Automatic HVAC Control with Real-time Occupancy Recognition and Simulation-guided Model Predictive Control in Low-cost Embedded System

Muhammad Aftab, Chien Chen, Chi-Kin Chau and Talal Rahwan☆

Masdar Institute of Science and Technology

Abstract

Intelligent building automation systems can reduce the energy consumption of heating, ventilation and air-conditioning (HVAC) units by sensing the comfort requirements automatically and scheduling the HVAC operations dynamically. Traditional building automation systems rely on fairly inaccurate occupancy sensors and basic predictive control using oversimplified building thermal response models, which hinder the performance of energy saving. However, recent embedded system technologies provide viable low-cost computing platforms with powerful processors and sizeable memory storage in a small footprint. As a result, sophisticated computational tasks can be executed efficiently on these systems, such as real-time video processing and accurate building thermal response simulation. We designed and implemented an occupancy-predictive HVAC control system in a low-cost yet powerful embedded system (using Raspberry Pi 3) to demonstrate the following key features for building automation: (1) real-time occupancy recognition using video-processing and machine-learning techniques, (2) dynamic analysis and prediction of occupancy patterns, and (3) model predictive control for HVAC operations guided by real-time building thermal response simulation using on-board EnergyPlus simulator. We deployed and evaluated our system for providing automatic HVAC control in a large public indoor space of a mosque, which is observed to achieve satisfactory performance.

Keywords: Automatic HVAC control, embedded system, occupancy recognition, model predictive control

1. Introduction

Heating, ventilation and air-conditioning (HVAC) units consume a significant amount of energy in both residential and commercial buildings [1], which are a primary target of building automation. In general, building automation systems aim to intelligently control building facilities in response to dynamic environmental factors, while maintaining satisfactory performance in energy consumption and comfort. The primary functions of a building automation system include: (1) sensing of the environmental factors by measurements, and (2) optimizing control strategies based on the current and predictive states of building and occupancy. These tasks require an integrated process of sensing, computation and control.

Traditional building automation systems rely on fairly inaccurate occupancy sensors, which hinder the responsiveness of automation systems. For example, passive infrared and ultra-sound occupancy sensors produce poor accuracy, because they are unable to determine the occupancy state properly when occupants remain stationary for a prolonged period of time. They also have a limited detectable range in a large area. More accurate sensing technology, such as cameras that use visible or infra-red lights, can significantly improve the accuracy of occupancy recognition.

On the other hand, model predictive control, by which the future thermal response and external environmental factors are anticipated to make control decisions accordingly, has been considered in a number of studies [2–7] which are shown to be more effective than classical PID and hysteresis controllers that do not consider anticipated events. However, these studies are often based on time-invariant, first-principle linear models (also known as lumped element resistance-capacitance (RC) models [8]), considering only simple building geometry and single-zone setting in near-future time horizon. Although these linear models are easier for calibration (e.g., using frequency domain decomposition, or subspace system identification methods [8–10]), the error accumulates considerably when a longer time horizon is considered in model predictive control. While non-linear models are too complicated and impractical, other alternatives based on physical models of building thermal response can provide a feasible solution.

Recently, there have been remarkable advances in embedded system technologies, which provide low-cost platforms with powerful processors and sizeable memory storage in a small footprint. In particular, the emergence of *system-on-a-chip* technology, which integrates all major components of a computer into a single chip, can provide versatile computing platforms with low-power consumption and mobile network connectivity in a cost-effective manner for mass production. As a result, smartphones are

☆Email: {muhaftab, cchen, ckchau, trahwan}@masdar.ac.ae

able to rapidly evolve from single-core to multi-core processors with a low incremental production cost. Notably, the *Raspberry Pi* project [11], which originally aimed to provide affordable solutions for teaching of computer science, has rapidly evolved for a wide range of advanced scientific projects. Therefore, there are plenty of opportunities to harness recent embedded system technologies in intelligent building automation systems. Particularly, sophisticated computational tasks can be conducted on these embedded systems efficiently, such as real-time video processing and accurate building thermal response simulation.

Against this background, we designed and implemented an occupancy-predictive HVAC control system in a low-cost yet powerful embedded system (using Raspberry Pi 3) for building automation. We highlight the three key features of our system to be presented in this paper.

(Section 3) *Real-time Video-based Occupancy Recognition.* We apply advanced video-processing techniques to analyze the features of occupants from video cameras, and automatically classify and infer the states of occupancy. Moreover, we consider privacy enhancement using a frosted lens. Our system can achieve high accuracy for timely occupancy recognition by real-time video processing. We further improve the accuracy rate of occupancy recognition by using machine-learning, with good performance even in considerably crowded settings.

(Section 4) *Dynamic Occupancy Prediction.* We employ various linear and non-linear regression models to capture and predict occupancy trends according to different day-of-week, seasonal patterns, etc. We present general as well as domain-specific approaches for occupancy prediction. Our models are able to identify the future occupancy trends considering a variety of dynamic usage patterns.

(Section 5) *Simulation-guided Model Predictive Control.* We utilize simulation-guided model predictive control in our automatic HVAC control system. We employ *EnergyPlus* simulator [12] for real-time HVAC control, which can provide accurate building thermal response simulation. We ported EnergyPlus simulator to the Raspberry Pi embedded system platform. We also release our Raspberry Pi version of EnergyPlus publicly [13] to enable other researchers to take advantage of our work for future building automation projects.

Our automatic HVAC control system is intended for public indoor spaces, such as corridors, libraries, or communal areas. Unlike private spaces such as homes, these public indoor spaces are not controlled by a particular occupant and can be affected by a diverse set of occupancy patterns. Such patterns tend to vary more dynamically in public spaces compared to private ones, posing challenges for effective occupancy sensing and prediction systems.

In particular, our system is deployed and evaluated for providing automatic HVAC control in a large public indoor space of a mosque (see Figure 1), which is the worship place for followers of Islam. We will present the testbed design and implementation in Section 6. Typically, mosques have large public spaces, and are open 24-hours a day and 7-days a week. There are nearly 5,000 mosques in the UAE [14], and over 55,000 mosques in Saudi Arabia [15]. Due to the hot climate in this region, HVAC is required on a regular basis. This work demonstrates how automatic HVAC control system can lead to significant energy saving for public indoor spaces without human administration.

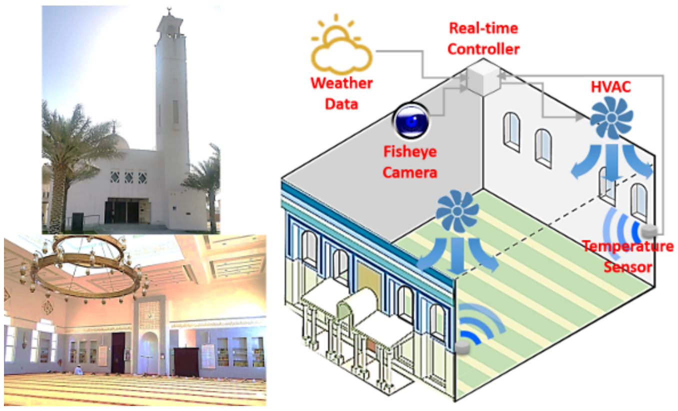


Figure 1: A large public indoor space of a mosque is used as a testbed for our automatic HVAC control system. Fisheye video camera, temperature and humidity senors, as well as real-time controller for HVAC have been deployed in our testbed.

2. Background and Related Work

In this section, we present the background and related work of our system. First, a brief review of occupancy recognition and prediction is provided below. Then, we compare three major approaches of model predictive control for HVAC control in the literature.

2.1. Occupancy Recognition and Prediction

Any object with temperature higher than perfect zero emits heat in the form of radiation. The conventional passive infra-red (PIR) sensors can be used to detect a certain wavelength when a person is near the sensor. Previous papers [16–18] demonstrated the possibility of applying this in lighting systems. However, the disadvantage of the PIR sensor becomes evident when the occupant remains stationary for a certain period of time. PIR sensors are designed to detect changes in the measurement. If a person remains stationary in front of the sensor, then as far as the sensor is concerned, there will be no change in the measurement, leading to erroneous observation.

Video based occupancy detection can provide better accuracy. The authors in [19] developed a technique for occupancy recognition based on video feed from cameras. They extract a set of features (edges, textures, etc.) from the video and use them in a regression model to estimate the number of occupants. However, their method is unable to count people in real-time and can only work with pre-recorded videos. The authors in [20] proposed a practical algorithm for occupancy detection using cameras. They focused on finding the head contours to determine occupancy in indoor environments. In contrast, our system is able to count occupants in considerably crowded settings in real time with good accuracy.

In addition to occupancy recognition, occupancy prediction is required to anticipate building usage and control pre-cooling in advance. A variety of techniques to predict occupancy have been proposed in the literature, including statistical analysis, machine learning, agent-based techniques, and stochastic modelling, etc. Reviews of occupancy prediction techniques are provided in [21, 22].

2.2. Model Predictive Control

There are three major approaches of model predictive control in the literature:

- *LTI Model Predictive Control*: This uses a linear time-invariant (LTI) mathematical model, which is a simplified thermal dynamics model considering only a near-future short time horizon. A common approach is to use an RC model to capture the first-order heat transfer dynamics. It is suitable for simple settings, such as a single zone with simple building geometry There are usually a small number of parameters in the model. LTI model predictive control is explored in [8–10].

- *Non-linear Model Predictive Control*: Many real-world systems exhibits rich non-linearity. There are many general non-linear mathematical models of system dynamics, such as Volterra series, neural networks and NARMAX models. However, most non-linear models require a large parameter space, which is difficult to calibrate from measurements. The use of non-linear models for predictive control is investigated in [23, 24].

- *Simulation-guided Model Predictive Control*: In this approach, model predictive control is guided by real-time physical model simulators considering future anticipated events. For buildings, there are a number of simulators, such as *EnergyPlus, TRNSYS*, that are much more accurate than LTI models, and also easier to calibrate than general non-linear models. Reviews of different building simulators and their merits are presented in [5, 25]. Existing studies that used simulation programs to facilitate model predictive building control can be found in [5, 7, 26].

For the above control approaches to be effective, it is crucial to calibrate the model parameters so that the model response is consistent with the empirical data. Several model calibration methods have been proposed in the literature. These methods can be broadly categorized as *manual* or *automated. Manual* approaches require the modeler to intervene repeatedly and make adjustments, whereas *automated* approaches use mathematical and statistical models to automate the calibration process. A review of model calibration can be found in [27].

3. Occupancy Recognition

Occupancy information is crucial for many applications, such as building management and human behavior studies. We develop an occupancy recognition system based on real-time video processing of a video stream to infer the occupancy patterns dynamically. This raises a number of challenges. First, structures differ from one building to another, leading to the possibility of occupants being obscured by various obstacles, such as pillars. Hence, we need to track the movements of occupants to determine the occupancy more accurately. Second, our algorithm is executed on an embedded system (e.g., Raspberry Pi), which has limited processing power and memory space compared to a typical desktop computer; this is particularly challenging since the typical video-processing algorithms are computationally demanding.

The basic idea of our occupancy-recognition algorithm is to count the number of people crossing a virtual reference line in the video, captured by a fisheye camera. Objects are identified as moving blobs (i.e., a set of connected points whose position is changing during the video stream); every such blob is interpreted as a person. Whenever a moving blog is detected, the algorithm keeps track of its movement to determine whether it passes the virtual reference line, and then updates the number of occupants accordingly. In particular, we position the line near the entrance of the space. Whenever the moving blob passes inward, the total number of occupants is increased by 1, and whenever the moving blob passes outward, the total number of occupants is decreased by 1.

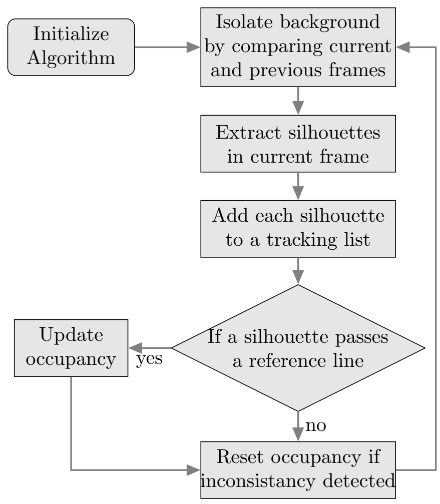

Initialize Algorithm
Isolate background by comparing current and previous frames
Extract silhouettes in current frame
Add each silhouette to a tracking list
If a silhouette passes a reference line
yes
no
Update occupancy
Reset occupancy if inconsistancy detected

Figure 2: Flowchart of occupancy-recognition algorithm.

Our algorithm consists of the following five steps: (1) background isolation, (2) silhouette detection, (3) object tracking, (4) inward/outward logging, and (5) inconsistency resolution. The flowchart of our algorithm is presented in Figure 2. In our implementation, two open-source projects are used: *OpenCV* [28] and *openFramework* [29]. Next, we will explain each step in details.

3.1. Background Isolation

The purpose of background isolation is to identify a background image in the current video frame. Note that the appearance of the background may vary over the course of the video, depending on the time-of-day (e.g., turning on the lights at night may significantly alter the appearance of the background compared to natural light). The shadows of occupants must also be taken into consideration during the background isolation. To overcome these challenges, we employ a Gaussian mixture-based background/foreground segmentation algorithm proposed in [30, 31], and implemented on OpenCV.

3.2. Silhouette Detection

In our setting, the term "silhouette" is used to refer to the border of a set of continuous points. Silhouette detection is based on an algorithm proposed in [32]. The silhouettes of moving objects are extracted by comparing the current frame with the previous one. See Figure 3 for some examples. Only the silhouettes larger than a certain threshold area, A, are considered. The threshold is adjusted depending on the viewpoint and orientation of the camera. Furthermore, we use different values of A for different parts of space, as to reflect the fact that individuals appear smaller as they move farther away from the camera. Importantly, the silhouettes of two or more people may overlap, and may therefore be interpreted as only one individual. To overcome this challenge, we use a machine-learning technique which will be explained later on in Section 3.6.

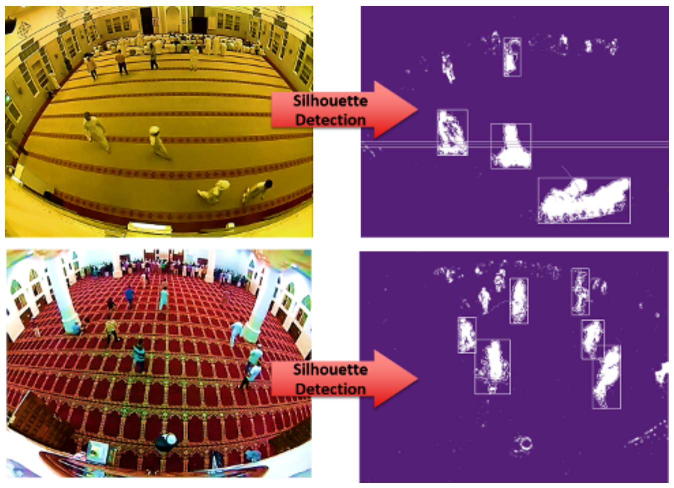


Figure 3: Examples of silhouette detection, taken from different buildings in which our system is deployed.

3.3. Object Tracking

After obtaining the silhouettes of occupants, their bounding boxes are logged for tracking. To this end, the list T_a of silhouettes from the last frame is maintained. The algorithm then obtains the list T_b of silhouettes from the current frame. For each bounding box B_b in T_b, the algorithm determines whether there exists a box B_a in T_a that is within a certain distance, d, from B_b. If so, then B_b is assumed to be the new location of B_a. Consequently, B_a is updated in T_a, and its location is set to be that of B_b, in preparation for the next iteration of the algorithm. Note that the silhouettes are detected when they are moving. However, in case that people might stop for few seconds then continue to walk, those objects will be removed from T_a when they are missed for a certain number of seconds.

3.4. Inward/Outward Logging

Given the collection of moving objects and their locations, a virtual reference line is used to count occupants. Specifically, whenever the locations are updated, the algorithm checks every object to determine whether that object has crossed from one side of the line to the other. If so, then the number of occupants is updated according to the direction of the movement. For inward movement, the number of occupants decreases by 1, otherwise it increases by 1. Furthermore, since the silhouettes of any two moving objects may overlap, the width of the bounding box can be used to infer the number of occupants contained in each object.

3.5. Inconsistency Resolution

With all the techniques described thus far, the performance of the algorithm may not be satisfactory, due to one major challenge: *the silhouettes of different occupants may overlap*. In this case, some of the overlapping occupants may go undetected by the algorithm. This problem becomes even more evident the movement patterns are affected by whether the occupants are entering or leaving the building, e.g., due to the fact that occupants arrive one by one, but leave all at once. For instance, in our application domain, the pace at which people leave is consistently faster than the pace at which they enter. Consequently, when all occupants leave at once, the number of overlapping silhouettes increases, leading to a larger number of people going undetected by the algorithm. As a result, the algorithm on average misses more occupants leaving than entering the building, leading to the erroneous conclusion that there are still occupants in the building when in fact there are none.

To resolve such inconsistency, two simple techniques are used. First, if the number of occupants becomes negative, the occupant counter is frozen until another object crosses the reference line inward. Second, if no moving object is detected for a certain period of time, the occupant counter is reset to zero. While these simple techniques reduce the error, they are clearly insufficient. In Section 3.6, we propose a dedicated technique to address this issue.

3.6. Improvement by Machine Learning

As mentioned earlier, one of the major challenges that we have encountered while deploying our occupancy-detection algorithm is to resolve the problem of overlapping silhouettes. One solution is based on the width of bounding box; the wider the box, the more occupants it contains. However, we observe that such a solution is insufficient, especially when an occupant happens to be directly in front of another. To resolve this issue, we employ machine learning and image classification techniques, coupled with randomized principal component analysis (PCA) [33], which is implemented in [34].

In more details, we collected thousands of blobs from video footages spanning a period of one week, and segmented each such blog as a separate image to create our training dataset. The number of occupants in each image was manually identified. See Figure 4 for some examples. Next, the images are transformed to gray scale to be analyzed, the reason to transform it to gray scale is to reduce the computational load. To apply the machine learning, all the images are rescaled to the same size (30x15 pixels, which is the minimum size of the images) and then converted to monochromatic. In this case, the size of those images will be 450 pixels. Second, randomized PCA is used to project the pixels in the original array to a smaller array that preserves the characteristics of the images, as a set of 25 features for each image in this case. The projection aims to reduce the computational complexity. Moreover, we add the original width, height and the ratio of black pixels to the set of features. After obtaining the features, Gaussian Naive Bayes is used to classify the blobs with respect to the number of occupants.

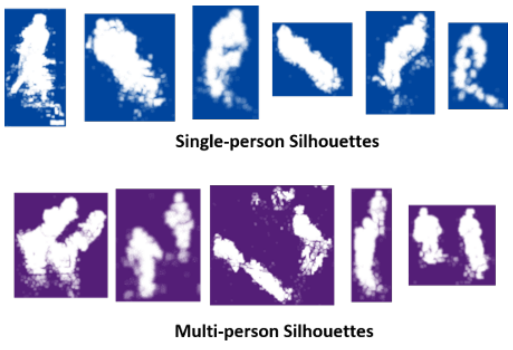


Figure 4: Sample images from the training dataset used for machine learning and image classification.

3.7. Privacy Enhancement

Privacy invasion is always a concern for most of video based detection methods. In our system, the videos and images are discarded immediately after processing. Also, the camera stream can be only accessed by one process at the same time, which means if the system is running, all the other programs are blocked from accessing the camera. In addition, hardware solution has been tested. We use frosted lens by overlaying a semi-transparent layer in front of the camera, so it can blur the camera. As a result, the faces of occupants are not recognizable. See Figure 5 for an illustration. Our occupancy-recognition algorithm can still apply in such a setting with good accuracy.

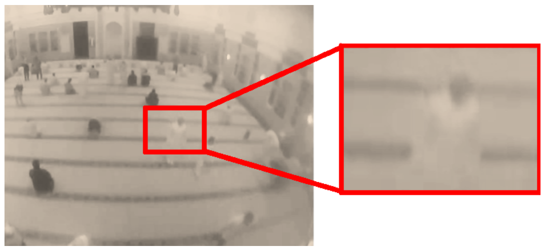


Figure 5: Enhance privacy by using frosted lens on camera.

3.8. Evaluation Results

In this section, we evaluate the accuracy of the occupancy recognition methods. We collected video footages from our testbed, with 800×600 pixels frame size and 30 frames per second sample rate. The true occupancy is obtained by manual counting in each frame.

First, define the accuracy rate by:

$$\mathsf{AccuracyRate} \triangleq 1 - \frac{\mathsf{Missing}_{\mathrm{in}} + \mathsf{Missing}_{\mathrm{out}}}{\mathsf{Total}_{\mathrm{in}} + \mathsf{Total}_{\mathrm{out}}} \tag{1}$$

where $\mathsf{Missing}_{\mathrm{in}}$ is the number of occupants that have entered but not detected, and $\mathsf{Missing}_{\mathrm{out}}$ the number of occupants that have exited without being detected.

Also, define the false alarm rate by:

$$\mathsf{FalseAlarmRate} \triangleq 1 - \frac{\mathsf{OverCount}_{\mathrm{in}} + \mathsf{OverCount}_{\mathrm{out}}}{\mathsf{Total}_{\mathrm{in}} + \mathsf{Total}_{\mathrm{out}}} \tag{2}$$

where $\mathsf{OverCount}_{\mathrm{in}}$ is the number of occurrences of over-counting inward occupants, and $\mathsf{OverCount}_{\mathrm{out}}$ is the number of occurrences of over-counting outward occupants.

Table 1: Results of occupancy recognition comparing morning and night intervals.

Type	Video Length	Total Num. of Occupants	Accuracy Rate	FalseAlarm Rate
Night	40 mins	127	90%	3.1%
Night	40 mins	154	90%	4.5%
Morning	20 mins	407	84%	2.4%

The first evaluation results are presented in Table 1 for certain periods in morning and at night. AccuracyRate is up to 90%. Typically, the algorithm is able to recognize occupancy accurately in a considerably crowded environment, and AccuracyRate reaches 84%. Note that FalseAlarmRate is relatively low in all settings.

Table 2: Results of occupancy recognition comparing weekend, weekday, Friday.

Type	Video Length	AccuracyRate	R-square
Weekend	1 day	88%	0.956
Weekday	1 day	86%	0.939
Friday	1 day	81%	0.609

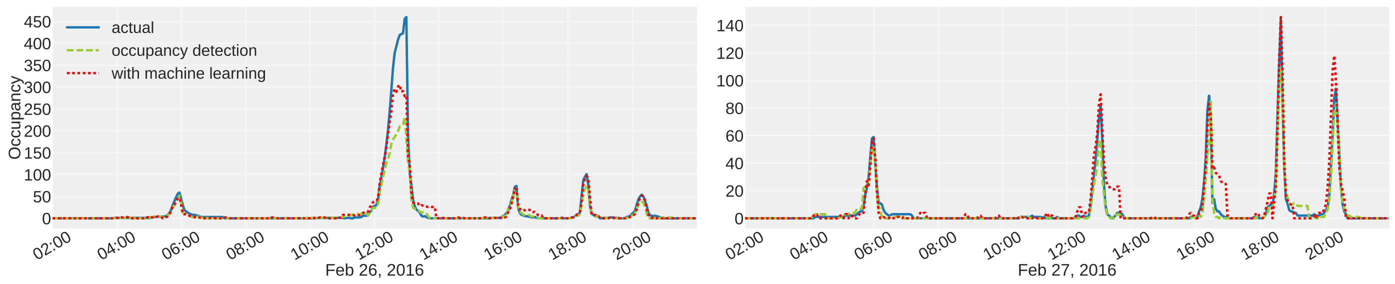


Figure 6: 26-Feb-2016, is a Friday, with the peak at midday showing the number of occupants attending the Friday prayer. On the other hand, 27-Feb-2016 is not a Friday, and so all five peaks are comparable

We define R-square as the difference of the output of algorithm against the true data. R-square is always between 0 and 1, where 1 means a perfect match between the algorithm output and true data, and 0 means total disagreement between the two. The second evaluation results are presented in Table 2 for one-day periods during weekend, weekday, Friday. Note that in our testbed Fridays are busiest day in the week, because of Friday praying session for mosques. The time-varying occupancy is plotted in Figure 6. AccuracyRate is up to 88%, and R-square is up to 0.956. Table 3 shows that frosted lens only reduces the accuracy slightly.

Table 3: Results of occupancy recognition comparing normal lens and frosted lens.

Type	Video Length	Total Num. of Occupants	Accuracy Rate	FalseAlarm Rate
Normal Lens	160 mins	322	87.8%	3.3%
Frosted Lens	160 mins	339	80.2%	2.1%

The blobs of 13,223 people were collected for machine-learning technique. Table 4 shows the results before and after applying machine-learning technique. We use 10-fold cross validation to evaluate the results. We define several common metrics. Define Precision as the number of occupants classified by the algorithm as either inward or outward that are true positives, Recall as the fraction of the total inward and outward occupants that are correctly detected, and F1-Score as the harmonic mean of Precision and Recall. Machine-learning technique can improve the performance considerably as indicated by AccuracyRate and F1-Score in Table 4.

Table 4: Results of occupancy recognition comparing with and without machine-learning technique.

Type	AccuracyRate	Precision	Recall	F1-Score
Original	63%	0.97	0.27	0.42
With ML	85%	0.69	0.78	0.73

4. Occupancy Prediction

This section describes our methodology for occupancy prediction. Accurate prediction of occupancy patterns is useful for optimal HVAC control. It allows forecast of building usage and pre-cooling before the occupied periods. We propose both general and domain specific approaches of prediction and evaluate their effectiveness.

4.1. General Approach

One simple approach is to take the past occupancy data to be the future occupancy prediction. Another approach is to use the entire occupancy data to train a linear regression model. This approach is denoted by $\widehat{\mathsf{LR}}_{\mathrm{AllData}}$.

A more useful approach is to develop a linear regression model considering special events of unusual occupancy patterns. These special events may be recurrent to a certain degree with slightly flexible timing and durations. Examples of such events are scheduled gatherings in a hall or daily prayers in a mosque. Knowledge about these special events (e.g., timing and durations) can be acquired in a-prior or inferred from the past occupancy data. We denote this special event based approach by $\widehat{\mathsf{LR}}_{\mathrm{SpEv}}$, and the corresponding regression model is given by Eq. 3.

$$\widehat{y}^{(t)} = \beta_0 + \beta_1 x^{(t)}_{\mathrm{offset}} + \beta_2 x^{(t)}_{\mathrm{event}} + \beta_3 x^{(t)}_{\mathrm{dayOfWeek}} \qquad (3)$$

where $\widehat{y}^{(t)}$ is the target variable, which is the predicted occupancy at the time t. β_{i} is a set of unknown coefficients to be determined. $x^{(t)}_{\mathrm{offset}}$ is the time difference between time t and a known special event. $x^{(t)}_{\mathrm{event}}$ is an indicator, taking a value of 1 if time t is the occurrence of a special event, and a value of 0 otherwise, and $x^{(t)}_{\mathrm{dayOfWeek}}$ is the day of the week at time t.

4.2. Domain-specific Approach

Specifically, we apply the above occupancy prediction approach to our testbed in mosque. The occupancy data collected from our testbed (see Section 6) contains occupant count and timestamp. One useful knowledge is the known prayer times throughout the year. In particular, there are five prayer times daily, for which the Muslims will normally pray in the mosque: (1) at dawn, before sunrise; (2) at midday, after the sun passes its highest; (3) at the late part of the afternoon; (4) just after sunset; and (5) between sunset and midnight. The precise prayer times vary over the course of the year, because days tend to be longer in summer and shorter in winter. Note that the Friday prayer (which takes place every Friday at midday) is consisted of a sermon, and it is mandatory for Muslims.

As such, the number of occupants increases significantly compared to any other prayer throughout the week.

We employ a linear regression model that takes the domain-specific knowledge into consideration to improve the prediction accuracy. We consider the following elements: (i) the offset in minutes from the first prayer of the day, (ii) the difference in minutes between the current time and the nearest prayer time, (iii) whether is weekend or weekdays, and (iv) the day of the week. To predict occupancy for the given time $\bar{t}$, we consider historical data for which the timestamp is within certain threshold from from time $\bar{t}$. The threshold varies according to time of the year due to the shifting prayer time.

We extract the above elements from the historical occupancy data, and use them to obtain a linear regression model by standard regression analysis, which is denoted by $\widehat{\mathsf{LR}}_{\text{DomSp}}$ and given by Eq. 4.

$$\widehat{y}^{(t)} = \beta_0+\beta_1 x^{(t)}_{\text{offset}}+\beta_2 x^{(t)}_{\text{nearestPrayer}}+\beta_3 x^{(t)}_{\text{weekend}}+\beta_4 x^{(t)}_{\text{dayOfWeek}} \tag{4}$$

where $x^{(t)}_{\text{offset}}$ is the offset in minutes from the first prayer till time t and $x^{(t)}_{\text{nearestPrayer}}$ is the difference in minutes between time t and the nearest prayer. The other variables are similar to those in Eq. 3.

In addition to the linear regression models, we also train a polynomial regression model (denoted by $\widehat{\mathsf{PR}}_{\text{DomSp}}$) using the same elements used by $\widehat{\mathsf{LR}}_{\text{DomSp}}$.

4.3. Evaluation Results

We use two metrics to evaluate the accuracy of the proposed prediction models: R-square and RMSE (root mean squared error). RMSE measures the spread of prediction error. It is between 0 and 1, where the value 0 implies a perfect match between the predicted and actual data. If RMSE is 0, then R-square will be 1.

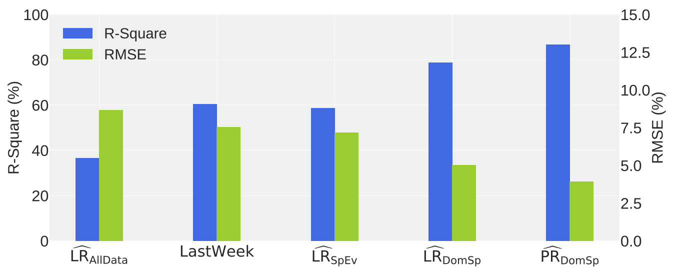


Figure 7: Results of occupancy prediction

The results are shown in Figure 7. A horizon of one week is used for prediction in the evaluation. For LastWeek, we take the data from the previous week, and assume that it will be similar to that of this week. For $\widehat{\mathsf{LR}}_{\text{AllData}}$, we take all the available training data to obtain the regression model. For the rest of approaches, the data from only the last 30 days was used to obtain the regression models, because it is observed to give better results than using the entire data. Typically, we ignore the prediction errors that lie within $10\% \pm 2$ of the actual values. This is because approximate values of occupancy suffice in the context of HVAC control.

While all three approaches are able to predict the overall occupancy trends, there are certain differences among them. First, $\widehat{\mathsf{LR}}_{\text{AllData}}$ can predict the overall trends, but it always gives occupied status because it often predicts non-zero number of occupants. The approach using the last week's data shows comparatively better accuracy, but it will be affected by any special circumstances not encountered before. We observe that the domain-specific approaches (i.e., $\widehat{\mathsf{LR}}_{\text{DomSp}}$ and $\widehat{\mathsf{PR}}_{\text{DomSp}}$) outperform the general approach $\widehat{\mathsf{LR}}_{\text{SpEv}}$, because they exploit domain-specific knowledge (e.g., by taking the shifting prayer times into consideration). Also, $\widehat{\mathsf{LR}}_{\text{DomSp}}$ performs relatively, despite being much simpler than $\widehat{\mathsf{PR}}_{\text{DomSp}}$. In summary, our prediction approaches are able to predict the overall trends and occupancy status with good accuracy.

5. Simulation-guided Model Predictive Control

Our system relies on simulation-guided model predictive control for automatic HVAC control, for which a virtual building model is needed to accurately simulate the thermal response of the building and its energy consumption. Traditional building modeling and control research is often based on simplified mathematical building models because of tractability and ease of use. However, simplified models, such as first-principle linear models, can capture only limited aspects of the dynamic nature of buildings and systems. On the contrary, sophisticated building simulation software uses more realistic physical models, which can provide accurate modeling to the thermal behavior of buildings. For example, *EnergyPlus*, a popular building energy simulation program, can create detailed building envelope and system models. EnergyPlus can perform extensive conductivity analysis using a layer-by-layer approach. It can consider local weather conditions by incorporating user-supplied weather information or EnergyPlus standard weather dataset [35]. It also provides well-defined interfaces, with which external tools/programs can interoperate for co-simulation. Co-simulation is a concept that integrates different subsystems for simulation and calibration simultaneously. In co-simulation, multiple simulators and software tools are coupled, where the strengths of each tool are exploited to overcome their individual weaknesses.

Currently, EnergyPlus is commonly used for planning and energy auditing. Using EnergyPlus for real-time HVAC control presents both opportunities and challenges. In this paper, we utilize a framework of co-simulation to adapt EnergyPlus for HVAC control.

5.1. Basic Building Geometry

First, the basic geometry of building is created for EnergyPlus. We consider the testbed of a large indoor space in a mosque of the size of 18m×18m×7m. The building is installed with packaged rooftop HVAC units with ceiling-based cool air distribution. We employ *SketchUp* for 3D modeling, and basic material properties for walls, windows and doors. Then, the 3D model is imported to *OpenStudio*, where the parameters of HVAC system are incorporated to the model, based on vendor's specifications and datasheet. The SketchUp building model and the corresponding OpenStudio model are visualized in Figure 8. Note that apart from the basic geometry and properties of building, many parameters are required to be calibrated according to the measured data.

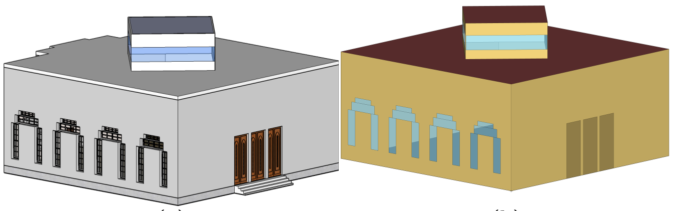


Figure 8: Basic building models. (a) 3D *SketchUp* building model, (b) the corresponding *OpenStudio* model.

5.2. Building Model Calibration

To attain accurate simulation of the thermal response of building, the parameters of building model needs to calibrated to minimize the gap between the simulated indoor temperature and the measured temperature. We employ inverse calibration approach, where the model outputs are used to calibrate the model parameters. The calibration is performed using a co-simulation framework shown in Figure 9. The framework couples the EnergyPlus building model with the calibration algorithm implemented in Python through BCVTB (Building Controls Virtual Test Bed) software tool [36], which provides data exchange interface between EnergyPlus and Python. BCVTB allows EnergyPlus to incorporate real-world data measured from our testbed (e.g., observed occupancy and temperature).

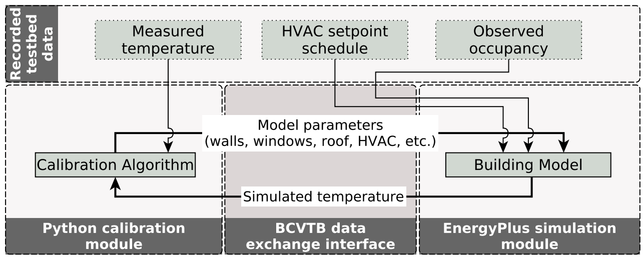


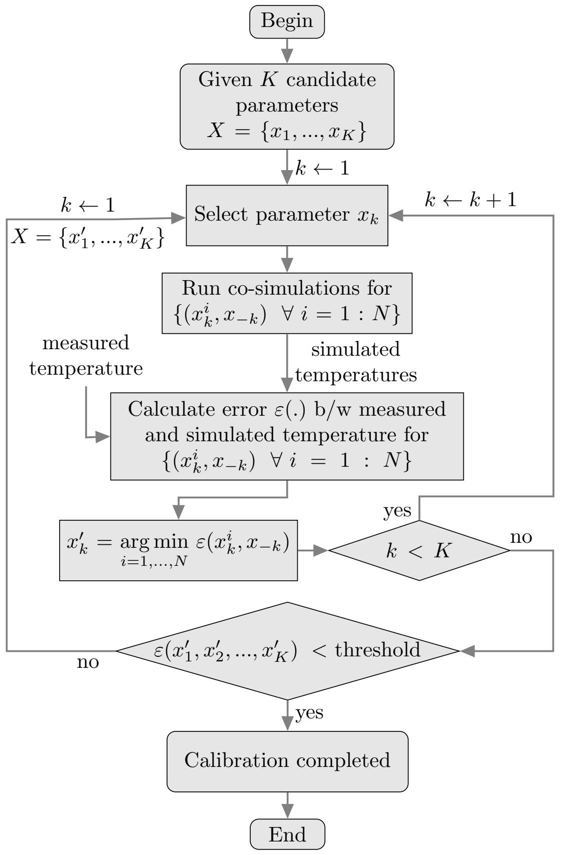
Figure 9: Co-simulation framework using BCVTB.

While there are a large number of possible parameters in EnergyPlus, we identify a set of uncertain parameters (e.g., thickness and conductivity of wall, windows, and roof, cooling capacity and air flow rate of HVAC) as candidates for calibration, because they are either unknown or likely to deviate from datasheet. The EnergyPlus parameters for model calibration are listed in Table 5.

The flowchart of the model calibration algorithm is presented in Figure 10. The algorithm is based on the notion of gradient descent, by which one starts with the initial set of parameters estimates and iteratively adjusts parameter values to minimize the error between the model output and the measured data. Similarly, the model calibration algorithm iteratively updates the EnergyPlus parameter values until the error between the simulated indoor temperature and the measured temperature is within an acceptable threshold.

Table 5: EnergyPlus parameters for model calibration.

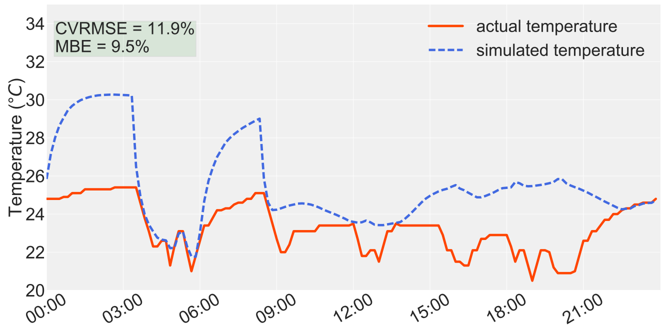
Parameter	Field		Initial Value	Calibrated Value
Exterior Walls	Outer Layer (Stucco)	Thickness	2.53cm	4.81cm
		Conductivity	0.69W/(m-K)	0.069W/(m-K)
	Layer 2 (Concrete)	Thickness	20.32cm	38.61cm
		Conductivity	1.31W/(m-K)	0.131W/(m-K)
	Layer 3 (Gypsum)	Thickness	1.27cm	2.413cm
		Conductivity	0.16W/(m-K)	0.016W/(m-K)
HVAC	Cooling Capacity		87920W	157500W
	Air Flow Rate		3.78m^3/sec	8.505m^3/sec



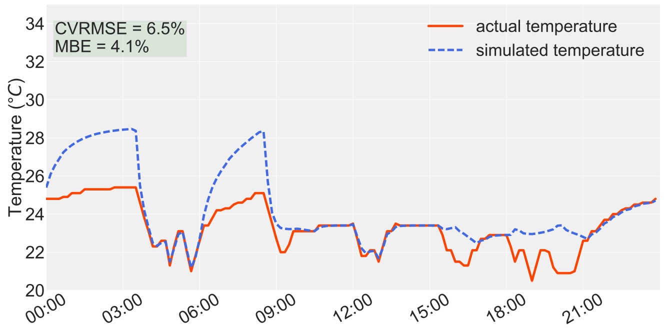
Begin
Given K candidate parameters $X = \{x_1, ..., x_K\}$
$k \leftarrow 1$
$k \leftarrow 1$
$X = \{x'_1, ..., x'_K\}$
Select parameter x_k
$k \leftarrow k+1$
Run co-simulations for $\{(x^i_k, x_{-k}) \;\; \forall\, i = 1:N\}$
measured temperature
simulated temperatures
Calculate error $\varepsilon(.)$ b/w measured and simulated temperature for $\{(x^i_k, x_{-k}) \;\; \forall\, i = 1:N\}$
$x'_k = \arg\min_{i=1,...,N} \varepsilon(x^i_k, x_{-k})$
yes
$k < K$
no
no
$\varepsilon(x'_1, x'_2, ..., x'_K) < \text{threshold}$
yes
Calibration completed
End

Figure 10: Flowchart of model calibration algorithm.

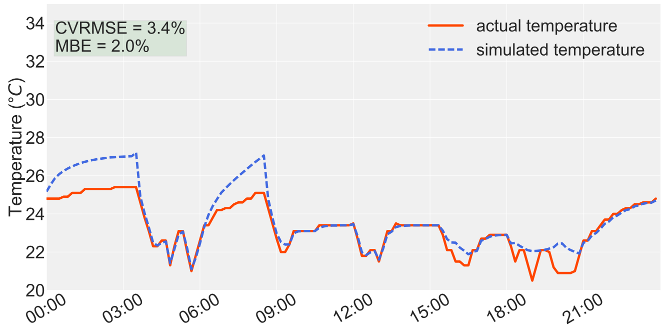
First, the algorithm chooses a set of K parameters $\{x_1, x_2, ..., x_K\}$ as candidates for adjustments. Then, it selects the first parameter x_1 and runs a series of N co-simulations, updating the model in every iteration with a different value of x_1 in a small feasible range, while keeping the values of the remaining parameters ($x_{-1} \triangleq \{x_2, ..., x_K\}$) unchanged. For each iteration (i.e., for each value of x_1), the algorithm calculates the error between the measured temperature and the simulated model tem-



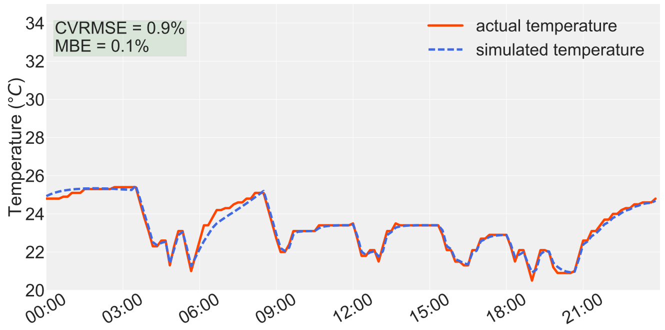
(a) **Wall thickness**(cm):[layer1 = 2.53, layer2 = 20.32, layer3 = 1.27], **Wall conductivity**($\frac{W}{m-K}$):[layer1 = 0.69, layer2 = 1.31, layer3 = 0.16], **HVAC**:[capacity = 87.9kW, air flow rate = 3.78m^3/sec]



(b) **Wall thickness**(cm):[layer1 = 3.79, layer2 = 30.48, layer3 = 1.91], **Wall conductivity**($\frac{W}{m-K}$):[layer1 = 0.35, layer2 = 0.65, layer3 = 0.08], **HVAC**:[capacity = 126.0kW, air flow rate = 6.80m^3/sec]



(c) **Wall thickness**(cm):[layer1 = 4.31, layer2 = 34.54, layer3 = 2.16], **Wall conductivity**($\frac{W}{m-K}$):[layer1 = 0.21, layer2 = 0.39, layer3 = 0.05], **HVAC**:[capacity = 136.5kW, air flow rate = 7.37m^3/sec]



(d) **Wall thickness**(cm):[layer1 = 4.81, layer2 = 38.61, layer3 = 2.41], **Wall conductivity**($\frac{W}{m-K}$):[layer1 = 0.07, layer2 = 0.13, layer3 = 0.02], **HVAC**:[capacity = 157.5kW, air flow rate = 8.51m^3/sec]

Figure 11: Results of model calibration.

perature, as described by Eq. 5:

$$\varepsilon(x_1^1, x_2, ..., x_K), \dots, \varepsilon(x_1^N, x_2, ..., x_K) \tag{5}$$

where $\varepsilon(.)$ is the error between the measured temperature and the simulated model temperature for the specific combinations of parameter values. The optimal value for x_1, denoted by x_1', is the one that minimizes the error:

$$x_1' = \underset{i=1,\dots,N}{\arg\min} \; \varepsilon(x_1^i, x_2, ..., x_K) \tag{6}$$

After identifying x_1' as the optimal value for x_1, the algorithm repeats the procedure for the second parameter x_2. Once again, the algorithm runs N number of co-simulations, where the value of x_2 is adjusted in a small feasible range, while other parameters ($x_{-2} \triangleq \{x_1, x_3, ..., x_K\}$) remain unchanged. The optimal value for x_2, denoted by x_2', is the one that minimizes the error:

$$x_2' = \underset{i=1,\dots,N}{\arg\min} \; \varepsilon(x_1, x_2^i, ..., x_K) \tag{7}$$

The algorithm repeats the procedure for other parameters. Next, we update the new parameters to be $(x_1', x_2', ..., x_K')$. This process goes on iteratively until the error $\varepsilon(x_1', x_2', ..., x_K')$ is within an acceptable threshold.

We define two metrics of calibration errors: Coefficient of Variation of Root Mean Squared Error (CVRMSE) and Mean Bias Error (MBE), as given by Eqs. 8-9:

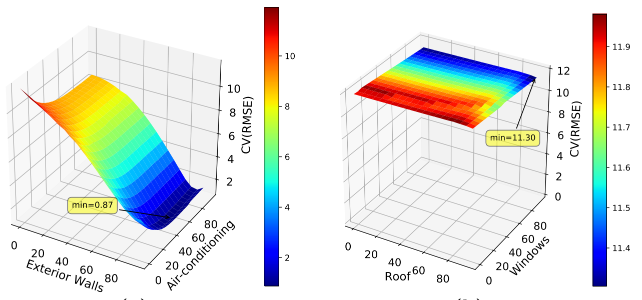
$$\text{CVRMSE} = \frac{\sqrt{\sum_{i=1}^{N}\left[\frac{(T_\text{i}-\hat{T}_\text{i})^2}{N}\right]}}{\frac{1}{N}\sum_{i=1}^{N} T_\text{i}} \tag{8}$$

$$\text{MBE} = \frac{\sum_{i=1}^{N}(T_\text{i}-\hat{T}_\text{i})}{\sum_{i=1}^{N} T_\text{i}} \tag{9}$$

where T_i and $\hat{T}_\text{i}$ are the measured and simulated temperature values at time i, and N is the total number of simulation time steps. As per ASHRAE guidelines, MBE and CVRMSE for a calibrated model must be less than 10% and 30% respectively for hourly data [37]. Similarly, the acceptance criteria for monthly data are MBE<5% and CVRMSE<15%. The model calibration algorithm presented here is able to achieve MBE<1% and CVRMSE<1% for sub-hourly data with sampling time period of 10 minutes over 24 hours. We do not consider longer periods because 24 hours look-ahead horizon is sufficient to realize optimal HVAC control. Both MBE and CVRMSE provide insights to the calibration of EnergyPlus parameters. Note that CVRMSE is a superior metric because unlike MBE it does not suffer from cancellation effect [38].

The calibration results are visualized in Figure 11, where each sub-figure compares the measured and simulated temperature during different stages of the model calibration algorithm. The behavior of the initial parameter values is shown in Figure 11a, while that of the final parameter values in Figure 11d. It is observed that the simulated temperature of calibrated EnergyPlus model closely matches the measured temperature. The calibrated EnergyPlus model was then evaluated on various subsets of collected data on a different day. For each subset of data, the simulated temperature produced by the calibrated EnergyPlus model always closely matches the corresponding measured temperature.

The initial and calibrated values of EnergyPlus parameters for calibration are listed in Table 5, while Figure 12 visualizes the effectiveness of these parameters with the remaining two parameters (i.e., roof and windows). The x-axis and y-axis represent the percent change in parameters relative to their original values, and z-axis represents the corresponding CVRMSE index. Comparing the minimum CVRMSE in Figure 12a and Figure 12b, it is evident that the model calibration algorithm is able to achieve a significantly lower CVRMSE when using the selected parameters for calibration.



(a) **(b)**

Figure 12: Examples of effectiveness of adjusting the parameter to reduce calibration error.

The model calibration algorithm can be implemented efficiently on Raspberry Pi. It is observed that Raspberry Pi 3 takes about 45sec on average per simulation. Hence, the model calibration algorithm algorithm can be executed on a regular basis to calibrate EnergyPlus simulator in dynamic environment. In summary, our calibration methodology can automatically correct the building model continuously for effective real-time HVAC control.

5.3. Model Predictive Control Algorithm

Using the building model simulated in EnergyPlus, we design a real-time control algorithm based on the forecast from the building model. Figure 13 illustrates the framework of model predictive control.

The model predictive control algorithm (Algorithm 1) is implemented in *Python* programming language, and provides HVAC setpoint schedule to the EnergyPlus building model through the BCVTB layer. EnergyPlus carries out simulations based on the input HVAC setpoint schedules and occupancy prediction, and returns the simulated indoor temperature to the model predictive control algorithm. Then, the model predictive control algorithm adjusts the HVAC setpoint schedule subject to acceptable thermal comfort when the controlled area is occupied, and the simulation is carried out with the adjusted schedule. This process goes on iteratively until the optimal HVAC setpoint schedule is determined. Note that real-time measured temperature, humidity, and occupancy data is utilized by the algorithm to calculate the actual thermal comfort caused by its control decisions.

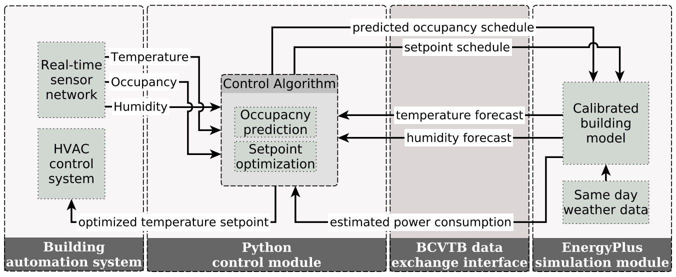


Figure 13: Framework of model predictive control algorithm.

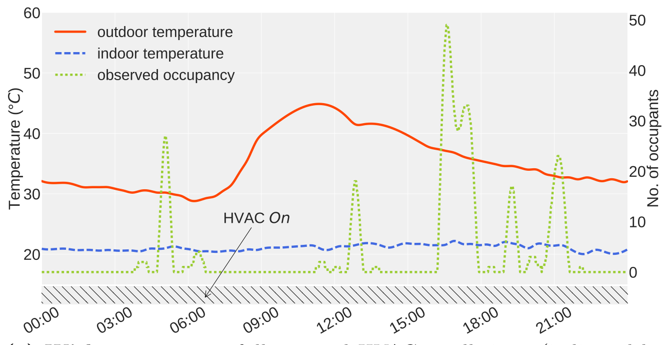
Algorithm 1 presents the HVAC model predictive control algorithm (HVAC-MPC). If the building becomes unoccupied at current time t_{now}, HVAC-MPC instructs the HVAC system to increase the temperature setpoint (subject to satisfying the feasible temperature range). The temperature setpoints used by HVAC-MPC are shown in Table 6. HVAC-MPC then looks at the occupancy prediction to find out when the building is likely to be occupied in the future and decides the optimal time for pre-cooling. The procedure of determining pre-cooling time in HVAC-MPC is given in Figure. 14.

Table 6: Setpoints used by HVAC-MPC.

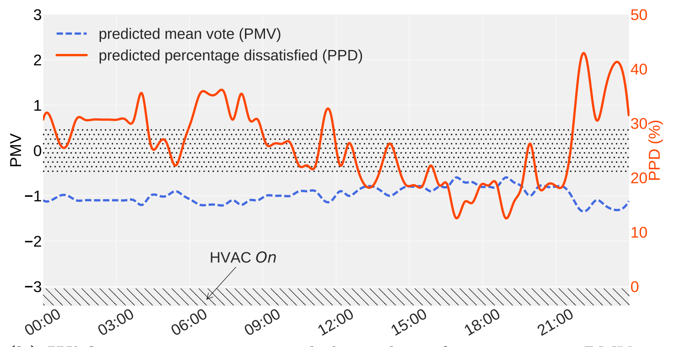
Occupied ($\mathsf{setpoint}_{\text{oc}}$)	Unoccupied ($\mathsf{setpoint}_{\text{uo}}$)
24°C	28°C

```
Algorithm 1: HVAC-MPC
   Input: $t_{\text{now}}$
   Initialization: $\tau \leftarrow 0$, timer $\leftarrow 0$
1  if $\mathsf{occup}_{\text{now}} = 0$ and timer $= 0$ then
2      $T_{\text{target}} \leftarrow \mathsf{setpoint}_{\text{uo}}$
3      Set HVAC setpoint to $T_{\text{target}}$          ▷increase setpoint
4      $\tau \leftarrow$ Pre-cooling($t_{\text{now}}, t_{\text{pd\_oc}}$)       ▷pre-cooling time
5      timer $\leftarrow t_{\text{now}} + \tau$
6  end
7  if $t_{\text{now}}$ = timer then
8      $T_{\text{target}} \leftarrow \mathsf{setpoint}_{\text{oc}}$
9      Set HVAC setpoint to $T_{\text{target}}$          ▷start pre-cooling
10     timer $\leftarrow$ 0
11 else
12     Wait for timer expiry                    ▷no pre-cooling yet
13 end
```

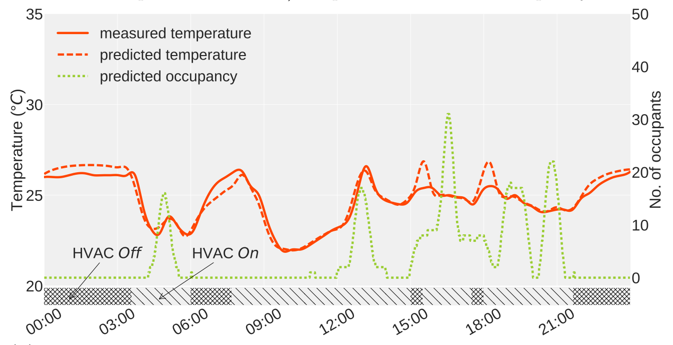
In Figure. 14, we employ binary search to decide the optimal time for pre-cooling. Binary search is an efficient way to find a target value in a sorted search space. In this case, the target value is the time at which pre-cooling should be started. Initially, the search space is the entire time sequence from current time (t_{now}) to time of predicted occupancy ($t_{\text{pd\_oc}}$). At each iteration, it runs a simulation in EnergyPlus, such that HVAC is off from t_{now} to the median time in the search space t_{mid}, and pre-cooling is started (i.e., HVAC is on) from t_{mid} to $t_{\text{pd\_oc}}$. Using the simulated temperature T_{sim}, it finds the exact time when the temperature drops from temperature $T_{\text{sim}}[t_{\text{mid}}]$ to target temperature $\mathsf{setpoint}_{\text{oc}}$. If the temperature reaches



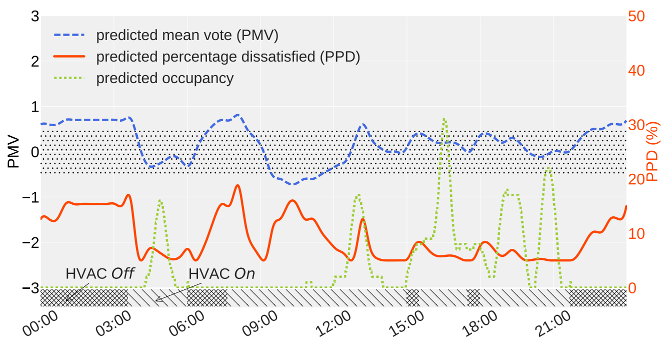
(a) Without `HVAC-MPC`: full-powered HVAC at all times (indicated by the hatched strip at the bottom) despite intermittent occupancy.



(b) Without `HVAC-MPC`: incurred thermal comfort sensation. PMV index is always outside the recommended [-0.5,+0.5] range

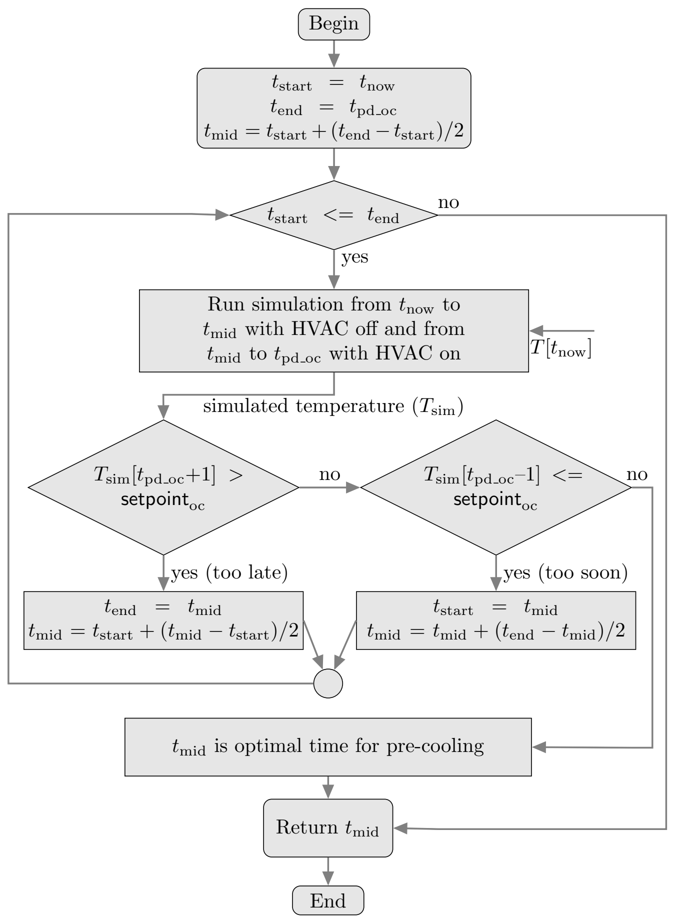


(c) With `HVAC-MPC`: occupancy and temperature forecasts are used to find the optimal pre-cooling schedule as indicated by the sequence of HVAC state transitions from *off* to *on* in the hatched strip.



(d) With `HVAC-MPC`: PMV index is almost always within the recommended [-0.5,+0.5] range during predicted occupied periods.

Figure 15: Results of HVAC model predictive control (`HVAC-MPC`) in our testbed.



Begin
$t_{\text{start}} = t_{\text{now}}$
$t_{\text{end}} = t_{\text{pd\_oc}}$
$t_{\text{mid}} = t_{\text{start}} + (t_{\text{end}} - t_{\text{start}})/2$
$t_{\text{start}} <= t_{\text{end}}$
no
yes
Run simulation from t_{now} to t_{mid} with HVAC off and from t_{mid} to $t_{\text{pd\_oc}}$ with HVAC on
$T[t_{\text{now}}]$
simulated temperature (T_{sim})
$T_{\text{sim}}[t_{\text{pd\_oc}}+1] >$ setpoint$_{\text{oc}}$
no
$T_{\text{sim}}[t_{\text{pd\_oc}}-1] <=$ setpoint$_{\text{oc}}$
no
yes (too late)
yes (too soon)
$t_{\text{end}} = t_{\text{mid}}$
$t_{\text{mid}} = t_{\text{start}} + (t_{\text{mid}} - t_{\text{start}})/2$
$t_{\text{start}} = t_{\text{mid}}$
$t_{\text{mid}} = t_{\text{mid}} + (t_{\text{end}} - t_{\text{mid}})/2$
t_{mid} is optimal time for pre-cooling
Return t_{mid}
End

Figure 14: Flowchart of determining pre-cooling time.

setpoint$_{\text{oc}}$ before $t_{\text{pd\_oc}}$ then it is too soon to start pre-cooling, which eliminates the first half of the search space and the process is repeated. Similarly, if the temperature can not reach setpoint$_{\text{oc}}$ by $t_{\text{pd\_oc}}$ then it is too late to start pre-cooling at time t_{mid}, which eliminates the second half of the search space and the process is repeated. By repeatedly eliminating half of the search space, it will eventually find the optimal time for pre-cooling such that the temperature reaches the desired thermal setpoint setpoint$_{\text{oc}}$ just in time when the control area is being occupied.

5.4. Evaluation Results

The results of `HVAC-MPC` algorithm are illustrated in Figure 15. The HVAC operations with `HVAC-MPC` deactivated are shown in Figures 15a and 15b. The default HVAC operations is to run full-powered at all times even though the building is occupied only intermittently (see Figure 15a). This inevitably results in unnecessary energy consumption as well as thermal discomfort. Figure 15b shows the resulting thermal comfort sensation as PMV and PPD indices. Predicted Mean Vote (PMV) refers to the human perception of thermal sensation on a scale that runs from -3 to +3, where -3 is very cold, 0 is neutral, and +3 is very hot [39]. The recommended PMV range for thermal comfort, which is highlighted as the dotted pattern in Figure 15b and 15d, is between -0.5 and +0.5 for indoor spaces [40]. It is observed that the PMV index is always outside the recommended range when `HVAC-MPC` is not activated. Figure 15b also provides PPD (Predicted Percentage Dissatisfied) metric that describes the percentage of occupants that are not satisfied with the current comfort requirements. It is observed that a significant percentage is dissatisfied at any given time when the HVAC is operated without using `HVAC-MPC`.

Figure 15c illustrates the results with `HVAC-MPC` activated. Whenever the building becomes unoccupied, `HVAC-MPC` increases the temperature setpoint and makes occupancy and temperature forecasts to find the optimal time for pre-cooling. The hatched strip at the bottom of Figure 15c indicates the start time of each pre-cooling (i.e., HVAC state transition from *Off* to *On*). The thermal comfort results are shown in Figure 15d. It is observed that the PMV index is mostly within the recommended range from -0.5 to +0.5 whenever the building is occupied. The PPD index is also much lower than before, keeping less than 10% during occupied periods. It is worth mentioning that PPD can not be less than 5% even if PMV index is zero [41]. `HVAC-MPC` is also able to significantly reduce the HVAC energy consumption as shown in Figure. 16, under three different days. In each case, `HVAC-MPC` is able to reduce consumption by at least 30%.

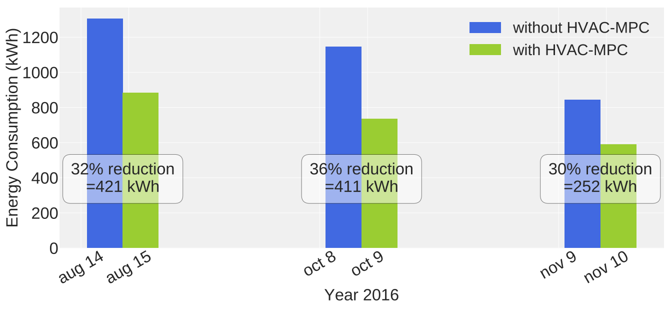


Figure 16: Reduction in HVAC energy consumption achieved by `HVAC-MPC` (Algorithm 1).

6. Testbed Design and Implementation

Our automatic HVAC control system with real-time video-based occupancy recognition and EnergyPlus-guided model predictive control is implemented on Raspberry Pi 3 platform. Raspberry Pi 3 is a low-cost embedded system platform costing as low as $35 per unit, as of 2016. The specifications of Raspberry Pi include 1.2 GHz 64-bit quad-core CPU, GPU, 1 GB SDRAM memory, SD card based extensible external data storage, WiFi/Bluetooth connectivity. This section describes the testbed design and implementation for our automatic HVAC control system.

6.1. System Design

Figure 17 illustrates the overall design of our system. This system can be divided into (1) *building automation system* and (2) *cloud-based management system*.

The building automation system consists of the hardware components installed in the building, including temperature and humidity sensors, occupancy monitors, energy meters, and customized wireless HVAC controllers. The system is designed to be able to process the data locally for intelligent HVAC control.

The temperature and humidity sensors are based on *EnOcean* technology which are self-powered and free of maintenance. These sensors transmit the measured data via 868 MHz radio link to a base station implemented by a Raspberry Pi. The occupancy monitor is a dedicated Raspberry Pi with a fish-eye camera for real-time tracking of occupancy. The occupancy data is then transmitted over *WiFi* network to the base station. An Arduino based energy meter is used to measure the energy consumption of the HVAC units and transmit the data to the base station.

A data daemon running on the base station interfaces with the above components. The data daemon receives the data and stores it in a local database residing on base station Raspberry Pi. The data daemon also provides environmental and occupancy data to the control daemon, which processes the data to make HVAC control decisions. The control daemon interfaces with the HVAC system through wireless HVAC controllers implemented in a Raspberry Pi. Both daemons run as immortal services, such that they can recover from temporary connection disruptions or system restart due to power outage. The basestation module is shown in Figure 18d.

The cloud-based management system comprises of the remote server and the web graphical user interface (GUI). The remote server receives the collected data and stores it in a server database for later analysis and permanent storage. The web GUI retrieves data from the server database for visualization and analysis. The web GUI also provides remote access to building control system. Secure Shell (SSH) protocol is used as the underlying protocol to securely provide remote access.

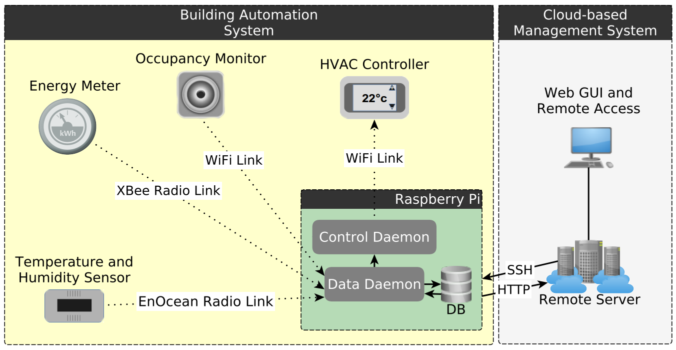


Figure 17: Outline of our system of *building control system* and *cloud-based management system*.

6.2. Occupancy Detection Module

Our system gathers detailed occupancy information to determine building usage patterns and the level of occupancy. Building occupancy is monitored using a Raspberry Pi with a fish-eye camera. A video processing algorithm is running on the Raspberry Pi that analyzes real-time video stream, captured by the fish-eye camera, in order to count the number of occupants in the building and time of occupancy. Figure 18a shows the occupancy

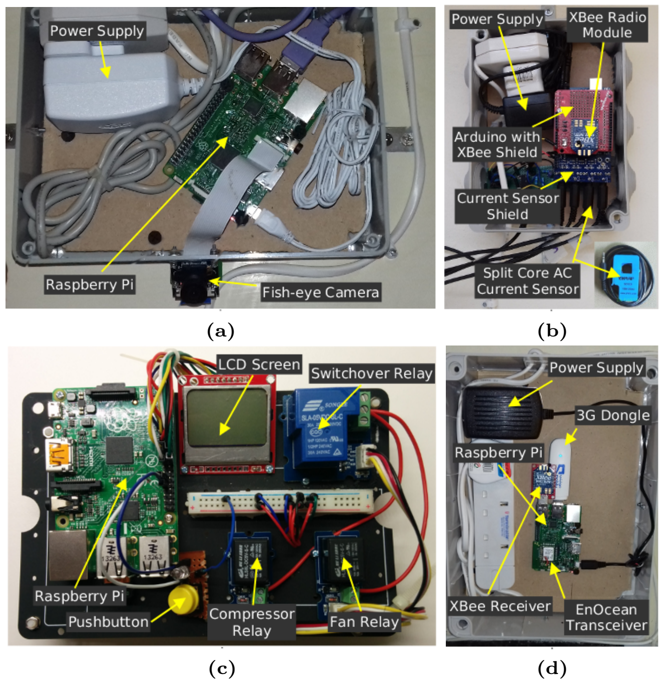


Figure 18: Testbed system components: (a) occupancy detection module, (b) HVAC energy measurement module, (c) HVAC controller, (d) base station module.

module developed for this research. The module should be installed above the building entrance door so that it can monitor the occupancy inside the building. Multiple modules may be needed if the building has multiple entry/exit points, where each module covers a single point.

6.3. HVAC Energy Measurement Module

We measure HVAC energy consumption as a fraction of total energy used by the building to quantify the energy savings achieved by our HVAC control algorithm over other control strategies. To collect HVAC energy consumption data, we developed a module shown in Figure 18b. The module design is adapted from *OpenEnergyMonitor* framework, an open source project for developing energy monitoring and analysis tools [42].

The Arduino based module uses non-invasive alternating current (AC) sensors to measure the HVAC energy consumption. These sensors can be clipped onto either the live or ground wire coming into the HVAC unit without needing to strip the wire thus avoiding any high-voltage work. We use a separate sensor for each HVAC unit installed in the building. An Arduino gathers the HVAC energy consumption data from the sensors and sends it to the base station through an XBee radio link. It was decided to use XBee radio technology because the HVAC unit power supply can be in a separate room or even on the roof. XBee modules with their extended transmission range can penetrate concrete walls and roofs and are able to transmit the sensor readings to the base station.

6.4. HVAC Controller

Figure 18c shows wireless HVAC controller implemented in Raspberry Pi. The module has three relays, two of which are used to open and close the compressor and fan circuits of the HVAC unit. The third relay (labeled as *Switchover Relay* in Figure 18c) is used to switch control between our wireless controller and the default HVAC thermostat controller, allowing the building occupants to disable automatic HVAC control if they are not satisfied with the current thermal comfort level or if the system is malfunctioning. The push-button can be used to turn the third relay On or Off. The module is also equipped with an LCD display to provide status information.

7. Conclusion

This paper presents an automatic occupancy-predictive HVAC control system, featuring real-time occupancy recognition, dynamic occupancy prediction, and simulation-guided model predictive control, implemented in low-cost embedded system (Raspberry Pi). We deployed and evaluated our system for providing automatic HVAC control in a large public indoor space of a mosque. We observe the real-time occupancy recognition system can reach 90% accuracy, whereas the prediction system achieves 85% accuracy to forecast the occupancy patterns in a mosque. We employ real-time HVAC control guided by on-board EnergyPlus simulator, after adaptive calibration, which shows more than 30% energy saving, while maintaining the comfort level within acceptable range. Remarkably, our system is sufficiently general that can also be deployed in other types of buildings with large public indoor spaces.

Notably, we ported EnergyPlus simulator to the Raspberry Pi embedded system platform. We release our Raspberry Pi version of EnergyPlus publicly [13] to enable other researchers to take advantage of our work for future building automation projects.

Our testbed has been evaluated in buildings with predictable regular occupancy patterns (e.g., mosques). It is challenging to be applied to the settings with irregular occupancy patterns. In future work, we will explore robust online HVAC control using minimal occupancy prediction. Online algorithms have been applied to wireless sensor based building control [43, 44]. Robust online control can ensure good performance in the presence of dynamic irregular environmental factors.

References

[1] L. Prez-Lombard, J. Ortiz, C. Pout, A review on buildings energy consumption information, Energy and Buildings 40 (3) (2008) 394 – 398.

[2] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, M. Morari, Use of model predictive control and weather forecasts for energy efficient building climate control, Energy and Buildings 45 (2012) 15 – 27.

[3] J. lvarez, J. Redondo, E. Camponogara, J. Normey-Rico, M. Berenguel, P. Ortigosa, Optimizing building comfort temperature regulation via model predictive control, Energy and Buildings 57 (2013) 361 – 372.

[4] S. Salakij, N. Yu, S. Paolucci, P. Antsaklis, Model-based predictive control for building energy management. i: Energy modeling and optimal control, Energy and Buildings 133 (2016) 345 – 358.

[5] Y. Kwak, J.-H. Huh, C. Jang, Development of a model predictive control framework through real-time building energy management system data, Applied Energy 155 (2015) 1 – 13.

[6] P.-D. Moroan, R. Bourdais, D. Dumur, J. Buisson, Building temperature regulation using a distributed model predictive control, Energy and Buildings 42 (9) (2010) 1445 – 1452.

[7] F. Ascione, N. Bianco, C. D. Stasio, G. M. Mauro, G. P. Vanoli, Simulation-based model predictive control by the multi-objective optimization of building energy performance and thermal comfort, Energy and Buildings 111 (2016) 131 – 144.

[8] H. Park, M. Ruellan, A. Bouvet, E. Monmasson, R. Bennacer, Thermal parameter identification of simplified building model with electric appliance, in: 11th International Conference on Electrical Power Quality and Utilisation, 2011, pp. 1–6.

[9] R. Blan, J. Cooper, K.-M. Chao, S. Stan, R. Donca, Parameter identification and model based predictive control of temperature inside a house, Energy and Buildings 43 (23) (2011) 748 – 758.

[10] J. Hu, P. Karava, A state-space modeling approach and multi-level optimization algorithm for predictive control of multi-zone buildings with mixed-mode cooling, Building and Environment 80 (2014) 259 – 273.

[11] Raspberry pi foundation, retrieved on 31-Jan-2017.
URL `https://www.raspberrypi.org`

[12] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. Buhl, Y. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, J. Glazer, Energyplus: creating a new-generation building energy simulation program, Energy and Buildings 33 (4) (2001) 319 – 331, special Issue: {BUILDING} SIMULATION'99.

[13] `https://dl.dropboxusercontent.com/u/44094567/eplus.zip`, retrieved on 31-Jan-2017.

[14] `https://www.awqaf.gov.ae/Affair.aspx?SectionID=3&RefID=18`, retrieved on 20-Jan-2017.

[15] `http://www.moia.gov.sa/menu/pages/statistics.aspx`, retrieved on 20-Jan-2017.

[16] D. T. Delaney, G. M. O'Hare, A. G. Ruzzelli, Evaluation of energy-efficiency in lighting systems using sensor networks, in: Proceedings of ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, ACM, 2009, pp. 61–66.

[17] A. Barbato, L. Borsani, A. Capone, S. Melzi, Home energy saving through a user profiling system based on wireless sensors, in: Proceedings of ACM workshop on embedded sensing systems for energy-efficiency in buildings, ACM, 2009, pp. 49–54.

[18] R. S. Hsiao, D. B. Lin, H. P. Lin, S. C. Cheng, C. H. Chung, A robust occupancy-based building lighting framework using wireless sensor networks, in: Applied Mechanics and Materials, Vol. 284, Trans Tech Publ, 2013, pp. 2015–2020.

[19] J. Li, L. Huang, C. Liu, Robust people counting in video surveillance: Dataset and system, in: Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on, IEEE, 2011, pp. 54–59.

[20] L. Chen, F. Chen, X. Guan, A video-based indoor occupant detection and localization algorithm for smart buildings, in: Emerging Intelligent Computing Technology and Applications, Springer, 2009, pp. 565–573.

[21] M. Jia, R. S. Srinivasan, Occupant behavior modeling for smart buildings: A critical review of data acquisition technologies and modeling methodologies, in: Winter Simulation Conference (WSC), 2015, IEEE, 2015, pp. 3345–3355.

[22] W. Kleiminger, F. Mattern, S. Santini, Predicting household occupancy for smart heating control: A comparative performance analysis of state-of-the-art approaches, Energy and Buildings 85 (2014) 493–505.

[23] P. Ferreira, A. Ruano, S. Silva, E. Conceio, Neural networks based predictive control for thermal comfort and energy savings in public buildings, Energy and Buildings 55 (2012) 238 – 251, cool Roofs, Cool Pavements, Cool Cities, and Cool World.

[24] Identification of nonlinear systems using narmax model, Nonlinear Analysis: Theory, Methods & Applications 71 (12) (2009) e1198 – e1202.

[25] D. B. Crawley, J. W. Hand, M. Kummert, B. T. Griffith, Contrasting the capabilities of building energy performance simulation programs, Building and Environment 43 (4) (2008) 661 – 673, part Special: Building Performance Simulation.

[26] J. Zhao, K. P. Lam, B. E. Ydstie, V. Loftness, Occupant-oriented mixed-mode energyplus predictive control simulation, Energy and Buildings 117 (2016) 362 – 371.

[27] D. Coakley, P. Raftery, M. Keane, A review of methods to match building energy simulation models to measured data, Renewable and Sustainable Energy Reviews 37 (2014) 123 – 141.

[28] G. Bradski, Dr. Dobb's Journal of Software Tools.

[29] openFrameworks Community, openframeworks.

[30] Z. Zivkovic, F. van der Heijden, Efficient adaptive density estimation per image pixel for the task of background subtraction, Pattern recognition letters 27 (7) (2006) 773–780.

[31] Z. Zivkovic, Improved adaptive gaussian mixture model for background subtraction, in: Proceedings of IEEE International Conference on Pattern Recognition, Vol. 2, IEEE, 2004, pp. 28–31.

[32] S. Suzuki, K. Abe, Topological structural analysis of digitized binary images by border following, CVGIP 30 (1) (1985) 32–46.

[33] N. Halko, P.-G. Martinsson, J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM review 53 (2) (2011) 217–288.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[35] Energyplus standard weather data, retrieved on 08-Feb-2017.
URL `https://energyplus.net/weather`

[36] M. Wetter, Co-simulation of building energy and control systems with the building controls virtual test bed, Journal of Building Performance Simulation 4 (3) (2011) 185–203.

[37] A. Guideline, Ansi/ashrae standard 14-2014, Measurement of energy, demand, and water savings.

[38] M. Royapoor, T. Roskilly, Building model calibration using energy and environmental data, Energy and Buildings 94 (2015) 109 – 120.

[39] P. O. Fanger, et al., Thermal comfort. analysis and applications in environmental engineering., Thermal comfort. Analysis and applications in environmental engineering.

[40] A. Standard, Ansi/ashrae standard 55-2004, Thermal Environmental Conditions for Human Occupancy.

[41] J. Cigler, S. Prvara, Z. Va, E. ekov, L. Ferkl, Optimization of predicted mean vote index within model predictive control framework: Computationally tractable solution, Energy and Buildings 52 (2012) 39 – 49.

[42] Open energy monitor – a project to develop and build open source energy monitoring and analysis tools, retrieved on 07-Feb-2017.
URL `https://openenergymonitor.org/`

[43] C.-K. Chau, M. Khonji, M. Aftab, Online algorithms for information aggregation from distributed and correlated sources, IEEE/ACM Transactions on Networking 24 (6) (2016) 3714–3725.

[44] M. Aftab, C.-K. Chau, P. Armstrong, Smart air-conditioning control by wireless sensors: An online optimization approach, in: Proceedings of ACM International Conference on Future Energy Systems (e-Energy), 2013, pp. 225–236.