Occupancy Monitoring in Smart Building: An Experimental Study

Ji Yang
University of Alberta
Edmonton, Canada
jyang7@ualberta.ca

Tianyu Zhang
University of Alberta
Edmonton, Canada
tzhang6@ualberta.ca

Xinlei Chen
University of Alberta
Edmonton, Canada
xinlei1@ualberta.ca

***Abstract*—With the development of Internet of Things (IoT) technologies, the IoT sensors are widely applied in buildings to make the building "smart". This paper purpose architecture for IoT devices to fetch, process, and store sensor data from various IoT devices in different areas, and use those data to track the location of each occupancy in the building. This paper also introduces a synthetic data generator that effectively generates synthetic sensor data, which is similar to the real-world setting. Our occupancy tracking system correctly tracks the location of all occupants within the building when we use the synthetic data. This is an open-source project, the code for synthetic data generator, backend server and frontend GUI is available on GitHub[1].**

***Index Terms*—Internet of Things, occupancy tracking, system architecture, synthetic data generation**

I. Introduction

As IoT technologies rapidly grow in the last decade, more and more existing commercial buildings with the integration of Building Management Systems (BMS) [1] have been turned into smart buildings by installing smart devices such as cameras, meters, RFIDs, sensors, and actuators. These IoT devices transform buildings into intelligent spaces able to significantly improve the living quality and productivity of occupants.

With the availability of IoTs in commercial buildings, building occupants and environment can be monitored in real-time. In this way, we can have access to real-time information of the building like environmental information in each room, occupancy counts in different rooms and zones, and even locate the occupants in the building. By combining and analyzing this information, a variety of applications can be developed. For example, based on the real-time occupancy counts in different areas, the smart building can dynamically schedule HVAC activity to help increase energy savings, therefore reducing the carbon footprint [2]. What's more, room-level real-time environmental sensor data can help building fire system detect and locate fire faster and occupant track information also help to rescue survivors in case of fire situation [3]. Finally, this information implies how the building is being used, which can be converted into insightful decision making, like improving space utilization.

In this paper, we build an occupancy monitoring system which is able to monitor the occupancy count in room level and track occupants in the building.

[1]Code available on https://github.com/skyu0221/660-iot

This paper is organized as follows. In section II, a brief overview of existing literature about occupant monitoring including occupancy count and occupant tracking. Then we describe the architecture and functionalities of our occupancy monitoring system in section III. Finally, the evaluation methods are provided in Section IV.

II. Background and Related Work

Occupancy Monitoring problems are referred to as three different interrelated problems — occupancy detection, tracking and event recognition [4]. In this section, we review the related work in occupancy detection and occupancy tracking since our occupancy monitoring system is built to address these two problems.

A. Occupancy Detection

A large number of occupancy detection models have been proposed to monitor building occupancy using wired and wireless sensor networks. For example, motion sensor data is fused with data from magnetic reed switches [5], thermal sensor arrays [6], carbon-dioxide sensors [7], cameras [8], passive infrared [9] and other ambient sensors [10] to estimate the room-level occupancy state (viz. occupant presence or count). However, these methods often require retrofits to install sensors. To address those issues, some recent studies focus on utilizing existing building infrastructure, such as the HVAC system [11], [12] and wireless network [13]. Today, the majority of medium and large commercial buildings are equipped with HVAC sensors sending their measurements to the central BMS. But these technologies have two main shortcomings. First, they achieve an acceptable level of accuracy only if physical sensors are installed in suitable locations in the building. Second, modeling occupancy using these techniques requires an abundance of labeled occupancy data collected from the same environment.

B. Occupancy Tracking

Occupancy tracking technology has attracted significant interest lately due to different applications in daily life, such as personalized heating/cooling systems and health care monitoring. A variety of technologies and tracking systems are studied, such as Wifi access points [14], Bluetooth localization [15], ultrasonic sensors [16], image-based tracking systems

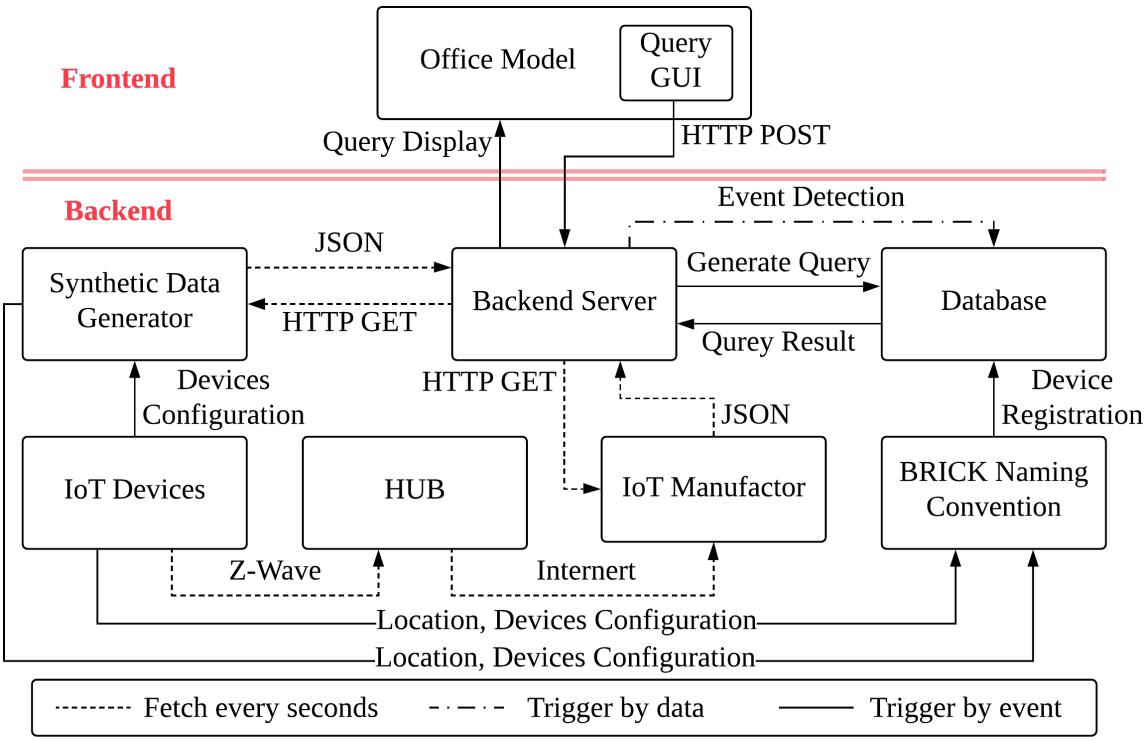

Fig. 1. The overall system architecture design.

[17], PIR sensors [18] and mechanical-based tracking system [19]. Wifi and Bluetooth techniques often require a great effort to deploy wireless transceivers and it is not convenient for occupants to carry the devices all the time. Also, indoor environment like walls or furniture will cause reflection and diffraction of radio waves, which will decrease the localization accuracy. The ultrasonic approaches are to send and receive ultrasonic waves. By measuring the echo intensity of the transmitted signal, the sensor can get the location information. However, vibrations in environments can easily affect the functionality of ultrasonic sensors. Mechanical-based tracking systems usually measure the footstep-generated vibration as a source of information for localization. But these methods require high density of sensor deployments and it is costly on installation and maintenance. PIR sensors are widely used in localization due to its advantages of low cost, less intrusive. However, compared to image-based tracking system, PIR has disadvantage on multiple target identification. Although the vision-based systems have privacy and high cost issue, they are still be widely used in localization due to its high accuracy. And with the development of edge computing [20] and video network, more and more tracking tasks has been done on the cameras rather than sending videos back to centralized system, which can address the privacy issue to a certain extent.

III. Methodology

A. Overall System Architecture

The overall system architecture design is shown in Figure 1. A list of IoT sensor devices is deployed at the building for our experimental purpose. Due to the limited budget for purchasing all the required devices for a large number of rooms, we have also implemented a synthetic data generator that mocks the data stream from the real devices. All the sensor data, after captured on-premise, are transferred to the cloud server that hosted a backend server written in Python Django framework. The Django backend server receives, processes and stores the data to a Postgre database. At the same time, the backend server also supports a list of RESTful APIs that will be used by the frontend, a real-time query, and visualization system that can demonstrate the building occupancy information.

B. Sensors

In this study, we utilize five different sensing nodes in each sensing area, which are the temperature sensor, motion sensor, lock sensor, power sensor, and camera unit.

1) **Temperature sensor**: We use Samsung SmartThings Smart Button GP-U999SJVLEDA as the temperature sensor. Each area has one temperature sensor, and we use this temperature as the area average temperature. This sensor reports temperature data to the hub every minute, or upon request.
2) **Motion sensor**: We use Samsung SmartThings Motion Sensor GP-U999SJVLBDA as the motion sensor. Each area has one or more motion sensors that can cover the whole area. We treat a set of sensors that cover the whole area as one node. This node reports binary motion data (active or inactive) to the hub when the motion state changes, or upon request.

3) **Lock sensor**: We use August Wi-Fi Smart Lock Pro AUG-SL03-C02-S03-C as the lock for each door. Each area only contains one lock sensor. For corridors, we use the door that separates the whole corridor, including the rooms connected to this corridor and the area outside the corridor. This sensor reports binary lock state data (locked or unlocked) to the hub when the motion state changes, or upon request.
4) **Power sensor**: We use Samsung SmartThings Outlet Smart Plug GP-U999SJVLDAA as the power sensor. Each area can have multiple power sensors. This sensor reports the power data in Watts to the hub every minute, or upon request.
5) **Camera unit**: Each unit contains two YI Dome Security Camera 2AFIB-YHS2016, and one Raspberry Pi 4 Model B. Figure 2 shows the architecture of the unit. Face recognition happens in the Raspberry Pi, and the Raspberry Pi sends the recognition result to the hub every second. We discuss this camera unit in detail in Section III-D.

There is one Samsung SmartThings HUB v2 GP-U999SJVLGDB installed in each sensing zone, which establishes the Z-Wave networks that connect all sensing nodes within the area. Z-Wave utilizes low-energy radio waves to help all sensing nodes communicate. The hub is connected to the internet, and it can receive the request through the Samsung SmartThings API. After the hub receives the request, it sends forward the request to the target sensing node and reports the response back to the client. Notice that the SmartThings API does not provide any data that can be used to identify the people who interacted with the sensors based on privacy considerations. In other words, we cannot know who opens or close the door. To fetch such identity data, we utilize the camera unit. In this study, the camera unit is only used to identify the people who interact with the lock.

C. Synthetic Data Generator

Due to the limited access to real-world data, a synthetic data generator is implemented to simulate the daily events in a commercial building. To closely simulate the real-world data, we simulate the number of employees, the number of guests, the events each person involved, the start time, event duration of each event for each person, and more as multiple stochastic processes, and each of them have different parameters.

In this synthetic data generator, we generate the locations of each person the next day at the end of each day and use these ground truth locations to generate the sensor data at each second. This generator assumes the number of the employee is fixed, and all employees never work overnights, and all other parameters below are changeable.

1) **Working time**: the period that employees should show up in the building. Assume the arrival time of each employee follows the exponential distribution with lambda equal to X minutes start from X minutes before the shift start time, and the leave time follows the exponential distribution with lambda equal to Y minutes start from a Y minutes before the shift ends. This lambda ensures the average arrival time, and the leave time is equal to the working time, and in most of the days, the employees come earlier and leave nearly the same as the time when the shift is over, with a very low possibility of working overtime or comes late. If the employee comes after time Z, the employee will ask for an absence and not show up in the building for the whole day.
2) **Call for absence**: the possibility of an employee to ask for daybreak. The employee will never show up in the building when they call for absence.
3) **Lunch break**: the period that employees should show up in the cafeteria or the rest area. Similar to the working time, we simulate the start time and end time of the lunch for each employee as two different exponential distribution. The eat time for each employee follows a beta distribution, which limits the eating time within α and β to ensure they can finish the lunch on time. If the employee finishes the lunch earlier, then the employee can remain in the cafeteria, goes to the rest area, or back to the office with a given possibility.
4) **Daily meeting**: the period of the meeting that all employees come to work must attend. The start time of the meeting is fixed with minimal variation, and the length of the meeting follows the normal distribution since the daily report should be the nearly equal length with some small variations.
5) **Customer visit and coffee break**: each employee can meet with multiple customers and take several coffee breaks during the working time. The number of meetings for the whole building and the number of coffee breaks each person take follows the Poisson distribution. The meeting and break length follow the normal distribution. Because the customer usually need to make appointments for the meeting, and the employee should only break when they are free, so the meeting time and the coffee break time is randomly selected after excluding the impossible periods. After decided the number of meetings in total, each meeting can be assigned to any employee who showed up in the building with even possibilities. The meeting can either hold in the office of the employee if only one customer comes or in the meeting room if two or more customers come together, and the meeting room is available. The arrival time of each customer follows the exponential distribution (similar reason to the working time).
6) **Moving speed and door open time**: each time a person moves from one area to another area, the moving speed follows the beta distribution, and the door open time is a fixed time that locks the door automatically after unlocked.

After generating the location data for each employee and guests, the generator simulates the data for each sensor at each second upon request.

1) **Temperature sensor**: Assume the central conditioning

system works well during the working hour, which means the temperature in each zone is tend to stay in a fixed value with small fluctuation. Moreover, when the number of people increases, the temperature slightly goes up since people emit heats, and with people leave, the temperature drops a little. And after a while, the temperature back to the set point.

2) **Motion sensor**: Whenever there is someone in the area, the motion sensor remains active, and when no one is in the area, the motion sensor is set to inactive.
3) **Lock sensor**: All locks automatically lock itself after a certain period of time unlocked. Unlock happens when a person moves from one area to another area.
4) **Power sensor**: We use a possibility to control the relationship between occupancy and power usage. If the occupancy is related to power usage, then the power is a fixed number during the time when someone within the area. Otherwise, the power usage randomly changes during the time.
5) **Camera unit**: We simulate the final recognition result of the camera by checking the location of each person. The camera report the name of the person a few seconds before the door is opened and after the person entered a different zone.

To better simulate the process of fetching data from the Samsung SmartThings API, the synthetic data generator is deployed on the Heroku server[2], and it requires the same header as fetch data through the Samsung SmartThings API. Moreover, to test the performance of the system querying system, the server is able to return the ground truth location information upon request.

D. Individual Identification: a privacy-preserved approach

With the limited identity information available from the sensor data stream, a face recognition system used for identifying individuals becomes necessary in our system. A face recognition system is developed software that is capable of identifying or verifying a person from a digital image or a video frame from a video capture source. Different implementations of facial recognition systems usually share the same generic architecture:

- a frontend to capture and, if necessary, pre-process the video captured.
- a backend that stores known individual information, such as their digital photo(s) of their face

However, a primary concern being raised with most face recognition systems is that people do not want their identification information such as photos to be hold by others. In addition, such a system needs to have a database to hold identity information, which may suffer from data leak and lead to unexpected results. With keeping such observations in mind, we developed our face recognition system by a privacy-protection approach that uses encoded latent face representations instead of directly storing user photos. This is similar to the way the IT industry used to store user passwords: hashing.

Thanks to the recent advances in deep learning, we can not only obtain a significantly accurate face recognition performance (99.7% reported in [21] but also leverage a "shortage" from deep models: inability to interpret and explain the model and its inference result. As the details of the deep model and its mechanism are beyond the scope of our work, a brief demonstration is included here. The frontend of our system consists of two webcams placed at both sides of the door being monitored, and a dedicated Raspberry Pi is used to process the video frame, extract the faces in the frames, and finally, encode the faces to send to the server for further usage. The process is shown in Figure 2. The cameras first capture the video frames, conventional image processing methods are used on the Raspberry Pi to detect and extract the faces presented in the frames. Then, the famous FaceNet [21] is used to transform the face images to a latent face feature vectors. Finally, the encoded face feature vectors are sent to the server and perform the recognition task.

The system is initiated by registering the building users to the system. To accomplish this task, a command-line based software is implemented to backbone the registration process. It asks the user for registration information and uses the exact same process as we have introduced above: the user photos are captured at the client-side, and only the latent feature vector is transferred to and stored at the server.

There are several benefits to using our implementation. First, for Close-circuit Television (CCTV) type of systems, with an increasing number of cameras being placed, a significant usage on the network bandwidth could raise due to a massive amount of video sources need to be transferred to the server. However, our implementation avoids all the transfer of video frames. Secondly, our origin intention is for knowing where and when the people move inside the building to understand the occupancy of the building. Therefore we do not require any video capture being stored and only compared the transfer face feature vector with the ones we stored in the database.

E. Backend Implementation

The backend server is written fully in Python Django framework with a few support package implemented with C++ (dlib used for face related functionalities). We implemented a list of APIs that support the frontend demonstration.

The implemented APIs are:

- Retrieve the number of people in each zone (in real-time)

```
GET  /room/
Response body
{
  "room_info": [
  ["Room_1_1_140", 2],
  ["Room_1_1_141", 3],
  [Room name, occupancy count]]
}
```

[2]Available on https://pacific-temple-42851.herokuapp.com/

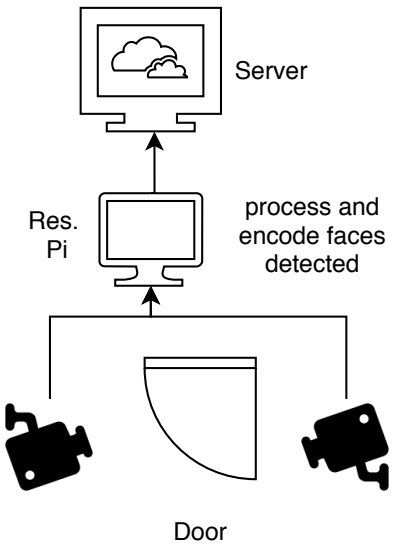

Fig. 2. Face recognition process.

- Retrieve the number of people in a specific zone (in real-time)

```
POST /room_info/{room_id}
Post Body:
{
  "room_id": "Room_1_1_140",
}
Response body
{
  "room_name": "Room_1_1_140",
  "occupancy_info": ["Ana", "Bob"]
}
```

- Count the number of people visited the building during a time interval

```
POST /people_building/
Post body
{
  "start" : "2019-06-29 08:15:27",
  "end" :  "2019-06-29 08:15:27"
}

Response body
{
  'count": 5
}
```

- Retrieve the name of people visited a specific zone during a time interval

```
POST /people_room/
Post body
{
  room":  "Room_1_1_140",
  "start" : "2019-10-31 11:30:45",
  "end" : "2019-10-31 12:30:48"
}
Response body
{
  "count": 3,
  "occupancy_info": ["Ana",
          "Ray", "Joseph"]
}
```

- Retrieve all zones that a specific person visited during a time interval

```
POST /person_room/
Post body
{
  "name":  "John",
  "start" : "2019-10-31 11:30:45",
  "end" : "2019-10-31 12:30:48"
}
Response body
{
  "room": {"Room_1_1_140"}
}
```

- Retrieve the utilization rate of a specific zone in a given day

```
POST /utilization/
Post body
{
  "room": "Room_1_1_140",
  "date" : "2019-10-31 00:00:00"
}
Response body
{
  "utilization": 0.59
}
```

The backend services are deployed at Amazon Web Services with a free-tier EC2 instance[3]. The frontend visualization software has access to the backend APIs and uses these predefined queries to obtain the information needed.

IV. Experiment

A. Experiment Setup

In this study, we do not have any available real data to use. Therefore, we use synthetic data to test our system. We generate the synthetic data by the generator we purposed in Section III-C.

Figure 3 shows the floor plan of our experiment. We use half of the floor in a real commercial building to generate synthetic data. In this floor plan, we have four offices, Room 140, 142, 143, and 144. Therefore, we assume the floor contains four employees, and each of them is assigned to one of those offices. Room 141 is a meeting room, where the appointment with two or more customers and the daily meeting is held here. Zone 150 contains the corridor as well as the rest lobby. The

[3]Available on http://3.133.122.194:8000

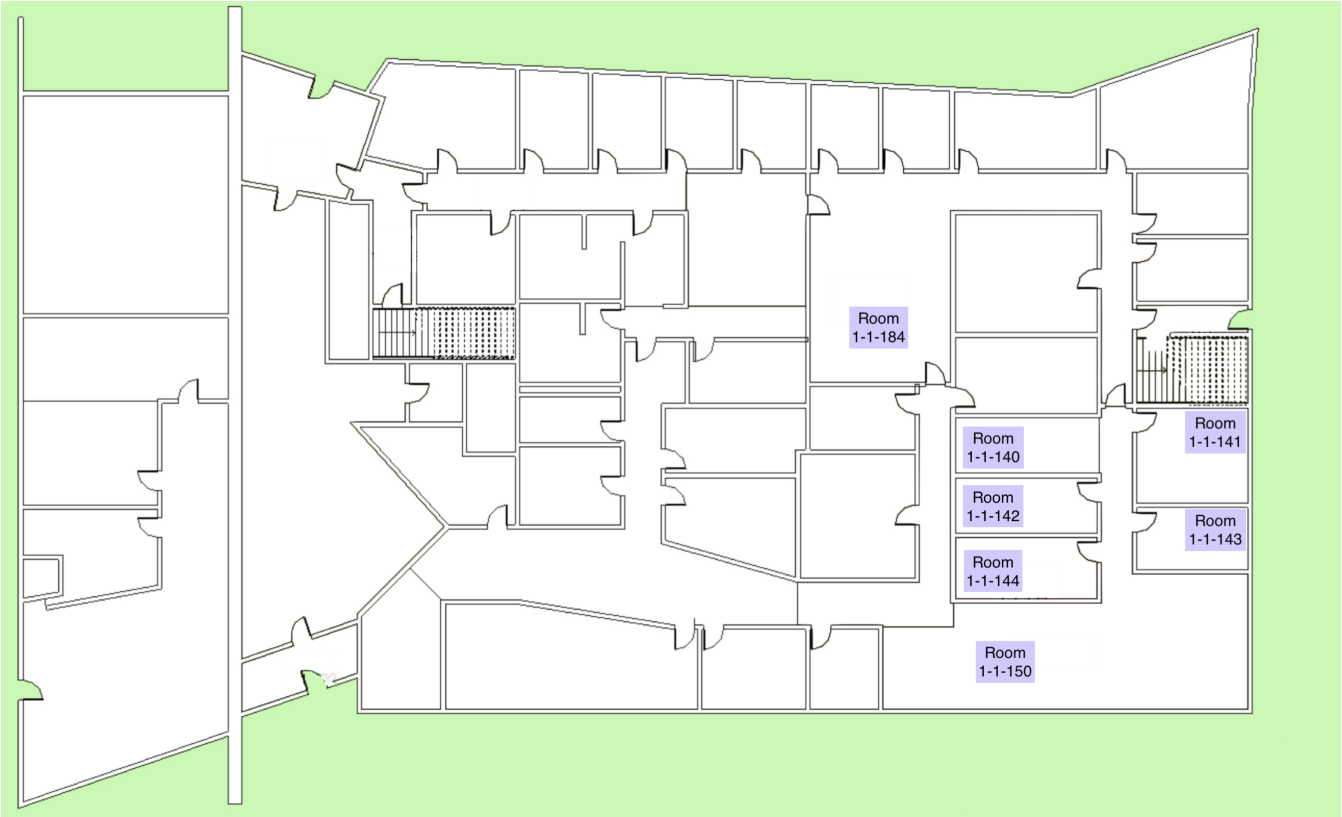

Fig. 3. Floor plan for the experiment.

door sensor for Zone 150 locates between Zone 184 and Zone 150. All people who want to leave the building or go to Zone 184 must pass Zone 150, and people can take a break in Zone 150. Zone 184 is the cafeteria, and all employee comes to 184 for lunch. The door sensor for Zone 184 locates between Zone 184 and the stairs. All people enter or leave this floor must open the door of Zone 184.

We set the working hours from 9 am to 5 pm every day and the daily meeting at 4 pm every day with an average length of 15 minutes. All workers start arriving at 8:30 am and leave after 4:45 pm. The lunch starts at noon and ends at 1 pm. The possibility of the employee call for absence is 1%, and the whole floor has a rate of 3 meetings per day.

B. Occupancy Detection and Tracking

It is non-trivial to retrieve the occupancy data only use data from some fixed period of time, since some person in this period of time might never reach any doors, and thus we cannot identify the person. To address this challenge, we utilize the advantage of our backend server data fetching method. Our backend server fetches the sensor data every one second, and save all data into the database. Because the person can never appear in two zones simultaneously, and people cannot teleport from one room or zone to another room or zone if no door connects them. Therefore, we track the location of each person that appeared in the building and update the location when they open a door. We save the location data into the database, which can be used to answer queries from the frontend.

We start from the time when all people are not in the building, such as the early morning, and assume we have an extra zone called “home”. For each second, we check the appearance of the front door and the door sensor that have someone inside the zone. If someone appears in the front door, we start tracking the location of this person by tracking the appearance of the person in the camera for each possible doors he/she able to open and mark this person an entered the building. If any person in the building appears in a camera that connected to the zone he/she is currently located, we mark the person move to the connected zone.

Figure 4 demonstrate the tracking result on 2020-04-04 for Employee 4. The result shows a perfect tracking on the view of a day. The estimated location usually has 2 4 seconds of differences with the ground truth location, with the person move from one area to another area. This is due to the truth that the time person showed up in the camera is not necessarily the time they move to another area; they need some time to move and open the door. After calculating the location of each person at each second, we can answer all frontend queries mentioned in Section III-E by some SQL statements.

1) Retrieve the number of people in each zone or a specific zone in real-time: Extract the locations of each people in the building at the current second in the location table, and count the number of entries in each zone as the number of occupancy.

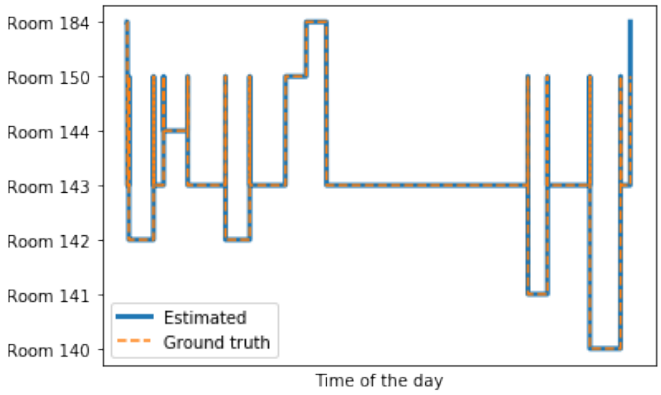

Fig. 4. Occupancy tracking result on 2020-04-04 for Employee 4.

2) Count the number of people visited the building during a time interval: Select all rows that within the given time period, and count the number of unique names appeared in the location table.

3) Retrieve the name of people visited a specific zone during a time interval: Record the unique names appeared in a specific zone during the given period from the location table.

4) Retrieve all zones that a specific person visited during a time interval: Record the unique zone names appeared during the given time period, where only select rows that represent the specified person from the location table.

5) Retrieve the utilization rate of a specific zone in a given day: Compute the total seconds during 9 am to 5 pm that a specific zone have motion sensor status as "active", and divide the value by the total seconds between 9 am to 5 pm, which is equal to 28800 seconds.

We demonstrate our result of queries using wxpython, which provide a graphical user interface for people to interact with our backend server and display the result of their queries[4].

V. Future Works

In this paper, we discussed the architecture of the occupancy tracking system. We introduced four main sections of the system: the camera unit, the backend server, the synthetic data generator, and the occupancy tracking algorithm. This system showed a very reliable tracking result when we use synthetic data. For future work, we plan to collect the real-world data and test the performance of our backend server as well as the occupancy tracking system. In the meantime, we can extend this system to some other applications, such as the escape route planning. We also plan to answer more queries related to the occupancy data in the building.

Acknowledgments

We would like to thank Prof. Dr. Eleni Stroulia and Dr. Faiza Nawaz, from the Dept. of Computing Science of the University of Alberta, for the invaluable support and guidance in the making of this this project.

[4]Code available on https://github.com/skyu0221/660-iot

References

[1] G. Levermore, *Building Energy Management Systems: an Application to Heating, Natural Ventilation, Lighting and Occupant Satisfaction*. Routledge, 2013.

[2] A. De Paola, M. Ortolani, G. Lo Re, G. Anastasi, and S. K. Das, "Intelligent management systems for energy efficiency in buildings: A survey," *ACM Comput. Surv.*, vol. 47, June 2014.

[3] K. Deve, A. Kumar, and G. Hancke, "Smart fire detection system based on the ieee 802.15.4 standard for smart buildings," in *Network Security and Communication Engineering - Proceedings of the 2014 International Conference on Network Security and Communication Engineering, NSCE 2014* (K. Chan, ed.), Network Security and Communication Engineering - Proceedings of the International Conference on Network Security and Communication Engineering, NSCE, pp. 649–652, CRC Press/Balkema, 2015.

[4] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri, "Iot-based occupancy monitoring techniques for energy-efficient smart buildings," in *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 58–63, 2015.

[5] Y. Agarwal, B. Balaji, S. Dutta, R. K. Gupta, and T. Weng, "Duty-cycling buildings aggressively: The next frontier in hvac control," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 246–257, 2011.

[6] A. Beltran, V. L. Erickson, and A. E. Cerpa, "Thermosense: Occupancy thermal based sensing for hvac control," in *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, (New York, NY, USA), p. 1–8, Association for Computing Machinery, 2013.

[7] V. B. Krishna, D. Jung, N. Q. M. Khiem, H. H. Nguyen, and D. K. Y. Yau, "Energytrack: Sensor-driven energy use analysis system," in *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, (New York, NY, USA), p. 1–2, Association for Computing Machinery, 2013.

[8] V. L. Erickson, S. Achleitner, and A. E. Cerpa, "Poem: Power-efficient occupancy-based energy management system," in *2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 203–216, 2013.

[9] Y. Peng, A. Rysanek, Z. Nagy, and A. Schlüter, "Using machine learning techniques for occupancy-prediction-based cooling control in office buildings," *Applied Energy*, vol. 211, pp. 1343 – 1358, 2018.

[10] S. Mamidi, Y.-H. Chang, and R. Maheswaran, "Improving building energy efficiency with a network of sensing, learning and prediction agents," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, (Richland, SC), p. 45–52, International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[11] O. Ardakanian, A. Bhattacharya, and D. Culler, "Non-intrusive techniques for establishing occupancy related energy savings in commercial buildings," in *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, BuildSys '16, (New York, NY, USA), p. 21–30, Association for Computing Machinery, 2016.

[12] I. B. Arief-Ang, F. D. Salim, and M. Hamilton, "Da-hoc: Semi-supervised domain adaptation for room occupancy prediction using co2 sensor data," in *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, BuildSys '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[13] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal, "Sentinel: Occupancy based hvac actuation using existing wifi infrastructure within commercial buildings," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, (New York, NY, USA), Association for Computing Machinery, 2013.

[14] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, (New York, NY, USA), p. 173–184, Association for Computing Machinery, 2010.

[15] F. S. Danis and A. T. Cemgil, "Model-based localization and tracking using bluetooth low-energy beacons," in *Sensors*, 2017.

[16] S. P. Tarzia, R. P. Dick, P. A. Dinda, and G. Memik, "Sonar-based measurement of user presence and attention," in *Proceedings of the 11th International Conference on Ubiquitous Computing*, UbiComp '09, (New York, NY, USA), p. 89–92, Association for Computing Machinery, 2009.

[17] V. A. Petrushin, G. Wei, and A. V. Gershman, "Multiple-camera people localization in an indoor environment," *Knowl. Inf. Syst.*, vol. 10, p. 229–241, Aug. 2006.

[18] L. Wu, Y. Wang, and H. Liu, "Occupancy detection and localization by monitoring nonlinear energy flow of a shuttered passive infrared sensor," *IEEE Sensors Journal*, vol. 18, no. 21, pp. 8656–8666, 2018.

[19] J. D. Poston, R. M. Buehrer, and P. A. Tarazaga, "Indoor footstep localization from structural dynamics instrumentation," *Mechanical Systems and Signal Processing*, vol. 88, pp. 224 – 239, 2017.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, pp. 637–646, 2016.

[21] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.