Check for updates

Journal of KIIT. Vol. 21, No. 10, pp. 137-145, Oct. 31, 2023. pISSN 1598-8619. eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2023.21.10.137

색상 필터와 딥러닝을 활용한 화재감지 구현

방한솔*, 이주형**, 최진구***

Implementation of Fire Detection with Color Filters and Deep Learning

Han-Sol Bang*, Joo-Hyung Lee**, and Jin-Ku Choi***

※ 이 논문은 2022년도 한국산업단지공단의 지원을 받아 수행된 산업집적지경쟁력강화사업임(No. IRKK2217)

요 약

비화재보와 늦은 검출 속도에 의한 신뢰성이 부족한 기존 화재 감지기의 문제점을 딥러닝 모델과 불꽃의 특징을 활용하여 해결하고자 하였다. 딥러닝 모델 YOLOv7을 이용하여 화재 영역을 검출하고, 색상 필터를 적용하여 배경을 제거한 불꽃 후보 영역만을 사용하였다. 이를 불꽃의 특징을 활용한 두 가지 불꽃 탐지 알고리즘으로 화재인지 확인하였다. 광학 흐름으로 겉불꽃 영역의 움직임을 확인하고, 이미지 중심부의 밝기가 임계치를 이상일 경우 속불꽃으로 판단하였다. 불꽃 후보 영역을 화재인지 판단하여 조기에 화재를 검출하고, 신뢰성을 개선한 시스템을 구현하였다. 화재 검출 시험의 결과는 평균 정확도 0.838이고 두 종류의 비화재 검출 시험의 경우 각각 평균 정확도 1.00과 0.976의 결과를 보였다.

Abstract

We tried to solve the problems of existing fire detectors that lack reliability due to false alarm and late detection time by using deep learning models and flame features. We used the deep learning model YOLOv7 to detect the fire area, and used only the flame candidate area with the background removed by applying a color filter. Two flame detection algorithms using flame features were used to verify the fire. Optical flow to check the movement of the outer flame area, If the brightness at the center of the image was above a threshold, it was judged as an inner flame. By determining the candidate flame area as a fire, the system detects fires early and improves reliability. The results of the fire detection test showed an average accuracy of 0.838, and the results of the two non-fire detection tests showed an average accuracy of 1.00 and 0.976 respectively.

Keywords

deep learning, image pre-processing, fire detector, object detection, YOLOv7

* 한국공학대학교 컴퓨터공학과 석사과정
- ORCID: https://orcid.org/0009-0001-1102-6113
** 한국공학대학교 컴퓨터공학과 석사과정
- ORCID: https://orcid.org/0009-0002-6632-3905
*** 한국공학대학교 컴퓨터공학부 교수(교신저자)
- ORCID: https://orcid.org/0009-0008-0659-1274

· Received: Jul. 12, 2023, Revised: Aug. 08, 2023, Accepted: Aug. 11, 2023
· Corresponding Author: Jin-Ku Choi
Dept. of Computer Science and Engineering, Tech University of Korea
237, Sangidaehak-ro, Siheung-si, Gyeonggi-do, Republic of Korea
Tel.: +8231-8041-0521, Email: jkchey@tukorea.ac.kr

Ⅰ. 서 론

소방청의 화재통계연감에 따르면 2017년부터 최근 5년간 화재 발생 건수는 2017년 44,178건에서 2021년 36,267건으로 꾸준한 감소 추세에도 불구하고 화재로 인한 재산 피해 금액은 2017년의 50억 6,914만원에 비해 2021년 109억 9,125만원으로 2배 가까이 증가하여 높은 상승 추세를 보여주고 있다[1]. 화재를 조기에 감지하여 피해를 줄이기 위해 탑재되어 있는 화재 감지기는 일반적으로 열 감지기와 연기 감지기로 구성되어있다. 이러한 화재 감지기들은 비화재로 인한 열과 연기에도 반응하므로 높은 오감지율을 보인다. 감지기의 비화재보(자동화재속보설비, 경보 오동작)에 의해 발생하는 서울시 오인출동 현황은 2020년 852건으로 같은 해 서울시 화재 발생 건수인 5,088건과 비교하면 비화재보에 의해 약 17%의 오인출동이 발생 됨을 확인하였다[2][3]. 화재 감지기의 작동 시간 비교실험에서 유염 화재의 발생 이후 연기 감지기는 최소 3분 31초 이후, 열 감지기는 5분 15초 이후에 감지되어 초기 화재진압이 어렵다[4].

본 논문에서는 비화재보와 늦은 검출 속도에 의한 신뢰성이 부족한 문제점을 해결하기 위해 YOLOv7로 화재 영역을 검출하고, 색상 필터를 적용하여 배경을 제거한 불꽃 후보 영역만을 남겼다. 이를 불꽃의 특징을 활용한 불꽃 탐지 알고리즘으로 화재인지 확인하여 조기에 화재를 검출하고 검출 신뢰성을 개선한 시스템을 구현하였다. 본 논문의 구성은 다음과 같다. 2장의 YOLOv7 및 색상 필터를 적용한 화재감지, 3장에서 시스템 구현 방식을 서술하였다. 4장에서 화재 데이터셋과 학습 환경을, 5장에서 화재 및 비화재 검출 결과를 서술하였다.

Ⅱ. 관련 연구

2.1 화재 감지기의 탑재 현황

화재 감지기는 감지 방식에 따라 열 감지기, 연기 감지기, 불꽃 감지기 등으로 나누어진다. 탑재된 화재 감지기 현황은 열 감지기 및 연기 감지기로 대다수 구성되어 있다[5]. 이러한 일반적인 화재 감지기를 대상으로 서울시 소방재난본부에서 실시한 연소물에 따른 작동 시간 비교실험에서 쓰레기통에 불꽃 점화하여 실험한 유염 화재의 경우 3분 31초 이후 연기 감지기가 작동되었고, 5분 15초 이후 차동식 열 감지기가, 5분 39초 이후 정온식 열 감지기가 작동되었다. 라텍스에 니크롬선으로 점화하여 실험한 무염 화재에서는 9분 46초 이후 연기 감지기가 작동하였고 두 종류의 열 감지기는 작동하지 않는 결과를 보였다.

2.2 딥러닝 알고리즘 분석

YOLO(You Only Look Once)는 기존의 주류 방식인 2단계 검출방식(Two-Stage Detection)의 검출 속도가 느린 문제점을 해결하기 위해 지역 제안(Regional proposal)과 분류(Classification)과정을 동시에 실시하는 모델이다. 이에 따라 연산량을 줄이고 향상된 검출 속도에 의해 실시간 객체 탐지를 할 수 있다[6]. 2022년 7월 발표한 YOLOv7 모델은 실시간 객체 검출 모델 중 State-of-the-art를 달성한 YOLOR의 후속 모델이다. Model reparameterization, Label assignment과 같은 YOLOv7에 적용된 Bag-of-freebies에 의해 같은 연산량으로 향상된 모델 정확도(Accuracy)를 보였다. 기존 ELAN 네트워크의 단점을 보완한 E-ELAN 네트워크를 사용하여 향상된 학습결과를 보였다. 기존의 YOLO 모델 중 YOLOR-CSP을 개선하여 AP는 0.4%, FPS는 55만큼 증가하여 YOLO 모델 군에서 빠르고 정확한 모델이나[7], 2단계 검출 방식에 비해 상대적으로 낮은 mAP를 보여 검출 속도보다 신뢰성이 중요시되는 경우에는 적합하지 않다.

2.3 HSV 색상을 이용한 화재 영역 분리

상을 표현하는 방식에는 RGB 모델, YCbCr 모델, HSV 모델과 같은 색상 모델을 사용한다. 이들 중 색도(Hue), 채도(Saturation), 명도(Value)로 이루어져 있어 사람이 색상을 인식하는 방식과 유사한 HSV 모델을 사용하였다. 논문 [8]은 색상 필터를 이용한 화재 영역으로부터 배경을 분리하는 알고리즘을 제안하였다. 위 논문에서 사용한 HSV 모델의 색상 범위는 아래와 같다.

$$0 \leq H \leq 0.35 \cup 3.0 \leq H \leq 3.14 \quad (1)$$

$$0.3 \leq S \leq 1.0 \quad (2)$$

$$0.8 \leq V \leq 1.0 \quad (3)$$

위 논문에서 제안된 화재 감지 시스템은 화재와 비슷한 적색 계열을 띄는 물체의 경우 오인할 가능성이 있으며, 제안된 색상 범위를 화재 영상에 적용한 결과 불꽃의 중심부가 손실되는 문제점이 있음을 확인하였다[9].

III. 제안 시스템 구현

본 논문에서 제안하는 시스템은 딥러닝 객체 검출을 이용한 빠른 검출 속도와 화재 특성을 이용한 색상 필터 및 불꽃 탐지 알고리즘을 적용하여 검출 정확도를 높여 비화재보를 줄이고자 하였다. 화재 검출 시스템은 추론(Inference)과 보정(Correction)과정으로 구성하였다. 추론 과정은 화재에 대해 학습된 YOLOv7 모델을 사용하여 화재 영역을 검출한다. 이미지 내 화재 영역은 경계 상자(Bounding box)로 나타내며, 각 경계 상자마다 신뢰 점수(Confidence score)를 표기하여 얼마나 신뢰할 수 있는지 나타낸다. 화재 영역 중 신뢰 임곗값(Confidence threshold)이상의 신뢰 점수를 가질 경우에만 화재로 판단한다. 화재로 판단된 전체 이미지에서 화재 영역만을 잘라내어 화재 영역의 중심 좌표와 함께 보정 과정으로 전달한다. 보정 과정은 화재의 특성을 응용하여 추론 과정에서 발생할 수 있는 비화재보를 줄이고자 하였다. 추론 과정의 화재 영역에 색상 필터를 적용하여 불꽃만 나타내었다. 불꽃은 크게 겉불꽃과 속불꽃으로 나누어진다. 일반적인 화재에서 속불꽃에 비하여 겉불꽃은 온도가 낮고 불규칙 적인 움직임을 보인다. 반면 속불꽃은 온도가 매우 높고 매우 밝은 빛을 내는 특성을 보인다. 보정 과정에서는 흐름을 사용하여 겉불꽃의 움직임을 검출하는 겉불꽃 탐지 알고리즘과, 속불꽃의 특성을 고려한 속불꽃 밝기 알고리즘으로 화재 영역을 검토하였다. 광학 흐름은 추론 결과의 연속되는 두 프레임으로부터 광학 흐름을 계산하여 겉불꽃을 검출한다. 광학 흐름은 이미지 크기가 같아야 연산할 수 있다. 두 불꽃 이미지의 크기를 일치시켜 광학 흐름을 계산한다. 이미지 외곽의 광학 흐름으로 움직임이 관찰되었을 시 겉불꽃으로 판단하였다. 속불꽃은 불꽃의 중심부를 지칭하므로 화재 영역의 중심부만 고려하였다. 속불꽃 밝기 알고리즘에서는 중심부의 평균 밝기가 임곗값 이상일 경우 속불꽃으로 판단하였다. 겉불꽃과 속불꽃이 검출될 경우 최종적인 화재로 판단하였다.

3.1 겉불꽃 탐지 알고리즘

광학 흐름의 연산은 Computer Vision 오픈소스 라이브러리인 OpenCV를 사용하였다. 그림 1의 좌측 이미지와 같이 8 픽셀 간격으로 격자를 만들어 광학 흐름의 연산 결과를 저장한다. 겉불꽃의 움직임만을 사용하기 위해 광학 흐름을 사용하였다. 광학 흐름은 물체의 밝기 항상성(Color constancy)과 작은 움직임(Small motion)을 전제로 간접적으로 움직임을 계산하는 방식이다. 그림 1의 우측과 같이 겉불꽃의 경계에 해당하는 격자의 광학 흐름만 평균 내어 임곗값 이상일 경우에 겉불꽃으로 판단한다. 본 논문에서는 식 (4)을 제안하여 기존 광학 흐름의 연산량을 줄인 Gunner Farneback의 광학 흐름을 사용하였다[10].

$$I_{t1}(x) = x^T A_1 x + b_1^T x + c_1 \quad (4)$$

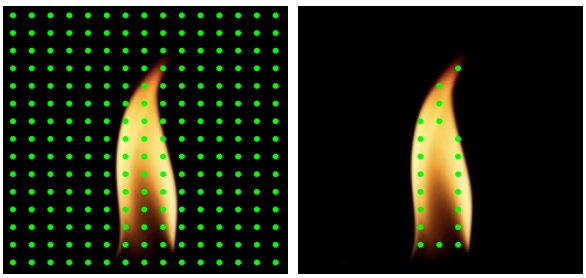

그림 1. 화재 이미지의 광학 흐름 예시
Fig. 1. Example of optical flow in a fire image

3.2 HSV 색상 필터 알고리즘

논문 [8]의 모델은 화재 영역의 중심부가 손실되는 문제점이 있다. 본 논문에서는 이를 해결하기 위해 백색 손실을 보완한 화재 색상 범위를 적용하였다.

화재는 일반적으로 노란색 계열부터 붉은색 계열의 색상으로 표현되므로 화재와 관련되지 않은 색상을 제거하여 오경보를 낮추었다. 불꽃의 중심부는 백색을 나타내므로 제거된 부분에서 백색을 표현하는 부분을 다시 추가하여 백색 손실이 발생하는 문제를 해결하였다. 이를 적용한 색상 범위를 그림 2에 흐름도로 나타내었다. 추가로 그림 3에 좌측의 원본 이미지와 우측의 필터 적용 이미지를 나타내었다.

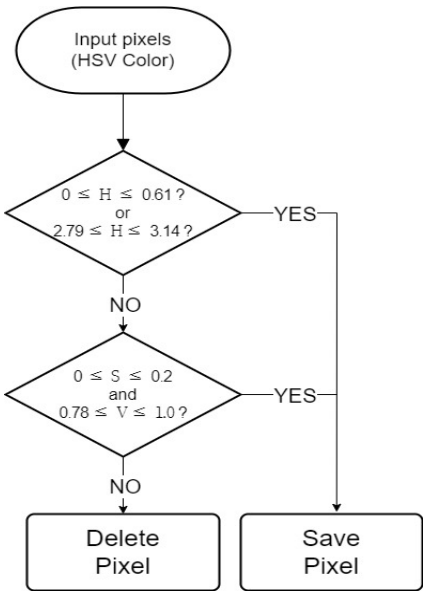

그림 2. HSV 색상 필터 알고리즘 흐름도
Fig. 2. HSV filter algorithm flowchart

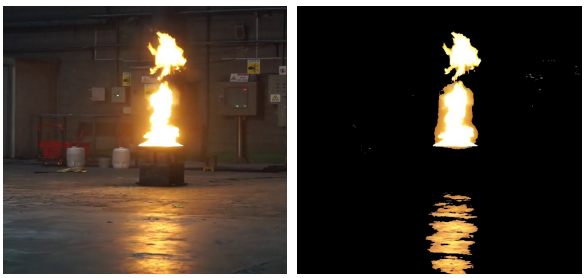

그림 3. HSV 필터 적용에 따른 모의 화재 영역
Fig. 3. Area of the simulated fire based on HSV filter application

3.3 속불꽃 밝기 알고리즘

추론 과정으로부터 전달받은 이미지는 색상 필터를 적용한 결과이다. 색상 필터를 적용하였을 때, 이미지에 필터 경계에 있는 색상이 다수 존재할 경우 그림 4의 좌측과 같이 이미지에 잡음(Noise)이 발생한다. 이 영역을 가우시안 필터를 적용하여 줄인다[11]. 속불꽃의 밝기를 측정하기 위해 이미지를 회색조(Grayscale)로 변환한다. 회색조 이미지의 각 픽셀당 밝기 임계치를 넘기면 백색으로, 나머지를 흑색으로 처리하여 최종적으로 그림 4의 우측의 결과를 얻는다. 속불꽃은 중심부에 있으므로 그림 5와 같이 이미지를 3×3 격자로 나누고 중심 이미지의 백색 픽셀의 개수가 임계치인 절반 이상이면 속불꽃으로 판단한다.

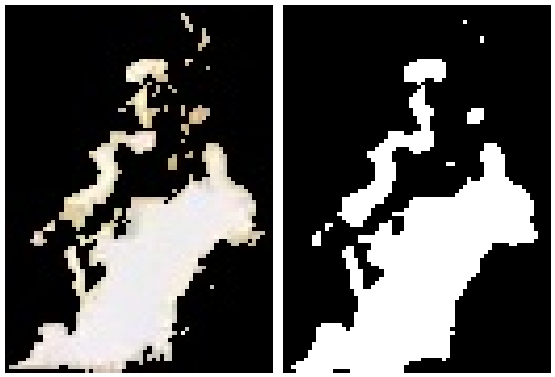

그림 4. 이미지 내 잡음과 잡음 제거 결과
Fig. 4. Noise in image and denoising image

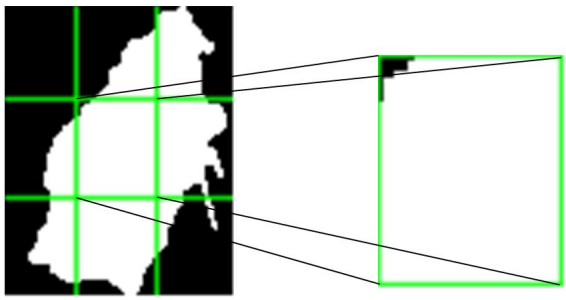

그림 5. 속불꽃 밝기 알고리즘
Fig. 5. Flame inner area brightness algorithm

Ⅳ. 화재 데이터 구성 및 학습

4.1 화재 데이터셋 구성

학습(Train), 평가(Valid) 및 시험(Test)을 위한 데이터셋의 모의 화재의 촬영기준은 다음과 같다. 가로 및 세로의 길이가 각각 300mm, 300mm, 높이가 50mm인 상단부가 개봉되어있는 직육면체의 금속 용기에 노말 헵탄(N-Heptane)과 물을 1:3으로 섞은 연료를 담아 점화한다[12].

화재의 촬영은 1m, 5m, 30m의 거리에서 여러 각도로 촬영하였다. 데이터셋을 구성하기 위한 영상은 높이와 너비의 길이가 같은 정방형으로 잘라내었고, YOLOv7x의 권장 학습 조건인 640×640 해상도로 변환하였다[7]. 변환한 영상을 30fps로 이미지로 저장하였고 무작위로 학습 데이터 4800장, 평가 데이터 1000장으로 구성하였다. 시험의 경우 보정 과정의 광학 흐름의 측정을 위해 연속되는 이미지로 구성해야 한다. 무작위 구성을 제외하고 학습 및 평가 데이터셋과 같은 조건으로 시험 데이터를 구성하였다. 시험 데이터는 5개 영상의 100프레임씩 저장하여 총 500장의 이미지로 구성하였다.

4.2 학습

학습 환경은 Python 3.7.15 버전을 사용하였으며, Anaconda 가상환경에서 CUDA 11.4 플랫폼과 오픈소스 머신 러닝 라이브러리인 PyTorch 1.11.0 버전을 이용하여 학습하였다. 시스템 사양은 아래의 표 1에 나타내었다[13]-[16].

표 1. 시스템 사양
Table 1. System specification

OS	Ubuntu 20.04
CPU	AMD ryzen 5600
GPU	RTX 3060 12GB
RAM	16GB

YOLOv7의 학습 조건은 640×640 해상도, Batch size 10, Epoch size 300, YOLOv7에서 제공하는 가중치 yolo7x_training.pt를 사용하여 학습하였다. 학습 후 YOLOv7에서 제공하는 Python 스크립트로 학습 결과를 측정하였다[17]. 측정 조건은 학습에 사용한 평가 데이터셋을 Batch size 1로 하였다. 측정결과로 mAP@0.5 0.999, mAP@[0.5:0.95] 0.912를 나타내었다.

V. 시험 및 성능 평가

성능평가는 추론 및 보정 과정을 시험하는 화재 검출 시험과 비화재를 의도적으로 검출시켜 보정 과정만 시험하는 비화재 검출 시험의 두 방법으로 실시하였다. 추론 과정의 결과는 평균 정밀도(mAP)인 mAP@0.5와 mAP@[0.5:0.95]로 나타내었다. 평균 정밀도는 정밀도(Precision)와 재현율(Recall)을 모두 고려하여 신뢰성이 높고 간단히 나타낼 수 있어 평가에 주로 사용되는 방식이다. 보정 과정의 결과는 CNN(ConvNet)의 평가에 사용되는 정확도(Accuracy)로 평가하였다[18][19].

$$(Accuracy) = \frac{TP + TN}{TP + FN + FP + TN} \tag{5}$$

화재 검출 시험은 각각 다른 시험 데이터셋을 사용하여 총 5회 실시하였고 추론 과정은 학습 결과와 동일한 방식으로 측정하였고, 보정 과정에서 화재 검출 발생 시 화재 영역을 그림 6의 우측과 같이 저장하였다. 화재 영역이 저장된 경우 화재(TP)로, 저장되지 않았을 경우 비화재(TF)로 분류하여 식 (5)에 대입하였다. 추론과 보정 과정에 따른 시험결과를 표 2와 표 3에 나타내었다. 그림 7에 각각 추론 과정의 예측과 정답의 예시를, 그림 8에 각 보정 과정의 데이터셋에 따른 시험 결과를 나타내었다.

그림 6. 추론 과정과 보정 과정의 화재 검출
Fig. 6. Fire detection in the inference and correction process

표 2. 화재 검출 시험 (추론)
Table 2. Fire detection test (Inference)

Data \ Evaluation metrics	mAP@0.5	mAP@[0.5:0.95]
1	0.911	0.238
2	0.991	0.345
3	0.956	0.264
4	0.996	0.375
5	0.992	0.342

그림 7. 추론 시험의 예측과 정답
Fig. 7. Predictions and answers on Inference test

그림 8. 데이터셋에 따른 보정 시험의 결과
Fig. 8. Results of correction test based on the datasets

표 3. 화재 검출 시험 (보정)
Table 3. Fire detection test (Correction)

Evaluation metrics / Data	Accuracy
1	0.83
2	0.85
3	0.85
4	0.86
5	0.80

추론 과정은 모든 프레임에서 화재가 옳게 검출되었고, 5회의 시험 평균은 mAP@0.5 0.969, mAP@[0.5:0.95] 0.313이다. 보정 과정 정확도의 평균은 0.838이고 30FPS의 영상 기준 초당 약 25프레임의 화재가 검출되었다.

비화재 검출 시험은 화재 검출 시험의 보정 과정과 동일한 방식이나. 그림 9과 같이 추론 과정에서 비화재를 의도적으로 검출시켜 보정 과정을 시험하였다. 시험 데이터셋의 이미지에 있는 화재와 비슷한 색상을 띄는 바스켓을 의도적으로 추론 과정에서 검출시켜 보정 과정에서 화재인지 판단하도록 하였다.

그림 9. 추론 과정의 바스켓 검출
Fig. 9. Detecting baskets in inference process

표 4. 비화재 검출 시험 (바스켓)
Table 4. Non-fire detection test (Basket)

Evaluation metrics / Data	Accuracy
1	1.00
2	1.00
3	1.00
4	1.00
5	1.00

바스켓은 정지된 물체이므로 움직임이 감지되지 않아 모든 시험에서 비화재로 검출되었다. 따라서 평균 정확도는 1.00으로 측정되었다.

추가로 화재 색상을 띄고 움직임이 있는 붉은 깃발을 검출시켜 동일한 방식으로 비화재 검출 시험을 실시하였다.

그림 10. 추론 과정의 깃발 검출
Fig. 10. Detecting flags of inference process

표 5. 비화재 검출 시험 (깃발)
Table 5. Non-fire detection test (Flag)

Evaluation metrics / Data	Accuracy
1	1.00
2	0.95
3	1.00
4	1.00
5	0.91

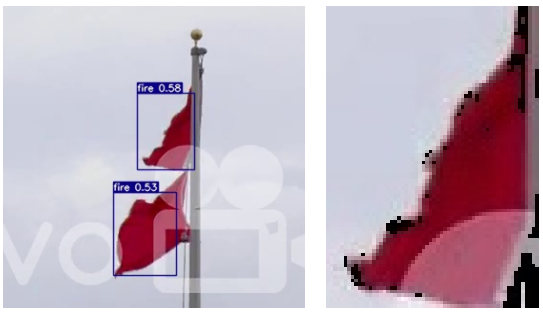

그림 11. 비화재 검출 시험의 오검출 예시
Fig. 11. False Positive Examples for Non-Fire Detection Tests

깃발의 비화재 검출 시험결과는 2번 데이터셋과 5번 데이터셋에서 그림 11과 같은 오검출이 발생하였다. 정확도 평균은 0.976으로 30FPS 기준 초당 약 1프레임의 오검출이 발생하였다. 오검출이 발생한 2번째, 5번째 시험의 경우 30FPS 기준 각각 초당 2프레임, 3프레임의 오검출이 발생하였다.

Ⅵ. 결 론

본 논문은 기존 화재 감지기가 가지는 비화재보와 느린 화재 검출 속도를 화재 검출 시스템을 구현하여 해결하고자 하였다. 화재 검출 시스템은 딥러닝 모델인 YOLOv7을 사용한 추론 과정, 색상 필터와 불꽃의 특성을 응용한 보정 과정으로 나누어진다. 추론 과정은 이미지 내 화재 영역을 실시간으로 검출하고 이를 보정 과정에 전달한다. 보정 과정은 HSV 색상 필터와 불꽃의 특징을 활용한 겉불꽃 탐지 알고리즘과 속불꽃 밝기 알고리즘으로 구성하였다. 기존 색상 필터의 백색 손실을 보완한 HSV 색상 필터로 화재 영역의 배경을 제거하여 불꽃 이미지만을 저장한다. 이를 Gunner Farneback 광학 흐름을 이용한 겉불꽃 탐지 알고리즘과 색상 필터에 의한 노이즈를 제거한다. 이미지 중심부의 밝기가 임계치 이상일 경우 속불꽃으로 판단하는 속불꽃 밝기 알고리즘으로 불꽃 후보 영역을 화재인지 판단하여 조기에 화재를 검출하고 검출 신뢰성을 개선한 시스템을 구현하였다. 성능평가는 화재 검출 시험과 비화재 검출 시험의 두 방법으로 다섯 종류의 데이터셋으로 각각 한번씩 실시하였다. 화재 검출 시험은 추론 과정의 경우 mAP@0.5 0.969, mAP@[0.5:0.95] 0.313을, 보정 과정은 평균 정확도 0.838을 나타냈다. 보정 과정의 결과를 FPS로 나타내면 30FPS 기준 초당 약 25프레임의 화재가 검출되었다. 비화재 검출 시험은 바스켓과 움직이는 깃발을 화재로 검출시켜 진행하였다. 바스켓의 경우 움직임이 감지되지 않아 5회 평균 정확도 1의 결과를 보였다. 깃발의 경우 5회 평균 정확도 0.976을 나타냈고 30FPS 기준 초당 약 1프레임을 화재로 검출하는 결과를 보였다. 향후 연구에서는 속불꽃 밝기 알고리즘의 중심 이미지가 겉불꽃과 중복되지 않고 속불꽃을 검출하는 방안과 화재 영상에서 다양한 화재의 특징점을 추출하고 딥러닝과 연계하여 오검출을 줄이기 위한 연구를 지속할 것이다.

References

[1] E-Country Indicators, "Fire Occurrence Status", https://www.index.go.kr/unity/potal/main/EachDtlPageDetail.do?idx_cd=1632 [accessed: Nov. 14, 2022]

[2] Seoul Statistics Integration Platform, "2020 Fire Occurrence Status in Seoul", https://stat.eseoul.go-.kr/statHtml/statHtml.do?orgId=201&tblId=DT_20104_O160026_2020 [accessed: May 7, 2023]

[3] Seoul Open Data Plaza, "2020 Statistics on fire misidentification in Seoul", https://data.seoul.go.kr/da-taList/10950/S/2/datasetView.do [accessed: May 7, 2023]

[4] "Seoul's 'Residential Fire Damage' Trend Analysis for the Last 5 Years", Seoul Information Communication Plaza, https://opengov.seoul.go.kr/press/167-47877, [accessed: May 7, 2023]

[5] S. W. Kim, R. S. Oh, Y. H. Bae, and J. H. Choi, "Analysis of Non-fire Alarm Causes using Malfunction Cases by Automatic Fire Detection Equipment Type in Large Shipyards", Fire Science and Engineering, Vol. 37, No. 1, pp. 51-56, Jan. 2023. https://doi.org/10.7731/KIFSE.0d7bf2e0.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, Jun. 2015. https://doi.org/10.48550/arXiv.1506.02640.

[7] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors", arXiv:2207.02696, Jul. 2022. https://doi.org/10.48550/arXiv.2207.02696.

[8] A. Munshi, "Fire Detection Methods Based on Various Color Spaces and Gaussian Mixture Models", Advances in Science and Technology Research Journal, Vol. 15, No. 3, pp. 197-214, 2021. https://doi.org/10.12913/22998624/138924.

[9] H. Bang, J. Lee, and J. Choi, "Color Filter and Deep Learning Based Fire Detection", Proceedings of KIIT Conference, Jeju, Korea, Vol. 27, pp. 452-455, Dec. 2022.

[10] F. Gunnar, "Two-frame motion estimation based on polynomial expansion.", Lecture Notes in Computer Science, Vol. 2749, pp. 363-370, Jun. 2003. https://doi.org/10.1007/3-540-45103-X_50.

[11] https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html [accessed: Jul. 29, 2023]

[12] https://www.kfi.or.kr/portal/kfiLaw/kfiLaw/kfiLawCont.do?seq=453&seqHistory=2454&lawGb=05&menuNo=500142&pageUnit=10&pageIndex=1 [accessed: Jul. 29, 2023]

[13] https://www.python.org/downloads/release/python-37-15/ [accessed: Jul. 29, 2023]

[14] https://anaconda.org/anaconda/conda/files?sort=d-ownloads&version=22.9.0&page=0 [accessed: Jul. 29, 2023]

[15] https://developer.nvidia.com/cuda-11-4-0-download-archive [accessed: Jul. 29, 2023]

[16] https://pytorch.org/docs/1.11/ [accessed: Jul. 29, 2023]

[17] https://github.com/WongKinYiu/yolov7 [accessed: Jul. 29, 2023]

[18] https://torchmetrics.readthedocs.io/en/stable/classification/accuracy.html [accessed: Jul. 29, 2023]

[19] https://torchmetrics.readthedocs.io/en/stable/detect-ion/mean_average_precision.html [accessed: Jul. 29, 2023]

저자소개

방 한 솔 (Han-Sol Bang)

2021년 2월 : 선문대학교
신소재공학과(공학사)
2022년 3월 ~ 현재 :
한국공학대학교
컴퓨터공학과 석사과정
관심분야 : Object Detection,
Image Segmentation

이 주 형 (Joo-Hyung Lee)

2021년 2월 : 한국공학대학교
ICT융합공학과(공학사)
2022년 3월 ~ 현재 :
한국공학대학교
컴퓨터공학과 석사과정
관심분야 : 인공지능, 화재감지,
IoT

최 진 구 (Jin-Ku Choi)

2003년 3월 : 와세다대학교
전자정보통신공학과(공학박사)
2003년 3월 ~ 현재 :
한국공학대학교
컴퓨터공학부 교수
관심분야 : Embedded system,
design IoT platform