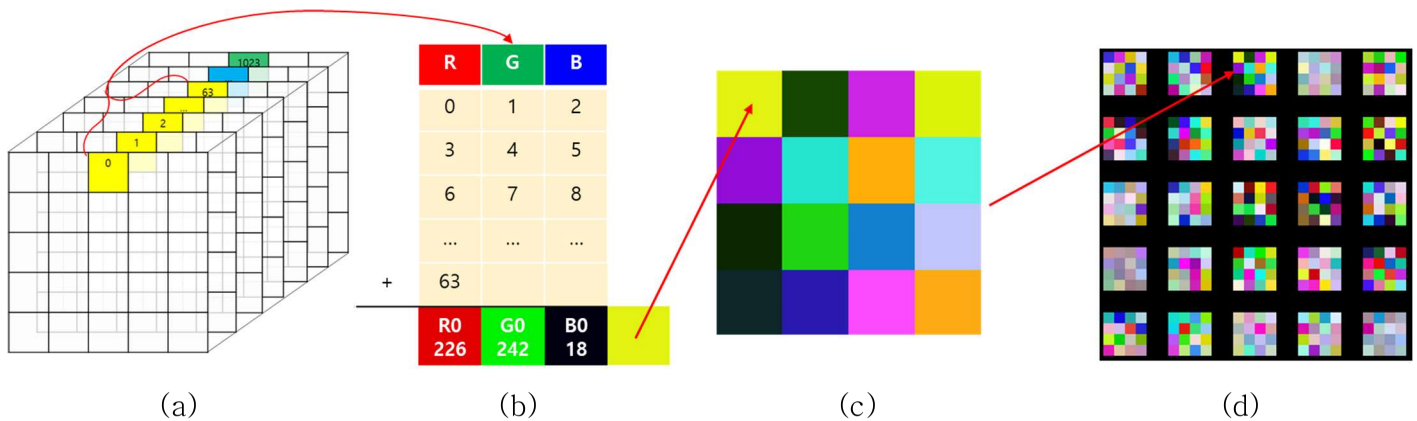


그림 4.5. 제안하는 feature 시각화 방법, (a) 5×5×1024 feature vector patch, (b) 첫 64개 channel 값의 합, (c) 하나의 feature vector의 시각화 표현, (d) 5×5 feature vector patch의 시각화 표현

4.3.2 기존 단일 카메라 YOLOv3 의 Feature 시각화

그림 4.6은 YOLOv3의 각 layer 별 feature vector를 시각화하여 보여준다. 그림에서, l_{74}~l_{80}의 feature vector의 시각화 및 l_{82}의 class probability를 보여준다. 시각화 이미지에서 선명하게 표현된 높은 밝기 픽셀은 해당 영역의 feature vector의 값이 크고 많은 정보를 제공한다는 것을 나타낸다. l_{74}의 I_v에서는 인식하고자 하는 물체인 Seagram이 있는 영역의 feature vector가 선명하게 시각화되어 많은 정보량을 포함하고 있음을 나타낸다. l_{75}, l_{77} 및 l_{79}의 I_v는 1×1 convolution의 결과를 보여준다. 1×1 convolution 후에도 이전 layer의 선명한 feature vector는 여전히 선명하다. 3×3 convolution에서 생성된 l_{76}, l_{78} 및 l_{80}의 시각화된 feature vector는 convolution에서 주변 feature vector에 의해 영향을 받는다는 것을 보여준다. Class probability는 해당 feature vector가 나타내는 grid에서 모든 class의 probability의 분포를 보여준다. Class probability 시각화의 경우 각 class의 classification probability를 시각화하기 위해 4×4 RGB에 각각 하나의 class의 probability를 mapping하여 I_v를 생성하였다. 그림 4.6의 class probability 시각화 image의 중앙의 빨간색 픽셀은 target object class인 Seagram의 class probability를 나타낸다. 두 번째 행의 grid에는 target object의 feature vector가 포함되지 않는다. 그러나 첫 번째 행의 target object의 feature vector와의 convolution을 통해 영향을 받아 target object class에 대해 높은 probability를 보여준다. 시각화 결과에서 알 수 있듯이, 정보량이 큰 feature vector는 높은 class probability를 생성하며 인접 grid의 class probability에도 영향을 미친다. 따라서 제안하는

방법을 통해 dominant feature vector를 pooling할 경우, 서로 인접한 feature vector들끼리 영향을 미쳐서 class probability의 정확도를 더 높일 수 있을 것으로 예상할 수 있다.

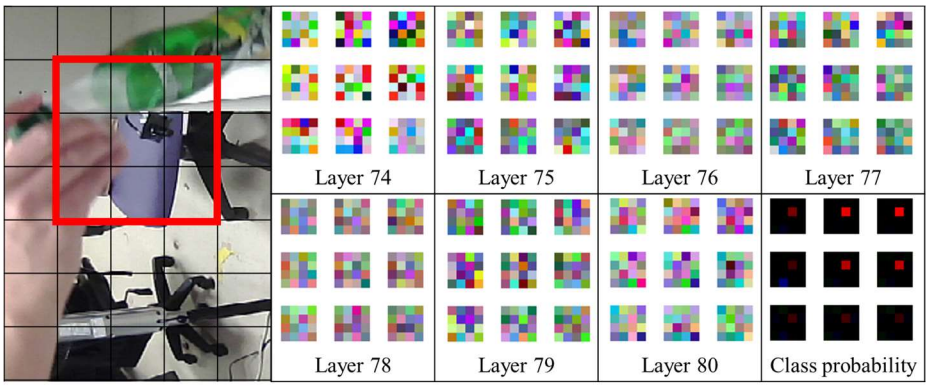
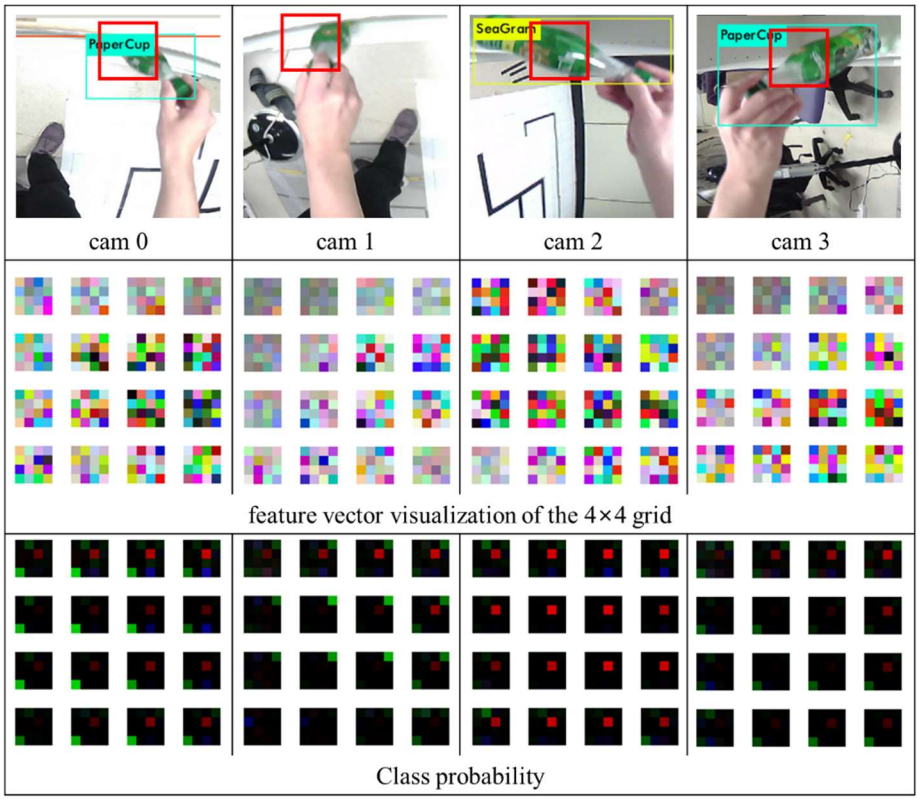


그림 4.6. YOLOv3의 layer 별 feature vector 시각화

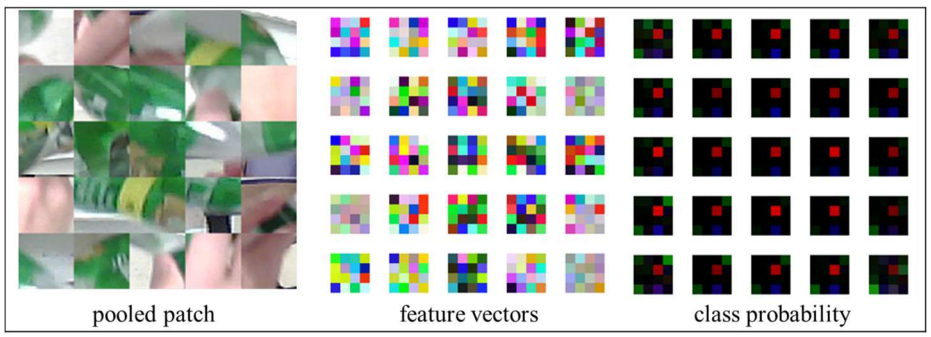
4.3.3. 멀티 카메라 Dominant Feature Pooling의 Feature 시각화

Dominant feature pooling의 효과를 관측하기 위해 서로 다른 카메라의 입력 및 pooling된 patch를 시각화한 image는 그림 4.7에 나타나있다. 그림 4.7 (a)에서와 같이 l_{74}의 feature vector는 target object의 영역에 선명하게 나타난다. Dominant feature pooling에 의해 얻어진 feature patch의 시각화는 그림 4.7 (b)에 나타나 있다. Target object 영역의 feature vector가 pooling 되어 선명한 시각화 픽셀로 pooled patch가 채워져 보다 정보량이 많고 유용한 feature vector가 pooling 되었음 보여준다. 이것은 제안된 dominant feature pooling 방법을 사용하면 feature vector가 성공적으로 pooling됨을 보여준다.

Dominant feature pooling이 class probability의 예측에 미치는 영향은 그림 4.7의 class probability 시각화로 나타난다. YOLOv3는 camera 2에서만 올바른 class인 Seagram을 인식한다. 그림 4.7 (a)에서 클래스 확률 I_v 의 빨간색 픽셀은 Seagram class의 확률을 나타내고, 왼쪽 하단 모서리의 녹색 픽셀은 paper cup class의 확률을 나타내며, 오른쪽 상단 모서리의 녹색 픽셀은 cider class의 확률을 보여준다. 이 feature vector들 중에서, 새로운 patch는 4개의 카메라들의 dominant feature vector들을 pooling하였고, 새로운 patch는 올바른 class인 Seagram이 가장 높은 probability를 보여 Seagram으로 분류되었다. 이 예시에서 볼 수 있듯이, 제안하는 dominant feature pooling 방법은 기존 단일 카메라의 잘못된 인식 결과를 수정할 수 있다.



(a)



(b)

그림 4.7. Dominant feature pooling의 시각화, (a) 각 카메라의 feature 및 class probability 시각화, (b) pooled patch의 feature 및 class probability 시각화

4.4 Dominant Feature Pooling 결과 및 분석

본 연구에서 제안하는 멀티 카메라 object detection을 위한 dominant feature pooling 방법을 grab-and-go 스타일의 무인 판매대 멀티 카메라 환경을 사용하여 실험하였다. 판매대에는 시야가 다른 카메라 6대가 설치돼 있으며, 48개 target 상품을 선정해 YOLOv3 network를 학습하였다. 그리고 48개 상품 중 임의로 18개의 object를 선택하여 dominant feature pooling을 검증하기 위한 test set을 구성하였다. 각 object class에 대해 순차적으로 카메라 6개를 활용하여 동시에 캡처한 후, 이 중 120개의 frame을 무작위로 선택하여 720개의 image를 생성하여 총 2160개의 frame, 12960개의 image로 test set을 구성하였다. 실험은 Intel Xeon E3-1245 CPU와 NVIDIA GeForce RTX 2080-Ti GPU로 수행하였다. 학습된 기본 YOLOv3 네트워크는 512×288 해상도 이미지에서 mAP50 42.77%의 성능을 보여준다. 이 연구에 사용된 probability의 threshold는 0.2에서 ground truth와 비교한 iOU 50% 박스의 검출 정확도는 39.57%이며, 검출된 총 bounding box의 분류 정확도는 41.53%이다.

또한 공개 dataset에서 결과를 검증하기 위해 다중 카메라 환경에서 COCO dataset [5]를 사용하는 새로운 test set을 구성하여 사용하였다. COCO dataset을 학습한 YOLOv3의 weight는 Redmon이 공개한 weight를 사용하였으며, 이는 mAP50 55.3%의 성능을 보여준다. 사용된 weight는 [40]에서 찾을 수 있다.

결과 비교를 위해 기존 멀티 카메라를 사용해서 결과를 개선할 수 있는 majority voting 방법과 multi camera CNN [6]의 feature pooling에서 사용되었던 max pooling 및 average pooling을 통한 방법의 결과를 함께 나타내었다.

4.4.1 COCO Dataset에서의 Dominant Feature Pooling 결과

공개 dataset에서 제안된 dominant feature pooling 방법을 검증하기 위해 COCO dataset의 object 중 12개의 object를 사용하여 멀티 카메라 test set을 구성하였다. 각 object에 대해 순차적으로 카메라 6대를 활용하여 동시에 캡처한 후, 100개의 frame을 무작위로 선택하여 class별로 600개의 image를 생성하여, 총 1200개의 frame, 7200개의 image를 사용하여 test set을 구성하였다. COCO test set에서의 dominant feature pooling의 결과는 표 4.1에 나타난다. Ground truth가 존재하는 1200개 프레임 중 majority voting method를 사용하면 1188개 frame이 detection을 출력한다. Feature pooling을 위한 clustering을 통해 false detection이 많은 frame을 제거하면 1017개 frame이 detection을 출력한다. 제안된 방법은 majority voting 대비 precision과 recall이 각각 14.69%, 4.17% 향상시킨다. 또한 average pooling을 적용했을 때에 비해 precision과 recall이 각각 10.33%, 8.75% 향상되었으며, max pooling을 적용했을 때에 비해 precision과 recall이 각각 3.64%, 3.09% 향상되었다. COCO 기반 test set의 결과를 통해 제안된 dominant feature pooling방법은 single camera detection의 false positive 결과를 제거하고 분류

정확도를 크게 향상시키는 데 효과적이라는 것을 볼 수 있다. 제안된 방법은 또한 recall의 향상을 통해 볼 수 있듯 정답을 획득한 frame 수가 증가하며, 잘못된 detection 결과는 dominant feature pooling을 통해 개선됨을 볼 수 있다.

표 4.1. COCO test set에서의 dominant feature pooling 결과 및 다른 방법과의 비교

	Majority voting	Average Pooling	Max Pooling	Proposed method
box detected	1188	1017	1017	1017
ground truth	1200			
correct boxes	663	608	676	713
Precision	55.80%	59.78%	66.47%	70.11%
Recall	55.25%	50.67%	56.33%	59.42%

4.4.2 Custom Dataset에서의 Dominant Feature Pooling 결과

무인 판매대를 위해 구성한 custom dataset을 활용한 dominant feature pooling 결과는 표 4.2에 나타난다. Ground truth가 존재하는 2160개 프레임 중 majority voting method를 사용하면 2048개 frame이 detection을 출력한다. Feature pooling을 위한 clustering을 통해 false detection이 많은 frame을 제거하면 1997개 frame이 detection을 출력한다. 제안된 방법은 majority voting 대비 precision과 recall이 각각 6.45%, 4.49% 향상시킨다. 또한 average pooling을 적용했을 때에 비해 precision과 recall이 각각 9.81%, 8.89% 향상되었으며, max pooling을 적용했을 때에 비해 precision과 recall이 각각 5.00%, 4.63% 향상되었다.

표 4.2. Custom dataset에서의 dominant feature pooling 결과 및 다른 방법과의 비교

	Majority voting	Average Pooling	Max Pooling	Proposed method
box detected	2048	1997	1997	1997
ground truth	2160			
correct boxes	1280	1181	1277	1377
Precision	62.50%	59.14%	63.95%	68.95%
Recall	59.26%	54.68%	59.12%	63.75%

4.4.3 Scoring Method 별 Dominant Feature Pooling 결과

본 연구에서는 다양한 scoring method를 통해 feature vector의 중요도를 평가하여 pooling하였다. 이러한 scoring method의 효과성을 보기 위해 다양한 scoring method를 적용하여 COCO test set에 dominant feature pooling을 수행한 결과는 표 4.3에 나타나있다. 표에는 중앙점과의 거리를 우선으로 pooling하는 center first, regression에 활용했던 다양한 score들인 absolute sum, mutual information score, view divergence 등의 값과 제안하는 regression score의 dominant feature pooling 결과값이 나타나있다. Regression score로 pooling을 하면 다른 방법들 대비 precision과 recall에서 모두 향상된 결과가 나타남을 볼 수 있다. 이를 통해 다양한 scoring method를 linear regression을 통해 하나의 score로 나타내어 이를 기준으로 pooling하는 것이 매우 효과적임을 볼 수 있다.

표 4.3. COCO test set에서의 score 별
dominant feature pooling 결과

	Center first	Absolute sum	Mutual information score	View divergence	Regression score
Correct Detections	628	684	581	634	713
Precision	61.75%	67.26%	57.13%	62.34%	70.11%
Recall	52.33%	57.00%	48.42%	52.83%	59.42%

4.4.4 Dominant Feature Pooling의 수행 시간 결과

제안하는 dominant feature pooling의 timing overhead는 표 4.3 에 나타난다. 총 6대의 다중 카메라의 object detection 전체 수행 시간은 평균 0.2241초이다. 유효한 bounding box의 분류와 클러스터링의 수행 시간은 평균 0.0078초인데, 이는 6대의 카메라에 대한 YOLOv3 detection에 필요한 수행 시간과 비교하여 겨우 3.49%의 overhead만 가진다. Foreground detection을 통해 유효한 bounding box를 결정하고 feature score를 계산하는 수행 시간은 0.13% overhead인 0.0093초이다. 제안하는 방법은 YOLOv3 detection 시간의 7.62%의 timing overhead만 필요로 한다. 본 연구에서는 OpenCV의 내장 function인 MOG2로 foreground detection을 처리했는데, 이를 더 간단한 알고리즘으로 대체하면 timing overhead를 추가로 더 줄일 수 있다.

표 4.4. Dominant feature pooling의 timing overhead

	Average time (sec)	%
YOLOv3 on 6 Cameras	0.2070	92.38
Foreground detection	0.0093	4.13
Clustering	0.0003	0.15
Dominant feature pooling	0.0044	1.98
Detection layer reuse	0.0031	1.36
Total	0.2241	100

제 5 장 Retinex Applied Object Detection

Object detection CNN은 다양한 환경에서 robust하게 동작하도록 네트워크를 설계하고 학습 dataset를 만들어 학습을 한다. 그러나 다양한 환경에 잘 대응하더라도 조도의 변화에는 취약한 경향을 보인다. 특히 저조도 환경인 경우 object의 dynamic range가 적어 detail이 충분하지 못하면 제대로 detection을 하지 못하는 문제가 있다. 무인 판매대와 같은 상황에서도 사람이 조명을 가리거나 물체에 그림자가 가리는 등 필요한 영역의 조도가 낮아 제대로 detection을 하지 못하는 상황이 생길 수 있다. 또한 object detection을 적극적으로 활용하는 자동화 창고나 자율 주행 환경과 같은 분야에어서도 저조도에서도 대응할 수 있는 object detection 방법이 있다면 다양한 실상황에서의 대응 능력이 올라가 성능을 크게 개선할 수 있다.

본 연구에서는 저조도 상황을 대응 할 수 있도록 화질 개선 알고리즘인 Retinex 알고리즘을 활용하여 object detection을 개선하는 방법에 대하여 연구하였다. 저조도 환경도 다양한 요인에 의해 발생하는데, 전체 dynamic range의 차이가 나더라도 Retinex 개선 학습을 통해 대응할 수 있는 robust한 object detection의 학습이 가능하다. 제안하는 학습방법을 통해 image conversion을 통해 얻은 저조도 영상의 Retinex 개선 학습으로 실사용 상황에서 저조도 영상의 object detection의 개선이 나타난다.

5.1 기존 Retinex 적용 연구

Retinex 알고리즘은 기존 object detection 영역에서 저조도, haze, underwater image 등 조명 성분을 제거해야 하는 환경을 개선하는데 사용되었다. 특히 Retinex 알고리즘을 연구하는 분야에서는 해당 알고리즘의 성능 평가를 위해 input image를 제안하는 Retinex 알고리즘으로 개선 한 후 object detection 성능의 변화를 토대로 제안한 알고리즘이 효과적이다 라는 것을 보여주는 성능 평가 지표로 활용하였다 [47], [48], [49]. 또한 어두운 환경으로 인해 인식이 어려운 지하철역 내부 CCTV 영상에서의 보행자 인식 연구 [50], 그리고 뿌연 물 속에 있는 물고기 숫자를 세기 위한 object detection 연구 [51] 등 나쁜 환경으로 인해 object detection이 제대로 되지 않는 경우 이를 개선하기 위해 Retinex를 적용하는 연구가 진행되었다.

Xiao[13]는 저조도 환경을 가정한 object detection dataset인 ExDARK [52]를 활용하여 Retinex 개선과 저조도 영상의 학습에 대한 관계를 분석하고, 저조도를 효과적으로 학습하기 위한 network 구조를 새롭게 제안하였다. Xiao는 Retinex를 활용해서 개선을 하여 object detection을 하는 것 보다 저조도 영상을 그대로 학습하는 것이 더 효과적임을 실험을 통해 나타내었다. Retinex를 적용하여 학습하는 것과 어두운 image를 그대로 학습하는 경우, 어두운 image를 학습한 것이 mAP가 조금 더 높게 나왔으며, 전처리에 활용되는 알고리즘의 연산량이 매우 크기 때문에 어두운 image를 학습하는 것이 더 효과적이라고 주장하였다. Xiao는 Retinex 개선을 통해 어두운 부분이 잘 개선 되지만

noise도 함께 개선되어 object detection에 영향을 미칠 수 있다고 분석하였다. 그러나 이 결과는 비슷한 저조도 양상을 띄는 ExDARK dataset을 training set과 test set으로 나누어 결과를 얻었으며, training set과 전혀 다른 양상의 저조도 영상이 입력으로 들어오는 상황에 대해서의 대처는 명시되어 있지 않다.

본 연구에서 활용하는 object detection은 무인 판매대 환경을 기반으로 하고 있으며, 판매대 특성 상 설치 장소, 그리고 판매대 주변 환경에 따라 조도 상황이 실시간으로 변하며, 그 모든 변화에 대응할 수 있는 저조도 dataset을 미리 만드는 것은 불가능하다. 따라서 다양한 dynamic range에 대응하기 위해서는 저조도 환경을 비교적 일관되게 출력해주는 Retinex 알고리즘을 활용한다면 조도 환경이 바뀌어도 object detection이 대응할 수 있다. 또한 저조도를 위한 training dataset을 다시 촬영하는 것에 비해 기존 training dataset의 밝기를 조정하여 어두운 dataset을 만들고 Retinex를 적용하는 것이 새로운 network를 학습하는데 드는 burden이 훨씬 적다. 따라서 밝기를 조절하여 dataset을 만들고 Retinex를 적용하여 학습을 하면 다양한 조도 환경에 대해서 기존에 촬영된 dataset을 가지고 학습하여 대응할 수 있다.

5.2 Retinex Applied Object Detection

5.2.1 Retinex Applied Object Detection 학습

본 연구에서는 무인 판매대의 저조도 환경에서의 object detection 및 dominant feature pooling의 효과를 분석하기 위해 기존 training set의 image의 intensity 값의 scale factor를 변경하여 저조도 set을 만들어 object detection network를 학습하는데 활용하였다. 학습을 위한 scaling factor는 0.1, 0.2, 그리고 0.5를 사용하였다. 같은 image를 0.1, 0.2, 0.5로 변환한 image는 그림 5.1에 나타난다. 또한 해당 image를 Retinex로 개선한 결과는 그림 5.2에 나타난다. 어두운 image의 평균 조도가 0.1과 0.5가 5배 차이 나는 것에 비해, 그림 5.2에 나타난 Retinex 개선 결과의 조도 차이는 0.1과 0.5 image 간에 2.03배 차이로 줄어들어 어두운 정도 대비 일관된 결과값을 나타내는 것을 볼 수 있다. 약 2.5배 차이 나는 0.2와 0.5의 경우도 Retinex로 개선하면 1.5배 차이로 줄어들어 다른 dynamic range의 조도도 Retinex를 통해 비교적 일관되게 출력할 수 있음을 볼 수 있다. 본 연구에서는 Retinex의 적용과 어두운 image의 학습에 대한 비교를 위해 원래의 조도로 학습한 YOLOv3, 저조도 image로 학습한 YOLOv3, 그리고 Retinex로 개선하여 학습한 YOLOv3에 대해 각각 test를 진행하여 비교하였다. 학습 조건은 momentum 0.9, decay 0.0005, learning rate 0.001로 진행하였으며, 어두운 training set은 90,000번 iteration, Retinex를 적용한 training set은 60,000번 iteration 하여 학습한 weight를 사용하였다.

(a)

(b)

(c)

그림 5.1. 서로 다른 scale factor 로 변환한 image,
(a) 0.1, (b) 0.2, (c) 0.5

(a)

(b)

(c)

그림 5.2. 서로 다른 scale factor 로 변환한 image 의 Retinex 적용 결과, (a) 0.1, (b) 0.2, (c) 0.5

(a) (b)

그림 5.3. 저조도 영상 획득을 위한 무인 판매대 환경,
(a) 조명을 켰을 때, (b) 조명을 껐을 때

(a) (b)

그림 5.4. 저조도 test set 예시,
(a) 조명을 켰을 때, (b) 조명을 껐을 때

5.2.2 Retinex Applied Object Detection 결과

Retinex를 적용한 object detection의 효과를 확인하기 위해 training set과 다른 test set을 제작하여 기존 object detection network, 어두운 dataset으로 학습한 network, 그리고 Retinex를 적용한 network를 비교하였다. 어두운 training set은 무인 판매대 환경에서 기본 조명을 어둡게 한 후, 판매대 위에 스탠드 조명을 설치하여 이 조명을 켠 것과 끈 것으로 밝고 어두운 차이를 두어 dataset을 획득하였다. 스탠드 조명이 설치된 무인 판매대의 그림은 그림 5.3에 나타나있다. 스탠드 조명을 끄게 되면 인식하고자 하는 대상 object가 어두워져서 기존 object detection network로는 인식을 하기 어렵다. 또한 밝은 test set도 같이 결과를 비교하여 Retinex를 적용했을 때 다양한 조도 환경에 대응할 수 있는지를 검증하였다. 저조도 환경에서 촬영한 test set의 예시는 그림 5.4에 나타나있다. 그림 5.4 (a)는 스탠드를 켜서 밝게 촬영된 test set을 보여주고, 그림 5.4 (b)는 스탠드를 꺼서 어둡게 촬영된 test set을 보여준다.

기존 dataset으로 학습한 original YOLO, 어두운 dataset으로 학습한 darkened, 그리고 Retinex를 적용하여 학습한 Retinex network의 YOLO 실험 결과는 표 5.1에 나타나있다. Retinex를 적용하여 학습한 network가 original과 darkened에 비해 precision과 recall이 개선되었음을 볼 수 있다. 특히 조명을 꺼서 어두워진 test set의 경우, precision은 original과 비슷하지만 recall이 2.13배 개선되어 훨씬 더 많은 정답을 획득했다는 것을 볼 수 있다. 어두운 training set으로 학습한 경우, 어두운 test set에서

recall의 개선이 이루어졌지만 굉장히 낮은 precision을 나타낸다. 이는 배경에 있는 다른 물체들을 잘못 인식한 false detection이 매우 많기 때문이다. Original YOLO가 Retinex에 비해 밝은 경우에도 더 낮은 성능을 기록한 이유는 밝은 test set과 기존 training set의 환경이 차이가 있기 때문으로 분석된다. 전체적으로 밝은 상황에서 촬영된 training set과 달리, 새로운 test set은 어두운 환경에서 가까이에 있는 광원으로 밝게 만들었기 때문에 물체가 광원과 가까우면 밝게 나타나지만, 광원 뒤쪽에 있는 부분은 어둡게 나오는 등 전체적으로 dynamic range의 분포에 차이가 있다. 따라서 original YOLO에 비해 Retinex를 적용하면 이러한 차이로 생기는 인식률의 저하에도 성공적으로 대응할 수 있음을 볼 수 있다.

표 5.1. 조도 변화 test set의 YOLO detection 결과

YOLO	Training method	Original	Darkened	Retinex
Bright test set	Precision	38.36%	8.34%	45.86%
	Recall	47.93%	19.69%	56.50%
dark test set	Precision	31.15%	19.69%	32.10%
	Recall	14.10%	17.75%	29.98%
all	Precision	36.48%	13.80%	40.09%
	Recall	31.12%	12.52%	43.40%

Retinex를 적용한 YOLO를 활용한 dominant feature pooling의 실험 결과는 표 5.2에 나타나있다. 다양한 각도에서 얻은 feature를 종합하는 dominant feature pooling method의 특성 상 YOLO의 인식률이 어느정도 되면 높은 precision과 recall을 보여준다. 표 5.2를 보면 original YOLO를 활용한 dominant feature pooling이 가장 높은 precision을 보여주고, Retinex를 적용한 dominant feature pooling은 original YOLO와 비슷한 precision을 보여주고, 높은 recall을 보여준다. 특히 어두운 test set의 경우 original 대비 25.6%, darkened 대비 33.2% 더 높은 recall을 보여 더 많은 정답을 획득했음을 볼 수 있다. 그림 5.5에는 어두운 test set에서의 각 network의 dominant feature pooling의 precision-recall graph를 나타낸다. Precision-recall graph의 특성 상 그래프의 우상단에 위치할수록 더 높은 성능을 나타내는 것을 의미한다. Original YOLO를 사용한 dominant feature pooling이 더 높은 precision을 나타내지만, 더 높은 recall을 기록한 Retinex를 활용한 dominant feature pooling이 그래프의 우상단에 더 가깝게 위치하는 것을 볼 수 있다. 이를 통해 Retinex 적용이 dominant feature pooling에서도 다양한 조도 환경에서 더 효과적임을 알 수 있다.

표 5.2. 조도 변화 test set의 dominant feature pooling 결과

DFP	Training method	Original	Darkened	Retinex
Bright test set	Precision	72.73%	15.95%	71.97%
	Recall	67.20%	12.00%	68.80%
dark test set	Precision	94.19%	51.67%	86.83%
	Recall	32.40%	24.80%	58.00%
all	Precision	78.55%	29.87%	78.08%
	Recall	49.80%	18.4%	63.40%

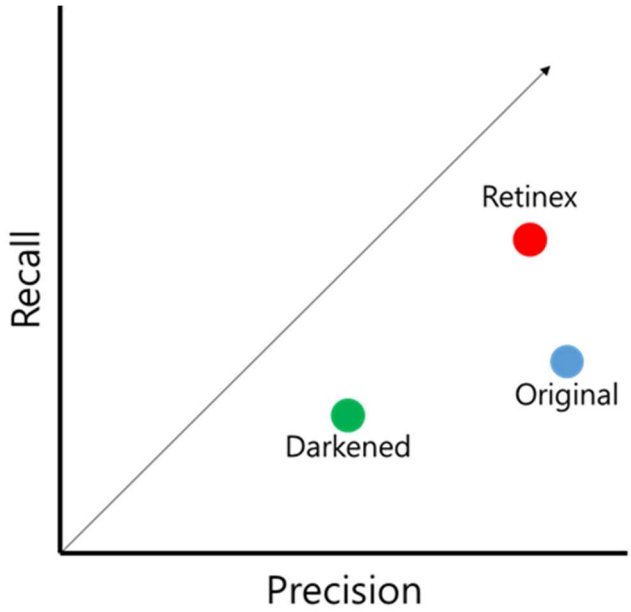
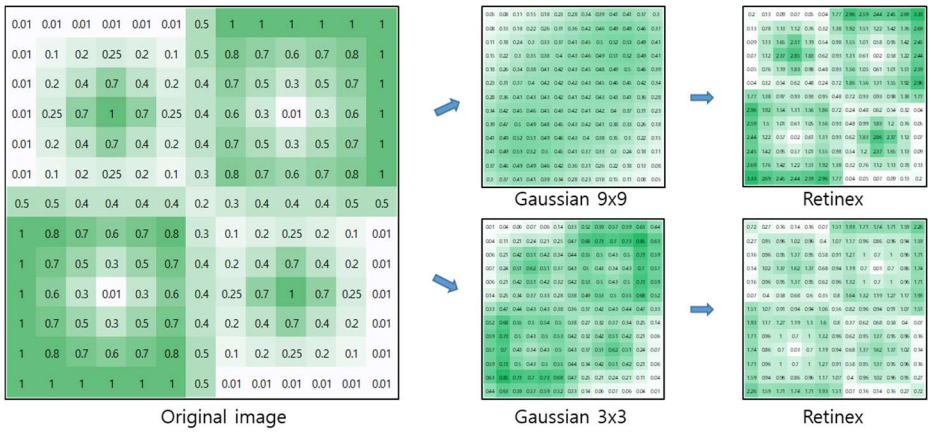


그림 5.5. Dark test set 에서의 precision 과 recall graph

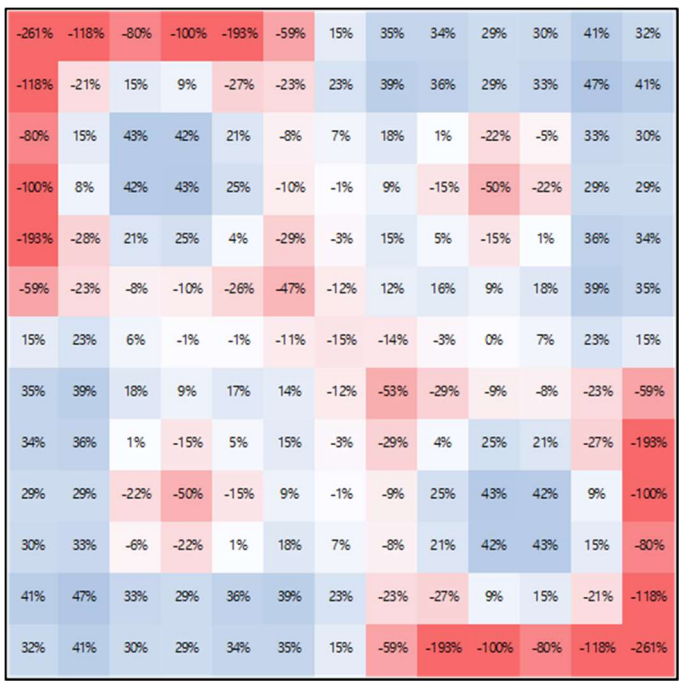
5.3 Object Detection을 위한 Retinex 최적화

5.3.1 Gaussian filter 크기에 따른 Retinex 효과 분석

Retinex 알고리즘에서 Retinex의 성능은 Gaussian filter kernel의 크기에 영향을 받는다. Gaussian kernel의 σ가 정해지고, 사용하는 kernel 크기가 변화하면 어두운 부분의 개선 효과에 영향을 미치며, 보통 kernel의 클수록 개선 효과가 크게 나타난다. 그러나 개선 효과가 큰 것은 출력 image를 사람이 눈으로 보았을 때의 결과이다. 출력 결과를 보면 kernel의 크기가 커지면 어두운 부분의 noise도 같이 밝게 되거나 halo effect가 생기는 등의 side effect도 생기게 된다. 또한 Gaussian filter를 연산하기 위한 kernel size가 커지면 이는 memory 사용량 증가와 연산 시간 증가로 이어지게 된다. Retinex 알고리즘을 통해 어두운 image를 개선하여 이를 최종 결과물을 사용하는 것과, object detection을 위해 network에 input으로 넣기 위한 전처리로 사용할 때 이러한 개선 효과와 side effect 간의 영향이 다를 수 있다. 따라서 object detection network에 활용하기 위한 Retinex의 최적화를 위해 kernel 크기에 따른 object detection의 결과를 분석하였다.



(a)



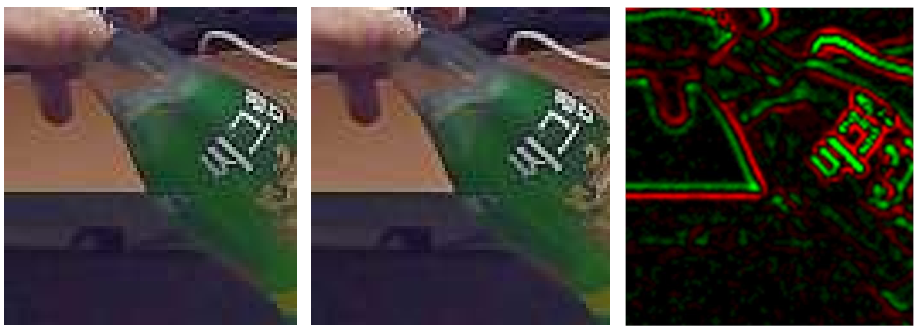
(b)

그림 5.6. Gaussian kernel size 에 따른 Retinex 의 차이,
(a) Gaussian filter 모양 및 Retinex 결과,
(b) 두 Retinex 결과 간의 차이

그림 5.6은 같은 σ값에 대해 서로 다른 크기의 Gaussian kernel이 Retinex에 미치는 영향을 어둡고 밝은 blob이 있는 example image에 대해 나타내고 있다. 그림의 입력 영상은 밝은 edge와 어두운 edge, 그리고 밝고 어둡게 blob을 생성하여 주변과 intensity 차이가 있는 대상 물체 주변에 Retinex가 어떤 영향을 미치는지를 볼 수 있게 만들었다. Gaussian kernel의 크기는 9×9와 3×3 두가지 종류를 사용하여 나타내었다. 그림 5.6 (a)는 두 크기의 kernel을 통과한 결과와, 해당 Gaussian filter 결과값을 사용한 Retinex 결과를 보여준다. 그림에서 볼 수 있듯이 Gaussian 크기가 크면 밝은 edge와 blob 부분이 더 변화가 큰 것을 볼 수 있다. 크기가 작은 Gaussian kernel을 사용하면 어두운 부분의 밝기 개선 효과는 볼 수 있지만 전체적으로 크기가 큰 kernel보다 어두운 edge가 더 밝아지는 것을 볼 수 있다. 그림 5.6 (b)는 두 크기의 kernel의 Retinex 결과값의 차이를 보여준다. 푸른 색으로 표기된 부분은 9×9 kernel의 결과값이 더 큰 것을 보여주고, 붉은 색으로 표기된 부분은 3×3 kernel의 결과값이 더 큰 것을 보여준다. 각 위치의 채도가 높을수록 그 차이 값이 더 크다는 것을 의미한다. 그림 5.6 (b)를 보면 밝은 edge와 blob 부분에서 크기가 큰 kernel을 썼을 때 밝기의 변화가 매우 크다는 것을 알 수 있다.

그림 5.7은 서로 다른 크기의 두 filter를 실제 image에 적용했을 때의 결과의 차이를 나타낸다. 그림 5.7 (a)는 3×3 filter를 적용했을 때의 Retinex 결과값을 나타내고, 그림 5.7 (b)는 7×7 filter를 적용했을 때의 결과를 나타낸다. 그림에서 볼 수 있듯이 7×7 filter의 결과값이 조금 더 선명하게 보이지만 아래 어두운 부분의 noise가 더 크게 부각되고, 물체 경계 부분의 halo

effect가 조금 더 뚜렷하게 나타나는 것을 볼 수 있다. 두 결과값의 오차는 그림 5.7 (c)에 나타나있다. 빨간 부분은 3×3 filter의 결과값이 더 큰 부분을 나타내고, 초록 부분은 7×7 filter 부분의 결과값이 더 큰 부분을 나타낸다. 그림에서 볼 수 있듯이 filter 크기가 크면 경계 부분에서 밝은 쪽에 halo effect가 더 크게 나타나고, filter 크키가 작으면 어두운 쪽에 halo effect가 나타나게 된다. 크기가 큰 filter가 사람 눈에는 더 선명하게 보일 수 있으나 이러한 noise 강화와 object 영역에 더 크게 나타나는 halo effect 등의 결과가 object detection CNN에는 다른 영향을 미칠 수 있다.



(a) (b) (c)

그림 5.7. 실제 촬영 영상에 대해 Gaussian kernel size 에 따른 Retinex 의 차이, (a) 3×3 filter, (b) 7×7 filter, (c) 오차

5.3.2 Gaussian filter 크기에 따른 Object Detection 결과

크기가 다른 filter를 쓴 Retinex로 개선한 object detection CNN의 결과를 위해 filter 크기를 변화하며 Retinex를 활용하여 실험을 통해 나타내었다. 크기가 큰 Gaussian filter를 활용한 Retinex 알고리즘으로 만든 training set으로 학습 한 후, 더 작은 크기의 filter 들로 test 했을 때의 영향을 통해 Retinex 알고리즘 전처리의 실사용 상황에서 더 작은 filter를 통해 연산량 감소 효과를 얻을 수 있는지 확인하였다. 표 5.3은 Gaussian kernel 크기에 따른 YOLO와 dominant feature pooling 결과를 나타낸다. 표에서 볼 수 있듯이 어두운 test set에 대해서 더 작은 크기의 filter를 적용하여도 YOLO와 dominant feature pooling의 성능에는 큰 차이가 없음을 볼 수 있다. YOLO의 경우, 크기가 작은 filter를 썼을 때 오히려 소폭 precision과 recall이 증가하는 양상을 보여주며, dominant feature pooling의 경우 precision은 소폭 하락하였으나 recall은 소폭 증가하는 등 전체적으로 1% 이내의 차이를 보여준다. 7×7 크기의 filter 대비 5×5 filter와 3×3 filter는 각각 Gaussian filter 수행 시간을 75.8%, 58.9%로 효과적으로 단축시킬 수 있다. 따라서 object detection CNN에 Retinex를 전처리로 적용하는 경우, filter size를 줄여서 적용하면 성능에는 큰 변화가 없이 연산량 감소로 인한 수행 시간 단축의 효과를 얻을 수 있다.

표 5.3. Gaussian kernel 크기에 따른 YOLO와 dominant feature pooling의 결과

Dark test set	YOLO		DFP	
	precision	recall	precision	recall
7×7	37.45%	25.39%	86.50%	53.80%
5×5	37.58%	25.59%	85.80%	54.40%
3×3	37.60%	25.64%	85.44%	54.00%

5.4 Retinex 하드웨어 시스템의 필요성 및 기존 연구

Retinex 알고리즘은 어두운 영역을 개선하며 동시에 image의 detail을 개선하는데 탁월한 성능을 가진 것으로 알려져 있으며 영상에서 활용될 가치가 매우 높다고 여겨진다 [53], [54]. Object Detection과 결합했을 때에도 Retinex는 저조도 환경에서의 detection 성능 향상에 도움이 되며 제안하는 dominant feature pooling의 성능 향상에도 도움이 되는 것을 볼 수 있다. 그러나 Retinex 알고리즘은 computational cost가 매우 높아서 실시간으로 동작하기가 어렵다는 문제를 갖는다. Object detection CNN보다 Retinex 알고리즘의 수행 시간이 약 15배 더 많이 들게 된다. Retinex 알고리즘을 최적화하더라도 object detection CNN의 전처리로 활용하기에는 지나치게 연산량이 많다.

그러므로 이러한 Retinex algorithm을 비롯한 adaptive approach들을 전용 hardware로 (HW) 구현하여 높은 computational cost 문제를 해결하는 연구들도 활발히 진행되었다. Y. Li [55], D. Ustukov [56], 그리고 S. Marsi [57] 등의 연구는 Retinex 알고리즘을 기반으로 한 영상처리 알고리즘이 실시간으로 동작될 수 있도록 FPGA를 활용하여 HW로 구현하였다. 이러한 기존 연구들은 throughput 측면에서는 당시 실시간 동작을 달성했지만 다음과 같은 3 가지 이유로 여전히 상대적으로 많은 HW resource를 사용하기 때문에 다른 application과 함께 사용되거나 모바일 기기 내부에서 사용되기에는 무리가 있다. 첫 째, 입력 영상을 memory에 저장한 후 사용하기 때문에 (frame buffer) input과 output 간의 latency가 존재하는 단점이 있다. Li 와 Marsi

는 external memory인 SRAM을 사용하여 image를 저장하였고, D. Ustukov 는 내부 BRAM에 image를 저장하여 사용하였다. External memory를 쓰게 되면 memory access를 위한 overhead가 생기고 추가적인 SRAM controller를 활용 해야한다. 또한 내부 BRAM에 image를 저장해서 사용하게 되면 처리 가능한 image size가 FPGA의 최대 BRAM 크기로 제한되며 한정된 BRAM을 많이 사용하기 때문에 다른 모듈의 구현을 어렵게 만드는 문제점이 있다. 둘 째, 기존의 HW 구현 연구들은 모두 성능 향상을 위해 Multi-Scale Retinex (MSR)를 대상으로 하였는데 MSR과 같은 경우, 여러 Retinex의 결과를 활용해야하기 때문에 연산량이 많아서 결국 상대적으로 많은 HW resource를 필요로 한다. 셋 째, 성능과 HW resource 간의 trade-off를 고려하지 않은 채 구현이 되었기 때문에 구현의 효율성이 매우 낮다.

이런 기존 Retinex 알고리즘의 HW 관련 연구들의 문제점을 보완하기 위하여 본 연구에서는 MSR이 아닌 Y. Shin [27]이 제안한 Efficient Naturalness Restoration 알고리즘을 FPGA target으로 하여 구현함으로써 low-complexity real-time Retinex HW IP를 제안한다. 선택된 알고리즘은 자연스러운 색감, 뛰어난 색 복원력, 그리고 MSR을 비롯한 다른 Retinex algorithm 대비 적은 연산량의 세 가지 장점을 갖기 때문에 저연산 real-time으로 의료용 영상을 개선하고자 하는 목적에 잘 부합한다. 뿐만 아니라 제안하는 HW는 기존의 연구들의 과도한 memory 사용 및 latency 문제를 stream 방식으로 image data를 주고 받는 방법을 적용하여 해결하였다. 또한 본 연구에서는 hardware resource를 효율적으로 사용하기 위해 성능과 HW resource 간의 trade-off를 고려하여

approximate computing [58] 개념을 적용함으로써 내부 모듈들의 경량화를 진행하였다. 알고리즘에서 가장 연산량이 많은 부분인 Gaussian filter에서는 최적의 σ값을 정하여 filter size를 맞췄고, 외부 memory를 사용하지 않도록 line buffer만을 이용하여 latency를 최소화 할 수 있도록 구현하였다. 또한 resource를 많이 차지하는 multiplication 기반의 Gaussian filter를 shifter와 adder로 구현하여 approximation을 하였다 [59]. 그리고 naturalness restoration의 경우, exponentiation 연산이 중요한 역할을 하게 되는데 이 연산이 non-trivial operation일 뿐만 아니라 HW로 구현하기에 굉장히 어렵기 때문에 이를 효율적으로 구성하는 새로운 방법을 도입하여 구현을 하였다.

본 연구의 또 하나의 강점은 실용성을 고려하였다는 점이다. 전체 시스템이 FPGA 상에서 동작할 때, HDMI/DVI 형식의 영상 입출력 장치와 호환 되도록 구현되었기 때문에 보편적으로 사용되는 1920×1080 (FHD) 해상도의 60fps 영상 입출력 장치와 실시간 호환이 가능하므로 활용도 및 활용 범위가 매우 넓다.

5.5 제안 하드웨어 시스템 구현 개요

본 연구에서는 Y. Shin [27] 이 제안한 naturalness restoration algorithm을 RTL 구현을 통해 저연산으로 설계하였으며 FPGA 상에서 실시간 동작을 검증하였다. Naturalness restoration 알고리즘의 FPGA 구현이 challenging한 이유는 Gaussian filter와 exponentiation과 같은 HW 구현이 까다로운 연산들을 포함을 하기 때문이다. 뿐만 아니라 naturalness restoration 알고리즘에는 frame 전체 단위로 보고 정해야 하는 변수들이 있어서 frame memory를 사용하지 않고는 구현이 어렵다. 그림 5.8은 이런 어려운 점을 효율적으로 처리한 제안하는 화질 개선 알고리즘 IP의 block diagram은 보여준다.

제안된 IP는 총 7개의 main module로 구현하였다. 각 module은 화질 개선 알고리즘을 최소한의 HW resource 만으로 실시간 처리 할 수 있도록 구현하였으며, 특히 latency를 줄일 수 있도록 frame buffer 없이 최소한의 line buffer만을 사용하도록 data를 입력단계부터 각 module 사이에서 stream으로 처리하였다. 또한 image 전체를 읽은 후에 나온 값을 사용해야 하는 data는 이전 frame의 결과를 사용하여 frame 단위의 latency가 생기지 않도록 구현하였다. 뿐만 아니라 영상 입력을 받은 시점부터 1 cycle당 1 pixel의 높은 throughput을 유지할 수 있도록 구현하였다. 전체 시스템은 Verilog HDL과 Xilinx 제공 IP 로 구현하였으며 각 모듈 간의 data의 처리는 AXI4 Stream Bus [60]로 구현하였다.

각 모듈들의 역할은 다음과 같다. separate RGBL module은

image를 입력 받은 후 intensity channel을 분리하는 역할을 한다. 이때, intensity channel은 RGB channel 중 최대값을 구하여 생성한다. Gaussian filter module은 intensity channel을 Gaussian kernel과 convolution 하여 조명성분을 estimation을 하기 위한 F channel을 생성한다. Weighting map generation module은 calculate F_m module에서 F channel을 modify 하여 modified illumination channel인 F_m을 계산할 수 있도록 weight를 계산하는 역할을 한다. RGB creation module은 원본 영상의 R, G, B channel에서부터 F_m channel을 각각 나누어주어 반사성분 channel인 R_R, G_R, 그리고 B_R을 생성하는 module이다. Calculate F_{enh} module은 exponentiation 연산을 활용하여 F_m과 기존의 intensity channel의 평균 값을 이용하여 조명성분 channel에 adaptive gamma correction을 적용한 channel인 F_{en} 을 계산한다. 본 연구는 SW에서 double precision floating point로 구현된 알고리즘을 RTL로 구현하였다. 따라서 double precision floating point를 오차가 가장 최소화 되는 범위 안에서 HW 효율성을 높일 수 있도록 13bit fixed point로 구현하였다. RGB 입력을 8bit로 받았을 때 각 단계에서 가장 최적의 fixed point precision을 정하여 구현하였다. 각 module의 fixed point precision을 Q−format으로 나타낸 표는 표 5.4에 나와있다.

그림 5.9는 전체 block diagram에서 각 module을 거친 후의 출력 결과가 어떤지를 나타낸다. 그림 5.9의 $R(x,y)$ channel을 보면 Retinex의 효과로 원본 영상에서는 어두워서 보이지 않던 부분의 색이 확연하게 드러나는 것을 확인할 수 있다. 그러나 $R(x,y)$ channel의 결과값은 자연스러운 색감이 사라져 있고 밝기가

과장되어 나타나기 때문에 $F_{en}\ (x,y)$ channel과 종합하여 최종적으로 어두운 부분이 개선된 자연스러운 색감의 $I_{enh}(x,y)$ 를 얻게 된다.

표 5.4. 각 module의 출력에 대한 fixed point precision

Module	Integral bit	Fractional bit	Q−format
Gaussian filter	0	13	Q0.13
Weighting Map	0	13	Q0.13
Calculate F_m	1	13	Q1.13
RGB creation	5	13	Q5.13
Calculate F_{enh}	0	8	Q0.8

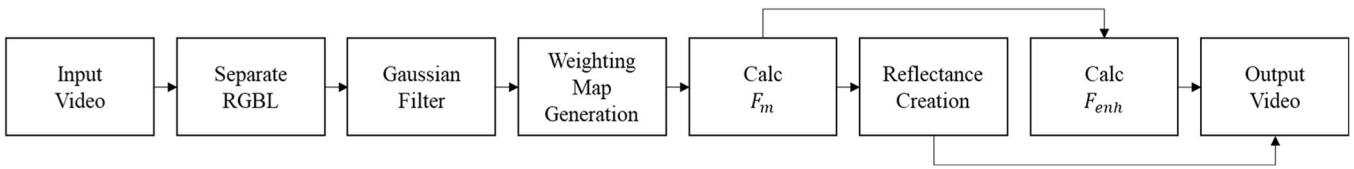


그림 5.8. 제안하는 하드웨어 시스템의 block diagram

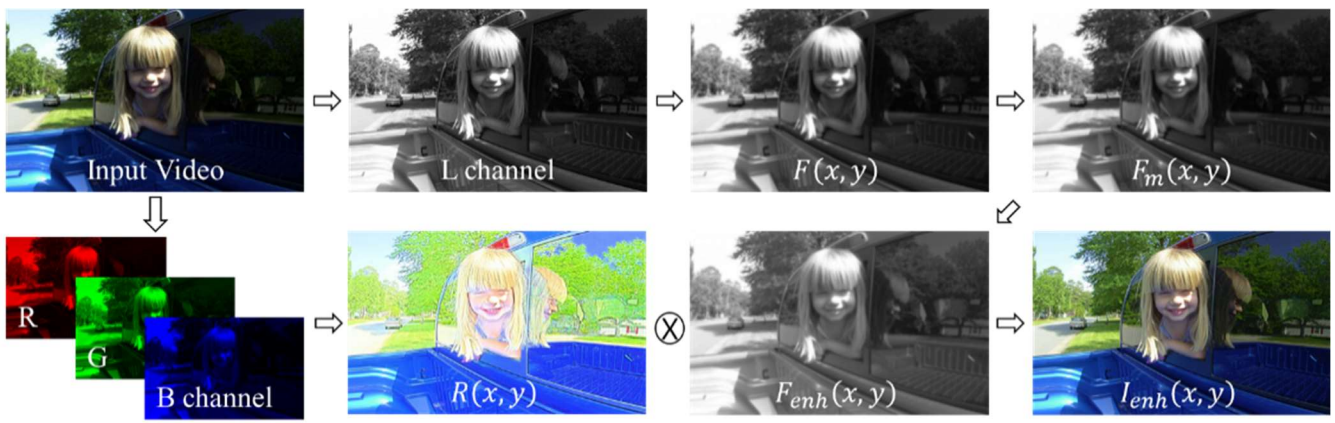


그림 5.9. Block diagram상의 각 module의 출력 결과

5.6 제안 하드웨어 시스템 구현 특장점

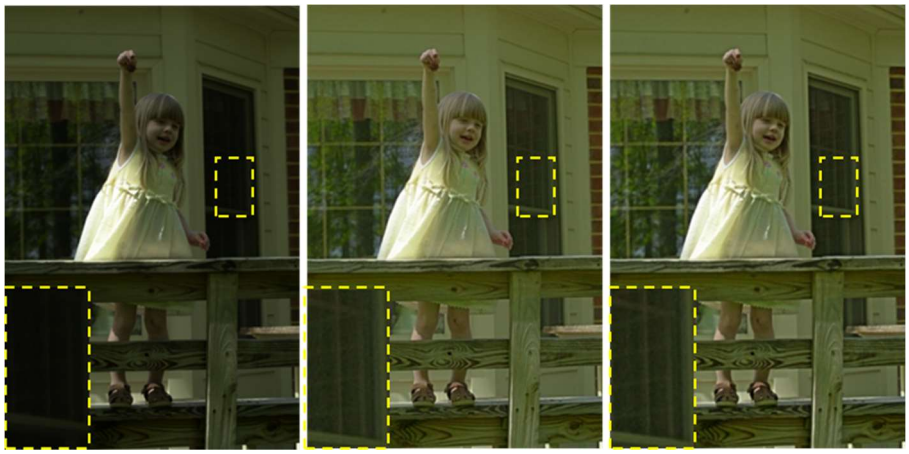
본 연구에서는 Retinex algorithm을 HW로 구현하기 위해 HW로 구현이 어려운 Gaussian filter, exponentiation을 효율적으로 approximation하였으며, 또한 모든 module이 최소한의 latency를 가지도록 최소한의 line buffer만을 사용하였다. 또한 실시간으로 동작하는 HW가 다른 상용 영상 장치들과 호환이 쉽도록 HDMI/DVI를 통해 data를 주고 받을 수 있도록 구현하였다.

5.6.1 Gaussian filter의 구현

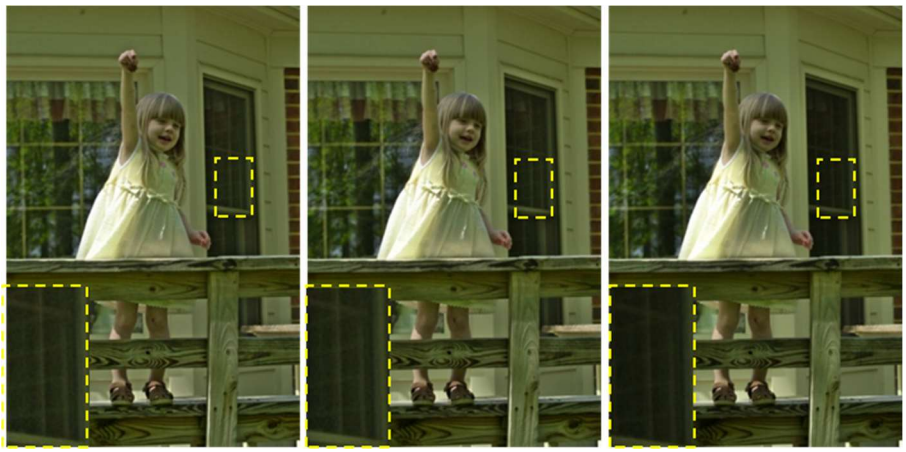
Retinex algorithm에서 화질 개선에 큰 영향을 미치는 부분이 Gaussian filter module이다. 특히, Gaussian filter의 σ 값이 전체 시스템의 화질 개선 정도에 큰 영향을 미치게 되는데 σ 의 크기가 클수록 어두운 부분을 개선하는 효과가 커지게 된다. 그러나 큰 σ 값을 사용하기 위해서는 큰 filter size가 필요하게 되고, filter size를 늘릴수록 memory, DSP 등의 resource를 많이 사용하게 된다. 또한 filter size에 해당하는 pixel의 수만큼 line buffer에 저장해야 하므로 line buffer의 크기가 커지는 만큼 입력부터 출력까지 걸리는 latency도 커지게 된다.

본 연구에서는 화질 개선 효과가 뚜렷하게 나타나도록 Gaussian kernel을 $\sigma=10$으로 정하였다. 그림 5.10을 보면 Gaussian filter의 sigma 값에 따른 Retinex algorithm의 결과를 볼 수 있다. σ 값이 커지면 contrast가 살아나는 효과가 있지만 어두운 부분 개선에는 그만큼 성능 하락이 생긴다. 본 연구에서는

어두운 부분을 개선하는 것이 가장 핵심이므로 과도하게 높은 σ 값은 필요하지 않다. 그림 5.10의 창문에 비친 어두운 부분의 detail이 살아나는 정도로 보았을 때 σ 값이 10인 경우 어두운 부분이 가장 잘 보인다고 볼 수 있다. 본 연구에서는 Gaussian kernel의 값을 13bit fixed point로 정했기 때문에 σ=10인 경우의 Gaussian coefficient가 모두 유효한 값을 갖게 하는 filter size는 53×53이다. 그러나 중앙에서 멀리 떨어진 부분의 coefficient는 매우 작고 Filter 결과 값에 미치는 영향이 적으므로 approximation computing 개념을 적용하여 더 작은 filter로도 구현하더라도 subjective quality에 큰 영향이 없다. 작은 filter로 구현하게 되면 그만큼 line buffer를 덜 사용하여 내부 BRAM을 적게 사용할 수 있고, 연산에 사용되는 resource도 크게 줄일 수 있다는 장점이 있다. 그러므로 본 연구에서는 성능과 resource 간의 trade-off를 고려하여 29×29 filter size를 사용하였으며, coefficient의 값은 normalize 하여 kernel의 합이 1이 될 수 있도록 하였다.



(a) (b) (c)



(d) (e) (f)

그림 5.10. Gaussian filter의 σ 값에 따른 Retinex 결과,
(a) 입력 영상, (b) $\sigma=2.5$, (c) $\sigma=5$, (d) $\sigma=10$,
(e) $\sigma=20$, (f) $\sigma=40$

그림 5.11를 보면 σ 값을 10으로 정했을 때 filter size에 따른 normalize factor 값을 볼 수 있다. Normalize factor 값은 filter kernel의 모든 값의 합으로, 이 값으로 kernel을 normalize 해주어 kernel의 합이 1이 되도록 나누어 주는 값이다. 이 값이 커질수록 filter size가 변할 때 shape가 많이 달라져서 오차가 크게 생기게 된다. Filter size가 29×29 일 때 normalize factor값이 0.9599이므로 0.95 이상이 되어 오차가 크게 생기지 않으므로 효과적으로 resource를 줄일 수 있도록 29×29 size로 filter를 선정하였다. 그림 5.12에는 approximate한 kernel의 shape와 기존 kernel shape가 나타나 있는데, 두 kernel간에 차이가 거의 없음을 확인할 수 있다. 29×29보다 더 줄이게 되면 full size filter와의 shape 차이가 커지게 되어 최종 값에서 차이가 발생하기 때문에 29×29로 선정을 하였다.

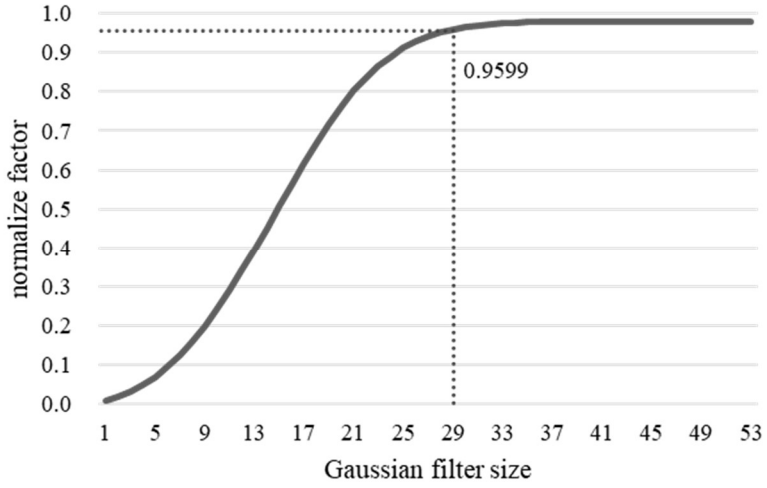


그림 5.11. Gaussian filter size에 따른 normalize factor

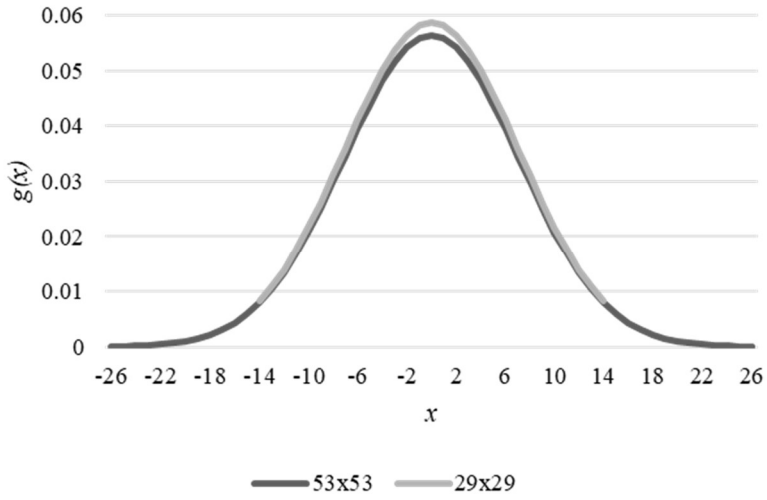


그림 5.12. Full size와 approximated size의 Gaussian kernel shape

또한 Gaussian filter가 사용하는 resource를 최소화 하기 위해 fixed point multiplication 연산 대신에 shifter와 adder를 사용하는 Gaussian filter를 구현하여 사용하였다. Gaussian filter kernel의 coefficient 값을 fixed point 13bit로 round한 값을 pixel 값에 곱하는 방법 대신 coefficient를 최대 3개의 shifter와 2개의 adder만을 사용하도록 coefficient를 조정하였다. i번째 Gaussian coefficient의 값 f_i는 식 (5.1)과 같이 표현되고, pixel p와 f_i의 곱은 식 (5.2)와 같이 표현할 수 있다.

$$f_i = 2^{a_i} \pm 2^{b_i} \pm 2^{c_i} \tag{5.1}$$

$$p * f_i = p \ll a_i \pm p \ll b_i \pm p \ll c_{\mathrm{i}} \tag{5.2}$$

조정된 kernel의 1번째부터 15번째까지의 13bit fixed point kernel coefficient value와 f_i의 표현식은 표 5.5에 나와있다. 16번부터 29번 coefficient는 이전 값과 대칭된다. Shifter와 adder 2개를 사용한 연산을 1 cycle에 완료하게 구현하여 Gaussian filter를 적은 resource로 구현하였다. 29×29 Gaussian filter를 구현하기 위해 1920×32 크기의 line buffer를 활용하여 filter를 구현하였다. HW resource를 절약하기 위하여 1D filter 2개를 각각 horizontally and vertically 사용하여 2D filter를 구현하였으며, vertical filter를 먼저 계산하고, horizontal filter 결과가 연산 되면 다음 module로 filtering된 pixel data를 stream으로 전달할 수 있도록 효율적으로 설계하였다.

표 5.5. Gaussian filter approximation

Real Value	13bit Fixed Point Value	Approximated Kernel Value	Approximated Kernel Notation
0.0082794	68	63	$2^6 - 2^0$
0.0108457	89	88	$2^6 + 2^4 + 2^3$
0.0139262	114	112	$2^7 - 2^4$
0.0175275	144	144	$2^7 + 2^4$
0.0216233	177	176	$2^7 + 2^5 + 2^4$
0.0261479	214	216	$2^8 - 2^5 - 2^3$
0.0309933	254	254	$2^8 - 2^1$
0.0360091	295	292	$2^8 + 2^5 + 2^2$
0.0410082	336	336	$2^8 + 2^6 + 2^4$
0.0457765	375	376	$2^8 + 2^7 - 2^3$
0.0500875	410	416	$2^9 - 2^6 - 2^5$
0.0537192	440	440	$2^9 - 2^6 - 2^3$
0.0564735	463	464	$2^9 - 2^6 + 2^4$
0.0581934	477	478	$2^9 - 2^5 - 2^1$
0.0587782	482	482	$2^9 - 2^5 + 2^1$

5.6.2 Exponentiation의 구현

F_{enh}는 F_m과 기존의 intensity channel의 평균 값을 이용하여 조명성분에 channel에 adaptive gamma correction을 적용한 channel이다. 해당 channel을 구현하기 위한 식은 식 (5.3)과 같다.

$$F_{enh}(x,y) = F_m(x,y)^{\gamma(x,y)} \tag{5.3}$$

$\gamma(x,y)$ 는 F_{enh} 의 이전 module인 F_{enh} 의 값과, intensity channel의 평균값을 통해 계산되는 adaptive gamma 값이며, 이 값은 pixel마다 다르게 정해지는 변수이다. 식 (5.3) 에서 볼 수 있듯 F_{enh}를 계산하기 위해서는 exponentiation 연산이 필요하다. 이 연산은 base와 exponent가 모두 변수인 연산이며 RTL로 구현할 시 복잡도가 매우 높다. 본 연구에서는 exponentiation 연산을 효율적으로 구현하기 위해 필요한 bit width에 적합한 최적의 exponentiation hardware module을 구현하였다.

본 연구에서 제시하는 exponentiation hardware module은 base 2 logarithm과 exponential을 사용하여 base와 exponent가 모두 변수인 exponentiation 연산을 decompose 하여 구현하였다. F_{enh}를 구현하기 위한 exponentiation 식의 base와 exponent를 각각 x와 y라고 두고, 이를 base 2 logarithm과 exponential로 표현하면 아래 식과 같다.

$$x^y = 2^{y*\log_2 x} \tag{5.4}$$

식 (5.4) 에서 exponent 부분은 y 와 $\log_2 x$ 의 곱셈으로 이루어져 있다. Exponent 부분의 값을 얻기 위해서는 $\log_2 x$에 대한 연산과 곱셈 연산이 필요하게 된다. Exponent 부분을 정수부(I_{exp})와 소수부(F_{exp})의 합으로 표현하면 식 (5.5)와 같다.

$$y * \log_2 x = I_{exp} + F_{exp} \tag{5.5}$$

이렇게 분리하게 되면 exponentiation 연산은 식 (5.6)과 같이 표현할 수 있게 된다.

$$2^{y*\log_2 x} = 2^{I_{exp}+F_{exp}} = 2^{I_{exp}} * 2^{F_{exp}} \tag{5.6}$$

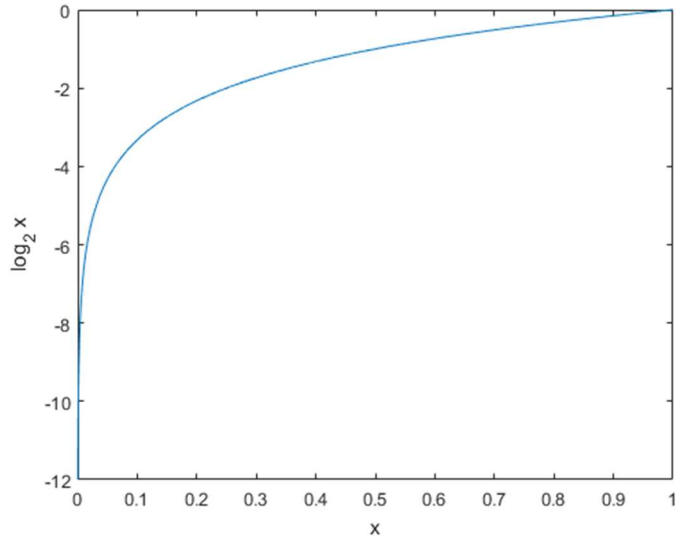
정수부에 대한 base 2 exponential은 shift 연산으로 대체할 수 있으므로 소수부에 대한 연산을 하면 전체 연산은 식 (5.7)과 같이 정리할 수 있다.

$$2^{I_{exp}} * 2^{F_{exp}} = 2^{F_{exp}} \gg |I_{exp}| \tag{5.7}$$

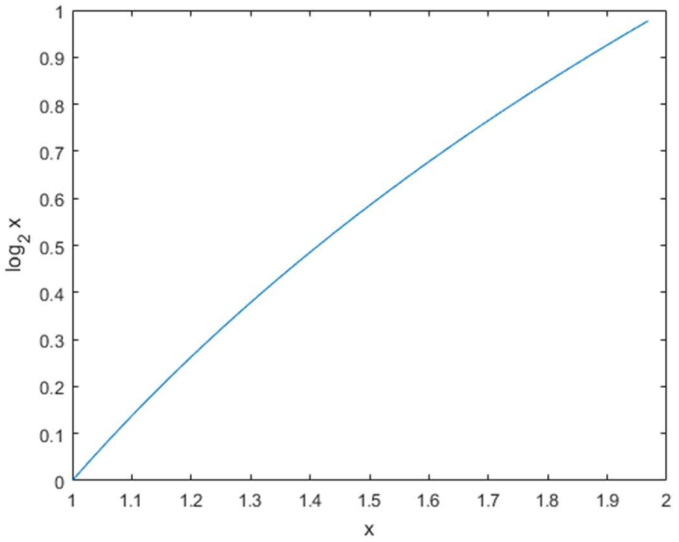
Exponentiation의 input value인 x는 항상 1보다 작은 범위를 가지므로, I_{exp}의 절대값을 취한 후 right shift를 하여 식 (5.7)와 같이 나타낼 수 있다. 이렇게 exponentiation을 decompose하면 곱셈과 덧셈, shift, 그리고 base 2 logarithm과 exponential의 연산을 통해 exponentiation 연산이 가능해진다.

곱셈과 덧셈, shift는 RTL에서 1 cycle 이하로 처리할 수 있는 연산이지만, logarithm과 exponential은 여전히 복잡한 구현이

필요하게 된다. 본 연구에서는 logarithm과 exponential에 사용되는 값의 범위가 정해져 있고, 결과값 역시 정해진 bit width 가 있다는 특징을 기반으로 LUT와 1차식으로의 근사를 통해 두 연산을 처리하였다. 본 시스템에서 사용된 exponentiation의 base에 해당하는 F_m 은 항상 0에서 1 사이의 값을 갖게 된다. 따라서 $\log_2 x$의 0에서 1 사이 값을 근사하면 원하는 값을 얻을 수 있다. 그러나 0에서 1사이의 $\log_2 x$ 의 그래프는 그림 5.13 (a)와 같이 기울기 변화가 심한 복잡한 경향을 띄어 근사 하기가 어렵다. 반면에 1에서 2 사이의 $\log_2 x$는 그림 5.13 (b)와 같이 기울기가 완만하여 근사 하기 쉬운 형태로 나타난다.



(a)



(b)

그림 5.13. 범위에 따른 $\log_2 x$의 그래프,

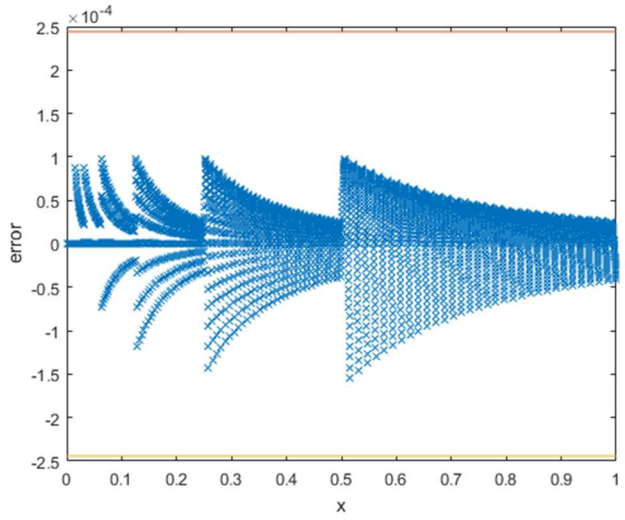
(a) $0 \leq x < 1$, (b) $1 \leq x < 2$

따라서 $\log_2 x$의 입력인 x의 범위를 1에서 2가 되도록 $\log_2 x$를 변환하여 근사를 시도하였다. 범위를 변환하기 위해 x의 fixed point precision에서 첫 nonzero bit의 위치인 k를 찾아 식 (5.8)과 같이 두 부분으로 나눈다.

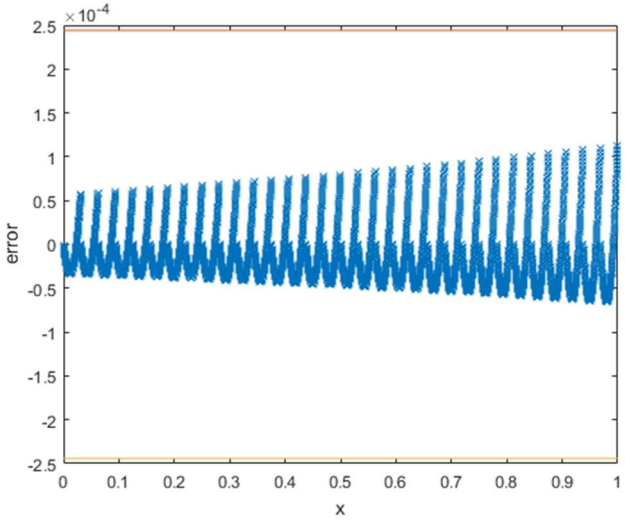
$$\log_2 x = \log_2 2^{-k} * (2^k x) = -k + \log_2(2^k x) \text{ i.e. } 1 \le 2^k x < 2 \qquad (5.8)$$

식 (5.8)에서 볼 수 있듯이 x의 첫 nonzero bit을 찾아 $1 \le 2^k x < 2$가 되는 k를 찾으면 $\log_2 x$를 정수 k의 뺄셈과 1에서 2의 범위의 logarithm으로 변환할 수 있다. 변환한 범위에서의 logarithm은 기울기가 매우 완만한 형태를 지니고 있기 때문에 매우 작은 오차만 갖도록 부분선형으로 근사할 수 있다.

부분 선형으로 근사 하기 위해 12bit precision의 x를 어떤 범위로 근사하면 될지 오차 분석을 통해 계산하였다. 2^5 크기의 entry를 가지는 seed와 기울기의 LUT를 사용하여 부분 선형으로 근사하면 전체 범위를 전부 LUT로 불러오는 2^{12} 크기보다 64배 줄여서 구현을 할 수 있다. 2^5 크기로 줄인 LUT를 활용한 부분 선형 logarithm의 오차는 그림 5.14 (a)에 나타나있다. 2^x 는 이미 정수부와 소수부로 나누어져 있기 때문에 0에서 1의 범위로 한정되어 있으며, 이는 1에서 2 범위의 logarithm과 같은 범위에서의 역함수 관계이므로 같은 방식으로 부분 선형 근사가 가능하다. 마찬가지로 오차 분석을 통해 정한 최적의 LUT 크기는 logarithm과 동일하며, 이때의 오차는 그림 5.14 (b)에 나타나있다.



(a)



(b)

그림 5.14. 2^5 크기의 LUT를 활용한 부분선형 근사의 오차. 파란색은 오차를 나타내고 주황색과 노란색 선은 12bit precision에서의 최소단위를 나타낸다. (a) $\log_2 x$, (b) 2^x

그림 5.15에는 exponentiation을 구현하기 위한 모든 approximation을 적용한 module의 block diagram을 나타낸다. 이러한 새로운 구조의 exponentiation 연산 HW IP는 성능의 변화가 거의 없으면서도 최소한의 resource만으로 효율적으로 구현되었기 때문에 image enhancement hardware Gaussian Filter와 같이 BRAM과 DSP를 많이 사용해야 하는 module이 충분히 resource를 utilize할 수 있도록 하는 장점이 있다. 또한 복잡한 연산을 간단한 단계 몇가지로 decompose하여 해당 연산에 소모되는 cycle 수도 매우 적어 실시간 image processing에 적합하다. 또한 fixed point precision에서 구현되었지만, 2의 exponential을 기반으로 하는 floating point에도 같은 원리로 근사를 적용할 수 있어 floating point를 필요로 하는 상황에도 어렵지 않게 제안하는 방법을 적용할 수 있다.

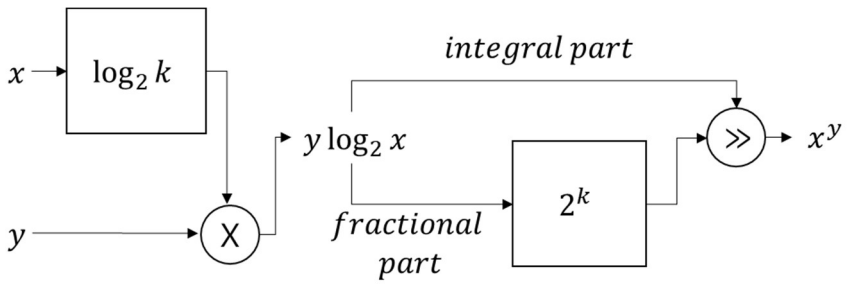


그림 5.15. Exponentiation module의 block diagram

5.6.3 HDMI/DVI 지원 및 영상 latency 최소화

상용화된 많은 영상 device들은 입력과 출력을 위해 HDMI나 DVI 단자를 통해 영상 data를 주고 받기 때문에 본 연구에서 구현한 시스템도 HDMI와 DVI 형식의 입력과 출력을 지원하도록 설계하여 범용성을 높였다. ZC706 board에서 제공하는 HDMI output을 사용하였고, HDMI input을 받기 위해 Digilent FMC-HDMI expansion card를 사용하였다. HDMI input을 받아 external memory와 같은 buffer에 저장하지 않고 바로 AXI4 Stream bus로 image enhancement module에 전달한 후, 출력도 AXI4 Stream bus로 HDMI 단자를 통해 내보낸다. 모든 data는 video stream data 전송 방식과 동일하게 1 pixel per 1 cycle로 전달한다.

이런 범용성이 장점으로 작용되기 위해서는 device와 HW IP 간의 통신에서 latency가 없어야 하는 것이 매우 중요하다. 본 연구에서는 frame buffer대신 line buffer를 이용하였는데, 이 방식을 택함으로써 매우 작은 latency의 image enhancement module을 구현할 수 있다. Frame buffer를 사용하게 되면 buffer의 사용 방식에 따라서 입력 영상과 출력 영상 간에 latency가 1 frame 이상 생기게 된다. Line buffer를 사용함으로써 1 frame 이상의 delay 대신 line buffer의 절반 크기만큼의 latency만 발생하게 된다. 이 값은 매우 작아서 상용 display device에서는 입력 영상과 비교했을 때 무시할만한 수치다.

또한 F_{enh} channel에서 필요한 α는 intensity channel의 전체 평균값을 사용하여 구해야 한다. 평균값을 구하기 위해서는 image 전체의 intensity를 더해야 하는데, 현재 frame의 평균값을

구하려면 1 frame의 delay가 생기게 된다. 본 연구는 작은 latency를 목표로 구현을 하였기에 1 frame의 delay가 생기지 않도록 α의 계산에 필요한 평균값을 이전 frame에서 가져오도록 하였다. Target video가 60 fps 이므로 인접한 두 frame 간의 intensity channel의 평균 값은 차이가 거의 없다. 따라서 이전 frame에서의 값을 사용하여도 결과에 영향이 없다. 그림 5.16을 보면 60 fps video 6 종류에서 각 frame에서의 α 값을 보여준다. Frame이 변화할수록 α 값은 비교적 큰 폭으로 변화하지만, 인접한 두 frame 간의 차이의 평균 값은 0.001272로 평균 alpha 값의 0.08% 수준밖에 되지 않는다. 100 frame중 가장 차이가 많이 나는 부분도 인접한 두 frame의 α값의 차가 0.01687이며 이는 해당 frame의 alpha 값의 1.14%이다. 이렇게 차이가 큰 상황은 인접한 두 frame이 급격한 화면 전환이 있을 때이며, 이런 상황에서는 이전 frame에서 적절치 못한 α값을 가져와 color restoration은 제대로 수행하지만 전체 밝기에 영향을 미칠 수 있다. 그러나 급격한 화면 전환의 상황에서 이러한 밝기 변화는 영상을 보는 관측자에게 미치는 영향이 크지 않다. 따라서 이전 frame의 α값을 사용해도 그 차이가 매우 작기 때문에 사용할 수 있다.

제안된 HW IP는 이런 방식으로 상용화된 device와 호환되는 HDMI/DVI 입출력을 매우 작은 latency로 지원하기 때문에 image 개선이 필요한 상황에서 timing loss 없이 본 연구의 system이 적용될 수 있는 큰 장점을 갖는다.

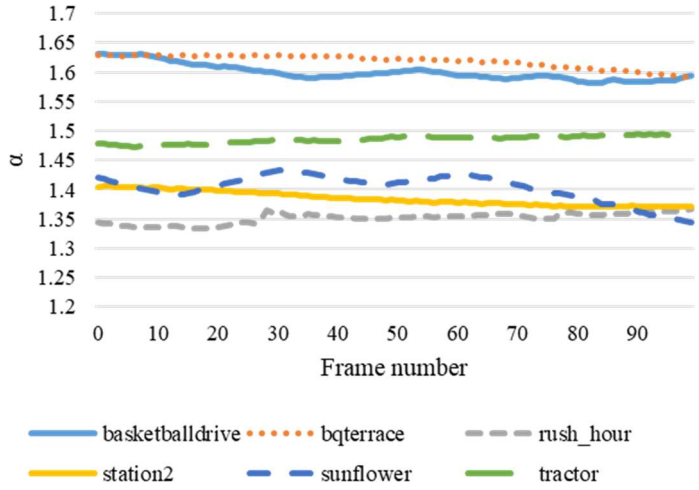


그림 5.16. 60fps 영상에서 연속된 frame의 α값 변화

5.7 제안 하드웨어 시스템 구현 결과 및 분석

5.7.1 실시간 동작 및 낮은 latency에 대한 분석

본 논문에서 제안한 Retinex 기반 이미지 개선 시스템은 FHD 해상도, 60 fps의 input을 실시간으로 처리하여 동일한 해상도와 frame rate의 output을 낸다. 구현한 module은 video stream을 1 pixel / 1 cycle로 전달하는 AXI4 stream bus에서 동작하기 때문에 해당 throughput으로 FHD 60fps를 만족하기 위해서는 최소 1920*1080*60≒125MHz 의 clock frequency가 필요하다. video processing module에서는 148.5MHz의 clock frequency를 사용하는 것이 보편적이므로 [61] 제안하는 시스템도 148.5MHz에서 동작하도록 설계되었다. Clock frequency를 높이면 throughput을 높일 수 있지만 그만큼 전력 소모가 증가한다. 본 연구에서는 전력 소모가 불필요하게 높아지는 것을 방지하면서 원하는 throughput을 얻기 위해 적당한 동작 clock을 설정하였다.

앞서 언급했듯이 이미지 개선 시스템의 특성상 input video와 output video 간에 latency가 작은 것이 매우 중요하기 때문에 본 연구에서는 제안한 frame buffer를 사용하지 않는 설계 기법을 통해 latency를 최소화하였다. 표 5.6에는 입력 처리 방법에 따른 입력 대비 출력 사이 필요 cycle 수와 latency의 비교가 제시되었다. 외부 frame buffer를 사용하는 방식에서는 필연적으로 1 frame 이상의 delay가 발생할 수 밖에 없기 때문에 결국 FHD 60fps, 148.5MHz clock 기준, 최소 13.96ms의 latency가

유발된다. Double, 또는 triple frame buffer를 사용하는 경우 이 latency는 더욱 길어지게 된다. 대신 line buffer를 사용하고, line buffer에 pixel이 필요한 만큼 stream으로 쌓인 후 바로 처리하게 되면 pixel을 처리하는 과정에서 생기는 latency를 최소화할 수 있다.

모듈 내부 연산의 관점에서 보았을 경우, 본 연구에서 사용하는 module 중 내부적으로 latency를 가장 많이 유발하는 module은 가장 복잡한 구조를 갖는 Gaussian filter module이다. 본 연구에서 설정한 σ 값을 오차 없이 충족하는 53×53 크기의 filter를 사용하면 line buffer를 사용하더라도 Gaussian filter module에서 최소 0.349ms의 latency가 발생하게 된다. Frame Buffer를 사용하는 것에 비해 2.5% 수준의 매우 작은 latency로 Gaussian filter를 처리하기 위한 window 구성 및 연산을 처리할 수 있게 된다. 본 연구에서는 29×29 크기의 window를 통해 구성하도록 approximation을 적용함으로써 추가적으로 latency를 0.2ms 수준으로 53×53 filter 사용 대비 57.3%만큼 대폭 감소하였다. 기타 모듈들로 인해 latency가 약간 증가했지만 최종적으로 제안된 기법은 이런 line buffer의 활용과 Gaussian filter의 approximation을 통해 latency를 frame buffer의 1.7% 수준인 0.241ms까지 감소하였으며 이런 최소한의 latency로 구현이 되었기 때문에 148.5MHz의 높지 않은 clock에서도 FHD 60fps를 만족시키며 상용 display device에서 input과 output 사이에 latency를 제거하는 것이 가능하다. 0.241ms의 값은 상용화된 대부분의 display device의 응답 속도인 5ms 의 4.82% 밖에 되지 않으므로 실제로 input과 output을 동시에 출력했을 때, latency가

전혀 나타나지 않는다.

표 5.6. Frame buffer와 line buffer,
제안하는 시스템의 latency 비교

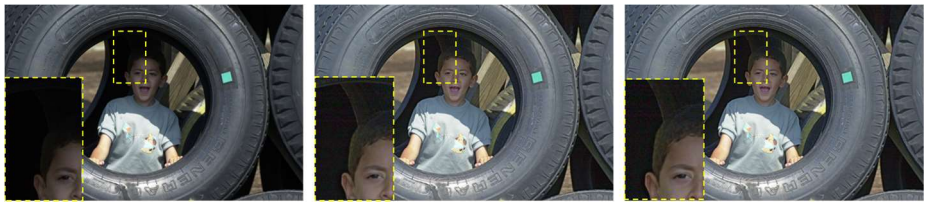
Architecture	Cycles to Process	Latency (ms)
Frame buffer*	2073600	13.96
Line buffer – 53×53 filter*	51840	0.349
Line buffer – 29×29 filter*	28800	0.194
Proposed system**	35732	0.241

* cycle과 latency 값은 연산을 통해 구하였다.

** FPGA 구현 후 측정을 통해 구하였다.

5.7.2 제안한 시스템의 영상 처리 성능 결과 분석

Approximation computing의 적용으로 발생되는 subjective quality의 저하가 없다는 것을 보이기 위해 3장의 full resolution image와 의료용 phantom 내부를 복강경 장비로 캡처한 image 1장에 대해서 얻은 결과를 그림 5.17-20에 나타냈다. 그림 5.17-20에 보면 원본 영상과 full Gaussian / approximate Gaussian 간에는 확연한 차이가 드러나며, Retinex의 적용으로 인해 어두운 부분이 확실히 향상되었음을 확인할 수 있다. 반면에 full size Gaussian filter를 사용한 결과와 approximation computing이 적용되어 29×29 크기의 Gaussian filter를 사용한 결과 간에는 눈으로 보이는 차이가 없음을 볼 수 있다. 각 image에서 확대된 부분을 보아도 어두운 부분을 개선한 효과가 같을 뿐 아니라, 세밀한 detail도 그대로 유지됨을 확인할 수 있다. 그림 5.17의 뒤에 존재하는 타이어의 edge 부분과 남자 아이의 머리의 detail이 살아나는 것이 유지되고, 그림 5.18에서는 여자아이 얼굴 및 배경의 보이지 않던 나뭇잎의 detail이 복원된 효과가 approximated filter에서도 유지됨을 볼 수 있다. 그림 5.19에서도 나뭇잎의 detail과 건물 벽의 색이 살아나는 효과가 유지됨을 볼 수 있고, 그림 5.20에서는 phantom 내부 인공 장기의 detail이 approximated filter를 적용하여도 개선되는 효과가 차이가 없음을 볼 수 있다. 이런 결과를 통해 approximation computing을 Gaussian filter에 적용하더라도 Retinex 알고리즘을 통해 얻고자 했던 어두운 부분을 개선하는 효과를 확실하게 유지되며 subject quality의 저하를 거의 야기하지 않음을 확인할 수 있다.



(a) (b) (c)

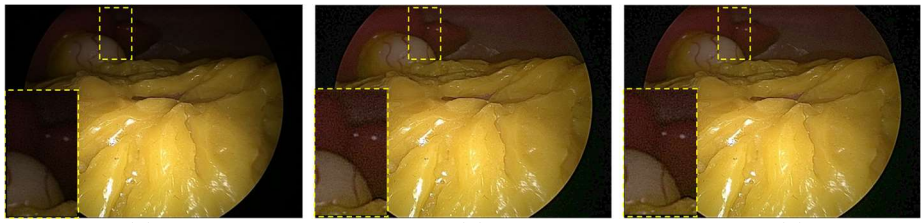
그림 5.17. 강한 햇빛과 그늘이 있는 영상,
(a) 원본 영상, (b) full size filter, (c) approximated filter

(a) (b) (c)

그림 5.18. 창문에 어둡게 반사된 영역이 있는 영상,
(a) 원본 영상, (b) full size filter, (c) approximated filter

(a) (b) (c)

그림 5.19. 그림자로 인한 저조도 영역이 있는 영상,
(a) 원본 영상, (b) full size filter, (c) approximated filter



(a) (b) (c)

그림 5.20. 적은 광량으로 인해 저조도 영역이 있는 영상,
(a) 원본 영상, (b) full size filter, (c) approximated filter

표 5.7. Full size filter 와 approximated filter 간의 absolute difference 의 평균값

Figure	Boy	Building	Girl	Phantom	Average
Average of Absolute Difference	0.4143	0.4082	0.3622	0.1366	0.3303

수치적인 차이 역시 비교하기 위해서 표 5.7 에는 동일한 4가지 image에 대해 full size Gaussian filter를 사용하여 출력된 Retinex 알고리즘의 결과와 approximation computing을 적용한 결과 사이의 pixel 당 평균 absolute difference를 제시하였다. 결과를 살펴보면 모든 영상에서 0.42이하의 pixel 차이를 보이며 평균적으로 pixel 당 0.33이하의 차이를 보이는 것을 확인할 수 있다. 이를 통해 approximate를 통해 작은 크기의 Gaussian Filter를 사용하여도 수치적 성능에도 큰 차이가 없음을 알 수 있다.

5.7.3 제안한 시스템의 FPGA Resource Utilization

제안하는 FPGA 이미지 개선 시스템은 Xilinx ZC706 evaluation 보드에 구현하였다. 모든 image processing logic은 programmable logic에 있는 FPGA resource만 사용하였고, 외부 메모리는 사용하지 않았다. 우선 Gaussian filter에 approximation computing의 개념을 적용하여 줄어든 크기의 filter와 shifter/adder만으로 저비용으로 구현한 resource saving 효과가 표 5.8에 제시되었다. 결과를 살펴보면 approximation computing을 적용하면 기존 대비 slice LUT는 35.61%, slice register는 58.42%, memory는 42.86%, DSP는 37개 사용하던 것을 전혀 사용하지 않게 된다. 그럼에도 불구하고 그림 5.17-20과 표 5.7에 제시된 것처럼 성능 저하는 매우 미비함을 볼 수 있다.

표 5.9는 제안하는 시스템의 FPGA 구현 resource 결과를 나타낸다. 전체 system에서 utilization을 가장 많이 차지하는 부분은 Gaussian filter임을 확인할 수 있다. 본 연구에서는 효과적으로 Gaussian filter module에서 사용되는 resource를 줄였고, 이는 전체 system의 resource를 큰 폭으로 줄이는 결과로 이어짐을 볼 수 있다.

표 5.8. 제안하는 시스템의 Gaussian filter approximation으로 감소한 resource utilization 및 비교

	53×53 Filter	29×29 Filter	Decreased %
Slice LUTs	36498	12998	35.61
Slice registers	10416	6085	58.42
Memory (RAMB36)	140	60	42.86
DSP (DSP48E)	37	0	–

표 5.9. 제안하는 시스템의 주요 module의 resource utilization

	Slice LUT		Slice Register		RAMB36		DSP	
	Util.	%	Util.	%	Util.	%	Util.	%
Gaussian Filter	12,853	5.88	6,254	1.43	60	11.01	0	0
Weighting Map Generation	3,565	1.63	5,094	1.17	24	4.40	6	0.67
Calculate F_m	285	0.13	1079	0.25	0	0	2	0.22
RGB creation	1,893	0.87	4,263	0.98	0	0	0	0
Calculate F_{enh}	1,684	0.77	1,420	0.32	0	0	5	0.55
Full system	24,176	11.06	22,787	5.21	93	17.06	13	1.44

5.7.4 제안한 시스템과 다른 시스템의 Resource Utilization 비교

제안하는 시스템의 HW resource 우수성을 보이기 위해 표 5.10에는 Gaussian filter를 사용한 Retinex 알고리즘을 FPGA로 구현한 Y. Li [55], D. Ustukov [56], 그리고 S. Marsi [57] 의 연구들과 성능 비교를 수행하였다. 다른 시스템과 제안한 시스템의 target FPGA가 다르기 때문에 다음과 같은 타당한 근거에 의해 비교를 수행하였다.

비교를 위해 각 FPGA의 user guide에 명시된 fabric 단위에 따라 각 FPGA logic unit들을 상응하는 memory 단위로 변환하였다. Li의 연구는 Xilinx Virtex4 FPGA에 구현하였고, Virtex4의 각 LUT와 register는 16 bit memory와 같은 크기를 지닌다 [62]. Ustukov의 연구와 제안하는 시스템은 각각 Virtex7과 Zynq7000 FPGA에 구현하였는데, 두 FPGA는 7-series FPGA에 속하며, 각 LUT와 register는 32-bit memory와 같은 크기를 지닌다 [63]. Marsi의 연구는 Cyclone III FPGA에 구현되었다. 서로 다른 제조사의 FPGA fabric을 비교한 article [64]에 따르면 Cyclone III에 쓰인 logic element의 한 unit은 Virtex4의 LUT의 한 unit의 약 1.3배 size가 크다고 나타나있다. 따라서 이러한 FPGA fabric의 차이를 바탕으로 각 시스템의 utilization을 상응하는 memory로 치환하여 normalize하여 비교하였다. 이러한 방식의 상이한 fabric의 FPGA resource 비교는 J. Choi의 연구에서도 활용되었다 [65].

표 5.10의 10번째 줄을 보면 normalized 된 HW resource가 Mbit의 단위로 표기되어 있다. 표에서 볼 수 있듯이 전체 시스템의

throughput을 고려하지 않더라도 제안하는 시스템이 다른 연구에 비해 작은 양의 resource를 사용했음을 볼 수 있다. Resource 크기에서 큰 차이를 보이는 것은 다른 연구들은 모두 frame buffer를 도입하여 영상을 처리하여 frame을 저장하기 위한 memory가 과도하게 사용되었기 때문이다. 제안하는 시스템은 Li의 연구에 비해 89.37%, Ustukov의 연구에 비해 20.78%, 그리고 Marsi의 연구에 비해 41.51%만의 resource를 사용하여 FGPA로 구현하였다.

제안하는 연구는 다른 연구에 비해서 높은 throughput을 가진다. 표 5.10의 11번째 줄에는 throughput 대비 normalize 된 HW resource를 1000 pixel 처리 당 Kbit으로 나타내고 있다. Throughput은 각 시스템의 해상도와 color space, 그리고 frame rate를 고려하여 계산하였다. Throughput을 고려했을 때의 FPGA resource는 Li, Ustukov, 그리고 Marsi의 연구의 각각 5.96%, 3.08%, 6.92%만의 resource만 사용하여 구현하였다

표 5.10. 제안하는 시스템과 다른 시스템과의 Resource 비교

	Li [55]	Ustukov [56]	Marsi [57]	Proposed
FPGA	Xilinx Virtex4	Xilinx Virtex7	Cyclone III	Xilinx Zynq7000
Resolution	720×576	640×480	720×576	1920×1080
Color space	Gray	RGB	RGB	RGB
Frame rate	60 fps	60fps	50 fps	60 fps
LUT	18186	20043	21608*	24176
Register	12724	420331	–	22787
SRAM	1616 Kb	9252 Kb	1179 Kb	3348 Kb
External memory	3317 Kb	0	9953Kb**	0
Normalized HW resource (Mbits)	5.43	23.34	11.69	4.85
Normalized HW / Throughput (Kbits / 1000 pixels)	39.26	75.99	33.81	2.34

* converted to LUT based on Virtex-4, where Cyclone III utilizes Logic Element instead of LUT and Register.
** resource usage is estimated because the paper stated the use of frame memory, but the utilization information is omitted.

제안하는 시스템과 비슷한 역할을 하는 알고리즘인 local tone mapping을 구현한 FPGA 연구와의 [66], [67] resource를 비교한 결과는 표 5.11에 나타나있다. Retinex 알고리즘과 local tone mapping 알고리즘의 복잡도가 다르고, 알고리즘의 결과물에 대한 성능도 차이가 나므로 직접적인 비교는 어렵지만 낮은 dynamic range의 image를 개선하는 역할을 하는 공통점이 있다는 측면에서 비교해볼 수 있다. 염두 할 것은 local tone mapping 알고리즘이 Retinex 알고리즘의 복잡도보다 훨씬 낮다는 점이다 [68].

Tone mapping과 HDR을 활용한 두 연구는 제안하는 시스템보다 작은 크기의 Gaussian filter를 사용한다. Licciardo [66]의 연구는 4×4 크기를 사용하였고, Ambalathankandy [67]의 연구는 5×5 크기를 사용하였다. Gaussian filter의 사이즈가 FPGA 시스템 전체의 resource에 큰 영향을 미치므로, 제안하는 시스템의 29×29 크기의 Gaussian filter를 5×5 크기로 사용했다고 가정하고 resource를 비교하였다. Resource normalize는 같은 방법을 사용하였다. Licciardo의 연구에 사용된 Virtex6는 제안하는 시스템과 같은 단위의 logic element를 사용한다 [69]. Ambalathankandy의 연구에서 사용된 Cyclone III는 Marsi의 연구와 같은 방법으로 normalize하였다.

표 5.11에 나와있듯이 제안하는 시스템은 Licciardo의 연구와 비교했을 때 94.51%, Ambalathankandy의 연구와 비교했을 때 120%의 resource를 사용하여 구현하였다. Resource 사용량은 비슷하거나 살짝 증가하였다는 것을 볼 수 있다. 그러나 Retinex 알고리즘의 복잡도가 local tone mapping 연구에 비해 최소 168% 더 복잡하다는 것을 염두 한다면 [68] 제안하는 시스템이 더

복잡한 알고리즘을 덜 복잡한 알고리즘의 FPGA 구현과 비슷한 수준의 resource만 사용하여 구현했다는 것을 볼 수 있다.

표 5.11. 제안하는 시스템과 비슷한 알고리즘을 구현한 연구와의 Resource 비교

	Licciardo [66]	Ambalathankandy [67]	Proposed 5×5
Algorithm	Inverse tone mapping	Tone mapping with halo-reducing	Retinex
FPGA	Xilinx Virtex6	Cyclone III	Xilinx Zynq7000
Resolution	1920×1080 RGB	1024×768 RGB	1920×1080 RGB
Frame rate	60 fps	126 fps	60 fps
LUT	51300	122186*	14888
Register	9200	-	21627
SRAM	1460 Kb	87 Kb	2052 Kb
Normalized HW resource (Mbits)	3.40	2.04	3.22
Normalized HW / Throughput (Kbits / 1000 pixels)	1.64	1.30	1.55

* converted to LUT based on Virtex-4, where Cyclone III utilizes Logic Element instead of LUT and Register.

5.7.5 제안한 시스템의 영상 처리 성능 결과 분석

본 연구에서 제안된 IP의 호환성의 우수함을 검증하기 위하여 제안된 FPGA 시스템을 computer HDMI출력, DVI 출력, 그리고 일반적인 display device에 연결하여 FHD 60fps에서 동작하도록 실험을 진행하였다. 특별히 Karl Storz의 Image 1 Hub image processor 와 Sony LMD-2760MD medical display 에 연결하여 동작을 검증하였다. Karl Storz는 내시경 및 복강경 장비의 leading manufacturer로 많은 병원에서 Karl Storz의 장비를 채택하고 있으며 의료 연구에서도 많이 사용되고 있다 [70], [71]. Sony의 medical display 역시 좋은 성능으로 인해 널리 사용되고 있는 의료용 기기이다. 그러므로 이런 장비들과의 검증을 통해 제안된 모듈의 호환성 및 상용화 가능성의 타당한 검증이 가능하다. Medical display에서는 입력 영상과 출력 영상을 PIP mode로 동시에 출력하여 입력과 출력 간의 latency가 나타나지 않음을 볼 수 있다. 상용 medical device와 연결되어 실시간으로 latency 없이 동작하는 demo 영상은 http://capp.snu.ac.kr/?p=research#Demos 에서 확인할 수 있다. 본 연구에서 구현한 FPGA는 HDMI/DVI로 영상 입출력이 가능한 다른 상용화된 장비들도 호환이 가능하다.

제 6 장 결 론

본 논문에서는 다중 카메라 환경에서의 object detection의 classification 정확도를 높이기 위한 방법과 저조도 환경에서의 object detection을 개선하기 위한 Retinex 알고리즘의 도입 및 최적화, 그리고 연산량이 많은 Retinex 알고리즘의 HW 구현을 통한 가속화를 제안한다. 다중 카메라 환경에서 다양한 각도의 feature들을 효과적으로 모으는 dominant feature pooling 방식을 제안하며, 더 중요한 feature를 판단하기 위한 scoring 방법을 linear regression을 통해 학습하는 방법을 제안한다. 또한 다양한 저조도 환경에 대응을 할 수 있도록 Retinex로 전처리를 한 영상에 대해 object detection을 학습하여 사용하는 것이 더 효과적임을 실험을 통해 나타내었다. 그리고 효과적인 Retinex 알고리즘이 실시간으로 동작할 수 있도록 HW로 구현하였으며, HW 구현을 최적화하기 위해 Retinex에서 사용되는 연산량이 많은 Gaussian filtering과 exponentiation 연산을 효율적으로 approximate하여 구현하는 방법을 제안한다. 제안하는 시스템은 무인 판매대 환경에서 실험하고 검증하였고, 실험을 진행한 환경 외에도 무인 자동차, CCTV, 무인 점포, 무인 창고 등 영상 처리를 통해 서로 다른 물체들을 인식하는 것이 필요한 환경에 적용할 수 있다.

참고 문헌

[1] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 10781−10790, Jun. 2020.

[2] Y. Liu, Y. Wang, S. Wang, T. Liang, Q. Zhao, Z. Tang, and H. Ling, "Cbnet: A novel composite backbone network architecture for object detection,” *in Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 34, no. 07, pp. 11653−11660, Apr. 2020.

[3] X. Du, T.−Y. Lin, P. Jin, G. Ghiasi, M. Tan, Y. Cui, Q. V. Le, and X. Song, “X. Spinenet: Learning scale−permuted backbone for recognition and localization,” *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11592−11601, Jun. 2020.

[4] J. Redmon and A. Farhadi, “ Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[5] T.−Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” *In European Conference on Computer Vision,* pp. 740−755, Sep. 2014.

[6] H. Su, S. Maji, E. Kalogerakis, and E. Learned−Miller, “Multi−view convolutional neural networks for 3d shape recognition,” *In Proceedings of the IEEE International Conference on Computer Vision*, pp. 945−953, Dec. 2015.

[7] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas,

"Volumetric and multi-view cnns for object classification on 3d data," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5648-5656, Jun. 2016.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652-660, Jul. 2017.

[9] E. Ristani, and C. Tomasi, "Features for multi-target multicamera tracking and re-identification," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6036-6046, Jun. 2018.

[10] Y. Fu, Y. Wei, Y. Zhou, H. Shi, G. Huang, X. Wang, Z. Yao, and T. Huang, "Horizontal pyramid matching for person re-identification," *in Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33 no. 01, pp. 8295-8302, Jul. 2019.

[11] E. H. Land and J. J. McCann, "Lightness and Retinex theory," *Journal of the Optical Society of America,* vol. 61, no. 1, pp. 1-11, 1971.

[12] Z. Rahman, D. D. Jobson, and G. A. Woodell, "Retinex processing for automatic image enhancement," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 100-110, Jan. 2004.

[13] Y. Xiao, A. Jiang, J. Ye, and M.-W. Wang, "Making of night vision: Object detection under low-illumination," *IEEE Access*, vol. 8, pp. 123075-123086, 2020.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,*" pp. 580-587, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 37, no.9, pp. 1904-1916, 2015.

[16] R. Girshick, "Fast r-cnn," *In Proceedings of the IEEE International Conference on Computer Vision*," pp. 1440-1448, 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *In Neural Information Processing Systems*, pp. 91-99. 2015.

[18] 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.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *in European Conference on Computer Vision*, pp. 21-37, 2016.

[20] D. Jobson, Z. Rahman, and G. Woodell, "A multiscale Retinex for bridging the gap between color images and the human observation of scenes," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 965-976, Jul. 1997.

[21] D. Terzopoulos, "Image analysis using multigrid relaxation methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 129-139, Mar. 1986.

[22] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Soble, "A variational framework for Retinex," *International Journal of Computer Vision*, vol. 52, no. 1, pp. 7-23, Apr. 2003.

[23] E. Land, "An alternative technique for the computation of the

designatorin the retinex theory of color vision," *in Proceedings of the National Academy of Sciences*, vol. 83, pp. 3078-3080, May. 1986.

[24] A. Moore, J. Allman, and R. M. Goodman, "A real-time neural system for color constancy," *IEEE Transactions on Neural Networks*, vol. 2, pp. 237-247, Mar. 1991.

[25] D. J. Jobson, Z. Rahman, and G. A. Woodell, "Properties and performance of a center/surround Retinex," *IEEE Transactions on Image Processing*, vol. 6, no. 3, pp. 451-462, Mar. 1997.

[26] D. Jobson, Z. Rahman, and G. Woodell, "A multiscale Retinex for bridging the gap between color images and the human observation of scenes," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 965-976, Jul. 1997.

[27] Y. H. Shin, S. Jeong, and S. Lee, "Efficient naturalness restoration for non-uniform illumination images," *IET Image Processing*, vol. 9, no. 8, pp. 662-671, 2015.

[28] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[29] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346-359. 2008.

[30] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3," *in 2019 1st International Conference on Unmanned Vehicle Systems Oman (UVS),* IEEE, pp. 1-6. 2019.

[31] J. H. Park, H. W. Hwang, J. H. Moon, Y. Yu, H. Kim, S. B. Her, G. Srinivasan, M. N. A. Aljanabi, R. E. Donatelli, S. J. and Lee, "Automated identification of cephalometric landmarks: Part 1-Comparisons between the latest deep-learning methods YOLOV3 and SSD," *The Angle Orthodontist*, vol. 89, no. 6, pp. 903-909, 2019.

[32] Y. Xiang, A. Alahi, S. and Savarese, "Learning to track: Online multi-object tracking by decision making," *in Proceedings of the IEEE International Conference on Computer Vision*, pp. 4705-4713, 2015.

[33] R. Hartley, A. and Zisserman, "Multiple View Geometry in Computer Vision," Cambridge: Cambridge University Press, 2011.

[34] Y. Lee, C. Lee, H. J. Lee, and J. S. Kim, "Fast Detection of Objects Using a YOLOv3 Network for a Vending Machine," *in 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems*, pp. 132-136. 2019.

[35] L. Dagum, and R. Menon, "OpenMP: An industry standard API for shared memory programming," *Computing in Science & Engineering*, vol. 1, pp. 46-55, 1998.

[36] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," *in Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736-2744, 2017.

[37] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018

[38] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating

very deep neural networks," *in Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389-1397, 2017.

[39] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2016.

[40] J. Redmon, "Darknet: Open source neural networks in c," [online]. Available: http://pjreddie.com/darknet/ [Accessed: Aug. 3. 2021].

[41] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," *in Proceedings of the 17th International Conference on Pattern Recognition*, vol 2, pp. 28-31, 2004.

[42] H. H. Aghdam, A. Gonzalez-Garcia, J. van de Weijer, and A. M. Lopez, "Active learning for deep detection neural networks," *in Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3672-3680, 2019.

[43] Y. Siddiqui, J. Valentin, and M. Nießner, "Viewal: Active learning with viewpoint entropy for semantic segmentation." *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 9433-9443, 2020.

[44] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, pp. 131-163, Nov. 1997.

[45] N. Houlsby, F. Huszar, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," *arXiv preprint arXiv:1112.5745*, 2011.

[46] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active

learning with image data," *International Conference on Machine Learning*, pp. 1183-1192, Jul. 2017.

[47] H. E. D. Mohamed, A. Fadl, O. Anas, Y. Wageeh, N. ElMasry, A. Nabil, and A. Atia, "Msr-yolo: Method to enhance fish detection and tracking in fish farms." *Procedia Computer Science*, vol. 170, pp. 539-546, 2020.

[48] H. Qu, T. Yuan, Z. Sheng, and Y. Zhang, "A pedestrian detection method based on YOLOv3 model and image enhanced by Retinex." *in 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, IEEE, pp. 1-5, 2018.

[49] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao, "Dehazenet: An end-to-end system for single image haze removal." *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5187-5198, 2016,

[50] H. Qu, T. Yuan, Z. Sheng, and Y. Zhang, "A Pedestrian Detection Method Based on YOLOv3 Model and Image Enhanced by Retinex," *in 11th International Congress on Image and Signal Processing, Bio Medical Engineering and Informatics (CISP-BMEI)*, IEEE, pp. 1-5, 2018.

[51] Y. Wageeh, H. E. D. Mohamed, A. Fadl, O. Anas, N. ElMasry, A. Nabil, and A. Atia, "YOLO fish detection with Euclidean tracking in fish farms," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 5-12. 2021.

[52] Y. P. Loh and C. S. Chan, "Getting to know low-light images with the exclusively dark dataset," *Computer Vision and Image Understanding*, vol. 178, pp. 30-42, Jan. 2019.

[53] S. Setty, N. K. Srinath, and M. C. Hanumantharaju, "Development

of multiscale retinex algorithm for medical image enhancement based on multi-rate sampling," *in 2013 International Conference on Signal Processing, Image Processing & Pattern Recognition*, pp. 145-150, Feb. 2013.

[54] W. Ma, J.-M. Morel, S. Osher, and A. Chien, "An L1-based variational model for Retinex theory and its application to medical images," *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 153-160, Jun. 2011.

[55] Y. Li, H. Zhang, Y. You, and M. Sun, "A multi-scale retinex implementation on FPGA for an outdoor application," *in 2011 4th International Congress on Image and Signal Processing*, pp. 1788-1792, Oct. 2011.

[56] D. I. Ustukov, Y. R. Muratov, and V. N. Lantsov, "Modification of retinex algorithm and its stream implementation on FPGA," *in 2017 6th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1-4, Jun. 2017.

[57] S. Marsi and G. Ramponi, "A flexible FPGA implementation for illuminance-reflectance video enhancement," J*ournal of Real-time Image Processing*, vol. 8, no. 1, pp. 81-93, Mar. 2011.

[58] J. Han, and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," *in 2013 18th IEEE European Test Symposium (ETS)*, pp. 1-6, May. 2013.

[59] H. Seong, C. E. Rhee, and H. Lee, "A novel hardware architecture of the Lucas-Kanade optical flow for reduced frame memory access," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 6, pp. 1187-1199, Jun. 2016.

[60] "AMBA AXI4-Stream Protocol Specification." [online].

Available: http://www.arm.com [Accessed: Aug. 3, 2021].

[61] "Xilinx Intellectual Property Video and Image Processing Pack." [online]. Available: http://www.xilinx.com [Accessed: Aug. 3. 2021].

[62] "Virtex-4 FPGA User Guide," [Online], Available: https://www.xilinx.com/support/documentation/user_guides/ug070.pdf, [Accessed: Aug, 3, 2021].

[63] "7 Series FPGAs Configurable Logic Block," [Online], Available: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf, [Accessed: Aug. 3, 2021].

[64] "FPGA Logic Cells Comparison." [Online], Available: http://ee.sharif.edu/~asic/Docs/fpga-logic-cells_V4_V5.pdf, [Accessed: Aug. 3, 2021].

[65] J. Choi, B. Kim, H. Kim, and H.-J. Lee, "A high-throughput hardware accelerator for lossless compression of a DDR4 command trace," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 1, pp. 92-102, Jan. 2019.

[66] G. D. Licciardo, A. D' Arienzo, and A. Rubino, "Stream processor for realtime inverse tone mapping of full-HD images," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2531-2539, Nov. 2015.

[67] P. Ambalathankandy, A. Hore, and O. Yadid-Pecht, "An fpga implementation of a tone mapping algorithm with a haloreducing filter," *Journal of Real-Time Image Processing*, pp. 1-17, Sep. 2016.

[68] G. M. S. Nunes, "Evaluation of tone-mapping algorithms for

focal-plane implementation," M. S. thesis, Electrical Engineering, Federal University of Rio de Janeiro, Rio de Janeiro, 2018. Accessed on Aug. 3, 2021.

[69] "Virtex-6 FPGA Configurable Logic Block," [Online], Available: https://www.xilinx.com/support/documentation/user_guides/ug364.pdf, [Accessed: Aug. 3, 2021].

[70] G. H. KleinJan, N. S. van den Berg, O. R. Brouwer, J. de Jong, C. Acar, E. Sm. Wit, E. Vegt, V. van der Noort, R. A. V. Olmos, F. W. B. van Leeuwen, and H. G. van der Poel, "Optimisation of fluorescence guidance during robot-assisted laparoscopic sentinel node biopsy for prostate cancer," *European Urology*, vol. 66, no. 6, pp. 991-998, Dec. 2014.

[71] R. Dutta, R. Yoon, R. M. Patel, K. Spradling, Z. Okhunov, W. Sohn, H. J. Lee, J. Landman, and R. V. Clayman, "Clinical comparison of conventional and mobile endockscope videocystoscopy using an air or fluid medium," *Journal of Endourology*, vol. 31, no. 6, pp. 593-597, Jun. 2017.

Abstract

Dominant Feature Pooling for Multi Camera Object Detection and Optimization of the Retinex Algorithm

Jin Woo Park

Department of Electrical and Computer Engineering

The Graduate School

Seoul National University

This paper proposes a novel dominant feature pooling method utilized in the detection phase for multi-camera object detection CNNs. Multi-camera systems can capture images of objects from various perspectives and utilize more of the important features of objects for detection. Thus, the detection accuracy can be improved by pooling the features of the multiple cameras. The proposed method constructs a new feature patch by selecting and pooling the dominant features that provides more information among the feature vectors obtained from various viewpoints of objects. The proposed method is based on the YOLOv3 network for a single camera, and does not require additional learning processes for multi-camera systems. To

show the effectiveness of dominant feature pooling, a novel method of visualizing feature vectors is also proposed in this work.

Furthermore, a method of utilizing Retinex algorithms that can improve response to low-light environments for object detection CNN is proposed. Although improvements can be made by learning low-light images as they are, experimental results show that Retinex improvements are essential because the degree of illumination cannot be predicted accurately to create new datasets in practical environments. This work proposes a method to optimize Retinex algorithms through HW designs. An efficient implementation of the exponentiation operation and the Gaussian filtering, which are essential for Retinex algorithm operations is proposed to implement HW that can operate in real time at high resolution.

Keywords : Multi-camera, object detection CNN, CNN features, Retinex algorithm, FPGA

Student Number : 2014-21658