Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020)
IEEE Xplore Part Number:CFP20K74-ART; ISBN: 978-1-7281-4876-2

Fire Detection using Artificial Intelligence for Fire-Fighting Robots

Sreesruthi Ramasubramanian
School of Engineering and Physical Sciences
Heriot-Watt University
Dubai, United Arab Emirates
snr1@hw.ac.uk

Senthil Arumugam Muthukumaraswamy
School of Engineering and Physical Sciences
Heriot-Watt University
Dubai, United Arab Emirates
m.senthilarumugam@hw.ac.uk

A.Sasikala
Department of Electrical and Electronics Engineering
Srisairam Institute of Technology
Chennai, India
sasikala.eee@sairamit.edu.in

***Abstract*— Fire-fighting robots are used in indoor environments to detect fires and extinguish them. Sensors such as flame sensors are currently used to detect fire in fire-fighting robots. The disadvantage of using sensors is that fire beyond a threshold distance cannot be detected. Using artificial intelligence techniques, fire can be detected in a wider range. *Haar Cascade Classifier* is a machine-learning algorithm that was initially used for object detection. The results obtained using *Haar Cascade Classifier* were not very accurate, especially when multiple fires had to be detected. Transfer learning from a pretrained YOLOv3 model was then used to train the model for fire detection to improve accuracy. The benefits and drawbacks of using deep learning for object detection over machine learning are highlighted. The algorithm used to obtain the target location the robot must move to use bounding box coordinates is also discussed in this paper.**

Keywords*— *Fire detection, Machine Learning, Deep Learning, Location finding

I. INTRODUCTION

Fire accidents cost lives and damage property. Having an autonomous fire-fighting robot that can detect fire and extinguish it will be extremely helpful in such situations. Most of the fire-fighting robots constructed in the past used sensors such as flame sensors [1] to detect fire. Fire-fighting robots also had ultrasonic sensors to detect obstacles in its path. The time taken for the pulse emitted by the sensor to travel from the object back to the sensor was used to determine the distance of the obstacle from the robot [1]. This distance was compared to a threshold value. If the distance was less than the threshold value, the robot turned in the direction of the least obstacle path and continued to move forward towards the fire.

IoT has been included in these robots [2] to communicate to the authorities about the incident. A water-based extinguisher is used for ordinary combustible material such as paper or wood and a carbon-dioxide based extinguisher is used for fires in flammable liquids such as petrol. Fire-fighting robots have been designed to have both types of extinguishers so that an appropriate type of extinguisher can be used [2][3].

Whether the sensors detect fire or not depends upon the distance between the sensor and the fire. Sensors cannot detect fire when it is beyond a certain threshold distance. Using artificial intelligence techniques, fire can be detected at a wider range which is the motivation behind exploring object detection using machine learning and deep learning techniques for fire detection. Object detection is used to find whether the object of interest is present, the location of the object, the number of objects of interest detected and the relative size of the objects.

Haar Cascade Classifier is a machine learning algorithm proposed by Paul Viola and Michael Jones that can be used to detect objects from images, video and camera feed [4]. *Haar Cascade Classifier*s have three important stages- *Integral image*, *AdaBoost* and Cascading Classifiers. The classifier is initially trained with a lot of positive and negative images. Haar features such as the two-rectangular, three-rectangular and four-rectangular features are identified for the particular object to be detected. The use of *Integral image* makes fast feature evaluation of these features possible. *AdaBoost* is then used to select the most important features from a large number of features extracted since all of the features are not useful. The use of *integral image* and *AdaBoost* ensures that the *Haar Cascade Classifier* works efficiently.

The Cascade Classifier has several stages. Different stages of the classifier are responsible for detecting different features. A strong classifier is formed combining the results of the weak classifiers. A window is slid over the image to identify positive regions containing the object using the features of the object it has been trained to recognize previously. If that particular region fails a stage, the window slides to the next region of the image and this region is no longer considered. In this manner, the *Haar Cascade Classifier* can be used to detect objects.

Deep learning-based algorithms can also be used for image classification to detect objects. *Convolutional Neural Network (CNN)* is a type of deep learning neural network [5]. Filters or kernels are applied to an input image. The purpose of each filter is to determine a particular characteristic such as the shape of eyes, ears etc.

The sliding kernel matrix is convoluted with each input matrix obtained from the image to produce an output kernel map. Several filters are used to obtain all the important features which are essential for classification. Different feature maps obtained due to the different convolution operations are combined to obtain the output. It is essential to add padding or zeros around the original matrix image accordingly to ensure that the size of the output feature map is the same as the size of the input image.

In *CNN*, a large number of regions are needed to find whether the object is present. This is because the object may have different spatial locations within the image. This increases the computational time. *R-CNN*, *Fast R-CNN*, *Faster R-CNN* and *You Look Only Once (YOLO)* were subsequently developed to reduce the testing time. *YOLO* [6] is the fastest algorithm compared to the other algorithms and is hence widely used for real-time detection. Although *YOLO* is fast, it isn't accurate as *Faster R-CNN*. The *YOLO* algorithm was developed in subsequent years to improve its accuracy. The latest development, *YOLOv3* shows substantial improvement in accuracy especially in small object recognition and is hence used widely for object detection [7].

The rest of the paper is organized in the following manner. Previous works regarding the usage of machine learning and deep learning for fire detection is described in Section 2 and the methodology followed is described in Section 3. The results obtained are presented and discussed in Section 4 and the paper is concluded in Section 5.

II. LITERATURE REVIEW

Haar Cascade Classifier was used to detect fire from CCTV footage [8]. Raspberry Pi Camera was used to capture images and Raspberry Pi 3 Model B was used to run the model and detect fires. The model was trained with a lot of positive images with the object and negative images without the object. The features were extracted from the objects to be detected such as edge features (two rectangle features), line features (three rectangle features) etc during training. The trained model was then used to identify fires from CCTV footage. In this system, IoT was also incorporated to inform the user about the fire after detection. An overview of the systemis shown in Fig. 1.

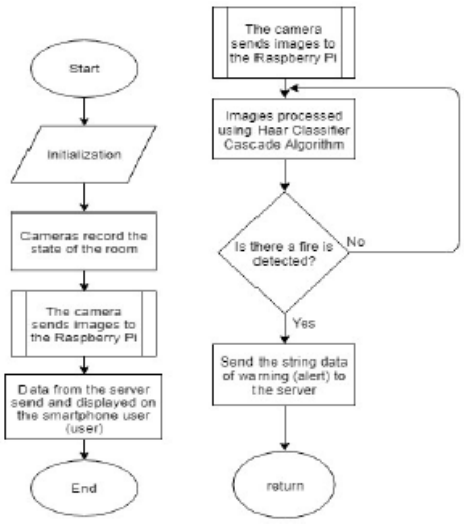

Fig. 1. Flowchart of the working of the system[8]

Based on results obtained, it was found that when the intensity of light present in the room was higher, the accuracy of detection was lower. It was concluded that the distance of the fire from the camera does not affect the level of accuracy but the brightness of the light present in the environment of fire detection affected the accuracy of measurements obtained.

Shen *et al* [9] had researched and performed deep learning for object detection. Deep learning was used rather than colour-based, motion-based or shape-based models alone as different flames may have different properties. Deep learning could be used to identify all these properties instead of a single property alone for fire detection. YOLO was used to perform flame detection. YOLO created an *n by n* grid where each grid was responsible for obtaining the probability and bounding box for the object that was present in it. The training procedure was divided into pre-training and formal training. Pre-training identified the grid where the center of the object was and classified the object. Formal training fixed the correct width and height for the bounding box. 172 images were used for training. For creating the dataset, 10 samples of each image were used by varying brightness and saturation. 60 epochs were used to train the model on the dataset.

From the results obtained, it was found that when other bright objects were present, the accuracy of the detection was compromised. The accuracy of detection was higher when the background was simpler. Hence, it is essential to not have very bright objects during detection so that accuracy does not get compromised while following both machine learning and deep learning for fire detection.

III. METHODOLOGY

The methodology followed to detect fires in an indoor environment is presented. A camera is placed in each room which is focused to capture most of the floor. The camera is placed at ground level, at a horizontal angle of 90 degrees. Detection of fires present on the floor is focused upon. The fire was detected using the respective object detection method. The steps followed to train the model and perform fire detection using *Haar Cascade Classifier* initially and *YOLOv3* later are outlined in Sections A and B respectively.

Often, the fire spreads to other locations. Using the bounding boxes obtained at the output, multiple fires in the room was detected. The specific bounding box coordinates of the fires detected was then used to obtain the local target position. Using the local target position computed concerning the room, the global target position with respect to the floor was found. The bounding box coordinates was also used to compare the relative sizes of the multiple fires to find the largest fire which should be extinguished first.

Path-planning algorithms can be implemented so that the shortest path between the robot's current position in another room on the same floor and the required target position can be found. The steps followed to compute target positions and relative sizes of fires is outlined in Section C.

A. Haar Cascade Classifier

The Cascade Trainer GUI [10] was used to train the cascade classifier model. To train the *Haar Cascade Classifier*, lots of positive and negative images were used. Positive images are images containing the object to be detected whereas

negative images are images containing the background or images without the object. Various background images were used as negative images. For the *Haar Cascade Classifier* to work well, the number of negative images must be at least twice the number of positive images. The *Haar Cascade Classifier* was trained with 147 positive images and 372 negative images and then an *OpenCV* code was used to detect fire from the images.

It was necessary to convert the image to grayscale and chose an appropriate *scale factor* and *minNeighbours* value. The *scale factor* was chosen as 1.01 to reduce the size of the image by 1%. This is done to make sure that the fire is detectable by the algorithm. The *minNeighbours* value was chosen as 10 to obtain high-quality detections.

B. Deep Learning Using ImageAI Library

Transfer learning from a pre-trained *YOLOv3* model was used to train the model to detect fire along with its co-ordinates using the *ImageAI* library in this paper [11]. Transfer learning was used to speed up the training process and ensure that the trained models have a better object detection accuracy. The dataset used had 533 images in total [12]. *LabelImg* was used to label bounding boxes around objects in images used in the dataset.

- The training code was run to train the model. An appropriate value for the batch size and number of experiments was chosen to train a model which can detect objects with good accuracy. The batch size was chosen as 8 and the number of experiments was chosen as 100 to train the model.
- As the model trains, *ImageAI* generates a json file which was saved in *fire-dataset/json folder* and generates models which were saved in *fire-dataset/models folder*.
- The best model was selected based on the *mean average precision* (*mAP*) of the model. The model generated in the 89th epoch gave the highest value of all models generated and was hence selected.
- The model then output bounding box regions having a probability of greater than 30% .

C. Obtaining the Location of Fire Using Bounding Box Coordinates

The bounding box coordinates were used to obtain the location to which the robot should move. Fig. 2 shows a representation of an image. This image epitomizes the room in which multiple fires are present.

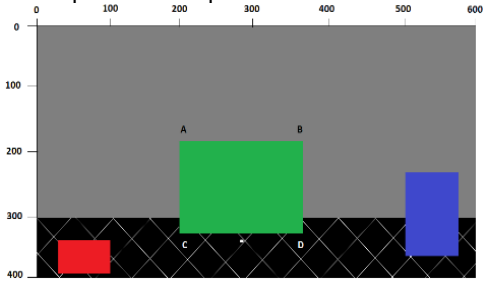

Fig. 2. Representation of an image where bounding boxes surround the object detected

The image captured by the camera is expressed in terms of image coordinates. The x-axis ranges from 0 to 600 and the y-axis ranges from 0 to 400. The red, blue and green rectangles represent the bounding boxes surrounding the three fires detected in the room. The bounding box coordinates of the fires are shown in Table 1. The bounding box coordinates describe the x and y coordinate of the top-left point (e.g. Point A in Fig. 2) of the bounding box and bottom-right point (e.g. Point D in Fig. 2) of the bounding box.

TABLE I. BOUNDING BOX COORDINATES

Bounding Box	Coordinates
Red	[30,340,100,390]
Green	[200,190,370,330]
Blue	[500,240,580,360]

The algorithm used to obtain the local and global target positions was implemented in MATLAB. The steps undertaken by the algorithm are-

- The number of fires identified in the image was one of the input parameters.
- The top-left and bottom-right point coordinates of each bounding box was another input parameter. The image coordinates of the top-left point are represented by x1, y1 and the image coordinates of the bottom-right point is represented by x2, y2. Equations (1) and (2) were used to obtain the length and height of the fire with respect to the image. Equation (3) was used to obtain the relative size of the fire with respect to the image. The size of different fires was compared to find the largest fire.

$$Length = x2 - x1 \quad \text{----- (1)}$$
$$Height = y2 - y1 \quad \text{----- (2)}$$
$$Size = length \times Height \quad \text{----- (3)}$$

- To obtain the x and y values in real-world of the target position (shown as a white dot in Fig. 2 for the fire enclosed by the green bounding box), the x-coordinate and y coordinate of the center of the lowest line of the rectangle was initially found. Equations (4) and (5) were used to obtain the x and y coordinate respectively.

$$x - coordinate = \frac{x2+x1}{2} \quad \text{----- (4)}$$
$$y - coordinate = 400 - y2 \quad \text{----- (5)}$$

- Coordinates between 0 to 600 cover the floor lengthwise and coordinates between 300 to 400 cover the floor widthwise in this case.
  For explanation purpose, let's consider that the area of the floor is 100 by 60 meters. This means that an increase in one coordinate in x direction increases the distance by 1/6 m (mapping parameter) and y

direction increases the distance by 3/5 (mapping parameter).

- The area of the floor covered by the camera is another parameter that was input to make the mapping possible.
- The x position of the target would be the value obtained by finding the product of the x-coordinate value and the mapped parameter. The y position of the target would be the product of y coordinate and the mapping parameter minus 1. It was essential to have a safe distance of 1 m since wouldn't want the robot to go too close to the fire.

In this manner, the target position of the fire in the room was obtained. The local target position obtained should be converted into a global target position to implement path-planning. The path between the global position of the robot and the global position of the fire can be identified using appropriate path-planning algorithms. Fig. 3 shows a sample Binary Occupancy Map of the entire floor. The robot used to create the Occupancy Map must-have range sensor to gather information about the obstacles present such as walls while moving around.

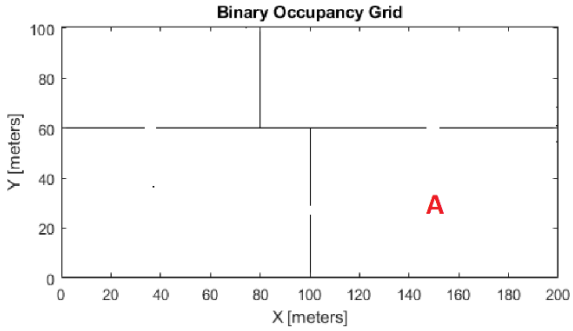

Fig. 3. Binary Occupancy Map of a floor with multiple rooms

The conversion of local position to global position has been explained for Room A (shown in Fig. 3).

- The global position of the target point can be between 100 to 200 meters x-wise and 0 to 60 meters y-wise in this case.
- The distance covered by room A is 100 meters x-wise and 60 meters y-wise.
- Equations (6) and (7) were used to obtain the global target position.

$$Global\ x\ position = Local\ x\ position + 100 \quad \text{--- (6)}$$
$$Glabal\ y\ position = Local\ y\ position \quad (7) \quad \text{----- (7)}$$

In this manner, the local target position was converted to the global target position for path-planning.

IV. RESULTS AND DISCUSSION

The results obtained using *Haar Cascade Classifier* and Deep Learning trained model is presented and analyzed. The reason for choosing the deep learning model is explained and the results obtained using *MATLAB* for obtaining global target position and size of the fire is presented. Finally, the limitations of the project are discussed.

A. Comparing results obtained using machine learning and deep learning

Haar Cascade Classifier was initially used for fire detection. The deep learning model trained using the *ImageAI* library was then used for fire detection in images. Different cases were considered for analysis, namely- small-scale single fire, large-scale single fire, small-scale multiple fires and large-scale multiple fires. The results obtained to detect the different kinds of fires are shown in Fig. 4A to 7B.

Fig. 4A. Small-scale single fire detection using Haar Cascade Classifier [13]

Fig. 4B. Small-scale single fire detection using Deep Learning trained model [14]

Fig. 5A. Large-scale single fire detection using Haar Cascade Classifier [15]

Fig. 5B. Large-scale single fire detection using Deep Learning trained model [15]

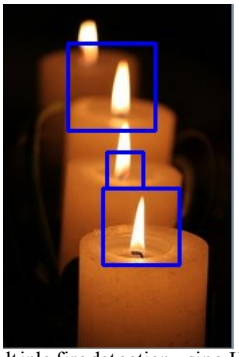

Fig. 6A. Small-scale multiple fire detection using Haar Cascade Classifier [16]

Fig. 6B. Small-scale multiple fire detection using Deep Learning trained model [16]

Fig. 7A. Small-scale and large-scale multiple fire detection using Haar Cascade Classifier [17]

Fig. 7B. Small-scale and large-scale multiple fire detection using Deep Learning trained model [17]

When *Haar Cascade Classifier* was used, most single fires were detected although the bounding box did not cover the entire region of the fire in some detections. When multiple fires were present, some fires were not detected in a few cases.

The model trained using deep learning detected most of the fires in the images with appropriate size bounding boxes. This means that using deep learning will give a more accurate result than using machine learning for a similar amount of data. However, it takes a lot of time to train a model using deep learning. Hence it is preferred to use GPUs instead of CPUs for training deep learning models. Even then, it takes a significantly long time to train the model. In contrast, you can train the model much faster while using machine learning and a CPU is sufficient enough for this process. Since accuracy is more important than the training period or computational resources, the deep learning trained model was used for fire detection.

Using the *ImageAI* library, the probability of detection along with bounding box coordinates was attained with each detection. The number of fires detected along with the probability of detection and bounding box coordinates is shown in Table 2 for Fig. 4B, 5B, 6B and 7B.

TABLE II. BOUNDING BOX COORDINATES AND PROBABILITY OF DETECTION FOR FIRE IN DIFFERENT IMAGES

Fig. Number	Fire Number	Probability (%)	Bounding Box Coordinates
4B	Fire 1	35.96	[1997,406,2323,1614]
5B	Fire 1	65.02	[108,141,262,267]
6B	Fire 1	37.57	[302,175,402,340]
	Fire 2	46.55	[210,49,303,173]
	Fire 3	38.25	[323,329,399,522]
	Fire 4	51.66	[366,562,438,755]
7B	Fire 1	36.61	[556,63,1241,691]
	Fire 2	45.61	[1357,699,1621,981]
	Fire 3	51.46	[1710,925,1970,1090]

Using the bounding box coordinates, the results obtained using MATLAB for the local and global target positions is discussed in the following section.

B. Finding the target position from MATLAB

The relative size of the fires along with both local and global target positions for fires shown in Fig. 2 was found using the MATLAB program. The results obtained are shown in Fig. 8 to 11. Fig. 8 to 10 show the x and y position locally and globally. Fig. 11 shows the largest fire obtained by comparing the size of various fires.

```
Please enter coordinates for fire 1
x = 1×4
      30   340   100   390

The local x position is 10.833333
The local y position is 5.000000
The global x position is 110.833333
The global y position is 5.000000
```

Fig. 8. Local and Global Target position for Fire 1

```
Please enter coordinates for fire 2
x = 1×4
     200   190   370   330

The local x position is 47.500000
The local y position is 41.000000
The global x position is 147.500000
The global y position is 41.000000
```

Fig. 9 Local and Global Target position for Fire 2

```
Please enter coordinates for fire 3
x = 1×4
     500   240   580   360

The local x position is 90.000000
The local y position is 23.000000
The global x position is 190.000000
The global y position is 23.000000
```

Fig. 10. Local and Global Target position for Fire 3

```
The largest fire is fire number 2
```

Fig.11. The largest fire found using bounding box coordinates

From the results, it can be seen that the robot should first move to the target location of the second fire and extinguish it using an appropriate amount of chemical before moving to the target position of other fires. The fires modelled are considered to be at a reasonable distance from the entrance of the room. If any fire is present at the entrance, the robot should extinguish it first before extinguishing the other fires.

C. Limitations

There are a few limitations regarding the proposed mechanism. The real-time detection using YOLOv3 cannot run well enough on Raspberry Pi. Therefore, computer systems would have to be used for running real-time detection instead of using a Raspberry Pi camera and the Raspberry Pi connected to the robot. The location of the target will be communicated wirelessly to the robot.

Fig. 12. Fires present one behind another being misinterpreted as a single fire [18]

Another problem is that depth cannot be interpreted from the images. This means that if another fire is present right behind, there is a possibility that the model would misinterpret the initial fire and the fire behind as a single fire as shown in Fig. 12. 3D reconstruction of the scene from a 2D image is essential for computing depth which is beyond the scope of this paper. Therefore, multiple fires which are not present directly behind one another were used to demonstrate how the location and size of the fires were obtained.

V. CONCLUSION

Machine learning and deep learning were used to train models for detecting fire from images. Deep learning gave a much accurate result compared to machine learning for a similar amount of data. Therefore, it can be concluded that adopting the deep learning approach for fire detection is superior to that of using machine learning even though the former takes significantly more time for the training process. The model trained using deep learning can be used to detect fire in real-time. The relative sizes of the fires were compared to identify the largest fire which must be extinguished first using an appropriate amount of the respective fire extinguishing substance. The local and global positions of the target location were also found for fire extinguishment using bounding box coordinates. It was essential to find the global target position to implement path-planning and find the shortest path between the robot and the target position.

REFERENCES

[1] Suresh, J., "Fire-fighting robot", 2017 International Conference on Computational Intelligence in Data Science(ICCIDS), 2017, pp.1–4.

[2] M. Kanwar and L. Agilandeeswari, "IOT Based Fire Fighting Robot," 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2018, pp. 718-723.

[3] Mittal, Shiva et al., "CeaseFire: The Fire Fighting Robot", 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, pp.1143–1146.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I.

[5] C. Kone, "Introducing Convolutional Neural Networks in Deep Learning", Available at: https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9, 2019

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788.

[7] A. Sonawane, "YOLOv3: A Huge Improvement", Available at: https://medium.com/@anand_sonawane/yolo3-a-huge-improvement-2bc4e6fc44c5,2020

[8] H. Pranamurti, A. Murti, and C. Setianingsih, "Fire Detection Use CCTV with Image Processing Based Raspberry Pi.",Journal of Physics: Conference Series, 2019

[9] Shen, D. et al., "Flame detection using deep learning", 2018 4th International Conference on Control, Automation and Robotics (ICCAR), pp.416–420.

[10] A. Ahmadi, "Cascade Trainer GUI", Available at: http://amin-ahmadi.com/cascade-trainer-gui/, October 2016

[11] O. Moses, "Train Object Detection AI with 6 lines of code", Available at: https://medium.com/deepquestai/train-object-detection-ai-with-6-lines-of-code-6d087063f6ff, 2019

[12] O.Moses,"FireNET dataset ",Available at:https://github.com/OlafenwaMoses/FireNET/releases/download/v1.0/fire-dataset.zip, 2019

[13] Pexels, Candle, Available at: https://www.pexels.com/photo/bright-burn-burnt-candle-278823/

[14] pixabay, Candle Light Church, Available at: https://pixabay.com/photos/candle-light-church-darkness-dark-2062861/

[15] pxfuel, Vehicle Fire, Available at: https://www.pxfuel.com/en/free-photo-xjdru

[16] Bigelow, Candles, Available at: https://bigelowteablog.com/2017/08/18/our-hearts-are-broken/

[17] SPANGDAHLEM AIR BASE, (2009). Harvest. Available at: https://www.spangdahlem.af.mil/News/Features/Display/Article/297217/illumination-of-cross-signifies-hope-for-a-good-harvest/