

16 Human–Machine Interface for Healthcare Technology Manufacturing	339
<i>N A Rahman and V Rajaratnam</i>	
16.1 The subsystems within the healthcare system	340
16.1.1 Primary care system	340
16.1.2 Secondary care system	341
16.1.3 Tertiary care system	341
16.1.4 Public health system	341
16.2 The patient journey	341
16.3 Healthcare stakeholders	342
16.4 Impact of technology on healthcare	343
16.5 Types of technology impacting healthcare	345
16.6 Human–machine interface (HMI) in healthcare	351
16.6.1 CPS	352
16.6.2 Nanomedicine and genomics	354
16.6.3 Robotic medicine	355
16.6.4 Rehabilitation and robots	360
16.6.5 3D printing	362
16.6.6 Case studies	366
16.6.7 Challenges of blockchain	367
16.7 Conclusion	367
References	368
17 Smart manufacturing workplace safety with virtual training, AR and haptic technologies	375
<i>Bhakti Yudho Suprpto, Ahmad Farhan Aristz, Eric Sean Kesuma and Suci Dwijayanti</i>	
17.1 Introduction	375
17.2 Robot design	376
17.3 Fire detection using YOLO	377
17.4 Shortest path using A* algorithm	379
17.4.1 A* algorithm	379
17.4.2 Map and node design	380
17.4.3 How the robots work	381
17.5 Results and discussions	382
17.5.1 Position of robots and fire	382
17.5.2 Collecting data for YOLO	382
17.5.3 Training YOLO	383
17.5.4 Evaluation of the shortest path	390
17.5.5 Receiving data	393
17.5.6 Evaluation of the entire system	395
17.6 Conclusions	398
References	399

Human Machine Collaboration and Interaction for Smart Manufacturing

Automation, robotics, sensing, artificial
intelligence, 5G, IoTs and blockchain

Edited by
Wai Yie Leong



IET CONTROL, ROBOTICS AND SENSORS SERIES 132

Human Machine Collaboration and Interaction for Smart Manufacturing

This page intentionally left blank

Human Machine Collaboration and Interaction for Smart Manufacturing

Automation, robotics, sensing, artificial
intelligence, 5G, IoTs and Blockchain

Edited by
Wai Yie Leong

Published by The Institution of Engineering and Technology, London, United Kingdom

The Institution of Engineering and Technology is registered as a Charity in England & Wales (no. 211014) and Scotland (no. SC038698).

© The Institution of Engineering and Technology 2022

First published 2022

This publication is copyright under the Berne Convention and the Universal Copyright Convention. All rights reserved. Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may be reproduced, stored or transmitted, in any form or by any means, only with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licences issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publisher at the undermentioned address:

The Institution of Engineering and Technology
Futures Place
Kings Way, Stevenage
Hertfordshire, SG1 2UA, United Kingdom

www.theiet.org

While the authors and publisher believe that the information and guidance given in this work are correct, all parties must rely upon their own skill and judgement when making use of them. Neither the author nor publisher assumes any liability to anyone for any loss or damage caused by any error or omission in the work, whether such an error or omission is the result of negligence or any other cause. Any and all such liability is disclaimed.

The moral rights of the author to be identified as author of this work have been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

British Library Cataloguing in Publication Data

A catalogue record for this product is available from the British Library

ISBN 978-1-83953-414-0 (hardback)

ISBN 978-1-83953-415-7 (PDF)

Typeset in India by MPS Ltd

Printed in the UK by CPI Group (UK) Ltd, Croydon

Contents

About the editor	xix
1 Introduction to HMI—current and future, systems, features, and benefits Human–machine interfaces in smart manufacturing	1
<i>Kamalakkannan R, Satheesh Kumar S, Tan Koon Tatt and Siva Sundar S</i>	
1.1 HMI on a growth drive	1
1.2 Origins of smart manufacturing	2
1.3 HMIs	3
1.3.1 Industry 4.0	5
1.3.2 Cause and development of the term	5
1.4 HMI features	6
1.5 HMI benefits	6
1.6 Disadvantages of HMI	7
1.7 Total global HMI dedicated AR/VR devices 2020–2030	7
References	7
2 Human–machine interaction (HMI) technology—Malaysia National Technology Roadmap Industry4WRD leading the human intelligence transformation in smart manufacturing	9
<i>Chee Fui, Wong</i>	
2.1 Smart manufacturing—a global overview	9
2.2 Malaysia smart manufacturing using HMI technologies—the call for a national policy in Malaysia	10
2.2.1 Industry4WRD: Malaysia national policy roadmap for HMI technologies	12
2.2.2 Shift factors in Malaysia national policy roadmap for smart manufacturing using HMI technologies	12
2.3 Convergence of emerging technologies	13
2.4 Malaysia readiness for Industry 4.0	15
2.5 Industry4WRD—framework	18
2.5.1 Industry4WRD objectives	18
2.5.2 Industry4WRD strategic enabler	19
2.5.3 Industry4WRD readiness assessment	19
2.6 Case study—Pentamaster—embracing Industry 4.0 automation	19
2.6.1 Background	19

Chapter 17

Smart manufacturing workplace safety with virtual training, AR and haptic technologies

Bhakti Yudho Suprpto¹, Ahmad Farhan Aristz¹, Eric Sean Kesuma¹ and Suci Dwijayanti¹

Accidents due to undetected fires have caused huge losses in various contexts in the world, such as in offices, manufacturing workplaces, residential areas, even in forest areas. The evacuation process in the firefighting system requires an effective and fast response. The source of fire must be found quickly to prevent it from spreading. Image processing based on an object detection system, it is believed, can circumvent this problem. The method for detecting objects is performed by the You Only Look Once (YOLO) algorithm in real-time. The most suitable YOLO model for the system is the Tiny YOLO VOC model which was built with the Darkflow framework. Besides detecting the fires, the robot can be used to ease the search process and firefighting efforts. However, it needs planning to fit the robots with the system optimized to do the best in the minimum time possible. In robotics, finding the fastest path can be solved by embedding the A* algorithm in the robot. A* algorithm is one of the artificial intelligence methods for finding the fastest path. Not many studies have applied this algorithm to the wheel robots to find the shortest path. Therefore, this A* method will be used for finding the fastest path to find the fire with accuracy by finding the best distance and route with the coordinate system. In this study, the wheel robots had three missions, finding the fire, extinguishing the fire, and returning to the starting points. The system accuracy for those missions was 100%, 83.33%, and 50%, respectively.

17.1 Introduction

Many accidents can occur from a human being's carelessness. Fire is one of the accidents that often occur in the factory. Thus, the manufacturing process needs to be smart to handle various types of accidents, including fire. The need for a fire detection system to be incorporated as a part of the process is growing rapidly [1]. There is a need for not only the evacuation process to be fast and effective, but also the search process to find the source of the fire. The firefighting system is efficient

¹Department of Electrical Engineering, Universitas Sriwijaya, Palembang, Indonesia

provided a robot is there to find the fire and extinguish it. In addition, a detection system is also important to find the source of the fire. Automatic systems of fire detection can utilize and detect temperature, smoke, and fire. Each sensor operates using different principles and gives different responses.

A robot can be utilized to search and detect fire by finding the best route. The procedure of finding the best route can be developed using artificial intelligence. The shortest path algorithm which is embedded in the robot may help the robot to complete its mission. However, the process of firefighting becomes a problem, especially for a wheeled robot [2]. In the previous research, some methods have been developed to find the shortest path, such as by Dijkstra [3], by Bellman-Ford [4], and research on the D* algorithm [5], and A* algorithm [6]. D* and A* algorithms have been implemented by Takayuki Goto and Takeshi in a robot and they showed that A* algorithm had better performances than D* algorithm [7]. Nevertheless, the A* algorithm has not been applied to the wheeled robot to extinguish the fire. Thus, this research utilized A* algorithm to find the best and shortest path to find the source of the fire by a robot. To do so, the robot needs to detect the fire. This can be implemented by using the image processing approach. The image processing method is applied to object detection by finding the characteristics of fire such as smoke; while fire detection is accomplished through images or real-time video. Various methods have been developed for object detection using image processing, such as R-CNN using PASCAL VOC [8], fast R-CNN for PASCAL VOC dataset and MS COCO, single shot multibox detector using PASCAL VOC, imageNet, and MS COCO [9], and YOLO using PASCAL VOC and ImageNet [10].

Some studies have shown that YOLO gives better performance in the frames per second, which is 91 fps faster than other methods often below 59 fps [11]. Even, PASCAL VOC 2007 Fast YOLO has the highest speed up to 155 fps which is twice the accuracy for real-time detector [10].

Thus, this study proposes an integrated system to detect and extinguish the fire. This system combines image processing and the shortest path algorithm. Here, YOLO is implemented to detect fire and some wheeled robots are used to extinguish the fire in a labyrinth which is assumed a factory. A* algorithm is then used to determine the shortest path passed by the robots based on the information given by the YOLO. This chapter is organized as follows: Section 17.2 describes the robot design and Section 17.3 explains the fire detection using YOLO, followed by the shortest path using A* algorithm in Section 17.4. Results and discussion are presented in Section 17.5. Finally, this chapter is summarized in Section 17.6.

17.2 Robot design

The design of the robot was divided into two parts, namely the mechanical part and the wiring. Acrylic was chosen as the material for the upper and lower part of robots. The sensors, components, and motors were put in place based on the design. The designed robot can be seen in Figure 17.1.



Figure 17.1 Firefighter robot

As shown in Figure 17.1, Robot_1 is yellow and Robot_2 is red. Both robots have the same sensors despite having different designs. The design was aimed to be distinguished by the name of the robot and by the sending of position data of the robot easily.

The hardware needed to build this robot is diverse such as a microcontroller, wireless module nRF24L01, flame sensor, driver motor DCMP H-bridge MOSFET, ultrasonic sensor, compass magnetometer 3 axis HMC5883L, relay, and LCD OLED. Each component has a different function.

Arduino Uno is the microcontroller used for radio communication with the robot. The robot receives the data sent by YOLO through wireless module nRF24L01. A flame sensor finds the location of fire or flame, driver motor DCMP H-bridge MOSFET drives the DC motor, and PING sensor is an ultrasonic sensor to detect the distance of an object in front of the sensor which beams an ultrasonic wave with a frequency of 40 kHz. The sensor detects the value of reflection received by it.

17.3 Fire detection using YOLO

A flowchart of the designed system for fire detection using YOLO can be seen in Figure 17.2. The data used for training were fire images and firefighter robots' images. In total, there were 1000 data where the fire and robot were placed randomly in the labyrinth. The data were taken above the path of the labyrinth using the external camera Logitech C525. Later, the image data were bounded box as a dataset for training YOLO.

The YOLO frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. It trains on full images and directly optimizes detection performances [12]. YOLO detects the object by

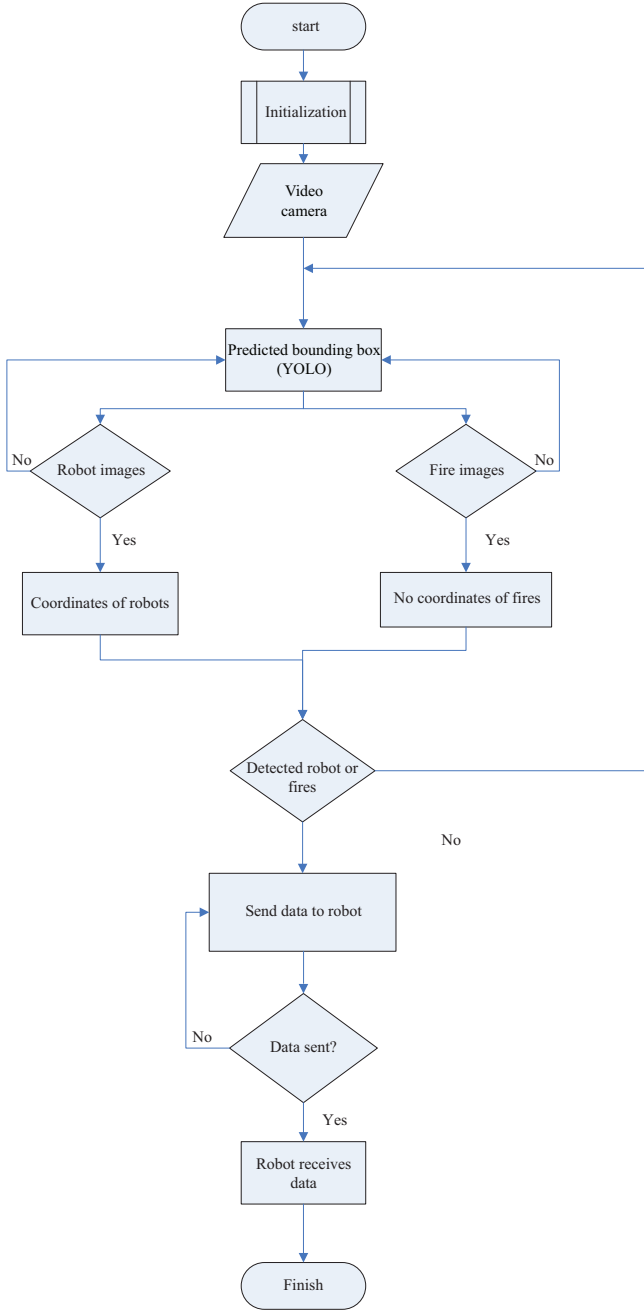


Figure 17.2 Flowchart of the designed system for fire detection using YOLO

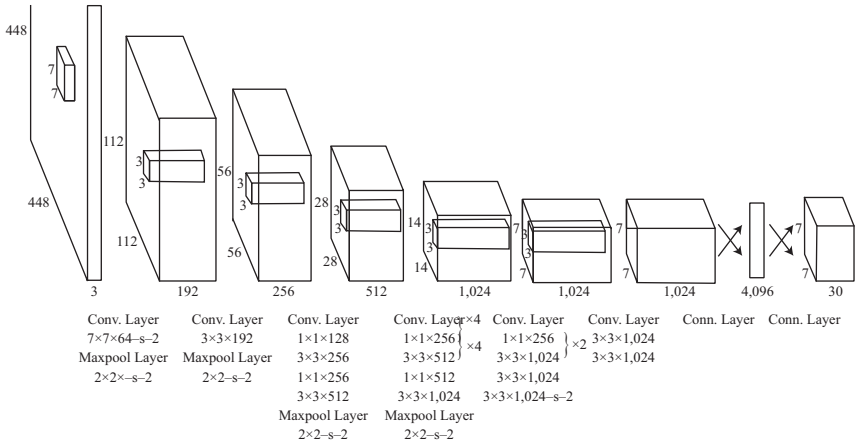


Figure 17.3 The architecture of YOLO [12]

dividing the image into some regions and predicts every bounding box and probabilities for each region. Those bounding boxes are then compared to every possibility to be predicted. YOLO has benefits compared to other classifiers since it can see the entire collection of images during training and test time. Thus, YOLO outperforms R-CNN by making less than half the number of background errors. The architecture of YOLO can be seen in Figure 17.3.

Some structures of YOLO were used in this study since the training process aims to get a model with appropriate architecture. Some models were compared to obtain the model with the smallest loss and shortest computation time. The most optimal model was then used to test the system.

Testing was performed to evaluate the designed system. In this study, two evaluation processes were done, including the detection test to find the robot and fire, and the sending data from the system to the firefighter robot.

17.4 Shortest path using A* algorithm

17.4.1 A* algorithm

A* algorithm is a computation algorithm to find the path by giving plots that can be passed from a point to several destinations. A map with several points may complicate the process to find the best path. Hence, the shortest path algorithm is needed to be embedded in a robot so it can find the best route to its destination. A* algorithm introduces heuristic in the algorithm. Hence, each step must be planned well to get the optimal decision.

A* algorithm works based on the shortest distance from the starting point to the destination point. It uses function $f(n)$ which approximates the distance from one point to another point. The distance traveled from one point to another point is

calculated as follows:

$$f(n) = g(n) + h(n) \quad (1.1)$$

where $f(n)$ is the total estimated distance from the starting to the destination point, $g(n)$ is the distance value to get the destination point, and $h(n)$ is the heuristic value that estimates value from the starting to the destination. Here, $h(n)$ is obtained by using a Euclidean distance heuristic which is formulated as follows

$$h(n) = \sqrt{(x_{destination} - x_{start})^2 + (y_{destination} - y_{start})^2} \quad (1.2)$$

17.4.2 Map and node design

In this study, the map of the labyrinth and nodes were designed to embed the A* algorithm in the robots. Each node has a code in letter form to get the position. The point was determined to ease the finding path algorithm. Coordinates serve to label positions were placed in the map so the A* algorithm can be embedded in the robots to navigate from the start point to the destination. Maps and nodes can be seen in Figure 17.4.

As shown in Figure 17.4, location points that represent the doors are A, C, F, G, and I. Besides, this map has four rooms, where room 1 has two doors C and I, room 2 with door A, room 3 has door F, and the last is room 4 with a door of G. Meanwhile, the paths are shown in the points of B, D, E, H, and J. These paths are routes which are passed by the robots to the rooms with fire inside. Coordinates in the map consist of 14 points on the x -axis and the y -axis. The coordinate points can be found using the Euclidian distance heuristic approach.

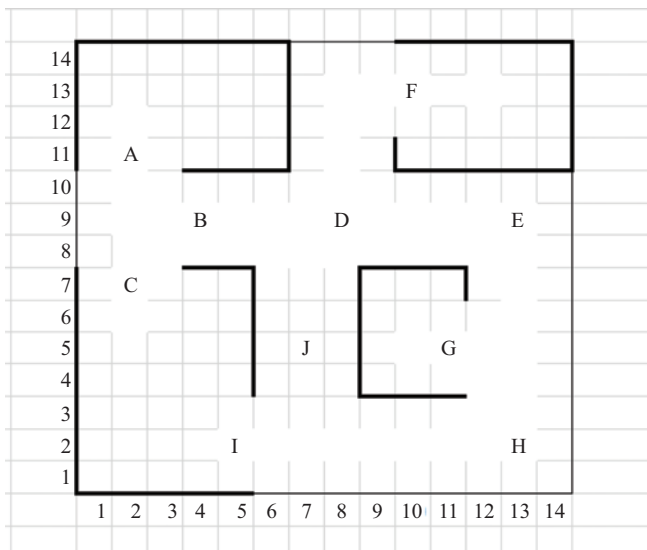


Figure 17.4 Map and obstacle nodes for firefighter robots

17.4.3 How the robots work

Figure 17.5 shows how the robot works. In the first stage, the system is initialized to find and use variable values given by the sensors and other components. Then,

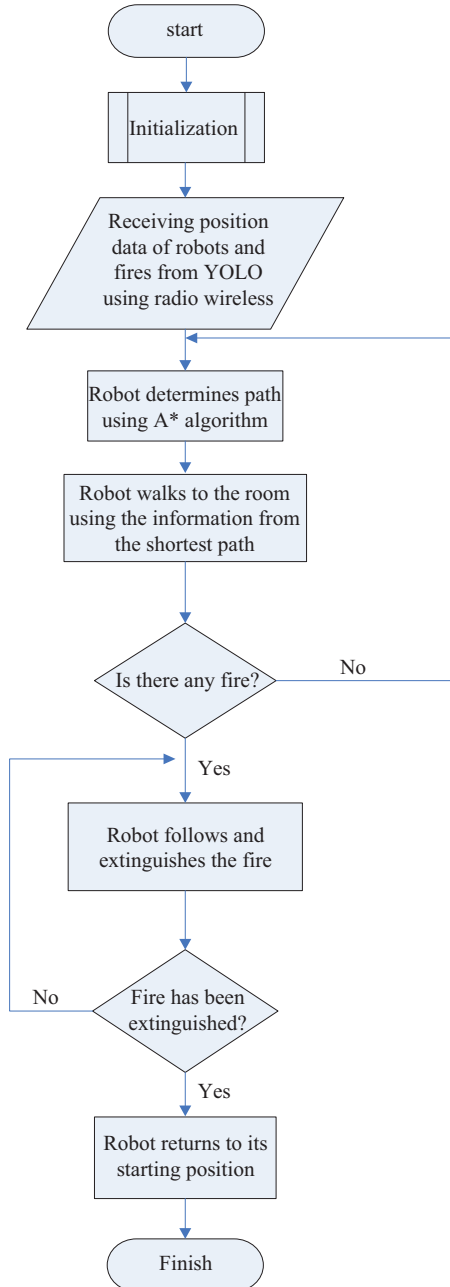


Figure 17.5 How the robot works

the robot receives the data which consists of its start position and position of flames using radio wireless 24L01. After receiving the codes, such as room and flames, both robots walk to the fire point with the shortest path given by the A* algorithm. The robot extinguishes the fire after finding the flames. Those two robots divide their work according to the rooms with fire using the shortest path given by the A* algorithm if the fire occurs not only in one room but also in other rooms. Finally, the robots return to their position once they have extinguished the fire.

17.5 Results and discussions

The workplace safety from the fire is an integrated process of object detection using YOLO and the shortest path calculation using the A* algorithm. Thus, both systems are examined in the experiment.

17.5.1 Position of robots and fire

Robots and fire were placed in the labyrinth as shown in Figure 17.6. The labyrinth is assumed as the real condition of a workplace. Each robot was placed in a different room. Then, three flames were used and placed in the different rooms. Some samples of robots' position and flames can be seen in Figure 17.7.

17.5.2 Collecting data for YOLO

The data for training YOLO was obtained by Logitech C525 camera which was placed above 3 m from the floor. The data were saved in the .PNG format with the file name 0.PNG to 1023.PNG sequentially. The data obtained were the combination taken of several possibilities of robots' position as well as flames' position in

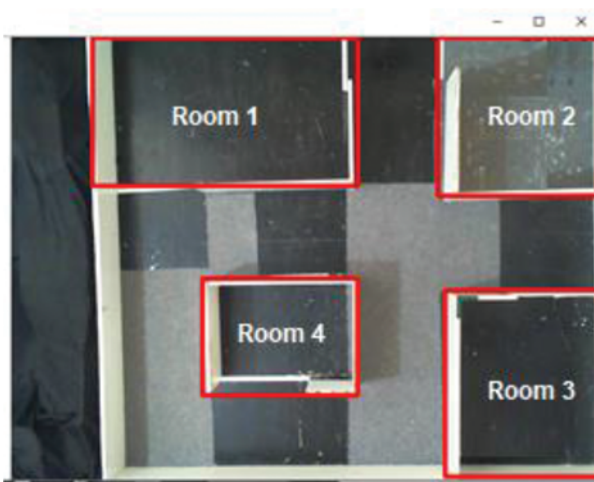


Figure 17.6 Room position in labyrinth

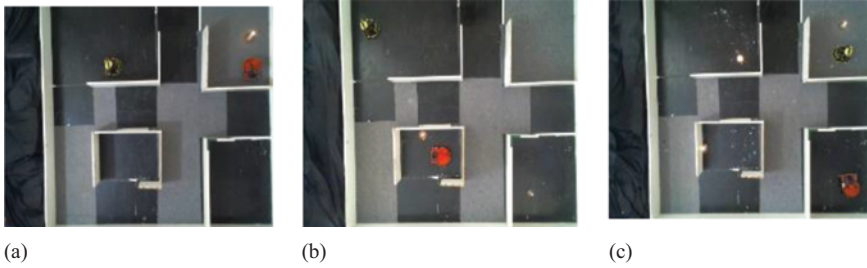


Figure 17.7 Some samples of robots' position (a) 2 robots and 1 fire (b) 2 robots and 2 fires, and (c) 2 robots and 3 fires



Figure 17.8 Labeling training data

the labyrinth. The number of robots and that of rooms is two and four, respectively. Meanwhile, the number of flames for training data may have 1, 2, or 3 flames.

The total data for training was 1,024 images with the labeled object of 3,921. The objects consist of 1,024 objects of Robot_1, 1,024 objects of Robot_2, and 1,873 objects of fire. The objects were labeled with a labeling program as shown in Figure 17.8.

The results of annotations consist of three classes, namely Robot_1, Robot_2, and Fire which were saved in XML file format.

17.5.3 Training YOLO

The training was performed using the YOLO algorithm utilizing Darkflow repository. Here, CUDA 9.0, Cudnn 7.0.5, OpenCV 3.4.6, Anaconda3 with Python 3.6, and TensorFlow 1.5 were used in the training process.

Two models were used in the training process, namely Tiny YOLOv2 and Tiny YOLO VOC. The number of epochs was set to 20, 50, and 100 epochs.

17.5.3.1 Tiny YOLOv2

The structure of Tiny YOLOv2 is shown in Table 17.1. Tiny YOLOv2 was a modified set of YOLOv2.

The training result using Tiny YOLOv2, an epoch of 20, and iteration of 1280 is shown in Figure 17.9.

As shown in Figure 17.9, the training results have large loss values. Meanwhile, a good model must have a loss value that approaches zero. Thus, the number of epochs was improved to 50 as shown in Figure 17.10.

Table 17.1 Structure of Tiny YOLOv2

Layer description	Output size
Input	(416, 416, 3)
conv 3x3p1_1 + bnorm leaky	(416, 416, 16)
maxp 2x2p0_2	(208, 208, 16)
conv 3x3p1_1 + bnorm leaky	(208, 208, 32)
maxp 2x2p0_2	(104, 104, 32)
conv 3x3p1_1 + bnorm leaky	(104, 104, 64)
maxp 2x2p0_2	(52, 52, 64)
conv 3x3p1_1 + bnorm leaky	(52, 52, 128)
maxp 2x2p0_2	(26, 26, 128)
conv 3x3p1_1 + bnorm leaky	(26, 26, 256)
maxp 2x2p0_2	(13, 13, 256)
conv 3x3p1_1 + bnorm leaky	(13, 13, 512)
maxp 2x2p0_2	(13, 13, 512)
conv 3x3p1_1 + bnorm leaky	(13, 13, 1024)
conv 3x3p1_1 + bnorm leaky	(13, 13, 1024)
conv 1x1p0_1 linear	(13, 13, 40)
max_batches	120,000
Step	-1,100, 80,000, 100,000

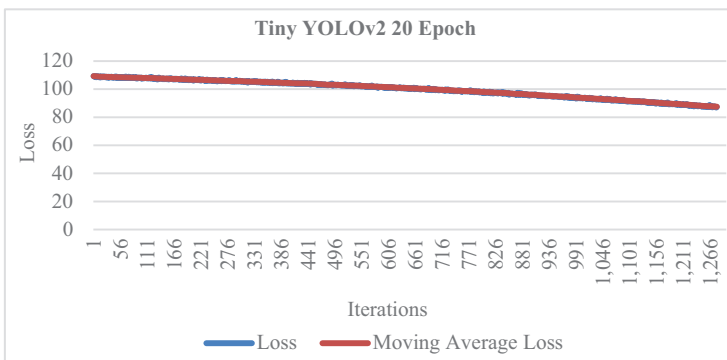


Figure 17.9 Training graph using Tiny YOLOv2 with 20 epochs

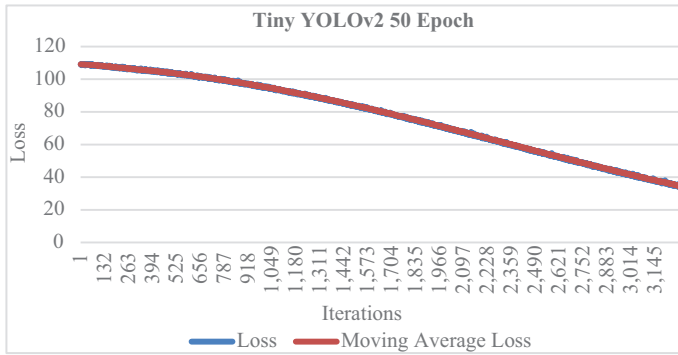


Figure 17.10 Training graph using Tiny YOLOv2 with 50 epochs

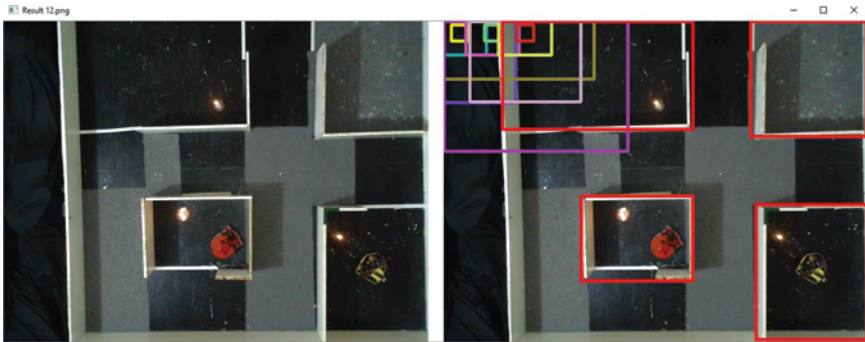


Figure 17.11 Some samples of testing results using Tiny YOLOv2 50 epochs model

Figure 17.10 shows an improvement of loss values with the final value of the loss being 34.84 in the 3,200th iteration. This value is lower than the 20 epochs. However, the model was not able to detect the robot as shown in Figure 17.11. Thus, the number of epochs is increased to 100 epochs.

The training result using 100 epochs can be seen in Figure 17.12. As shown in the figure, the performance of the model is much better than the previous two models. However, the model is not able to detect the fires and robots as shown in Figure 17.12.

These results indicate that the model is not able to detect the object. Given confidence has been decreased to 1%, the system was still not able to detect the fire or robot as shown in Figure 17.13. Thus, another structure of YOLO has to be implemented since the Tiny YOLOv2 is not able to detect the object well.

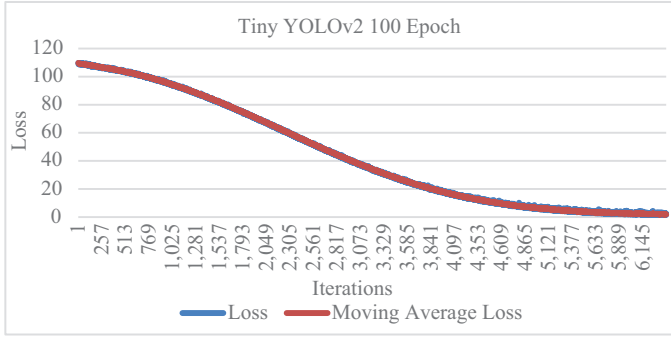


Figure 17.12 Training graph using Tiny YOLOv2 with 100 epochs

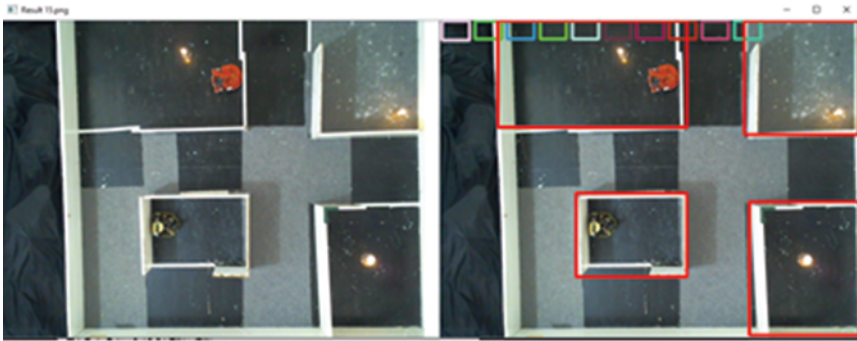


Figure 17.13 Some samples of testing results using Tiny YOLOv2 50 epochs model

17.5.3.2 Tiny YOLO VOC

Tiny YOLO VOC is a modified model of YOLOv2 and the structure of Tiny YOLO VOC can be seen in Table 17.2.

Here, three epochs were used, including 20, 50, and 100 epochs. The results of the training process using those epochs can be seen in Figures 17.14–17.16, respectively. As shown in Figure 17.14, the training result is much better than Tiny YOLOv2. The final loss value is 1.93. To validate this model, the testing was performed in 15 images as shown in Table 17.3.

As shown in Table 17.3, the class of Robot_1 has average confidence of 86.75%, Robot_2 has 76.28%, and Fire has 84.4%. These results indicate the Tiny YOLO VOC can be used to detect the object of fire and robots. However, the performance may improve by increasing the number of epochs. Figure 17.15 shows the results using 50 epochs. As shown in the figure, the loss value has decreased to 0.23 which indicates this model is good enough to be used in the

Table 17.2 Structure of Tiny YOLO VOC

Layer description	Output size
Input	(416, 416, 3)
conv 3x3p1_1 + bnorm leaky	(416, 416, 16)
maxp 2x2p0_2	(208, 208, 16)
conv 3x3p1_1 + bnorm leaky	(208, 208, 32)
maxp 2x2p0_2	(104, 104, 32)
conv 3x3p1_1 + bnorm leaky	(104, 104, 64)
maxp 2x2p0_2	(52, 52, 64)
conv 3x3p1_1 + bnorm leaky	(52, 52, 128)
maxp 2x2p0_2	(26, 26, 128)
conv 3x3p1_1 + bnorm leaky	(26, 26, 256)
maxp 2x2p0_2	(13, 13, 256)
conv 3x3p1_1 + bnorm leaky	(13, 13, 512)
maxp 2x2p0_2	(13, 13, 512)
conv 3x3p1_1 + bnorm leaky	(13, 13, 1,024)
conv 3x3p1_1 + bnorm leaky	(13, 13, 1,024)
conv 1x1p0_1 linear	(13, 13, 40)
max_batches	40,100
Step	1,100, 20,000, 30,000

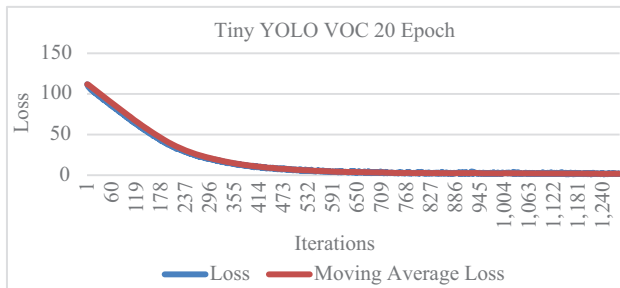


Figure 17.14 Training graph using Tiny YOLO VOC with 20 epochs

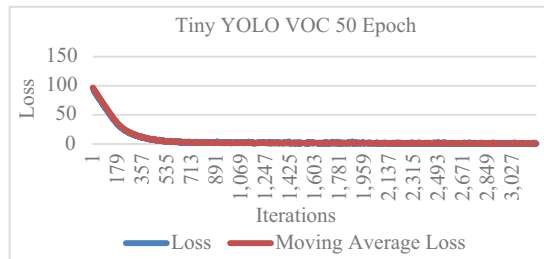


Figure 17.15 Training graph using Tiny YOLO VOC with 50 epochs

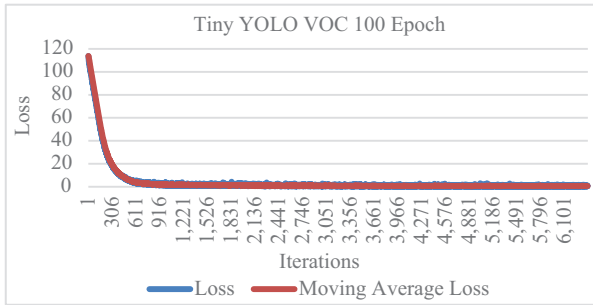


Figure 17.16 Training graph using Tiny YOLO VOC with 100 epochs

Table 17.3 Testing result using Tiny YOLO VOC 20 Epoch, threshold = 20%

Image	Robot_1		Robot_2		Fire_1		Fire_2		Fire_3	
	Loc.	(%)	Loc.	(%)	Loc.	(%)	Loc.	(%)	Loc.	(%)
1.png	1	99.1%	2	43.8%	4	96.1%	3	63.1%	3	58.3%
2.png	1	96.1%	2	76.4%	2	60.4%	3	99.5%	4	43.7%
3.png	1	79.5%	4	79.1%	–	–	–	–	–	–
4.png	1	97.3%	4	67.7%	4	27%	4	45.6%	3	97.7%
5.png	1	97.2%	4	81.9%	1	93.8%	1	26.8%	4	56.4%
6.png	3	94%	2	88.4%	2	98.8%	–	–	–	–
7.png	4	99.2%	3	91.9%	2	98.6%	–	–	–	–
8.png	4	89.8%	2	41.7%	2	86.1%	4	97.3%	–	–
9.png	4	96.7%	2	68.3%	2	92.8%	3	99.5%	–	–
10.png	4	67.6%	2	92.8%	1	99.4%	4	82.5%	–	–
11.png	4	96.9%	3	84.2%	2	99%	3	99.3%	–	–
12.png	3	61.4%	4	73.9%	1	97.9%	4	97.7%	3	98.8%
13.png	4	49.4%	1	86.4%	1	98.5%	4	88.9%	3	98.8%
14.png	3	91.8%	2	93.1%	1	87.9%	2	90.5%	4	99.3%
15.png	4	85.2%	1	74.6%	1	95.7%	2	97.7%	3	98.6%

testing. Thus, the test was done to see the performance of the model as given in Table 17.4.

As shown in Table 17.4, the system was able to detect the robots and fires. The average confidence for class of Robot_1, Robot_2, and fire are 87.06%, 87%, and 91%, respectively. The system did not have any error in detecting the robots and fires with an accuracy of 100%. To see whether this is the best model for detecting fires and robots, the number of epochs was increased to 100 epochs. The figure below shows the results of training using 100 epochs. The final loss value is 0.12 which approached zeros. Thus, this model has the lowest loss value. To test this model, 15 testing data were used as shown in Table 17.5.

Table 17.4 Testing result using Tiny YOLO VOC 50 Epoch, threshold = 20%

Image	Robot_1		Robot_2		Fire_1		Fire_2		Fire_3	
	Loc.	(%)	Loc.	(%)	Loc.	(%)	Loc.	(%)	Loc.	(%)
1.png	1	99.4%	2	53%	4	98.1%	3	93.0%	—	—
2.png	1	99.2%	2	92.1%	2	78.3%	3	99.5%	4	28%
3.png	1	89.9%	4	93.4%	—	—	—	—	—	—
4.png	1	98.9%	4	92.5%	4	70.7%	3	97.2%	—	—
5.png	1	95.8%	4	98.3%	1	95.1%	4	64.3%	—	—
6.png	3	98.1%	2	95.3%	2	99.4%	—	—	—	—
7.png	4	98.8%	3	99.2%	2	99.3%	—	—	—	—
8.png	4	89.5%	2	87.7%	2	85.8%	4	99.3%	—	—
9.png	4	97.4%	2	56%	2	93.7%	3	99.4%	—	—
10.png	4	61.2%	2	99.4%	1	99.7%	4	80.3%	—	—
11.png	4	99.4%	3	96%	2	99.8%	3	99.7%	—	—
12.png	3	65.85%	4	64%	1	98.6%	4	99.5%	3	99.8%
13.png	4	76.8%	1	73.1%	1	99.2%	4	98.8%	3	99.3%
14.png	3	81.9%	2	98.7%	1	98.7%	2	98.2%	4	99.5%
15.png	4	53.8%	1	99%	1	97.7%	2	98.9%	3	99.1%

Table 17.5 Testing result using Tiny YOLO VOC 100 Epoch, threshold = 20%

Image	Robot_1		Robot_2		Fire_1		Fire_2		Fire_3	
	Loc.	(%)	Loc.	(%)	Loc.	(%)	Loc.	(%)	Loc.	(%)
1.png	1	99.9%	2	22.7%	4	97.8%	3	87.3%	—	—
2.png	1	99.9%	2	92.9%	2	66.8%	3	99.8%	fail	fail
3.png	1	73.2%	4	95.7%	—	—	—	—	—	—
4.png	1	99.9%	4	96.4%	4	33.3%	3	96.9%	—	—
5.png	1	99.3%	4	99.8%	1	99.8%	4	30.6%	—	—
6.png	3	99.8%	2	98.7%	2	99.6%	—	—	—	—
7.png	4	99.9%	3	99.8%	2	99.8%	—	—	—	—
8.png	4	93.2%	2	84.6%	2	78.8%	4	99.8%	—	—
9.png	4	99.6%	2	17.6%	2	89.1%	3	99.8%	—	—
10.png	4	75.3%	2	99.8%	1	99.9%	4	42.8%	—	—
11.png	4	99.9%	3	99.1%	2	99.9%	3	99.9%	—	—
12.png	3	13%	4	29%	1	99.2%	4	99.7%	3	99.9%
13.png	4	60.2%	1	73.1%	1	99.3%	4	99.2%	3	99.7%
14.png	3	95.3%	2	99.8%	1	99.9%	2	98.6%	4	99.7%
15.png	Fail	Fail	1	99.8%	1	95.7%	2	99.7%	3	99.6%

As shown in Table 17.5, the model of Tiny YOLO VOC with 100 epochs has shown instability compared to the model with 50 epochs. For example, for image 1, the model with 100 epochs has a confidence of 22.7% which is lower than 50 epochs. Besides that, a few errors occurred in detecting the fires as in images 2 and

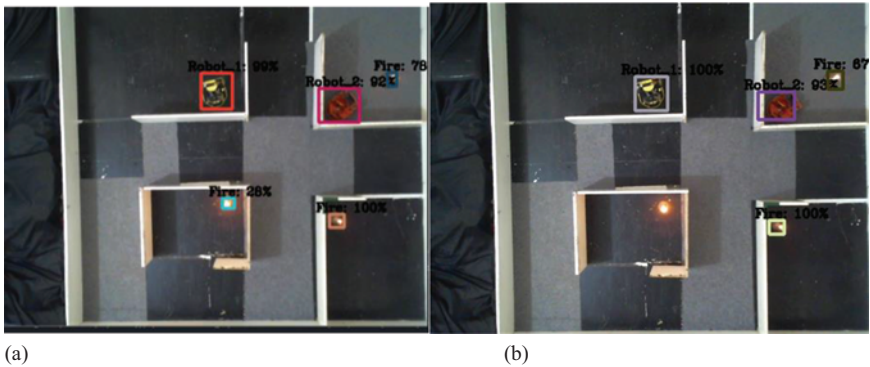


Figure 17.17 Testing results using Tiny YOLO VOC (a) 50 epochs and (b) 100 epochs

15. Comparison of testing results using Tiny YOLO VOC with 50 epochs and 100 epochs can be seen in Figure 17.17.

As shown in Figure 17.17, the Tiny YOLO VOC with 50 epochs can detect the fire in room 4. This result may be improved by decreasing the confidence threshold. However, it may influence the performance of the system as a whole. Thus, this study shows that Tiny YOLO VOC with 50 epochs is the most appropriate model to detect fire and robots.

Based on the structure of Tiny YOLOv2 and Tiny YOLO VOC as shown in Tables 17.14 and 17.15, the difference between them is the batch number. The Tiny YOLOv2 has 120,000 batches, meanwhile, the Tiny YOLO VOC has 40,100 batches. Batch shows the maximal value of sample data can be processed before the YOLO model is updated. Thus, the Tiny YOLOv2 has more cost computation than the Tiny YOLO VOC. Hence, this study utilized the model obtained by the Tiny YOLO VOC with 50 epochs.

17.5.4 Evaluation of the shortest path

Evaluation of the performance of A* algorithm was done manually by finding the shortest path. The results can be seen in Table 17.6.






As shown in Table 17.6, the A* algorithm can find the shortest path from the start point to the destination. The distance obtained by the A* algorithm is close to the actual distance which was calculated using a ruler. The distance in the A* for each box is 10.21341. The shortest distance was from A to C which is 85 cm.

An evaluation was also performed by taking the combination of the location of fire and robots. The robot used the A* algorithm to find the shortest path with some random combinations. There are some possibilities of 1 fire, 2 fires, and/or 4 fires.

17.5.4.1 Testing a robot with 1 fire

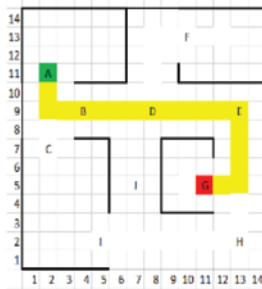
Here, the robots and fire were placed randomly. A sample of the position of robots and a fire can be seen in Figure 17.18. Green represents the start point of the robot

Table 17.6 Evaluation of the shortest path using A* algorithm

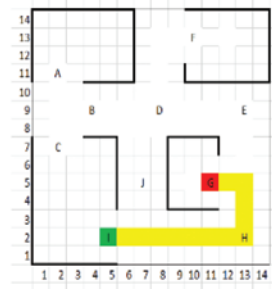
No.	View in OLED	Input location	Input destination	Route	Distance with A*	Actual distance
1		1	3	CBDF	25.30	259.7
2		2	4	ABDEG	35,30	348,8
3		3	1	FDBC	25.30	259.7
4		4	3	GEDF	30.94	325.9
5		1	4	IHG	24.61	239



(a)



(b)



(c)

Figure 17.18 Samples of robot evaluation with 1 fire (a) initial position of the robot, (b) Robot_1 walks to room 4, and (c) Robot_2 walks to room 4

and red is the door where the fire takes place. The results of using 2 robots and 1 fire as the means of obtaining the shortest path can be seen in Table 17.7. As shown in the table, both robots tried to reach the same room since there was only one source of the fire. The robot determined the shortest path to the fire.

Table 17.7 Sample of evaluation results using 2 robots and 1 fire

No	Location Robot_1	Destination Robot_1	Route Robot_1	Distance 1 (A*)	Location Robot_2	Destination Robot_2	Route Robot_2	Distance 2 (A*)
1.	A	G	ABDEG	35.30	I	G	IHG	24.61
2.	C	A	CA	8	G	A	GEDBA	35.30
3.	F	C	FDBC	25.30	A	C	AC	8
4.	G	F	GEDF	30.94	A	F	ABDF	25.30

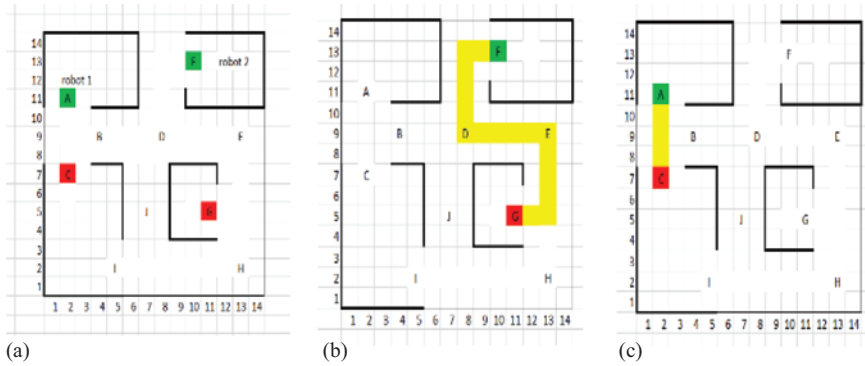


Figure 17.19 Samples of robot evaluation with 2 fires: (a) initial position of the robot, (b) Robot_1 walks to room 2, and (c) Robot_2 walks to room 4

17.5.4.2 Testing 2 robots with 2 fires

This test was performed using two robots and two sources of fire. The sample position of robots and fires can be seen in Figure 17.19. Robots and fires were placed in random locations. A sample of the position of robots and a fire can be seen in Figure 17.19.

The results for all the possible positions of robots and fires can be seen in Table 17.8. As shown in the table, two robots divided their duty to tend to the fire. The robot with the shortest position to the fire based on the A* algorithm looked for the fire, and the other robot found the other fire. The distance obtained by the A* algorithm became the information used by the robot to determine their location to the fire.

17.5.4.3 Testing 2 robots with 3 fires

An evaluation was also performed with 3 fires and the two robots placed randomly. The results can be seen in Table 17.8. As shown in the table, the robot with the shortest distance to the fire extinguished the fire. Then, both robots head to the farther fire. The distance calculated by the A* algorithm was used by robots to determine the shortest path to the fire so both robots can extinguish the fire efficiently. Besides, the A* algorithm was used to compare the distance to the fire between the robots.

17.5.5 Receiving data

Receiving data is performed by module NRF24L0, the data was received in the format data type of unsigned int [robot1, robot2, fire1, fire2, fire3]. As an example, if robot1 is in room 1, robot2 is in room 2, and the two fires are in rooms 3 and 4, the system may receive the data as a code of [12340]. The last digit is "0" since there is no fire in room 3. The result of receiving data can be seen in Table 17.9.

Table 17.8 Sample of evaluation results using 2 robots and 2 fires

No	Location Robot_1	Destination Robot_1	Route Robot_1	Distance 1 (A*)	Location Robot_2	Destination Robot_2	Route Robot_2	Distance 2 (A*)
1.	C	F	CBDF	25.30	A	A	A	0
2.	F	F	F	0	G	I	GHI	24.61
3.	G	I	GHI	24.61	A	F	ABDF	25.30
4.	G	F	GEDF	30.94	C	A	CA	8

Table 17.9 Sample of evaluation results using 2 robots and 3 fires

No.	Route Robot_1	Distance 1 (A*)	Route Robot_2	Distance 2 (A*)
1	G	0	AC	8
	GEDF	30.94	CBDF	25.30
2	C	0	F	0
	CHG	24.61	FDEG	30.94
3	GEDF	30.94	C	0
	FDBA	25.30	CA	8
4	A	0	G	0
	ABDF	25.30	GEDF	30.94
5	FDEG	30.94	A	0
	GHI	24.61	AC	8

As shown in Table 17.9, the module radio NRF24L01 succeeded to send and receive the data since the sent code is the same as the received code. Received code is processed as the location of robot position and fire position so the robot knows its position and the location of the fire. After sending the data, the robot calculates the distance using the A* algorithm and walks toward the fire's location.

17.5.6 Evaluation of the entire system

The evaluation of the system as the complete system of safety using the combination of the YOLO algorithm and the A* algorithm can be seen in Table 17.10. The robots tried to find the location of the fire and the fire was extinguished. This testing was done in ten experiments, where the fire and robots were placed randomly in the labyrinth. The number of fires for this testing was 1, 2, and 3 fires. Testing aims to see the performance of the YOLO model and A* algorithm.

As shown in Table 17.10, YOLO can detect the location of the fire and the robot's as well. The robot succeeds in finding the location of the fire. The whole system can send the code of robots' position and the location of fires using the A* algorithm. The result of detection and localization can be seen in Table 17.11.

In the evaluation of the entire system, the robots succeeded in finishing their mission to look for the fire. However, some experiments showed the robot failed to extinguish the fire because the fan installed on the robot was not strong enough to extinguish the fire. The percentage for the robot to detect and find the fire using the YOLO algorithm and shortest path of the A* algorithm is 100%. Meanwhile, the success of extinguishing the fire was 83.33%.

Analysis of the confidence value of YOLO to detect objects was also performed. This data can be seen in Table 17.13. It shows that the image processing approach can be utilized to detect fire. The average confidence value for Robot_1,

Table 17.10 *Evaluation of receiving data*

Experiment no,	Detection image by YOLO	Data location sent	Data location received	Success
1		<pre>Robot_1 diruang 1 Robot_2 diruang 2 Fire diruang 2 Fire diruang 2 Data Serial Disarkan : 12430</pre> [12430]	 [12430]	✓
2		<pre>Robot_1 diruang 1 Robot_2 diruang 2 Fire diruang 2 Fire diruang 2 Fire diruang 2 Data Serial Disarkan : 12243</pre> [12243]	 [12243]	✓
3		<pre>Robot_1 diruang 1 Robot_2 diruang 4 Fire diruang 4 Fire diruang 2 Data Serial Disarkan : 14430</pre> [14430]	 [14430]	✓
4		<pre>Robot_1 diruang 1 Robot_2 diruang 4 Fire diruang 4 Data Serial Disarkan : 14140</pre> [14140]	 [14140]	✓
5		<pre>Robot_1 diruang 4 Robot_2 diruang 1 Fire diruang 1 Data Serial Disarkan : 43200</pre> [43200]	 [43200]	✓
6		<pre>Robot_1 diruang 1 Robot_2 diruang 2 Fire diruang 2 Fire diruang 2 Data Serial Disarkan : 42240</pre> [42240]	 [42240]	✓
7		<pre>Robot_1 diruang 4 Robot_2 diruang 2 Fire diruang 2 Fire diruang 2 Data Serial Disarkan : 42230</pre> [42230]	 [42230]	✓
8		<pre>Robot_1 diruang 4 Robot_2 diruang 2 Fire diruang 1 Fire diruang 2 Data Serial Disarkan : 42140</pre> [42140]	 [42140]	✓
9		<pre>Robot_1 diruang 4 Robot_2 diruang 2 Fire diruang 2 Fire diruang 2 Data Serial Disarkan : 43230</pre> [43230]	 [43230]	✓
10		<pre>Robot_1 diruang 1 Robot_2 diruang 4 Fire diruang 4 Fire diruang 2 Data Serial Disarkan : 34143</pre> [34143]	 [34143]	✓

Robot_2, and Fire are 84%, 92%, and 88%, respectively, using the model given by Tiny YOLO VOC.

The results indicate that the combination of YOLO – to detect the fire and robot location – and the A* algorithm – to find the shortest path based on the location given by YOLO – is good enough to be implemented in various contexts,

Table 17.11 Evaluation of the entire system


Experiment no.	Detected images by Tiny YOLO VOC	Data sent	Remarks
1		21,400	Robot finished the missionRobot_1 failed to return to the start position
2		32,100	Robot finished the missionRobot_2 failed to return to the start position
3		12,230	Robot finished the mission
4		41,320	Robot_1 finished the missionRobot_2 found the fire
5		13,430	Robot finished the mission
6		23,140	Robot finished the mission
7		13,123	Robot finished the mission to fire_2Fire_3 was found
8		13,200	Robot finished the mission
9		21,400	Robot finished the missionRobot_2 failed to return to the start position
10		13,123	Robot finished the mission

Table 17.12 *Evaluation of the entire system in real-time*

Experiment no.	Robot_1		Robot_2		Fire	
	Location	Remark	Location	Remark	Location	Remark
1	2	True	1	True	4	True
2	3	True	2	True	1	True
3	1	True	2	True	2,3	True
4	4	True	1	True	3,2	True
5	1	True	3	True	4,3	True
6	2	True	3	True	1,4	True
7	1	True	3	True	1,2,3	True
8	1	True	3	True	2	True
9	2	True	1	True	4	True
10	1	True	3	True	1,2,3	True
Percentage of success		100%	–	100%	–	100%

Table 17.13 *Confidence value of system evaluation*

Experiment no.	Confidence value Robot_1	Confidence value Robot_2	Confidence value Fire
1	96%	99%	93%
2	99%	90%	99%
3	91%	93%	95%, 99%
4	100%	100%	98%, 99%
5	90%	91%	99%, 98%
6	90%	93%	94%, 85%
7	85%	90%	85%, 90%, 90%
8	36%	88%	27%
9	85%	90%	79%
10	67%	90%	85%, 86%, 77%
Average	84%	92%	88%

including manufacturing. The proposed method is effective in extinguishing the fire which may take place in the manufacturing with various rooms.

17.6 Conclusions

Image processing can be used to detect fire using a deep learning approach, namely the YOLO algorithm. Tiny YOLO VOC shows better performance than Tiny YOLOv2 to detect the object with an accuracy of 100%. The A* algorithm can be implemented in a multi-robot scenario to determine and compare the distance from the start point to the destination (fire location) by calculating the shortest distance. The system to detect and extinguish the fire can find the shortest path to the fire with a success rate of 100%. A combination of YOLO and A* algorithm can be implemented in real-time conditions, such as manufacturing workplace.

However, further study is still needed to improve the capability of the robot to extinguish the fire and return to the start position since the current system has an 83.33% success rate to extinguish the fire and 50% to return to the start point. Besides, various data for training based on the environment of the robot's acting out to extinguish the fire can be improved.

References

- [1] S. Wilson, S.P. Varghese, G.A. Nikhil, I. Manolekshmi, and P.G. Raji, "A comprehensive study on fire detection," *Proceedings of IEEE Conference on Emerging Devices and Smart Systems ICEDSS 2018*, no. March, pp. 242–246, 2018.
- [2] F. Duchon, A. Babinec, M. Kajan, *et al.*, "Pathplanning with modified A star algorithm for a mobile robot," *Procedia Eng.*, vol. 96, pp. 59–69, 2014.
- [3] M.A. Javaid, "Understanding Dijkstra's Algorithm Abstract: Introduction," pp. 1–27.
- [4] D. Walden, "The Bellman-Ford Algorithm and 'Distributed Bellman-Ford,'" pp. 1–12, 2003.
- [5] D. Ferguson and A. Stentz, "The Field D * Algorithm for Improved Path Planning and Replanning in Uniform and Non-Uniform Cost Environments."
- [6] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, 2nd ed., vol. 7, 2001.
- [7] T. Goto, T. Kosaka, and H. Noborio, "On the Heuristics of A* or A Algorithm in ITS and robot path-planning," *Proc. 2003 IEEE/RSJ International Conference on Intelligent Robots and System (IROS 2003)*, Vol. 2, pp. 1159–1166, 2003.
- [8] R. Girshick, J. Donahue, T. Darrell, J. Malik, U. C. Berkeley, and J. Malik, "R-CNN: region based convolutional neural networks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, p. 2–9, 2014.
- [9] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015.
- [10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.
- [11] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.
- [12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016.