[[1]](#footnote-1)



## Fig. 1. Exemplary visualization of AMG8833 sensor reading as a heatmap. Three red spots are people in the room which are captured by the sensor on the ceiling at 3 m height.

Privacy-Preserved Social Distancing System Using Low-Resolution Thermal Sensors and Deep Learning

Aisha Fahad Alraeesi, Hanan Fekri Kharbash, Jawaher Saif Alghfeli, Shamma Sultan Alsaedi, and Munkhjargal Gochoo, Member, IEEE

*Abstract*— During the current COVID-19 pandemic, it was obvious that life slowed down, and people became less productive with remote working through online platforms. It has become a necessity to find solutions to facilitate our daily routines such as going to school, work and travel while abiding by the guideline for social distancing and cope with the existence of COVID-19 or any upcoming health pandemics. We employed an IoT and deep learning method to propose an indoor privacy-preserved social distancing wireless sensor network-based system using 8×8 low-resolution infrared sensor AMG8833, which can be used by companies and organizations to ensure that social distancing is practiced at all time. To the best of our knowledge, there is no practical privacy-preserved technology in the market. We collected a total of 6606 infrared low-resolution images using four wireless sensor nodes to cover as many as 200 cases. We employed YOLOv4-tiny object detection model for detecting persons which is trained on our dataset, and achieved mAP@0.5 equal to 95.4% and inference time of 5.16ms. Thus, we conclude our proposed novel social distancing system has a potential in fighting COVID19.

***Index Terms—*Human localization, privacy-preserved, smart office, infrared sensors, unobtrusive, AMG8833, YOLO**

# INTRODUCTION

With the emerge of the COVID-19 crisis, the world had been permanently reshaped. COVID-19 has affected the way we live and interact with each other, the way we work and study, and the way we travel.

In March 2020, COVID-19 had become a global pandemic, which caused most countries to start shutting down their public activities, services, and businesses. This shutdown had a big impact on the global economy which forces many companies to either declare bankruptcy or reduce their employee significantly. In both cases, a lot of people lost their jobs. Therefore, there were urgent demands to find other ways to work around the restrictions imposed by this pandemic. For instance, moving from complete shutdown status to working in distance and practicing social distancing allowed us to contain the virus while continuing our daily lives and jobs.

We are proposing a solution to help limit the spread of COVID-19 while keeping facilities and workplaces open to welcome back their staff, customers, or students in a safer supervised environment. The main goal of this work is to provide a system for monitoring and alerting social distancing violations that do not capture any identifying information about the monitored people, and by so does not invade people’s privacy.

In our solution we apply deep learning techniques on data collected by multiple low-resolution infrared sensors from multiple wireless sensor nodes, each node covers an area of 2.5m×2.5m, a sample sensory reading is illustrated as a heatmap in Fig. 1. A wireless sensor node consists of a microcontroller (PyCom WiPy3.0), an ultra-low resolution infrared sensor (AMG8833), and a buzzer, that will be attached to the ceiling. All wireless sensor nodes and the local server are connected to the same WiFi network and use MQTT messaging protocol to exchange messages between nodes and the local server. The wireless sensor nodes will send their sensor readings to the local server, the local server will preprocess data received and perform deep learning object detection techniques to check for proximity rule violations, if found, it will send an alarming message to the wireless nodes involved to initiate the buzzer and red light. A system administrator can use our provided web application to view system statistics such as the number of violations over time.

# Related Works

## Related Works

Due to the current situation, many social distancing solutions [1-15] are developed. Those solutions can be divided into three categories: smartphone-based [1-5], wearable devices-based [6-9], and camera-based [10-15].

## Competitive advantage

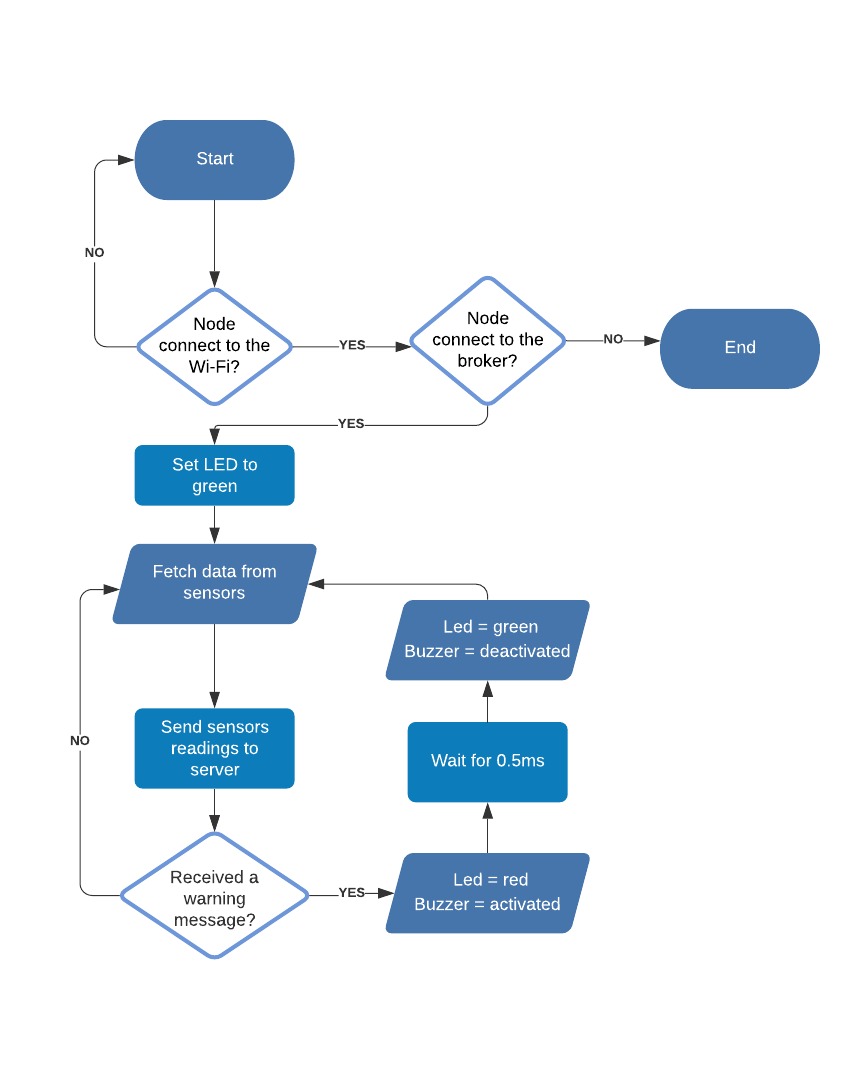


Fig. 3. Program flowchart of the wireless sensor node.

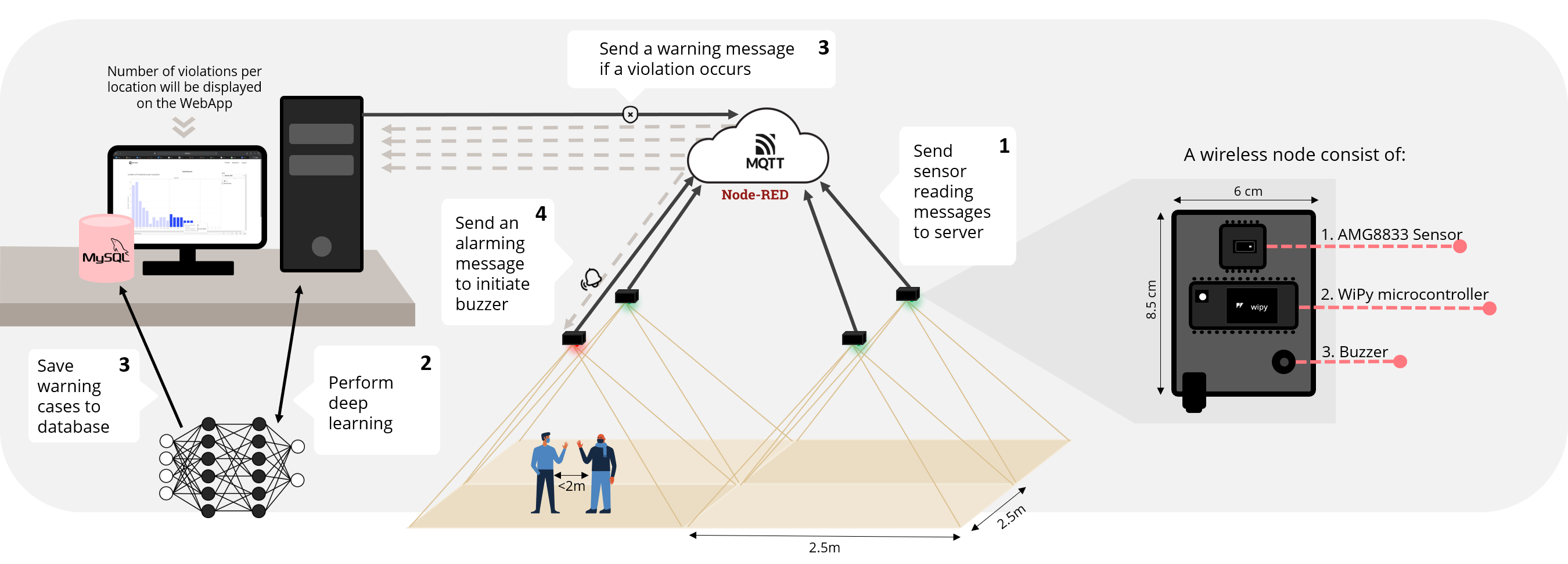


Fig. 2. A framework of the proposed system.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TABLE I. RELATED WORKS | | | | | | | | | | | | |
| **No.** | **Methods** | **Technology used** | **Advantage** | | **Disadvantage** |  | **No.** | **Methods** | **Technology used** | **Advantage** | **Disadvantage** |
| *Smartphone-based* | | | | | |  | 7 | ZoneSafe P2P Social Distancing [9] | RFID | Each ZoneSafe P2P VibraTag is hermetically sealed to provide long-term protection against water and dust ingress and is rated to IP67. | Not that accurate( accuracy: up to 2 m). |
| 1 | NOVID [1] | Ultrasound and Bluetooth | Data are encrypted and anonymized (No need to provide personal information like phone number, email, etc.), ultrasound technology provides better accuracy than Bluetooth alone. | | Uses a microphone; the level of privacy is questionable. |  |
|  |
|  | *Camera-based* | | | | |
| 2 | FastSensor ADAM [2][3] | Turnkey sensors | It is not PII (not a Personally Identifiable Information system) based platform. | | Everyone needs to download the app, and does not work with people not carrying their phones or having them turned off |  | 8 | KOGNIZ [10] | AI and ML detection algorithms | Check the temperature, violation of wearing masks, and social distancing all in one, everything you need is on the camera. | No personal privacy |
|  |
|  |
| 3 | 1point5 [4][5] | Bluetooth | Simple and free. | | Bluetooth penetrates walls and can falsely record interactions between people who are physically separated |  | 9 | Landing AI [11] | Faster R-CNN + non-max suppression (NMS) | integrated into existing security camera, no need to buy new hardware. | did not specify how accurate is their system. There is no specific way to alarm when violation happens. |
|  |
|  |
| *Wearable devices-based* | | | | | |  | 10 | Distance Assistant by Amazon [12] | machine learning and augmented reality. | Open source, gives immediate visual feedback, can be deployed easily anywhere | Privacy issue, the monitor will display what the camera captures so everyone in the place will be able to see it. |
| 4 | Social Distancing Sensor (SDS) [6] | UWB | | The tag will not store any data. | Cost more for larger quantities and need to carry the tag all the time |  |
|  |
|  | 11 | AXXONSOFT [13] | using AI-powered behavior analytics | detect all sort of violation Mask, social distancing, temperature, employee access by face recognition, control the occupancy of the building. | it sends a screenshot throw email as sort of alarm which is not practical in real time. Did not specify the accuracy. |
| 5 | RIGHT CROWD Social Distancing Monitoring [7] | BLE | | Continuous use of up to 80 hours | Not very accurate, badges are not waterproof |  |
|  |
|  |
| 6 | TSINGOL UWB Social Distancing [8] | UWB | | High precision | need to carry the tag all the time. |  | 12 | Social Distancing AI by Actuate [14][15] | AI algorithm that detects people within 6 feet of each other. | budget-friendly; requires no new hardware. | Alerts are sent to the staff so no immediate action is taken. |
|  |
|  |

## Our solution uses multiple low resolutions 8x8 thermal sensors to return an array of 64 individual infrared temperature readings, which makes it impossible to have identification information of people in the room, but just enough to detect it as a human. The wireless nodes will be attached up in the ceiling and the sensor viewpoint will show people as red dots. Fig. 1 shows an exemplary visualization of a heat map produced by sensor readings. Our solution will also cost less than wearable technologies since our system cost is independent of the number of users but fixed for a certain area. Moreover, we do not need to worry about people taking off their wearable devices, not carrying their phones, or turning off their Bluetooth. Battery life is also no longer a concern since our system will be hooked directly to electricity.

# Methods

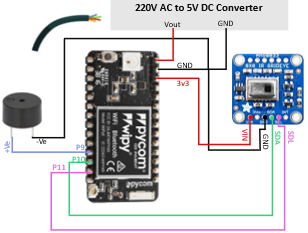


Fig. 4. Wireless sensor node circuit diagram connection.

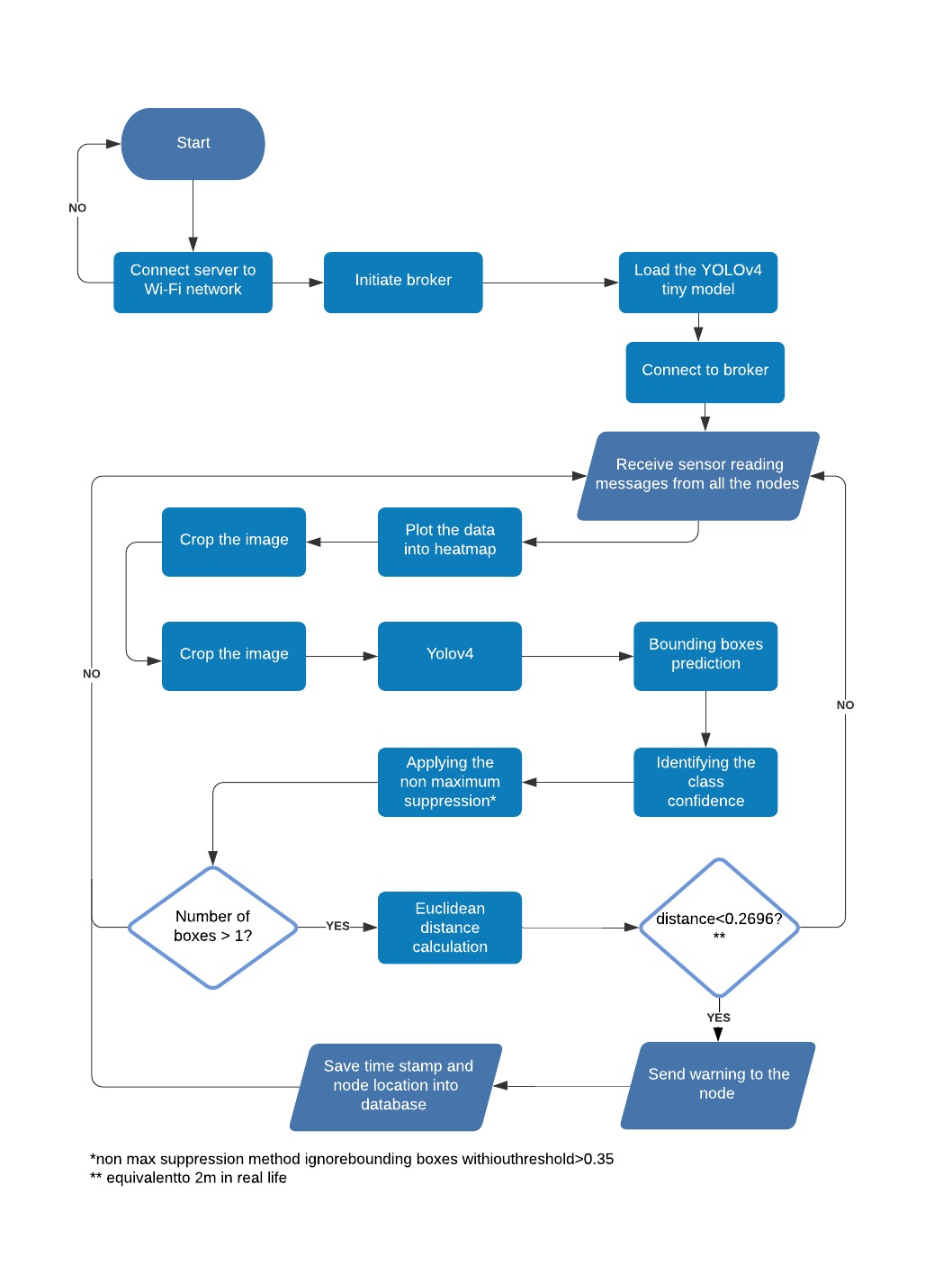


Fig. 5. Flowchart of the local server

The system framework is summarized in Fig. 2. It consists of several wireless sensor nodes and the local server where messages are exchanged using MQTT protocol. The system components features are described below.

## Wireless sensor nodes

The microcontroller will fetch data from the low-resolution sensor and send the sensor readings as messages to the local server. The wireless node will receive the alarming message that will be sent by the local server only if any social distancing violations happen, to initiate alarming in the node side. In Fig. 3 shows the flowcharts that explain what happens on the wireless sensor node.

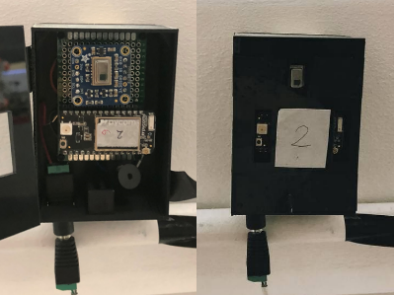


Fig. 6. Wireless sensor node.

## Local server

The local server is responsible for doing all the computations, it will perform deep learning on the data received from the nodes and if a violation found, the local server will then send an alarming message to the nodes involved to initiate their buzzer and red light while also saving the timestamp and location of violation in the database. In Fig. 5 the flowchart explains what is happening on the local server side.

## MQTT protocol

Wireless sensor nodes and the local server are exchanging messages using MQTT which is a lightweight IoT messaging protocol over Wi-Fi. An MQTT broker which is responsible for routing all messages is deployed locally using Node-RED.

## Deep learning

YOLO, you only look once, is a real-time object detection model, that apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. Making it much faster than other types of models like R-CNN and Fast R-CNN [16].

Darknet library was used for training and developing our deep learning model, and TensorFlow library for deploying it in real time.

## Web Application

The data retrieved from the database will be visualized in the web application using Tableau. The web application can be accessed only by the system administrator, using email and password.

# Dataset and Experimental Results

The implementation part of the project includes the following: wireless sensor node, dataset, deep learning, distance calculation and the web application.

## Wireless sensor nodes

A 3D printer was used to create the enclosure case. As shown in Fig 6, the case is of the size of only 6x8.5 cm, the cover of that case has three openings, one for the sensor’s head (to not cover the viewing angle which is 60°) [17], the other two are for the RGB led and antenna.

The circuit in Fig.4 was soldered into a solderable board then wired into a DC power jack, all wireless sensor nodes are supplied using a 220v to 5v converter.

Fig. 7 shows nodes attached to the ceiling at a height of 3m and are 2.5m apart from the other nodes.

Wireless sensor node’s code is written using MicroPython language on the WiPy 3.0 microcontroller.

The retrieved sensor data is returned as a string that contains an 8x8 matrix and ending with two values, the minimum and maximum heats. Below is an example of the returned value. Only the 8x8 matrix is sent to the local server using the MQTT protocol. When the node receives an alarming message from the server, it will initiate the buzzer and the red light.

## Dataset

العنوان: العنوان: العنوان: العنوان: Title: صورة تحتوي على داخلي, المصباح

تم إنشاء الوصف تلقائياً

Fig. 7. Four wireless sensor nodes attached to the ceiling.

العنوان: العنوان: العنوان: العنوان: Title: Graphical user interface, application, Word

Description automatically generated

العنوان: العنوان: العنوان: العنوان: Title: Diagram

Description automatically generated

Fig. 8. YOLOv4 dataset with bounding boxes.

TABLE II. YOLOv4 Object Detection Models Results.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | mAP@0.5 | Precision | Recall | F1-score | Inference time (ms) | FPS |
| YOLOv4 | 92.96% | 0.89 | 0.94 | 0.91 | 43.9 | 22.78 |
| Scaled-YOLOv4 | 97% | 0.836 | 0.976 | 0.90 | 19 | 52.63 |
| **YOLOv4-tiny** | **95.4%** | 0.93 | 0.96 | 0.94 | **5.16** | 193.7 |

To train the deep learning models, we created our dataset. The low-resolution images were collected using four AMG8833 sensors from four wireless sensor nodes, covering more than 200 cases each recorded for 10 seconds and we ended up with 6606 data samples.

As a preprocessing step we started with normalizing the data to reduce the noise, add tables as white pixels to eliminate non-human hot objects, then converted this data into images and lastly cropped them into squares. Object detection data set is created using Roboflow [18] online tool which was used to create bounding boxes on red areas, and to label them as a class person as shown in Fig. 8.

Dataset was divided as train 70%, test 20%, and validation 10%.

## Deep Learning

We trained three different YOLOv4 object detection models for class ‘person’, with a threshold of 0.5, and got the following results presented in Table II, these results obtained from testing validation set on Google Colab GPU, because of hardware limitation, we were not able to test the model with a GPU on our local server.

We chose YOLOv4-tiny since it got a mAP accuracy of 95.4% which is very high, and the shortest inference time of only 5.16 ms.

العنوان: العنوان: العنوان: العنوان: Title: Chart, line chart

Description automatically generated

Fig. 9. mAP and loss graph for YOLOv4-tiny.

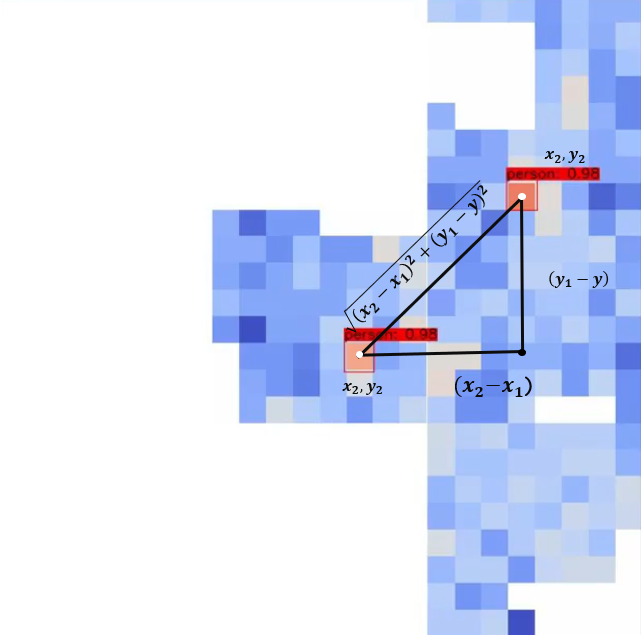


Fig. 10 Example of predicted bounding boxes using YOLOv4-tiny, and distance calculation between the two persons.

## Distance calculation

After the YOLOv4-tiny model predicts the bounding boxes for each detected person, Fig. 10 shows an example output of that. If two or more people were detected, the algorithm will get the center location of the bounding boxes and calculate the distance between every pair using Euclidean distance. Then compare that distance with the value that corresponds to 2 meters in real distance (this value equals 0.269). Euclidean distance is calculated as follows:

Euclidean distance =  (1)

Furthermore, predicted bounding boxes are also used to locate under which node the violation happened, to then send the alarming message to the nodes involved.

## Web application

The web app is designed using Flask framework since we cannot integrate python into the browser directly. This site's goal is to provide the administrator with statistics regarding violations. After we finished the YOLOv4-tiny model, we linked it to the MySQL database to store violation records.

## Testing

TABLE V. Testing System Distance Calculation Accuracy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case #** | **Real distance (cm)** | **Calculated distance (cm)** | **Difference (cm)** | **Percent error (%)** |
| **1** | 205 | 198 | 7 | 3.41 |
| **2** | 463 | 414 | 49 | 10.58 |
| **3** | 130 | 86 | 44 | 33.84 |
| **4** | 204 | 198 | 6 | 2.94 |
| **5** | 118 | 130 | -12 | 10.10 |
| **6** | 293 | 284 | 9 | 3.07 |
| **7** | 144 | 152 | -8 | 5.56 |
| **8** | 266 | 275 | -9 | 3.38 |
| **9** | 153 | 158 | -5 | 3.26 |
| **10** | 239 | 218 | 21 | 8.78 |
| **11** | 300 | 309 | -9 | 3.00 |
| **12** | 90 | 90 | 0 | 0 |
| **13** | 166 | 150 | 16 | 9.63 |
| **14** | 96 | 94 | 2 | 2.08 |
| **15** | 130 | 125 | 5 | 3.84 |
| **16** | 210 | 207 | 3 | 1.42 |
| **17** | 130 | 124 | 6 | 4.61 |
| **18** | 85 | 97 | -12 | 14.11 |
| **19** | 110 | 96 | 14 | 12.73 |
| **20** | 425 | 404 | 21 | 4.94 |
|  | **Average difference (cm)** | **12.9** | **Average percent error %** | **7.06** |

From the deep learning models, we are only testing YOLOv4-tiny model. All our real-life testing done using a PC running a CPU only, so it could be even faster if done with a GPU.

Experiment setup:

* Tools: Four wireless sensor nodes attached to the ceiling, 220v to 5v adapter and a PC
* Location of testing: AI & Robotics Lab, CIT, UAEU (Fig. 11 shows layout of the lab with location of attached wireless sensor nodes)
* Wireless sensor nodes are attached on areas with more likelihood of people passing through.
* Wireless sensor nodes are attached at a height of 3m from ground and covers an area of 2.5m x 2.5m.
* White pixels are tables.

## YOLOv4-tiny Confusion matrix

Table III. shows the confusion matrix of the YOLOv4-tiny model, results acquired from testing the model on a test dataset containing 435 detection objects. Accuracy is calculated using the following formula:

Accuracy % = (TP+TN) / (TP+TN+FP+FN) \* 100 (2)

## Testing YOLOv4-tiny output

TABLE IV. Testing System Alarming Accuracy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| case | Total number of iterations | # of correct output | # of false output | Accuracy % |
| empty | 100 | 100 | 0 | 100 |
| One person | 100 | 100 | 0 | 100 |
| Two people | 100 | 97 | 5 | 97 |
| Three people | 100 | 92 | 8 | 92 |
| Four people | 100 | 89 | 11 | 89 |
|  |  |  | **Avg. Accuracy** | **95.6%** |

TABLE III. YOLOv4-tiny Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **n=435** | **predicted positive.** | **predicted negative.** |  |
| **Actual positive** | TP = 364 | FP = 29 | 393 |
| **Actual negative** | FN = 17 | TN = 25 | 42 |
|  | 381 | 54 |  |
|  | **Accuracy %** | **96.1%** |  |

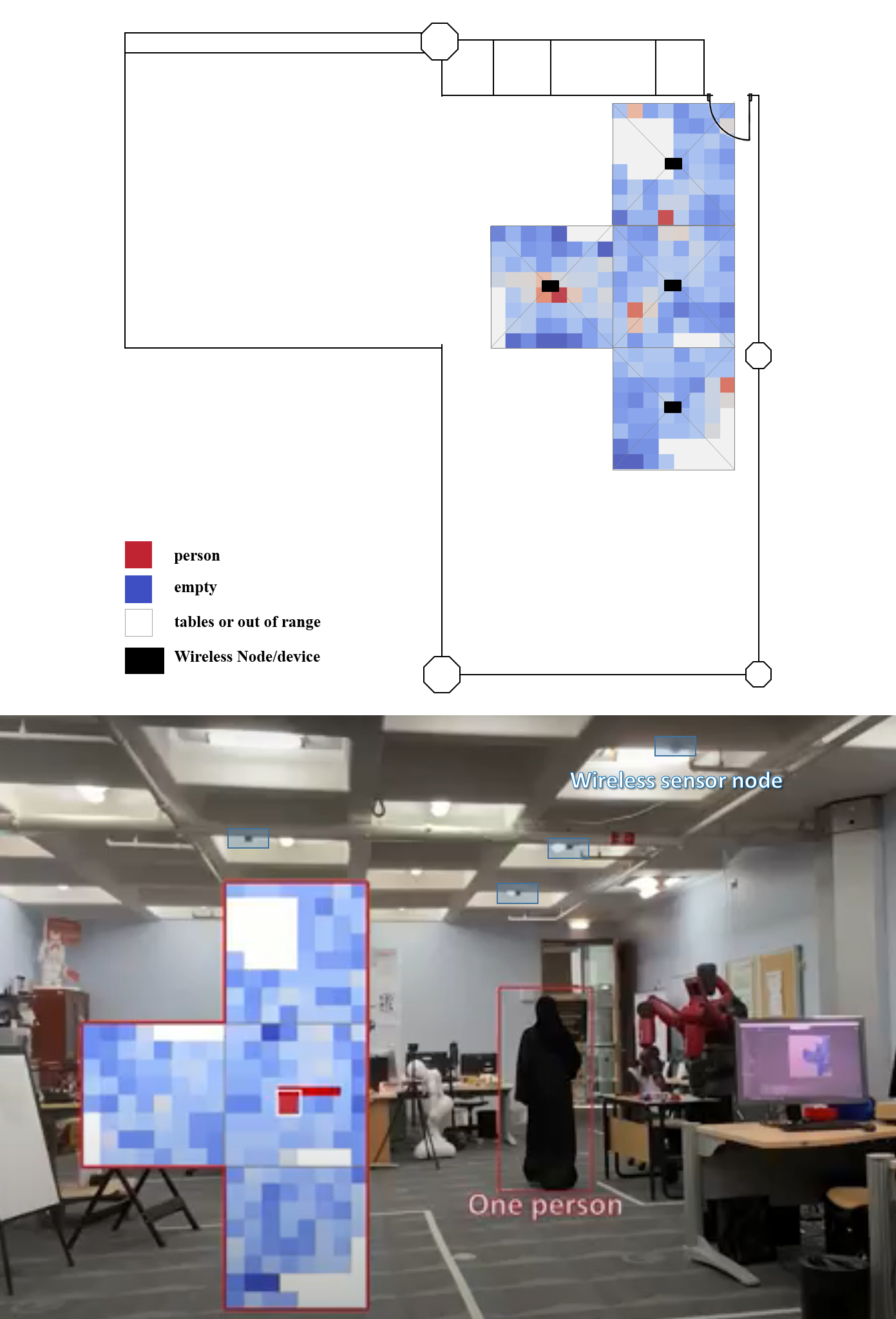


Fig. 11. AI & Robotics lab layout and a real-time person detection.

Test in Fig. 12 is done to verify model output for different cases, and all results obtained were correct.

## Testing system alarming accuracy

For each case in Table IV, we did 100 different scenarios and checked the output of the system alarming.

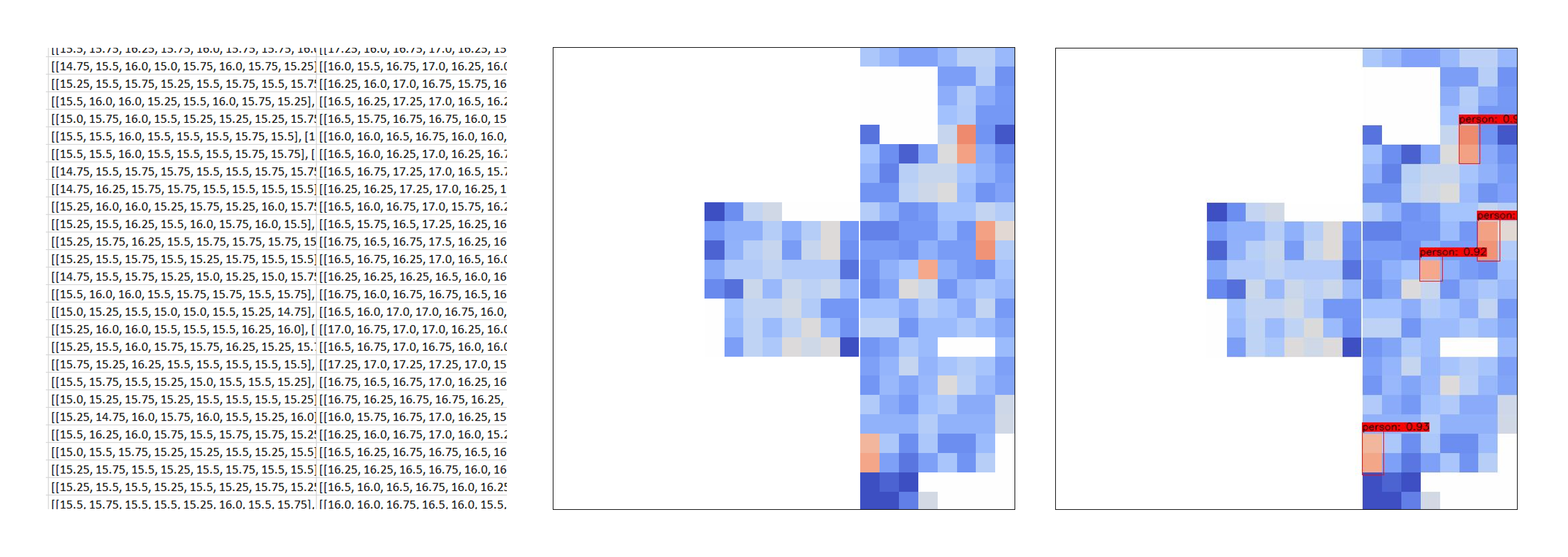


Fig. 12. YOLO output bounding boxes.

Below are descriptions of each parameter in the table:

Total number of iterations: 1 scenario is 1 iteration of the code.

* # of correct output: how many times in those 100 iterations, the alarm gave a correct output by initiating under the right node or not initiating in case of safe distances.
* # of false output: how many times of those 100 iterations, the alarm was falsely initiated, initiated under the wrong node, or not initiated when it was supposed to initiate.

## Testing system distance calculation accuracy

TABLE VI. Testing Time.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Part tested | Receiving message time (s) | Preprocessing time (s) | YOLOv4-tiny prediction time (s) | Distance calculation time (s) | Sending alarming message times (s) | Updating database time (s) | Total time (s) |
| Average time | 3.27879E-06 | 0.3122 | 0.0562 | 6.947E-05 | 9.333E-05 | 0.011581 | 0.378 |

Table V. shows testing system distance calculation accuracy by comparing it to the real distance. Real distance (cm): actual distance measured in real time in centimeter. Calculated distance (cm): distance calculated based on our code in centimeter. Difference(cm): Real distance (cm) - Calculated distance (cm). Percent error (%): | Real distance (cm) - Calculated distance (cm) | / Real distance (cm) \* 100.

## Testing time of each part in the computation process

Table VI. shows time for each process tested, and the average total response time is equal to 0.378. Below are descriptions of each process mentioned:

* Receiving message time (s): The average time it takes to the process received message in seconds.
* Preprocessing time (s): The average time it takes to process data, normalize, convert to heatmap image, remove tables/unneeded areas, crop, resize, and final format processing before inputting it to the model in seconds.
* YOLOv4-tiny prediction time (s): The average time it takes to predict the output bounding boxes in seconds.
* Distance calculation time (s): The average time it takes to calculate distance between every two detected human and classify it as a warning or not, in seconds.
* Sending alarming message time (s): The average time it takes to decide to which node alarm will be sent and sending it, in seconds.
* Updating database time (s): The average time it takes to insert queries of violation timestamp and its location, in seconds.
* Total time (s): The average total response time.

# Conclusions

The privacy preserved system for detecting and alarming any social distancing violations is proposed. Training results show that YOLOv4-tiny had the best results (mAP@0.5 of 95.4% and inference time of 5.16ms). Test results achieved 96.01% which verified our model accuracy. The response time is fast with an average of only 0.378 seconds and the system testing in real-time proved it was very responsive. Distance calculation accuracy has an average percent error of only 7.06% and an average difference from real measurements of 12.9 cm, this shows that our system succeeded in all aspects. The applications of this method could be in multiple areas other than social distancing, and there are many possible ways to improve it. It is the perfect solution to apply social distancing in sensitive places that consider privacy as the most important feature of any device.

References

1. NOVID. (2020). Stop COVID-19 with NOVID. https://www.novid.org/.
2. FastSensor Measuring indoor journeys for ROI and safety. (n.d). FastSensor ADAM active distance alert and monitoring. https://www.fastsensor.com/adam.
3. Social Distancing Solution: FastSensor ADAM Webinar (RCS). (2020, June 18). [Video]. YouTube. https://www.youtube.com/watch?v=qS6y68OEUkw.
4. 1point5. (n.d.). The app that helps you with social distancing. https://onepointfive.app/.
5. LTO NETWORK. (2019). COVID-19 Survivor Creates 1Point5 Social Distancing App. https://blog.ltonetwork.com/covid-19-1point5app-social-distancing-app/.
6. Sentech the sensors integrators. (2017). Keep your distance (in the workplace) with a Social Distancing Sensor. https://www.sentech.nl/en/social-distancing-sensor/.
7. RightCrowd eveyone on site is right. (2020). Social Distancing Monitoring for the Workplace. https://www.rightcrowd.com/social-distancing-monitoring/.
8. TSINGOAL. (2020). Social Distancing and contact tracing. https://www.uwb-social-distancing.com/?6gclid=EAIaIQobChMIgq2a-obG6gIVDPlRCh1EewDDEAAYASAAEgK5zvD\_BwE.
9. Zonesafe proximity warning & alert systems. (n.d.). ZoneSafe Person to Person Social Distancing Solution. https://www.zonesafe.net/proximity-warning-system-applications/person-to-person-alert-social-distancing/.
10. KONGNIZ SAFETY PLATAFORM. (n.d.). Creating safer, smarter spaces. https://www.kogniz.com/.
11. Synced AI TECHNOLOGY & INDUSTRY REVIEW. (2020, April 20). Landing AI ‘Social Distancing Detector’ Monitors Workplaces. https://syncedreview.com/2020/04/20/landing-ai-social-distancing-detector-monitors-workplaces/.
12. Porter, B. (2020, June 24). Amazon introduces “Distance Assistant.” About Amazon. https://www.aboutamazon.com/news/operations/amazon-introduces-distance-assistant.
13. AXXONSOFT. (n.d.). AXXONSOFT Meeting The challenge. https://stopcovid19.axxonsoft.com/?utm\_source=google&utm\_medium=cpc&utm\_campaign=stopcovid&utm\_term=distancing&gclid=Cj0KCQjw6ar4BRDnARIsAITGzlCYuss875\_Nq-Rgxa7Z\_PZahDOR1XxjUO54yOSr\_CQrIeWOqyMLQyoaAqjZEALw\_wcB
14. Actuate. (2020, October 17). COVID-19 AI Solutions. Actuate AI. https://actuate.ai/covid19
15. Actuate Analytics UI - Real-Time Security Camera AI. (2020, June 4). [Video]. YouTube. https://www.youtube.com/watch?v=o2D4qSQCFhM.
16. Redmon, J. (n.d.). YOLO: Real-Time Object Detection. https://pjreddie.com/darknet/yolo/
17. Miller, D. (2018, August 22). https://cdn-learn.adafruit.com/downloads/pdf/adafruit-amg8833-8x8-thermal-camera-sensor.pdf
18. Roboflow. (n.d.). Roboflow. https://roboflow.com/

1. \*Research is supported by CIT, UAEU.

   (*Corresponding author: Munkhjargal Gochoo*)

   A. F. is with the Department of Information Systems and Security, United Arab Emirates University, Alain, UAE (e-mail: 201704105@uaeu.ac.ae).

   H. F. Author is with the Department of Computer Science and Software Engineering, United Arab Emirates University, Alain, UAE (e-mail: 201702482@uaeu.ac.ae).

   J. S. is with the Department of Computer and Network Engineering, United Arab Emirates University, Alain, UAE (e-mail: 201703257@uaeu.ac.ae).

   S. S. is with the Department of Computer Science and Software Engineering, United Arab Emirates University, Alain, UAE (e-mail: 201702635@uaeu.ac.ae).

   M. G. is with the Department of Computer Science and Software Engineering, United Arab Emirates University, Alain, UAE (e-mail: mgochoo@uaeu.ac.ae [↑](#footnote-ref-1)