Jurnal Mandiri IT ISSN 2301-8984 (Print)
Vol. 12 No. 2, October (2023), pp. 72-81

Comparison of investor detection algorithm in internet of things based home security systems

Didit Karyadi
Department of Business Information Systems, Universitas Gunadarma, Indonesia

A R T I C L E I N F O

Article history:

Received Oct 9, 2023
Revised Oct 23, 2023
Accepted Oct 23, 2023

Keywords:

Home Security
Intruder Detection
Yolo

ABSTRACT

Empirical studies state that the environment is the main factor that influences crime patterns, so closed circuit television (CCTV) is an option to reduce the risk of crime. However, CCTV is less effective because it requires high bandwidth & storage and cannot provide notifications. Therefore, a technology called the internet of things (IoT) has emerged so that CCTV or webcams can work together with sensors to detect the presence of intruders and provide notifications. This research proposes a system that detects intruders and sends notifications to home owners without being tied to time and place. This system is usually referred to as a smart home security system. This research aims to compare intruder detection algorithms in IoT-based home security systems. This research method uses the internet of things (IoT) in smart homes or home security by comparing the accuracy and processing time of the HoG+SVM and Yolo V3 algorithms. The results of the system implementation show that the most accurate intruder detector is the Yolo V3 algorithm with an accuracy of 99% and a processing time of 14.852 seconds. This processing time can be accelerated by using a Graphics Processing Unit (GPU) with higher specifications.

Corresponding Author:

Didit Karyadi,
Department of Business Information Systems,
Universitas Gunadarma,
Jl. Margonda No.100, Pondok Cina, Beji District, Depok City, West Java 16431, Indonesia.
Email: diditkaryadi@gmail.com

1. INTRODUCTION

Surveillance cameras or what is often called CCTV (Closed Circuit Television) are often found in office buildings, banks, shopping centers and are even used by small to medium scale shops and in the homes of the upper middle class. This indirectly shows the increasing concern and awareness of the public regarding acts of theft (Rahman et al., 2018). The use of surveillance cameras so far has mostly been used as evidence of crimes or as a reference for law enforcers to identify perpetrators so they can dig up further information to catch the perpetrators, but using CCTV alone will be less effective in preventing crime because it will only monitor continuously, but does not provide a warning or initial reaction when capturing a suspicious object. Besides that, CCTV requires high bandwidth and storage because it will transmit large amounts of data (M. A. Hossain & Song, 2016).

Currently, several studies from academia, industry, and government have tried to connect everything in the world to the internet to provide a seamlessly integrated system to improve performance in information transmission, which is referred to as the Internet of Things (IoT). The development and growth of IoT is very fast and there are various types of IoT applications that are very helpful and contribute to making everyday human life better (Ray, 2016). IoT has been widely applied in social life applications such as smart grid, intelligent transportation, smart security, and smart home (Jing et al., 2014). New technologies continue to emerge in IoT that can increase the number of sensors and complexity of smart homes. One use of devices in IoT is passive infrared (PIR) sensors which can detect the displacement or movement of an object with simple information (Ahvar et al., 2016). Apart from that, the use of cameras to capture images or objects can be combined with PIR motion to detect unauthorized people in the house so that they can provide security notifications to the home owner. This can be a good combination in developing home security which can increase security and comfort and save more energy (Hossain Jewel et al., 2017).

Several studies regarding monitoring systems have been carried out. One of them is the human detection method with a video surveillance system. This system can detect humans using histogram of gradient (HoG) extraction features and classification using the support vector machine (SVM) algorithm. This method only gets an accuracy of approximately 89% (Seemanthini & Manjunath, 2018). Apart from that, the HoG and SVM methods have been implemented in a home security system which has human detection capabilities using a Raspberry Pi 3, webcam, PIR sensor, and buzzer. The results of these systems and methods can detect the presence of intruders with an average accuracy of 90% and an average processing time of around 2 seconds (Surantha & Wicaksono, 2019).

Apart from that, there is a method or algorithm called You Only Look Once (YOLO) which can usually be used to detect pedestrians and tracking systems. The YOLO algorithm has existed up to YOLO V3 which is more accurate and faster than the previous version (Guo et al., 2018). In some experiments, YOLO V3 was able to detect, track pedestrians, humans in a basketball game and/or various other objects successfully in every video frame. YOLO V3 is more accurate and faster than the previous version. YOLO V3 can detect various types of objects such as humans, cars and other objects with an accuracy level of greater than 90% (Lee & Seo, 2019); (Qu et al., 2019); (Yoon et al., 2019).

Based on the problems above, the author wants to develop a system that can provide notification to home owners anytime and anywhere more easily and comfortably via smartphone when they detect an intruder. Apart from that, the author wants to see the relationship between accuracy and the time needed to carry out the human detection process. Therefore, the aim of this research is to compare the accuracy and processing time of several algorithms that can be used to develop an effective system based on accuracy or processing time.

2. RESEARCH METHOD

2.1 *Research Phase*

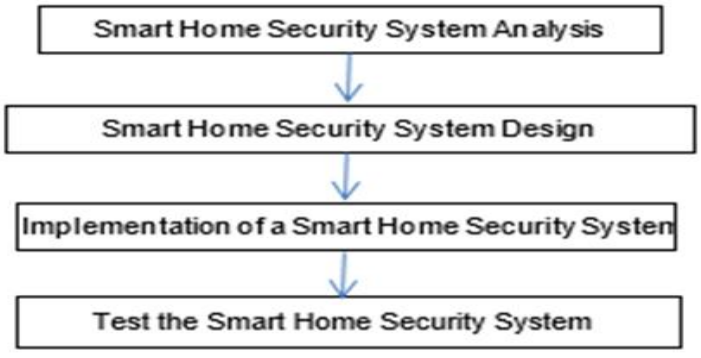


Figure 1. Research stage diagram

ISSN 2301-8984 (Print)

Figure 1 is the steps taken in this research, the first step the author carries out an analysis of the smart home system, after that the author creates a smart home system design, then after the system design is formed the author implements the design, the final step is to carry out a trial of the smart home system smart home system to find out whether the smart home system can work as expected.

2.2 *Smart Home Security System Design*

Smart Home Security system design consists of System Architecture, System Workflow, Hardware Architecture, Data Capture and Room Design

System Architecture

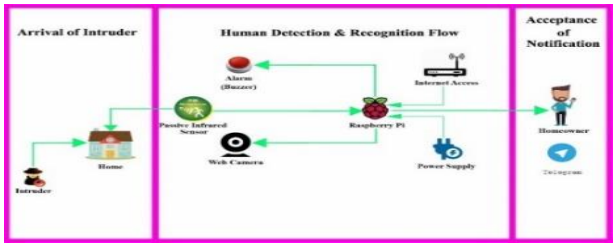


Figure 2. System Architecture

Figure 2 shows the system architecture design divided into three process areas, namely the arrival of an intruder, human detection & recognition flow, and receipt of notification. The author assumes that the intruder will enter from a point where the intruder is able to enter the house, such as doors, windows, attics.

System Workflow

In this research, the workflow of the system is described as follows:

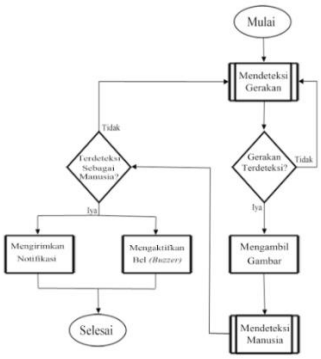


Figure 3. System workflow diagram

Figure 3 shows the system workflow starting when the homeowner starts activating the system by connecting the device to the power supply and through the application on the homeowner's smartphone. Then the Passive Infrared sensor will be active and will monitor the status of entry into the house whether movement is detected or not. If the Passive Infrared sensor detects movement, the web camera will capture an image of the object and process it further. However, if the sensor does not detect movement, the Passive Infrared sensor will continue to monitor the entry access status of the door. The workflow of the system if depicted in a use case diagram will be as in Figure 4.

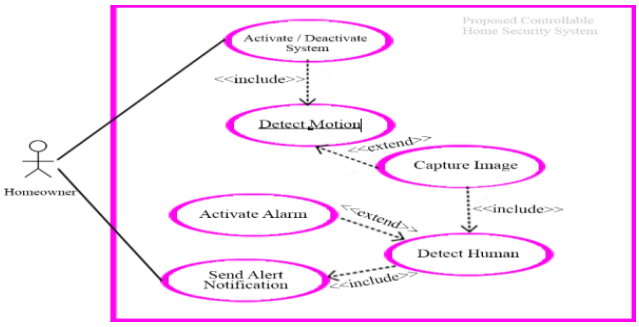


Figure 3. Use case diagrams

Data Retrieval & Room Design

In this research, the author conducted trials using the proposed smart home security system architecture. The data sources taken consist of the accuracy and time of the human detection process. Data collection was carried out 100 times on human objects and 100 times on non-human objects. Image data of human and non-human objects is taken from an active webcam after a trigger from the Passive Infra Red (PIR) sensor when the door opens. The process of collecting data in the form of images of human and human objects is shown in Figure 6 below.

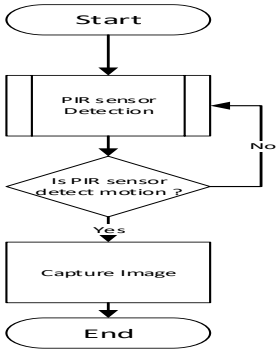


Figure 6. Data collection flow

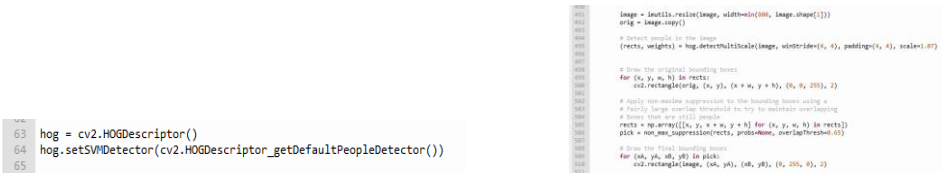
2.3 *System Implementation*

In this research, it was implemented using a Raspberry Pi 4 B connected to a Logitech C525 webcam, passive infrared sensor, and alarm or buzzer. The function of the Raspberry Pi 3 is to carry out all computing processes. To carry out this computing process, the Raspberry Pi 4 B has detailed specifications listed in Table 2.

Table 1. Raspberry Pi 4 B specifications

Operating System	:	Raspbian
Processor	:	Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
RAM	:	4 Gigabyte LPDDR4 RAM
GPUs	:	VideoCore VI 3D Graphics
WLAN	:	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN
Bluetooth	:	Bluetooth 5.0 with BLE
Ethernet	:	Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
GPIO	:	40 Pins
Display Output	:	HDMI ports support dual displays up to 4Kp60 resolution
USB	:	2x USB2 ports, 2x USB3 ports
Interfaces	:	CSI, DSI, 3.5mm audio jack
Storage	:	Micro SD (16GB)
Power Supplies	:	5V/2.5A DC

2.4 *Implementation of Dataset Use HoG + SVM*



```
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

```
image = imutils.resize(image, width=min(800, image.shape[1]))
orig = image.copy()

# Detect people in the image
(rects, weights) = hog.detectMultiScale(image, winStride=(4, 4), padding=(4, 4), scale=1.07)


# Draw the original bounding boxes
for (x, y, w, h) in rects:
    cv2.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)

# Apply non-maxima suppression to the bounding boxes using a
# fairly large overlap threshold to try to maintain overlapping
# boxes that are still people
rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)

# Draw the final bounding boxes
for (xA, yA, xB, yB) in pick:
    cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
```

Figure 8. Source code for using dataset in HoG + SVM (1) and source code for using dataset in HoG + SVM (2)

This research uses a training dataset provided by the OpenCV library. The way to use it is by calling a certain function. In Figure 8, it can be seen that the author uses the OpenCV library with the command "import cv2" to use the training dataset. Then "cv2.HOGDescriptor()" is a function from the OpenCV library to create a HoG descriptor and detector with several default parameters.

The function "imutils.resize(image, width=min(800, image.shape[1]))" is used to resize the image object accordingly, which in this case is 800 width. The hog.detectMultiScale(image, winStride=(4, 4), padding=(4, 4), scale=1.07) function is used to detect whether there are humans in the image object. Next, the function non_max_suppression(rects, probs=None, overlapThresh=0.65) is used to apply non-maxima suppression for bounding boxes that overlap too much than the threshold so as to keep the overlapping bounding boxes around objects that are detected as humans.

YOLOV3

```
labelsPath = os.path.join(DATA_YOLO , "coco.names")
LABELS = open(labelsPath).read().strip().split("\n")

# Initialize a list of colors to represent each possible class label
np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
    dtype="uint8")

# Define the paths to the YOLO weights and model configuration
weightsPath = os.path.join(DATA_YOLO, "yolov3.weights")
configPath = os.path.join(DATA_YOLO, "yolov3.cfg")

# Load our YOLO object detector trained on COCO dataset
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

imagePaths = list(paths.list_images(YOLO))
# Get the input image and grab its spatial dimensions
for imagePath in imagePaths:

    #Tolerance of object
    tolerance = 0.8
    threshold = 0.4
```

```
# apply non-maxima suppression to suppress weak, overlapping bounding
# boxes
idxs = cv2.dnn.NMSBoxes(boxes, confidences, tolerance, threshold)


# ensure at least one detection exists
if len(idxs) > 0:
    # loop over the indexes we are keeping

    for i in idxs.flatten():
        # extract the bounding box coordinates
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])

        # draw a bounding box rectangle and label on the image
        if classIDs[i] == 0:
            color = [int(c) for c in COLORS[classIDs[i]]]
            cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
            text = "{}: {:.4f}".format(LABELS[classIDs[i]], confidences[i])
            cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,0,255) , 2)
```

Figure 10. Source code for using dataset in YOLOV3 (1) and YOLOV3 (2)

In this research, the author used a dataset that had been trained by the inventor of the YOLOV3 algorithm. This dataset is available on the website https://pjreddie.com/darknet/yolo/ and the author uses a dataset called YOLOV3-608 on this website. Then the author uses the function os.path.join(DATA_YOLO, ”coco.names”) to determine the classification of the type of image object detected. The object type is contained in a file called "coco.names".

The function "cv2.dnn.NMSBoxes (boxes, confidences, tolerance, threshold)" is used to apply non-maxima suppression so that it can suppress overlapping bounding boxes. Then the function "if len(idxs) > 0:" is a repetition of the previously existing indexes to ensure whether at least one object is detected

3. RESULTS AND DISCUSSIONS

3.1 Test Results

HoG + SVM Human Detection Algorithm Results

The following images are some examples of detection results with the HoG + SVM algorithm on human objects or non-human objects:

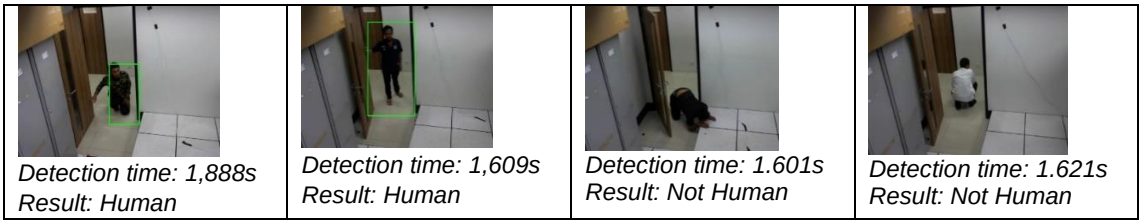


Figure 12. Human object image detection results: HoG + SVM

Figure 12 shows several examples of human object image detection results using the HoG + SVM algorithm. Based on this image, this algorithm is able to detect humans in various positions, namely standing, squatting, facing backwards and other poses. This algorithm can detect humans well if the detected object has a complete human body posture because this algorithm uses gradients or edge directions as the basis for detection so that it will produce the final detection result as "Human". These images are the results of processing the HoG + SVM algorithm which were detected as humans (true positives). The total true positives (TP) in this algorithm is 90 out of 100 trials.

Shows several examples of human object image detection results using the HoG + SVM algorithm. Based on this image, this algorithm produces incorrect detection results for human objects in prone or prone poses. Apart from that, if the detected object is not full or its posture is cut even slightly, it will produce the final detection result as "Not Human" because this algorithm uses gradients or edge directions as the basis for its detection. These images are the result of processing the HoG + SVM algorithm which is not detected as a human (false positives). The total false positives (FP) in this algorithm is 10 out of 100 trials.

Yolo Algorithm Human Detection Results

The following images are some examples of detection results using the Yolo V3 algorithm on human objects or non-human objects as follows:

Figure 16. Human object image detection results: Yolo V3 and image detection results of non-human objects: Yolo V3

Figure 16 shows several examples of image detection results for non-human objects using the Yolo V3 algorithm. Based on this image, this algorithm can detect various non-human objects at various distances very accurately, resulting in the final detection result as "Not Human". This happens because this algorithm performs object recognition and detection with a single neural network which predicts bounding boxes and class probabilities directly in one evaluation step. These images are the result of processing by the Yolo V3 algorithm which were detected as non-human (True Negatives). The total true negatives (TN) in this algorithm is 100 out of 100 trials. So there are no false negatives (FN) or 0 times the FN in this algorithm.

3.2 *Notification Delivery Results*

The process carried out after detection uses 2 algorithms, namely by sending notifications to the user's Telegram and email. Emails and telegrams can be opened from the homeowner's smartphone. The following is a detailed explanation regarding sending notifications to home owners via telegram and email.

ISSN 2301-8984 (Print)

Sending Notifications Via Telegram

Before sending notifications to Telegram, first register and officially create an API Bot on the @BotFather Telegram account. Figure 19 is the @BotFather account for creating Bot API.

Figure 18. @BotFather account on telegram, @BotFather telegram home page & features, and bot name & token on @BotFather telegram

Then type and send the word "/start" to display the menu and features of the Telegram Bot. The start page for starting the Bot and the menus and features on Telegram are in Figure 18 below.

Next, type and send the words "/newbot" to create a new Bot on Telegram. After that, type the desired Bot name and if the desired Bot name is already used, you can enter a new Bot name that is not yet used. In this system, the author uses the Bot name "PiSecurity".

If successful, you will get a notification that the Bot account has been successfully created on Telegram. Apart from that, there is a Bot link that has been created and also a token to be able to access the API. The token is used as an identity so that the system can send notifications to the home owner's telegram.

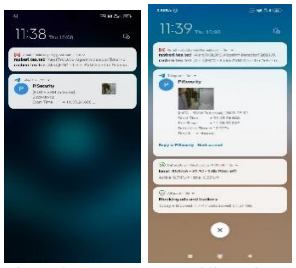


Figure 19. Example of pop up notification on a smartphone

Figure 19 is a pop up notification sent to telegram and email via the homeowner's smartphone. A pop up notification will be sent when the Passive Infrared sensor is triggered and the detection process for each algorithm has been completed. The content of the notification sent is in the form of a photo captured by the webcam and a log which displays the type of algorithm used to detect the object, date, start time of detection, end time of detection, time of detection process and detection results (results) whether the object detected was human or not human (not human).

Sending Notifications Via Email

Then, to send notifications via email, the only way to do this is to create or register an email account and password as usual. In this case, the author uses a Google email account or the domain @gmail.com. Then set "Less secure app access" to "active" (on) as in Figure 25 below.

Figure 20. Application access settings in gmail

3.3 *Testing & Evaluation Scenarios*

In testing the accuracy of human detection, 100 or more trials will be taken consisting of several conditions of human objects standing, facing the back, facing the side, or other conditions. Apart from that, the author will also try to detect non-human objects such as dogs, cats, chickens or other animals 100 times or more as a complement.

In this research, the evaluation carried out to measure the percentage of accuracy was by using the accuracy paradox formula. Measurements are carried out by detecting human objects that are detected as humans correctly and incorrectly. Apart from that, measurements are also carried out by detecting non-human objects which are detected as non-human correctly and incorrectly. Therefore, the accuracy percentage formula will be obtained as follows:

$$A = \text{x } 100\% \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Explanation of the symbols in the formula, namely:

Accuracy(A) = percentage accuracy.
True Positives(TP) = number of correct detections of the human object being tested.
False Positives(FP) = number of false detections of the human object being tested.
True Negatives(TN) = number of correct detections of non-human objects tested.
False Negatives(FN) = number of false detections of non-human objects tested.

Then an evaluation was also carried out to measure the processing time of each human detection algorithm. Process time measurements are taken from the start of detection, namely after the object is captured by the webcam, until the time the object detection is completed, namely when the object is declared human or non-human. Therefore, the formula for calculating the human detection process time will be obtained as follows:

$$DT = ET - ST$$

Detection Time(DT) = processing time required for one detection of an image object.
Start Time(ST) = start time for object detection, namely after the image object is captured by the webcam.
End Time(ET) = completion time for object detection, namely after the image object has been concluded as human or non-human.

3.4 *Comparison Results of Accuracy and Detection Process Time*

Based on the results of algorithm testing that has been carried out, there are two main data of concern. In this research, the main data of concern is accuracy data (A) and processing time (DT) of detection results from the two algorithms that have been tested. From 100 tests, for each human and non-human object and for both algorithms the following results were obtained:

Table 2. Confusion matrix

		Prediction: HoG + SVM	
Total Sample = 200		Negative	Positive
Actual	Negative	TN = 93	FN = 7
	Positive	TP = 90	FP = 10
Accuracy	= (TP + TN) / (TP+FP+FN+TN) = (90+93) / (90+10+7+93) = = 92 %		
		Prediction: Yolo V3	
Total Sample = 200		Negative	Positive
Actual	Negative	TN = 100	FN = 0
	Positive	TP = 97	FP =3
Accuracy	= (TP + TN) / (TP+FP+FN+TN) = (97 + 100) / (97+3+0+100) = 99 %		

Based on Table 2, it can be seen that the HoG + SVM algorithm produces an average accuracy of 92% and a detection process time of around 1.622 seconds. Then the Yolo V3 algorithm produces an average accuracy of 99% and a detection processing time of around 14.852 seconds. There are factors that influence the level of accuracy in each algorithm, resulting in different detections in the same image, these factors are:

*Datasets*The algorithm used in each algorithm is different because it is provided in the OpenCV library and other sources in the literature.

The YOLOV3 algorithm uses CNNs, while HoG + SVM does not use it so the accuracy of YOLOV3 is better because it consumes a higher GPU (Graphics Processing Unit) so the detection process is slower if run on devices that have low GPU specifications.

Table 3. Comparison of raspberry Pi 4 B specifications with laptops

No	*Items*	Raspberry Pi 4 B	Asus VivoBook X407UF
1	OS	Raspbian	Windows 10 Pro (64-bit)
2	*Processor*	*Quad core 64-bit ARM-Cortex A72 running at 1.5GHz*	Intel® Core™ i3-7020U CPU @ 2.30GHz
3	RAM	4 Gigabyte LPDDR4 RAM	8GB DDR4
4	GPUs	*VideoCore VI 3D Graphics*	Nvidia Geforce mx130
5	*Storage*	16 GB Micro SD	1TB HDD

However, tests carried out on a laptop which has different specifications from the Raspberry Pi 4 B as shown in Table 4.3, show the results in Table 4.4 that the fastest detection process time, namely the HoG + SVM and Yolo V3 algorithms, is slightly different, namely 1.401 seconds for HoG + SVM. and 1.405 seconds for Yolo V3. An interesting thing happened in the detection using Yolo V3 using a laptop. There was a very significant difference in processing time from the previous average detection process time of around 14.852 seconds using the Raspberry Pi 4 B to an average of around 1.405 seconds.

Table 4. Comparison of accuracy and detection process time

No	Works	Methods	Accuracy	Detection Times(in Raspberry Pi)	Detection Times(in Laptop)	TotalSample
1	Surantha, N., & Wicaksono, W. R	HoG + SVM	92%	1,952s	1,401s	200
2	*This works*	Yolo V3	99%	14,852s	1,405s	200

This can happen because Yolo V3 is very complex in detecting objects, resulting in high object detection accuracy, namely 99% on average. Under these conditions, the Yolo V3 algorithm requires better specifications, especially on the higher graphics processing unit (GPU) if you want to get a faster detection processing time. This is confirmed by the results of research from (Redmon & Farhadi, 2018)that the dataset that has been trained using the Darknet-53 model will achieve the largest floating point measurements per second. This means the network structure is better using GPU making it more efficient to evaluate and thus producing faster results.

Additionally, in research (S. Hossain & Lee, 2019) stated that object detection algorithms such as Yolo, SSD, and R-CNN will produce better and more efficient performance by using GPUs that have better performance as well. Then the experiments carried out by the author can be seen by comparing the specifications in Table 4.3. On the Raspberry Pi 4 B it only uses the standard GPU ieVideoCore VI 3D Graphics causes the detection process time using the Yolo V3 algorithm to be slow, namely an average of 14.852 seconds. However, the detection process time on the laptop has increased to be faster, namely an average of 1.405 seconds because it uses an Nvidia Geforce mx130 GPU.

4. CONCLUSION

In this research there are several steps carried out by the author, the first step is the author to analyze the smart home system, after that the author creates a smart home system design, then after the system design is formed the author implements the design, the last step is to test the smart home system. home.

Smart home security systems can be used to detect the presence of intruders and send alerts or notifications via telegram and email. The system proposed by the author is simple, economical and effective because it only consists of a Raspberry Pi 4 B, webcam, passive infrared sensor and buzzer. The Yolo and HoG algorithms respectively used have an average accuracy of

99% and 92%, as well as an average processing time of 14,852 seconds and 2 seconds. The Yolo V3 algorithm needs to be further modified and applied to devices that have a better Graphic Processing Unit to get a faster and more optimal detection processing time.

REFERENCES

Ahvar, E., Lee, G. M., Han, S. N., Crespi, N., & Khan, I. (2016). Sensor network-based and user-friendly user location discovery for future smart homes. *Sensors (Switzerland)*, *16*(7), 1–18. https://doi.org/10.3390/s16070969

Guo, H., Ren, J., Zhang, D., Zhang, Y., & Hu, J. (2018). A scalable and manageable IoT architecture based on transparent computing. *Journal of Parallel and Distributed Computing*, *118*, 5–13. https://doi.org/10.1016/j.jpdc.2017.07.003

Hossain Jewel, Md. K., Mostakim, Md. N., Rahman, M. K., Ali, Md. S., Dobir Hossain, S., Hossain, Md. K., & Ghosh, H. K. (2017). Design and Development of a Versatile and Intelligent Home Security System. *International Journal of Engineering and Manufacturing*, *7*(4), 60–72. https://doi.org/10.5815/ijem.2017.04.06

Hossain, M. A., & Song, B. (2016). Efficient Resource Management for Cloud-enabled Video Surveillance over Next Generation Network. *Mobile Networks and Applications*, 806–807. https://doi.org/10.1007/s11036-016-0699-3

Hossain, S., & Lee, D. J. (2019). Deep Learning-Based Real-Time Multiple-Object Detection and Tracking from Aerial Imagery via a Flying Robot with GPU-Based Embedded Devices. *Sensors*, *19*(15), 1–25. https://doi.org/10.3390/s19153371

Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: Perspectives and challenges. *Wireless Networks*, *20*(8), 2481–2501. https://doi.org/10.1007/s11276-014-0761-7

Lee, J. H., & Seo, C. J. (2019). Deep Learning based Pedestrian Detection and Tracking System using Unmanned Aerial Vehicle and Prediction Method. *International Journal of Innovative Technology and Exploring Engineering*, *8*(8), 794–799.

Qu, H., Yuan, T., Sheng, Z., & Zhang, Y. (2019). A Pedestrian Detection Method Based on YOLOv3 Model and Image Enhanced by Retinex. *Proceedings - 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2018*, *2016*, 1–5. https://doi.org/10.1109/CISP-BMEI.2018.8633119

Rahman, Md. W., Harun-Ar-Rashid, M., Islam, R., & Rahman, Dr. M. M. (2018). Embodiment of IOT based Smart Home Security. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, *6*(September), 4–14. https://doi.org/10.22214/ijraset.com/fileserve.php

Ray, P. P. (2016). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, *30*(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv*, 1–8.

Seemanthini, K., & Manjunath, S. S. (2018). Human Detection and Tracking using HOG for Action Recognition. *Procedia Computer Science*, *132*(Iccids), 1317–1326. https://doi.org/10.1016/j.procs.2018.05.048

Surantha, N., & Wicaksono, W. R. (2019). An IoT based house intruder detection and alert system using histogram of oriented gradients. *Journal of Computer Science*, *15*(8), 1108–1122. https://doi.org/10.3844/jcssp.2019.1108.1122

Yoon, Y., Hwang, H., Choi, Y., Joo, M., Oh, H., Park, I., Lee, K. H., & Hwang, J. H. (2019). Analyzing Basketball Movements and Pass Relationships Using Realtime Object Tracking Techniques Based on Deep Learning. *IEEE Access*, *7*, 56564–56576. https://doi.org/10.1109/ACCESS.2019.2913953