Home Automation System using Camera Control and Raspberry Pi

Md Nahid Hasan Shuvo
Computer Science
University of Nebraska Omaha
Nebraska, USA
nshuvo@unomaha.edu

Hasniuj Zahan
Computer Science
University of Nebraska Omaha
Nebraska, USA
hzahan@unomaha.edu

Arjon Das
Computer Science
University of Neberaska Omaha
Nebraska, USA
arjondas@unomaha.edu

Abstract—**With the advancement of IoT devices and technology, smart devices can communicate with each other which makes home automation systems more engaging. App-based and voice-based home automation is quite common these days though they have their own limitation (i.e.: not all smart devices are compatible with one app, command interpretation problems because of the accent). Acknowledging this problem, we have proposed a computer vision-based home automation system using Raspberry pi 4 and a camera module. In our home automation system, we demonstrated that gesture recognition can be a viable way to control the devices to avoid the wrong interpretation of commands. For a better understanding of the feasibility and efficiency of our system, we used three devices with multiple functionalities. Our evaluation shows that the gesture recognition is 90 percent which ensures the accuracy of command interpretation.**

Index Terms—**IoT, Deep Learning, Computer Vision, Home Automation, Raspberry Pi**

I. Introduction

One of the most rapidly expanding areas of the Internet of Things is home automation, which has the potential to fundamentally alter the way people live. In a home automation system, several devices are linked together to control various aspects of the residence [1]. Whether it's a small business or a major corporation, all are taking advantage of automation and eliminating human interaction to achieve greater operational efficiency and productivity.

The scope of a home automation system's potential applications is virtually limitless. Luxurious and technologically advanced systems appeal to affluent consumers, while others are geared toward the elderly and handicapped market. With these systems, residents can enjoy a safe, healthy, and comfortable way of life. [2]. Undoubtedly, the popularity of IoT-based home automation has exploded in recent years, owing to its simplicity and low cost. Henceforth, as the Internet expands, these network-enabled systems are extensively adopted for remote control and monitoring [3]. Because of the portability and wide variety of functionalities, mobile devices and websites are suitable for offering a better user experience in a home automation system. In fact, even when we are not at home, we can remotely monitor the condition of our home's automation system and manage a variety of smart equipment via the internet [2].

There are several existing Virtual Voice Assistants-based solutions for home automation with IoT devices. They often prove to be very useful and accurate at the specific sets of tasks they are tuned to perform. But using voice commands (or, on a broader level, NLP) for controlling connected devices can sometimes feel like gimmicks due to requiring unnatural interactions to initiate actions, e.g., uttering "Hey Siri" or "Alexa" for activating virtual assistant. On top of that, in the current state of virtual assistant solutions, the voice commands can often be misinterpreted due to accented language inputs or noisy backgrounds. Consequently, in this work, we are also exploring if computer vision can serve as an intuitive mechanism for controlling IoT devices with our home automation system.

Computer vision can be applied in a variety of novel ways in IoT. Computer vision deals with systems that require video or image analysis. As we all have certain distinct features, such as a facial system or a fingerprint. We have been employing these qualities for security purposes for a long time, but with the advancement of computing systems, computers can now utilize these methods to authenticate any person for home or company security without manually checking it. In a facial recognition system, a person's face is compared to a predefined database, and if the matching accuracy surpasses a certain level, the system grants that person required access [4]. Following an accurate face detection, it is necessary to incorporate liveliness features in the system, so that an unauthorized person cannot enter with the photo of an authorized person [5]. Along with that, after a camera and computer vision recognize something, it is important to take action so the system can communicate with other devices in the home [6]. Like we are being able to control the door with feedback that we receive from the installed camera Or we are being able to control the light, fan, or other smart devices. This improves the security and functionality of our house without controlling any switch manually.

A high-performance processor is required that can load and operate a deep learning model and control other devices through a single board in order to construct a camera control system based on deep learning. Raspberry Pi is an excellent development board because Linux operating systems could be used to run the necessary tools [7].In order to send and receive data over the internet, a user-friendly platform is required. Because of this, the raspberry Pi is an ideal platform for this project. Furthermore, since raspberry pi has the power of a full-sized computer, it is possible to use different programming languages here, such as python, which has a large library for building models using deep learning and computer vision, and thus it is easier to run and build a model to perform complex calculations. Also, it is difficult to operate, manage, or monitor home appliances or gadgets, intruder detection, and speech recognition at the same time using a micro-controller, i.e., it is very difficult to accomplish many functions at the same time [8]. The researchers can do this with the help of a computer but using a computer for this purpose is costly and needs a lot of electricity. The Raspberry Pi helps to solve these issues. Users can use the Raspberry Pi to operate home appliances utilizing sensors, speech recognition, and a video interface for intruder detection. Raspberry pi can also be used as a server to establish communication between user and appliances, and send notification in case of emergency [9].

A camera-based home automation system for controlling smart devices has been proposed in this study. This study makes the following significant contributions:

- A Smart IoT-based home automation system.
- Implementation of the whole system in the Raspberry Pi micro-computer.
- Integration of the computer vision to pose as viable alternative to Speech Recognition (NLP) based IoT device control module.

This paper is therefore organized as follows, the existing related works on computer vision, and home automation system are elaborated in section II. The methodology of developing camera control home automation system are demonstrated in section III. Experimental method and evaluation of the proposed home automation system are explained in section IV. In section V, we have concluded our project with the future work.

II. Background and Related Works

A. Related Works

The rapid adoption of IoT devices in daily consumers has accelerated the progress of Home automation system research. Additionally, emerging methods like Machine Learning, Natural Language Processing, and Computer Vision are opening ways for new mediums of interfacing with such devices. In this section, we have briefly reviewed some of the related works on IoT devices, Home Automation, Machine Learning, and how they intersect. The primary architecture of IoT systems involves the objects, the way they connect with the system, and the exposed attributes of the object. Every IoT system tries to the extent of the scope of the internet based on the exposed feature sets of these peripheral devices. To expand this generic organization, [10] proposed a unified IoT framework consisting of a seven-layer structure and concisely redefining IoT nodes model, virtual things, services established by things, and the hierarchical model of services, with their corresponding properties. The IoT based home automation systems can also be a rich source of data for analyzing behavior activity and consumer interactions. To better facilitate data collection, [11] demonstrates home automation, utilizing EmonCMS platform along with NodeMCU micro-controller.

IoT systems also need to concentrate on the seamless interaction between users. [12] introduces both mobile app interfacing along with chat-based recognition system, namely a dual-mode communication with the IoT hub for granular control of the connected devices. The control hub is based on a single chip general-purpose micro-controller, which has proven adequate for performing most of the rudimentary remote controlling tasks. The Amazon Alexa [13] has further bridged the gap for IoT device interaction via integrating voice commands for triggering control events. The AWS IoT combined with Amazon Cloud Services and Alexa Voice Services enable a more advanced controlling scheme for such IoT devices. Although NLP-based speech recognition by virtual assistants has unfolded an excellent way of human-computer interaction, it can often feel like a party trick rather than an actual feature. For instance, predefined activation sequences, poor recognition with accented language [14] and background noise [15], etc., make virtual assistants counter-intuitive and frustrating in many daily practical scenarios. As a result, in this paper, our IoT home automation system development is accompanied by an exploration of computer vision techniques for fine-grained controlling of the IoT devices. One of the core rationales is natural language commands present a lot of uniqueness and nuances which might be difficult for the AI systems to filter out, whereas computer vision-based interactions, e.g., gesture recognition [16], pose estimations [17], may constitute as a more general form of interaction among a large group of people.

III. Methodologies

A. An Overview of Our System

In this section, The core idea and the techniques to develop the home automation system using computer vision have been explained here. The system consists of two main modules, the vision unit, and the control unit. The vision unit takes a continuous video feed through a camera to recognize human hand motion gestures. On the other hand, the control unit takes sequential gesture inputs (as scalar values) to control multiple peripheral IoT devices. We have implemented both modules in Raspberry Pi 4 hardware due to its superb interfacing flexibility. The model of our camera controlled home automation system is illustrated in figure-1. Our whole system can be divided into three section that are Hardware Design, Gesture Recognition, Device Control. The description of these sections has been provided in detail.

B. Hardware Design

Hardware and software components are both included in the proposed Home Automation system. In this subsection we will discuss about design and setup of the hardware involved in our developed system.

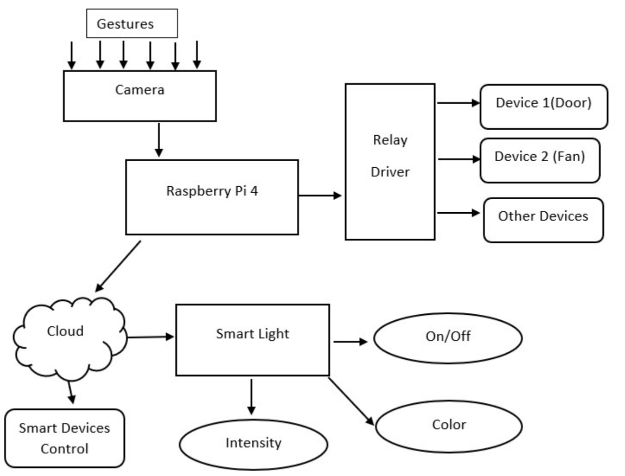


Fig. 1: Overview of the Computer Vision-based Home Automation System

1) Raspberry Pi 4: Electrical components along with the Raspberry Pi itself make up the hardware of the system. In addition, the system's software provides all of the algorithms and programs necessary to keep it running. Basically, Raspberry Pi 4 [18] single board microcomputer which have the 1.5 GHz 64-bit quad core ARM Cortex-A72 processor for giving higher performance than the previous version of this microcomputer. To make a computer work, it requires to have a microprocessor and I/O ports and memory. To facilitate higher frame-rate, we've configured the RPi to run at 2.1 GHz clock. For better power management due to overclocking, we've also disabled some hardware components like the HDMI controller and the Bluetooth chip situated on the RPi board.

2) Camera Module: Video and high-resolution photos have been captured with the Raspberry Pi's native Pi camera module with dedicated camera input port, namely the CSI Camera Connector. The Raspbian OS has pre-installed drivers to operate this camera module. We have implemented tools with OpenCV to capture the live video feed using Python and this Pi-specific module, allowing us to feed input into the gesture recognition alorithm.

3) Peripheral Device (Relay Switch): Several pieces of hardware have been attached to the Raspberry Pi in order to complete this project. The Relay module, a type of switch, was linked to the GPIO pin directly. When the RPi4 received a specific value from the camera, it triggered the GPIO pin to switch on or off the relay module, then the relay module connected with a door lock successfully unlocked the door. In fact, this relay module is also able to use as a switch for our electric fan. Here, we have connected a relay module which represent the door lock, as door lock switch can directly powered from the current and relay basically acts as a switch, thus we control the relay from the GPIO pin 17, and the it also connected with the $5V$ and Ground pin of the Raspberry Pi. The GPIO pin 17 have been set to the Output mode so that it can be used for triggering. As a result when we the Raspberry Pi send High signal to the GPIO pin it make the switch on, on the other hand by sending the low signal to the pin it make the relay off. This triggering to the GPIO pin have been done by the certain hand gestures detection from the camera.

4) Peripheral Device (Smart Light): We have used multiple Govee $H6008$ smart RGB bulb which is both Wi-Fi and Bluetooth compatible device. We have connected the smart light with the Raspberry Pi via wireless communication protocol. This smart bulb supports 2.4 GHz and 802.11 b/g/n bands. We have selected this smart bulb as this light has multiple functionality like turn on/off, changing of color intensity, and change to different colors etc.

Not only these, it is possible to connect and control as many devices as we want in our automated home. In the case of controlling the smart device, a developer account is needed to be created in the cloud of that brand which will be used to get the parameters of the device for making the API call. For changing the smart device modes, REST API's "POST"/"PUT" methods can be used. Different brands have different platforms and requirements to make the API call to control those devices. We have controlled the bulb via Wi-Fi and based on the particular input from the camera different action have been triggered.

C. Gesture Recognition

The gesture recognizer has two sub-modules: the hand landmark detector and the core gesture recognizer. The former detects hands and maps each finger within the field of view into 3-Dimensional coordinates. And the latter takes these coordinates and recognizes predefined hand gestures. For our implementation, we are using the MediaPipe [19] framework along with the OpenCV [20] library to handle the landmark detection task. The OpenCV library is specially built for real-time computer vision purposes, and we have utilized that to read camera frame captures efficiently. The MediaPipe provides cross-platform computer-vision tools to build applied Machine Learning pipelines. For our specific landmark detection purpose, the framework utilizes the TensorFlow-Lite [21] backend to facilitate the detection.

D. Gesture Design

Due to resource constraints, it is difficult to adopt any ML-based landmark-to-gesture recognizer. It is crucial to design a gesture recognition system that can accurately detect individual gestures while effortlessly serve our purpose. Consequently, we are implementing a counting-based gesture recognizer, based upon heuristics and conditions, with some intermediate motion sensing which don't require too many computation resources. To be particular, we are featuring two types of gestures. Firstly, one motion based gesture, namely "the hand wave motion", for recognizing the activation sequence, that prompts the Home Automation system to read hand signs for controlling peripheral devices. Secondly, five static hand signal based gesture, showing number of fingers on hand to execute a specific control operation. The Table I shows the device actions depending on various gestures. Fig **??**

TABLE I: Device Action based on Gesture Pattern

No	Gesture Pattern	Device Action
1	Hand Wave	Activate Gesture Input
2	Show 1 Finger	Turn On/Off Light-1
3	Show 2 Fingers	Turn On/Off Relay
4	Show 3 Fingers	Turn On/Off Light-2
5	Show 4 Fingers	No action assigned
6	Show 5 Fingers	No action assigned

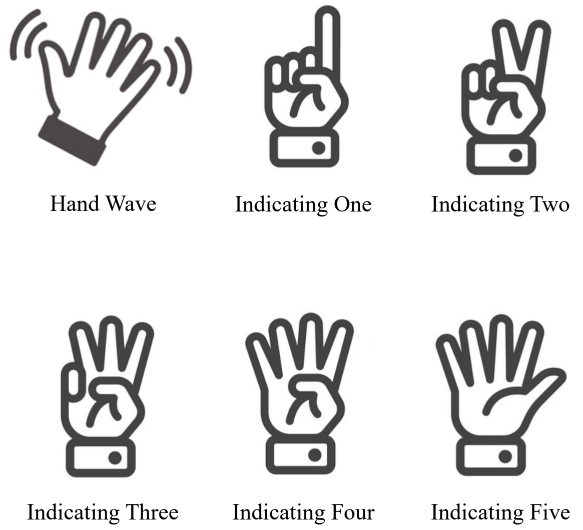


Fig. 2: Visualization of all the gestures recognized by the proposed Home Automation System

E. Device Control

Another part of home automation is controlling the smart devices without physical interactions of human. For controlling the smart devices, there are different types of apps for the different devices based on the brands, and managing them to control all the devices of a home is quite a handful. That's why we have made the REST API call to develop our controller for simplifying the home automation system for users. To control the devices (such as smart lighting, smart plug), we have used the camera as a controller, the Raspberry Pi has been used as a processing unit, and the REST API method have used to communicate commands between the smart light and the camera. Though. A REST (Representational State Transfer) API is a web application programming interface (API) that adheres to the requirements of the REST architectural style and allows users to communicate with RESTful web services. When a client request is made, a REST API sends a representation of the status of the resources to the requester or endpoint. JSON has been used to transmit this information or representation because it is language-independent and can be used by humans and machines. Headers and arguments have also been provided while making the API call, since they contain the key identification information such as the request's metadata, authorization, unified resource identifier (URI), caching, cookies, and other information. The "GET" method method is being used to get the device status and present brightness value of the bulb. The "PUT" method is being used to modify the action of smart light. We have used the REST API's "PUT" method with a varied request body to execute multiple illumination modes based on different movements.

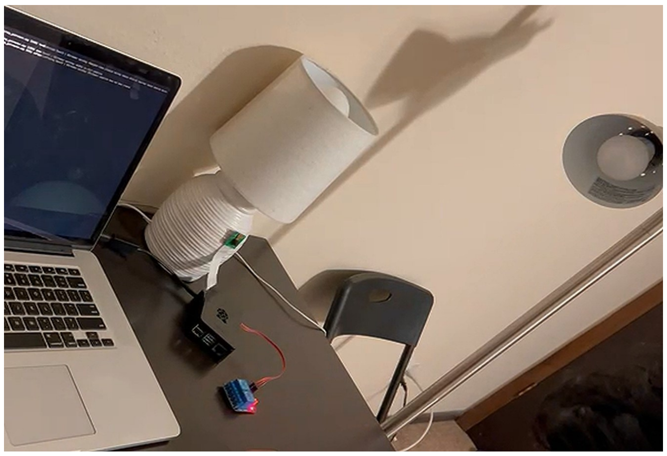


Fig. 3: Experimental Setup of our home automation system

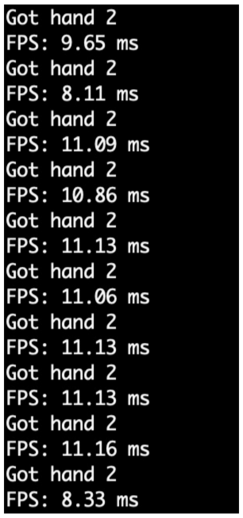
With a small form, computation capability is one of the major challenges in this project. To be able to pull off a computer vision-based gesture recognition system along with the control unit to automate IoT devices is very difficult in such a system. And our initial benchmark tells the same story. Our landmark detection implementation running natively on the RPi-4 achieves a frame rate of 7 FPS while detecting two hands. We established this value as our baseline metric and started building upon it. For a smoother operation and better frame-rate, we are adopting two strategies.

Firstly, we are utilizing overclocking [22] capability of the device to increase the detection frame-rate. But overclocking the CPU comes with some risks of failure and freezing. To mitigate this issue, we are implementing our detection frame-rate manager. The device manufacturer advises keeping the CPU temperature under 70 degrees Celsius. Since RPi-4 CPU clock changes require system boot, we adopted a static high clock frequency while dynamically managing the detection frame-rate per the CPU temperature to keep the hardware safe. For power management, we are also disabling some of the unnecessary features of the device to get a consistent high CPU voltage.

IV. Experiment and Evaluation

A. Experimental Settings

In this section we have discussed our experimental setup, result of our experimental method, and evaluation of our method. For this experiment, we used Raspberry Pi 4 model B as our central processing Unit. The feed from the Pi Camera is processed into the RPi4. The camera feed is processed by the MediaPipe detector to generate hand-landmarks. Then accord-



```
Got hand 2
FPS: 9.65 ms
Got hand 2
FPS: 8.11 ms
Got hand 2
FPS: 11.09 ms
Got hand 2
FPS: 10.86 ms
Got hand 2
FPS: 11.13 ms
Got hand 2
FPS: 11.06 ms
Got hand 2
FPS: 11.13 ms
Got hand 2
FPS: 11.13 ms
Got hand 2
FPS: 11.16 ms
Got hand 2
FPS: 8.33 ms
```

Fig. 4: MediaPipe hand detection and land-marking Frame-rate in RPi terminal

TABLE II: Raspberry Pi 4 model B specification

Parameters	Specifications
Processor	Quad core Cortex-A72 64-bit SoC @ 1.5GHz
Memory	4GB LPDDR4 SDRAM
WiFi	2.4 GHz AND 5 GHz
Ethernet	1 Gbps
Bluetooth	5.0
Access	Extended 40-pin GPIO header
Camera	MIPI CSI camera port
SD Card Support	Micro SD format for loading OS and data storage
Multimedia	OpenGL ES 1.1,2.0,3.0 graphics
Power Requirement	5V/3A
Operating System	Raspbian: A Debian-based Linux OS

ing to the detected landmarks our custom gesture recognizer algorithm identifies the specific gestures/signals. Based on these gestures the RPi makes a decision and send trigger to the smart device and door which are connected with the RPi via both wireless and wired connection. The specification of this device is shown in Table 1. For capturing video stream we are using OpenCV library. To keep a low computation complexity, we are capturing low resolution pictures (320×240) per frame.

Table II shows the base clock speed of the RPi Processor. Since we are doing both hand detection and hand landmark detection, which are very compute intensive work, we are clocking the processor at 2.1GHz for getting higher frame-rate on inference task. The configuration for overclocking the RPi processor is situated at boot/config.txt directory. Here simply changing the arm_freq to 2100 and $over\_voltage$ to 6 allows the RPi to overclock at a stable rate. Fig 4 shows the frame-rates for hand and finger landmark detection task on the RPi device after overclocking. Since this is a home automation system, our system leverages the idea of utilizing compute power of surrounding devices (e.g. Laptop, Smartphone). Obviously, we designed our system to work standalone without any support of any edge device with some limited capabilities.

As a smart home device, we are using smart RGB bulb

TABLE III: Success-rate on different hand gesture recognition

	# Trials	# Success
Hand Wave	100	87
Gestures 1	20	17
Gestures 2	20	19
Gestures 3	20	19
Gestures 4	20	18
Gestures 5	20	18

which can be controlled using both blue-tooth and WiFi. The model of the bulb is Govee $H6008$ and to access this device's API, a developer account has been created in the cloud which allowed us to get the state of the devices or execute the command for different action of the smart device. To control the action of the bulb (e.g.: turn on/off, change the brightness, change the color), REST API's "PUT" method is used that has been written in python using JSON library. Different functions have been created to describe the different actions of the smart bulb that are integrated with the hand gesture based action. For example, if the controller detects finger count 1, the light-1 is being toggled, for finger count 3, the light-2 is being toggled and if the controller detects the finger count 4, it changes the brightness of the light-1. In the JSON body, device id(unique id) and other necessary parameters have been used. Device id and other necessary information can be gathered using another API call using "GET" method. To make sure the authenticated control of the device, api-key is being used in the headers. Though it is a quite secured approach of controlling a device, the main disadvantage is latency. As the request/response for changing device action comes via cloud, latency can be observed. To evaluate our method we have designed three experiments. They are as follows:

1) Monitoring the FPS of the Raspberry Pi 4 with the MediaPipe hand detection and finger-landmark detection inference task.
2) Rate of successful activation sequence detection.
3) Rate of successful individual static gesture detection.
4) Rate of successful interaction between RPi Control Unit and IoT devices.

The former is a pre-implementation experimentation, designed for evaluating the feasibility of running real-time gesture detection on the RPi hardware, the latter three are post-implementation experiments. For monitoring inference FPS on the RPi4, we only ran the MediaPipe's hand detection algorithms, coupled with landmark detection. Please note that, this experimentation does not include our custom gesture recognition algorithm. For calculating activation sequence success rate, we counted the number of successful attempts opposed to number of trials. Similarly, for calculating individual static gesture detection success rate we logged individual gestures hit rate opposed to their corresponding trials. These two experimentation help us evaluate the efficacy of our custom gesture recognition algorithm. And last but not least, we derived the interaction success rate experiment which denotes the successful interfacing between RPi Control Unit and online IoT device and peripheral devices. For conducting such experiment we counted the number of device-controlling function invocation opposed to the number of actual action that took place on the peripheral devices.

B. Evaluation

Figure 4 shows the terminal log of FPS per frame while running our hand detection inference task at the RPi. We observe a speed of $8-11$ FPS which is sufficient for our gesture recognition to be real-time although true real-time performance demands at least 24 FPS. Since we wrote a template based gesture recognition algorithm, which isn't compute heavy, there won't be any drop in overall FPS. Hence this experiment concluded that we our RPi package (with overclocking) can be feasible for the detection workload. From table III we can observe the gesture recognition success rate of our template based gesture detection algorithm. The "Hand Wave" gesture is used as an activation sequence to alert our detection system to start recognizing static gestures. Out of 100 trials we've seen 87 successful detection of hand waves. As for the other Gestures $(1-5)$, since they are static in nature, the detection system was quite accurate on recognizing them. Hence we can conclude that the gesture detection algorithm paired with the MediaPipe hand-landmark detection algorithm is quite accurate to carry out real world applications. For our control module we have developed functional-APIs to toggle IoT-based Smart LED bulb, and Relay Switches. We've logged the number of times a we've successfully interfaced with these peripheral devices to identify the action failure rate. Of the 100 peripheral device interaction (turn on/off) function invocation, all of them were successful deriving a 100% interfacing success rate. Since the toggle request for Smart-light is executed online, it introduces some latency on action. On the contrary, the GPIO interface based Relay controlling instantaneously takes action.

V. Conclusion and Future Work

In this paper, the home automation system was designed and implemented using the camera control system rather than voice based assistant system. One of the major difficulties here was to implement the recognition and making decision system in real time on raspberry Pi, due to its lower configuration. However, using overclocking and optimizing algorithm those mitigation was removed. Thus, using our approach we were able to successfully deploy a system, which at first recognized hand wave and works as active sequence for getting gestures for controlling different devices. After that, when the camera got different gestures like 1, 2 or 3 fingers then it control predefined IoT devices. Like in our system, table lamp (light-1) and living room light bulbs (light-2) when we show 1 and 2 fingers subsequently, and by showing 2 fingers it toggle the door lock system. The main advantage here was, as we used raspberry pi, it significantly reduce the cost and can be easily modifiable for adding future upgrades. In short, this low cost, flexible and energy efficient system make this a suitable secure choice for home automation system instead of voice assistant.

In future, we will update our algorithm for obtaining more frame rate. We will also also offload the computation to the cloud edge server which will make our system more faster. We will extend our work to recognize more sophisticated dynamic gestures for controlling more devices at the same time Finally, we will, develop an mobile application to make it user friendly so that adding new devices or adding new command or gestures can be easily done through the app.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, p. 2787–2805, Oct 2010.

[2] B. Vaidya, A. Patel, A. Panchal, R. Mehta, K. Mehta, and P. Vaghasiya, "Smart home automation with a unique door monitoring system for old age people using python, opencv, android and raspberry pi," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Jun 2017, p. 82–86.

[3] S. Jain, A. Vaibhav, and L. Goyal, "Raspberry pi based interactive home automation system through e-mail," in *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*, Feb 2014, p. 277–280.

[4] R. Bhise, N. Phadnis, R. Bari, and V. Dhage, "Iot based door lock and unlock system using face recognition," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 12, pp. 1136–1138, 2018.

[5] S. Liu, Y. Song, M. Zhang, J. Zhao, S. Yang, and K. Hou, "An identity authentication method combining liveness detection and face recognition," *Sensors*, vol. 19, no. 2121, p. 4733, Jan 2019.

[6] M. Muthumari, N. K. Sah, R. Raj, and J. Saharia, "Arduino based auto door unlock control system by android mobile through bluetooth and wi-fi," in *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Dec 2018, p. 1–4.

[7] V. Patchava, H. B. Kandala, and P. R. Babu, "A smart home automation technique with raspberry pi using iot," in *2015 International Conference on Smart Sensors and Systems (IC-SSS)*, Dec 2015, p. 1–4.

[8] I. Kaur, "Microcontroller based home automation system with security," *International Journal of Advanced Computer Science and Applications*, vol. 1, 2010.

[9] D. Pavithra and R. Balakrishnan, "Iot based monitoring and control system for home automation," in *2015 Global Conference on Communication Technologies (GCCT)*, 2015, pp. 169–173.

[10] W. Lv, F. Meng, C. Zhang, Y. Lv, N. Cao, and J. Jiang, "A general architecture of iot system," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 1, 2017, pp. 659–664.

[11] M. Al-Kuwari, A. Ramadan, Y. Ismael, L. Al-Sughair, A. Gastli, and M. Benammar, "Smart-home automation using iot-based sensing and monitoring platform," in *2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG 2018)*, 2018, pp. 1–6.

[12] O. Hamdan, H. Shanableh, I. Zaki, A. R. Al-Ali, and T. Shanableh, "Iot-based interactive dual mode smart home automation," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, 2019, pp. 1–2.

[13] G. B. Phani Ram and B. Keerthana, "Iot based amazon alexa using raspberry pi," *Annals of the Romanian Society for Cell Biology*, pp. 20 437–20 447, 2021.

[14] D. Pal, C. Arpnikanondt, S. Funilkul, and V. Varadarajan, "User experience with smart voice assistants: the accent perspective," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2019, pp. 1–6.

[15] J. Meyer, L. Dentel, and F. Meunier, "Speech recognition in natural background noise," *PloS one*, vol. 8, no. 11, p. e79279, 2013.

[16] M. Asadi-Aghbolaghi, A. Clapes, M. Bellantonio, H. J. Escalante, V. Ponce-López, X. Baró, I. Guyon, S. Kasaei, and S. Escalera, "A survey on deep learning based approaches for action and gesture recognition in image sequences," in *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*. IEEE, 2017, pp. 476–483.

[17] M. Oberweger, P. Wohlhart, and V. Lepetit, “Hands deep in deep learning for hand pose estimation,” *arXiv preprint arXiv:1502.06807*, 2015.

[18] E. Upton and G. Halfacree, *Raspberry Pi user guide*. John Wiley & Sons, 2014.

[19] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee *et al.*, “Mediapipe: A framework for building perception pipelines,” *arXiv preprint arXiv:1906.08172*, 2019.

[20] G. Bradski and A. Kaehler, “Opencv,” *Dr. Dobb’s journal of software tools*, vol. 3, p. 2, 2000.

[21] M. S. Louis, Z. Azad, L. Delshadtehrani, S. Gupta, P. Warden, V. J. Reddi, and A. Joshi, “Towards deep learning using tensorflow lite on risc-v,” in *Third Workshop on Computer Architecture Research with RISC-V (CARRV)*, vol. 1, 2019, p. 6.

[22] A. Pajankar, “Overclocking raspberry pi,” in *Raspberry Pi Supercomputing and Scientific Programming*. Springer, 2017, pp. 81–86.