SPOT: A Smart Personalized Office Thermal Control System

by

Xiang Gao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2013

Abstract

Heating, Ventilation, and Air Conditioning (HVAC) accounts for about half of the energy consumption in buildings. HVAC energy consumption can be reduced by changing the indoor air temperature setpoint, but changing the setpoint too aggressively can overly reduce user comfort. We have therefore designed and implemented SPOT: a Smart Personalized Office Thermal control system that balances energy conservation with personal thermal comfort in an office environment. SPOT relies on a new model for personal thermal comfort that we call the Predicted Personal Vote model. This model quantitatively predicts human comfort based on a set of underlying measurable environmental and personal parameters. SPOT uses a set of sensors, including a Microsoft Kinect, to measure the parameters underlying the PPV model, then controls heating and cooling elements to dynamically adjust indoor temperature to maintain comfort. Based on a deployment of SPOT in a real office environment, we find that SPOT can accurately maintain personal comfort despite environmental fluctuations and allows a worker to balance personal comfort with energy use.

Acknowledgements

I would like to thank all the people who helped me on my thesis. I would like to thank my advisor Prof. S. Keshav, who always gives me useful life and academic advices. I would like to thank Prof. Bernard Wong and Prof. Lukasz Golab for being my thesis reader. I would also like to thank Prof. S. Keshav, Prof. George Labahn, Prof. Peter Forsyth, Prof. Bernard Wong, Prof. Reid Holmes for participating the experiment. Many thanks to Gail Chopiak, Ronaldo Garcia, and Lawrence E Folland for helping me get the equipments and components for SPOT. Thanks all my lab mates and friends from other labs for helping me on training and testing the clothing estimation component of SPOT.

Dedication

This is dedicated to my parents for their endless love.

This is also dedicated to my supervisor Prof. S. Keshav for his constant support and encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

About 30% to 50% of the residential and commercial energy consumption in most developed countries is used by Heating Ventilation, and Air Conditioning (HVAC) systems [4, 31, 27, 7]. Therefore, the energy footprint of commercial buildings can be reduced by improving the efficiency of the HVAC systems. For most office environments in commercial buildings, temperature is controlled by a centralized HVAC system, resulting in a constant temperature setpoint between different zones of a building. The ASHRAE [2] standard suggests to maintain the indoor temperature between 20°C to 24°C in winter, and 23°C to 26°C in summer.

By setting the temperature at the suggested value, an 'average person' feels comfortable in the indoor environment. However, people have different thermal preferences. A fixed temperature temperature setpoint can not satisfy all the people working in the building. We therefore suggest personalized thermal control, an control strategy that takes personal thermal preference into account. Assuming that workers in an office have work areas that are thermally isolated from each other, such as separate office rooms or cubicles, we aim to control the temperatures of the work areas such that every worker in the building feels comfortable.

We suggest the overall building temperature level be set to a value lower than normal in winter and higher than normal in summer. This setting will obviously make most of the workers in the building feel uncomfortable. We then set up a personalized thermal controller in each work area that could provide an offset to the base temperature. The amount of temperature offset is based on the personal thermal preference of the worker. For example, most commercial buildings today are heated to 23°C in winter. We instead suggest to heat the building only to 20°C, and equip each work area with a small computer-controlled radiant heater to provide the temperature offset. The work areas will be heated by the radiant heater to a higher level that meets the personal thermal preference of the worker.[1] In summer, symmetrically, a small fan or an air-conditioned fan can provide extra cooling below the building-wide setpoint of 26°C [34]. By using the personalized thermal control system, individual worker's thermal comfort can be maintained.

Previous researches have shown that human thermal comfort is not only a function of room temperature, other factors may be involved as well. Two people who are differently dressed may experience different levels of comfort in the same room environment. An ideal HVAC control system should control the room temperature to achieve a particular human thermal comfort level, rather than a fixed temperature setpoint. This motivates us to design SPOT: a Smart Personalized Office Thermal control system.

SPOT uses an ensemble of sensors to measure the six parameters that have been found to contribute to human comfort: air temperature, radiant temperature, humidity, air speed, clothing level, and activity level. This lets it compute human comfort according to the ISO 7730 standard called the Predicted Mean Vote (PMV) model [5]. We have extended this model to allow per-user personalization; we call our personalized model the Predicted Personal Vote (PPV) model. SPOT uses the PPV model to maintain a desired comfort level despite environmental fluctuations. We have deployed SPOT and evaluated its performance in a realistic office environment to control the heating in winter.

Our work makes it possible to trade off a decrease in human comfort for a reduction in energy usage. We model the environmental characteristics by using Learnig-Based Model Predictive Control (LBMPC). LBMPC automatically learns the thermal insulation and heat capacity of the work area and builds a model for accurate predictive control. The model can predict future room temperature given the current power of the heater as an input. We use optimal control framework to find the trade-off between human comfort and energy saving. With occupancy prediction, optimal control can find the best time to turn on/off the heater. For example, the heater can be turned on 10 minutes prior to the arrival of the worker such that the worker feels comfortable when he or she arrives. Symmetrically, the heater can be turn off earlier to the predicted departure time of the worker to save energy.

The major contributions of this thesis are:

- We extend the ISO 7730 standard [5] to define the PPV model for user comfort and use it to design SPOT, an HVAC control system that maintains user comfort, rather than merely air temperature

[1] Compare with manual control or a heating system with motion sensor.

- We have implemented SPOT and deployed it in a realistic environment. We find that SPOT can accurately maintain personal comfort despite environmental fluctuations and allows a user to balance personal comfort with energy use.

- We model the thermal environment using Learning-Based Model Predictive Control (LBMPC) and use optimal control framework to find desired trade-off between energy-saving and user comfort.

Chapter 2

Background

HVAC control systems traditionally put user comfort first, expending energy freely to achieve a given setpoint. ‘Dumb’ thermostats use the same setpoint all day, and smarter, programmable thermostats allow users to vary setpoints by time of day and day of week. Some thermostats allow remote control. For example, in Ontario, the PeakSaver [1] thermostat responds to an emergency broadcast radio signal and increases the cooling set point by up to two degrees, thereby reducing home electricity usage by up to 37%. Other ‘smart’ thermostats, such as the Nest [11], learn user occupancy patterns to intelligently control HVAC usage by means of proprietary algorithms. Nevertheless, none of these thermostats are aware of user comfort: they focus, instead, only on controlling room temperature.

2.1 ASHRAE Standard

The ASHRAE [2] Standard 55 evaluates the comfort level of the thermal environment in a 7-point scale (Table 2.1). According to the table, people give positive votes in warm environments and negative votes in cool environments. A vote of zero means the person feels comfortable.

Using ASHRAE Standard, people in the same thermal environment, such as a building, can vote based on their feelings. The average vote among all the people in that building is used for evaluating the thermal comfort. A average vote between -0.5 and 0.5 is usually considered as the acceptable range.

Although ASHRAE Standard helps building operators find the average comfort of the occupants, it is impractical to ask all the people in a building to vote on the temperature.

A more practical method is to model people's average comfort, such that the average vote can be predicted by variables such as temperature, air flow speed, and clothing. This is done by Fanger's PMV model [21], which will be discussed in the following section.

Vote	**Comfort Level**
+3	Hot
+2	Warm
+1	Slightly Warm
0	Neutral
-1	Slightly Cool
-2	Cool
-3	Cold

Table 2.1: 7-point ASHRAE scale

2.2 Predicted Mean Vote

The basis of our work is a quantitative model for human comfort called the PMV model that is defined in the ISO 7730 Standard [5]. The PMV model computes a numerical comfort level, called a *vote*, that describes the degree of comfort of a typical person in a moderate thermal environment. The PMV model predicts human comfort as a function of four environmental variables (air temperature, radiant temperature, air speed, and humidity) and two personal variables (clothing and physical activity). Given these variables, it predicts the mean value of a group of people's votes in a 7-point ASHRAE [2] thermal sensation scale.

The PMV model was first proposed by Fanger [21] in 1970 and it is widely used for evaluating thermal comfort. Although the model is based on a theoretically well-grounded physical thermal balance model, it has been found to be problematic to use in practice [24]. Many variations of the PMV model have been developed to fix these problems. For example, De Dear et. al. [17] developed a model to capture the sociological and geographical factors that may affect human's thermal preference, such as people living in warmer areas preferring warmer indoor temperature than people living in cooler areas. Similarly, Nicol et. at. [30] have shown that people can use physiological and psychological adaptations to be comfortable in a wider range of temperatures than supposed by the PMV model; their model reflects this observation.

Although these newer models improve the accuracy of the PMV model, they all predict the average thermal comfort of a large group of people. However, in a micro-climate such as an office work area, comfort is usually relevant only for one person or a small number of people. This motivates the design of a *personalized* thermal comfort model. In our work, we extend the PMV model to the Predicted Personal Vote (PPV) model to capture individual thermal preference. We use PPV model to automatically adjust an HVAC control system's temperature setpoint so that a worker always feel comfortable.

PMV model assigns a numerical comfort value $pmv(\mathbf{x})$ based on a vector $\mathbf{x}$ with six elements

$$\mathbf{x} = \{t_a, \bar{t_r}, v_{ar}, p_a, M, I_{cl}\}^\top$$

- t_a is the air temperature
- $\bar{t_r}$ is the mean background radiant temperature
- v_{ar} is the air velocity
- p_a is the humidity level
- M is the metabolic rate of a person
- I_{cl} is the clothing insulation factor of a person

We can evaluate PMV using the function:

$$pmv = f_{pmv}(\mathbf{x}) \tag{2.1}$$

The details of the function can be evaluated in practise is in the appendix (§A).

Chapter 3

Design

We now describe our design in more detail. Recall that SPOT's goal is to maintain a particular comfort level (PPV value) based on sensor measurements and its control over the operation of a small personal radiant heater or fan. We assume that the control plant is a room or a cubicle in a building owned by an office worker.

In order to evaluate the user's thermal preference, we first extend the PMV model to Predicted Personal Vote (PPV) model, as described in §3.1. The PPV model first requires a training stage, during which the office worker can give votes to the office temperature. With the training data, SPOT learns the thermal preference of the office worker. It then controls the room temperature as an agent of the office worker.

Note that human thermal comfort is not only depended on indoor air temperature. It is actually a function of four environmental variables and two personal variables. For example, two people wearing different cloth can experience differently in the same environment. In §3.2, we propose to use infrared sensors to estimate the clothing level of the office worker. We assume the worker's body temperature is maintained at a relatively constant level. The worker's cloth will attenuate the infrared emitted by his or her body. By measuring the attenuation, we estimate the clothing level of the user.

SPOT uses Learning-Based Model Predictive Control (LBMPC) to keep the room temperature at the desired level. Instead of changing the control input based on the control output, LBMPC builds a model of the control plant and predicts the control output given any control input. In §3.3, we explained the details of using LBMPC to control the office room temperature. At the training stage, SPOT learns office room's heat capacity and thermal leakage rate. After several hours training, SPOT can predict future room temperature given the current room temperature and heater power.

With occupancy prediction, as described in §3.4, we can use LBMPC to perform control action in advance to achieve a control objective. For example, SPOT can turn on the heater before the predicted arrival time of the office worker such that the room feels comfortable when he or she arrives. We use optimal control as a principled framework to make such control decision. The details of optimal control is described in §3.5.

3.1 Predicted Personal Vote Model

The PMV model reflects the thermal comfort of a large group of people. However, individual workers may have their own thermal preference. We have, therefore, modified the PMV model to create a model we call the Predicted Personal Vote (PPV) model. Like PMV model, PPV also maps thermal comfort into the 7-point ASHRAE scale.

For each person, the Predicted Personal Vote function has two parts, the PMV part and the personal part:

$$ppv(\mathbf{x}) = pmv(\mathbf{x}) + personal(\mathbf{x}) \tag{3.1}$$

where $pmv(\mathbf{x})$ is the output of the PMV model and $personal(\mathbf{x})$ models how the current user is different from an average person. We model $personal(\mathbf{x})$ as a linear function:

$$personal(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + b \tag{3.2}$$

where $\mathbf{a}$ is a vector of size 6 that models the users sensitivity to each variable.

$$\mathbf{a} = \{a_{temp}, a_{radiant}, a_{velocity}, a_{humidity}, a_{metabolic}, a_{clothing}\}^\top \tag{3.3}$$

. For example, a person who is more sensitive to humidity than average will have a relatively large $a_{humidity}$ value. Variable b denotes the thermal preference of the user. A person prefers warmer temperatures will have negative b value and vice versa.

Using the PPV model requires a training phase. In the training phase, SPOT measures the environmental variables $\mathbf{x}$ and also records the worker's personal vote *apv*. Suppose we have a training set $\{(\mathbf{x}_k, apv_k)\}_{k=1}^{K}$ of size K, where apv_k is the k-th actual personal vote, and $\mathbf{x}_k$ is the vector of environmental and personal variables when the user gives the k-th vote. This allows us to estimate parameters $\mathbf{a}$ and b using straightforward linear regression. In the absence of a training set, SPOT simply reverts to the PMV model. Similarly, when there are not enough data points to do a linear regression for Equation 3.2, we train a simpler linear function $g(\cdot)$ to estimate PPV:

$$ppv(\mathbf{x}) = g(pmv(\mathbf{x})) \tag{3.4}$$

The function $g(\cdot)$ is trained by least square regression.

3.2 Clothing Level Estimation

Five out of the six underlying parameters of the PPV model can be measured in a relatively straightforward manner using appropriate sensors (this is discussed in more detail in §4.2). However, measuring the 'clothing level' parameter is non-trivial (see Table 12), and the focus of this subsection.

The key idea behind our approach to clothing level estimation is the fact that most humans have a relatively constant skin temperature of about 34°C. The greater the level of clothing worn, the greater the degree of insulation, and the lower the temperature of the outermost layer of clothes. Thus, the clothing level can be estimated by measuring the temperature of the clothing using an infrared sensor as discussed in §4.2.2.

Specifically, we build a linear regression model to estimate the clothing level I_{cl} as:

$$I_{cl} = f(t_{cir}) \tag{3.5}$$

where $f(\cdot)$ is a linear function and t_{ir} is the clothing surface temperature measured by the infrared sensor. We fit the function $f(\cdot)$ using least square linear regression. The model is trained using a data set of I_{cl} estimates from Table A.2 and t_{cir} measured by the infrared sensor.

In practise, multiple problems could affect the accuracy of the simplest model appear in Equation 3.5. For example, it is impractical to assume that all people have the same and constant body temperature. In fact, human body temperature fluctuates during the day. Therefore, we also record the office worker's body surface temperature t_{fir} as a referencing point. We measure the body surface temperature from the face of the office worker. We will discuss the details of using Kinect and movable infrared sensor to measure clothing surface temperature and skin surface temperature in §4.

The infrared intensity measured by the movable infrared sensor is also attenuated with distance. Using sensors with smaller detection angle could solve this problem. We solve this problem by incorporating the sensor-to-worker distance into the model, such that the attenuation is compensated by the measured distance. The value of sensor-to-worker distance can be calculated by the data streamed from Kinect.

In practise, we are building a linear model with three inputs:

$$I_{cl} = f(t_{cir}, t_{fir}, dist) \tag{3.6}$$

The function $f(\cdot)$ is fitted using least square method.

3.3 Learning-Based Model Predictive Control

Traditional feedback control determines the control input based on the control output (feedback). For example, if the room temperature is lower then the setpoint, the heater will be turned on. In SPOT, we use Learning-Based Model Predictive Control (LBMPC) to model the thermal characteristics of a house, and use this model to make control decisions.

With this predictive control model, we can predict the outcome of a control action and do optimization with this knowledge. For example, we will be able to know that if we adjust the heater power to 800W, we can increase home temperature from 22°C to 23°C in 20 minutes. We can also use the inverse function to determine the appropriate heater power to increase the room temperature to a certain point.

The mathematical formulation of room thermal model is like the following. Given the outside temperature T_{out} and the indoor temperature T_{in}, by the Newton's Law of Cooling, the rate of thermal energy loss is proportional to the temperature difference. We use the following differential equation to model it:

$$P_{loss} = k(T_{in} - T_{out}) \tag{3.7}$$

where Q is the thermal energy of the house, and P_{loss} is the thermal energy loss rate. Model parameter k is the conduction factor of the house. A house with better insulation will have a smaller conduction factor.

To control the indoor temperature, a HVAC system with power P_{hvac} is used in the house. Let e denotes the efficiency of the HVAC system. The net heat input rate is:

$$P = eP_{hvac} - P_{loss} = eP_{hvac} - k(T_{in} - T_{out}) \tag{3.8}$$

The net heat input rate is the differentiation of thermal energy ($P = \frac{dQ}{dt}$), which is proportional to temperature change:

$$P = \frac{dQ}{dt} = C\frac{dT_{in}}{dt} \tag{3.9}$$

Model parameter C is the heat capacity of the house. Combining Eq. 3.8 and Eq. 3.9, we have:

$$\frac{dT_{in}}{dt} = \frac{eP_{hvac} - k(T_{in} - T_{out})}{C} \tag{3.10}$$

To enable digital control, we convert Eq. 3.10 to its discrete version:

$$T_{in}(s+1) = T_{in}(s) + \frac{eP_{hvac}(s) - k(T_{in}(s) - T_{out}(s))}{C} \tag{3.11}$$

where $T_{in}(s)$ is the temperature at the s-th timestep.

The model contains three parameters: the efficiency of the HVAC system e, the conduction factor k, and the house heat capacity C. Given tuples of $\{T_{in}(s), T_{out}(s), P_{hvac}(s), T_{in}(s+1)\}$ at different timesteps, the model parameters can be estimated by regression method. In the experiment, we use least square regression to find the parameters of house model. Note that in winter and summer, the house model is different, so we need separate models for controlling heating in winter and cooling in summer.

3.4 Occupancy Prediction

Temperature control usually has a delay. Therefore, if room occupancy could be predicted, we can use LBMPC to take the control action in advance to compensate the delay.

The prediction essentially finds similar previous days in the occupancy database and predicts based on those records in the database. Similar method is validated in PreHeat [33] so we believe the method can yield relatively accurate prediction.

We predict room occupancy using the K-nearest neighbour method. The system records the occupancy of the room for each half an hour timeslot. Let t be the identifier of that timeslot and m_t be the occupancy of timeslot t. For example, we can define timeslot t as the t-th half an hour timeslot after Jan 1st, 1970. Variable $m_t = 1$ if the room is occupied and $m_t = 0$ otherwise. Let the current time be t, we predict the future using past i timeslots' occupancy:

$$m_{t+j} = f_j(m_t, m_{t-1}, ..., m_{t-i+1}) \tag{3.12}$$

where $f_j(\cdot)$ is the j step ahead prediction function. We define a helper function $TOD(t)$ which returns the timeslot of the day. For example, $TOD(t) = 0$ means timeslot t is the timeslot between mid-night and 0:30 a.m. of a day.

Given the current timeslot t, we search in our database for all timeslots $s \in S$ such that $TOD(s) = TOD(t)$ and compare their similarity. The similarity of two timeslots is defined as :

$$sim(s,t) = \sum_{b=0}^{i} \delta_{s-b,t-b} \tag{3.13}$$

where $\delta_{s-b,t-b} = 1$ if $m(s-b) = m(t-b)$ and $\delta_{s-b,t-b} = 1$ otherwise. We select the K timeslots with highest similarity to t in S and we denote them as $s_k \in S_K$. To do j-th step ahead occupancy prediction, we calculate the occupancy probability using the following formula:

$$p(t+j) = \frac{1}{K}\sum_{k=1}^{K} m(s_k + j) \tag{3.14}$$

If the occupancy probability $p(t+j)$ is larger than 0.5, we predict that timeslot $t+j$ will be occupied.

3.5 Optimal Control Strategy

We use a principled optimal control framework to find out the optimal control decision at each timestep. Optimal control decides the time to turn on the heater before the predicted arrival time of the worker, and the time to turn off the heater before the predicted leaving time of the worker. Further, it provides a trade-off between the human comfort and energy saving. An energy-aware office worker can slightly reduce his or her comfort level in order to save energy.

The goal of the optimal HVAC control is to find the optimal operating temperature sequence over an optimization horizon that minimizes the total energy use and guarantees PPV is close to 0. For example, the office worker usually arrives her office at 9am. The system needs to decide when to start heating the office. Turning the heater on too early will be a waste of energy while turning it on too late will make the home owner uncomfortable. With the knowledge from PPV and LBMPC, the optimal controller can heat up the house just in time to guarantee user comfort.

Suppose we are optimizing over a horizon of S timesteps and let $\mathbf{x}(s)$ be the environmental and personal variables at time step s. Note that $x_{air}(s)$ and $T_{in}(s)$ are the same thing, both denotes the indoor air temperature at time step s. We also define the occupancy indicator $m(s)$ which is the predicted occupancy ($m(s) = 1$ if occupied, 0 otherwise). The optimal control sequence can be obtained by solving the following problem.

$$min \sum_{s=1}^{S} P_{hvac}(s) + \lambda \sum_{s=1}^{S} m(s)(\beta_c(s) + \beta_h(s)) \tag{3.15}$$

where λ is the weight on thermal comfort. To guarantee comfort, we usually set λ to be a value much larger than the maximum power of the HVAC system. The optimal control sequence has the following sets of soft constraints: PPV should be within [-ϵ, ϵ] if house is occupied.

$$\begin{aligned}\forall s, ppv(\mathbf{x}(s)) &\geq -\epsilon - \beta_c(s)\\ \forall s, ppv(\mathbf{x}(s)) &\leq \epsilon + \beta_h(s)\\ \forall s, \beta_c(s) &\geq 0\\ \forall s, \beta_h(s) &\geq 0\end{aligned}$$

where $ppv(\mathbf{x}(s))$ is the predicted personal vote at time s, $\xi_c(s)$ and $\xi_h(s)$ are the cold and hot penalty at time s.

Note that function $ppv(\mathbf{x})$ includes $pmv(\mathbf{x})$, which is a non-obvious function. In fact, solving $ppv(\mathbf{x})$ needs to use an iterative numerical method. Therefore, we convert the problem to a state diagram and use the shortest path algorithm to find the optimal control sequence.

Figure 3.1 shows an example of a state model. For r-th state in each timestep s, we denote it as $N_{s,r}$ and it represents a potential control outcome $\mathbf{x}_{s,r}$. The state is associated with a comfort penalty:

$$\beta(s,r) = \begin{cases} 0, \ if \ |ppv(\mathbf{x}_{s,r})| < \epsilon \\ |ppv(\mathbf{x}_{s,r})| - \epsilon, \ otherwise \end{cases} \tag{3.16}$$

There are edges between any pair of states in layer s and $s+1$. To change from the state $N_{s,r}$ to state $N_{s+1,r'}$, there will be state transition energy cost on the HVAC system $P_{hvac}(N_{s,r}, N_{s+1,r'})$, which can be calculated from Eq. 3.11. The distance between state $N_{s,r}$ and $N_{s+1,r'}$ is the energy cost from state $N_{s,r}$ to state $N_{s+1,r'}$ plus the weighted comfort penalty:

$$d(N_{s,r}, N_{s+1,r'}) = P_{hvac}(N_{s,r}, N_{s+1,r'}) + \lambda m(s+1)\beta(s+1, r') \tag{3.17}$$

For example, if $\lambda = 10k$ and $m(2) = 1$, the distance between $N_{1,1}$ and $N_{2,1}$ is $d(N_{1,1}, N_{2,1}) = 800 + 10000 * 1 * 0.5 = 5800$. The optimal control sequence is the shortest path from state $N_{1,1}$ to the virtual end state after the last step. The optimal control sequence need to be frequently updated in the case that variables in $\mathbf{x}$ has been changed or the predicted occupancy $m(s)$ is different from the actual observation.

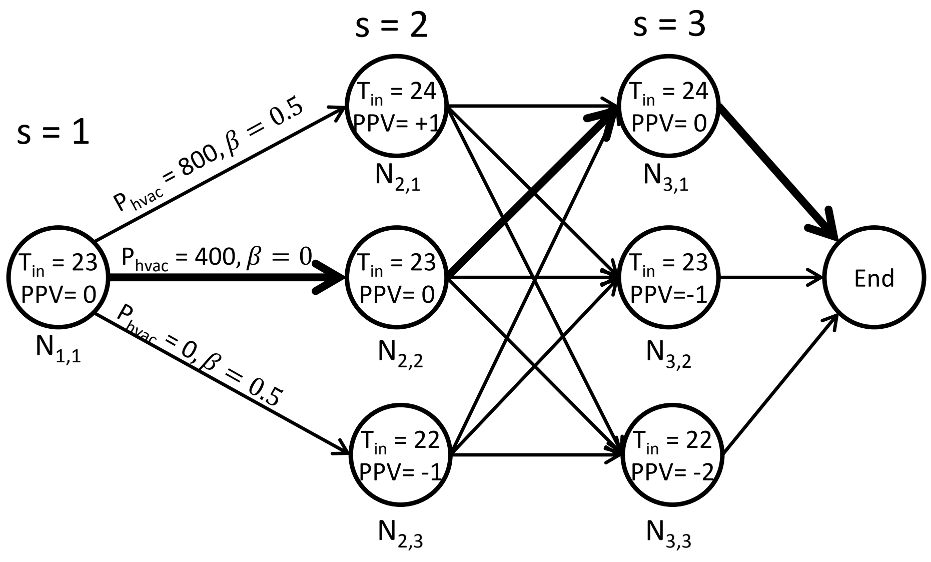

Figure 3.1: We use a state model to find the optimal control sequence. Each state in the state diagram represents a potential control outcome at step s. T_{in} is the potential indoor temperature and PPV is the Predicted Personal Vote given such an indoor temperature. For each edge, P_{hvac} is the state transition energy cost, and β is the comfort penalty of the destination state. The optimal control sequence is the shortest path from the leftmost state to the rightmost state.

Chapter 4

Implementation

We now describe the implementation of our system in greater detail. SPOT has three principal components: controller, sensors and actuators.

4.1 Controller

The SPOT controller is a PC with an Intel i5-3450 processor and 8GB of RAM, running Windows 7 Enterprise edition. The entire project code is written in C#. All sensors and actuators are connected to this PC, and all control logic is implemented on this machine.

4.1.1 Hardware Requirements

Microsoft Kinect has the following minimum system requirement:

- Microsoft Windows 7 or Microsoft Windows Embedded Standard 7
- Dual-core 2.66GHz or faster processor
- Dedicated USB 2.0 bus
- 2GB RAM

The PC needs to communicate with different components of SPOT using USB. In total, at least 4 free USB ports are required for the PC, which are used for connection with

Kinect, WeatherDuck Climate Monitor, Arduino Microprocessor, and Z-Wave Controller. The WeatherDuck Climate Monitor requires a serial to USB converter to communicate with the PC.

4.1.2 Software Prerequisites

SPOT has the following software prerequisites:

- **Visual Studio 2010** [13] **and .Net Framework 4** [12] are the software development environment used for SPOT.
- **Kinect for Windows SDK** [8] is the software requirement for developing Kinect Program. The SDK processes the color image stream, depth image stream, and skeleton stream and passes them to the application developer. The current SDK version is the Kinect for Windows SDK 1.6. However, Kinect for Windows Runtime 1.6 is sufficient if you only want to run SPOT on your machine.
- **Arduino Software** [3] is necessary for SPOT to communicate with the Arduino microcontroller. The Arduino software includes a driver that creates a virtual serial port on the PC and any program can communicate with the Arduino microcontroller using the virtual serial port. Arduino Software also contains a simple IDE for you to write and flush program onto the microcontroller.
- **Gurobi Optimization** [6] is an optimization software package to solve linear programming, quadratic programming and mixed integer programming. We use it for least square regression in SPOT. All the model parameters mentioned in §3.1, §3.2, and §3.3 are estimated using Gurobi.

4.2 Sensors

SPOT uses multiple sensors to measure the environmental variables (air temperature, background radiant temperature, humidity and wind speed) and personal variables (users' clothing level and activity level) that underlie the PPV model. We describe these next.

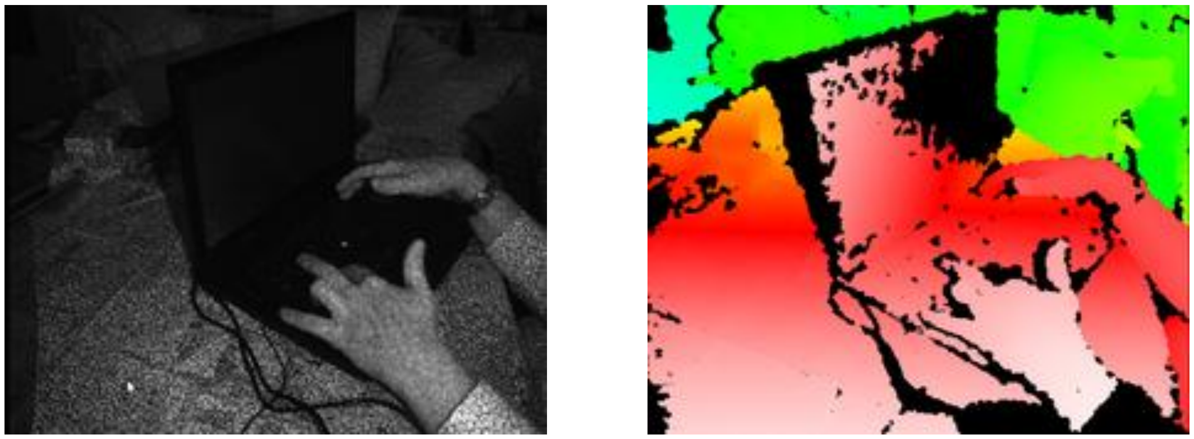

Figure 4.1: Left image is from the camera and right image is the depth image. The red part is closer to the camera.

4.2.1 Microsoft Kinect

A Microsoft Kinect sensor provides 3D information in real-time about the location of humans in a scene. The Kinect sensor was originally designed for the Microsoft Xbox as a natural user interface. By using a Kinect sensor, video game players can interact with Xboxes without actually touching the game controllers.

The Kinect has an RGB camera and an infrared camera. The RGB camera is similar to a normal webcam that captures image from the visible light spectrum. The infrared camera works together with an infrared projector, which emits infrared laser signal with a predefined pattern. The infrared camera collects the reflected laser beams and calculate the distance to the laser point by the time difference between sending and receiving the signal. The Kinect can generate 640×480 resolution depth images with a sensitivity of 1mm. Figure 4.1 shows the raw infrared image and the depth image generated by the Kinect sensor.

By using both the color images generated by the RGB camera and the depth images generated by the infrared camera, the Kinect can build a 3D motion model for the player. When a player enters the frame, the Kinect starts to track the skeleton points of the player, and report the locations of these skeleton points (such as head, hands, knees) as a *skeleton stream*. Gesture-based Xbox applications can use the skeleton stream as user input.

For our research prototype, we use Kinect for Windows, which is a special sensor designed for Windows developers. We implemented our system using Visual Studio 2010 and Kinect for Windows SDK v1.6.

SPOT uses the Kinect for three purposes:

1. It is used as an *occupancy sensor*. When a person is tracked in the skeleton frame, the work space is treated as occupied and the system starts to control the room temperature. We also use the Kinect skeleton tracking APIs to determine the worker's activity level.
2. It is used as a worker *location sensor*. We use the location information to point an infrared thermal sensor mounted on a tracking system at the worker to measure the worker's clothing surface temperature. This allows us to estimate the clothing level (see §3.2).
3. We also use the Kinect to allow workers to *customize their PPV model* parameters using simple gestures. To record a vote, a worker simply points to the Kinect and raises his or her hand in the air to indicate a particular comfort level (the selected comfort level is shown on a screen connected to the Kinect). The system then records one data point of the form $(\mathbf{x}_k, apv_k)$ (see §3.1).

4.2.2 Infrared Thermometer

A MLX90614 Infrared Thermometer (Figure 4.2 upper part) detects background radiant temperature between -40°C to +85°C with a resolution of 0.02°C. It is connected to an Arduino Uno[1] board, which reads the measured radiant temperature and sends the value to a PC via a USB cable every second.

To measure the surface clothing temperature, we mounted two servos[2] and an infrared sensor on top of the Kinect (Figure 4.4). The infrared sensor and a laser pointer are placed on the two servos such that they can face any direction. The laser pointer is used for calibration, and turned off during normal operation. The two servos, the infrared sensor, and the laser pointer are connected to an Arduino micro-controller. The micro-controller sends signals to control the angle of the servos and the on/off state of the laser pointer. The micro-controller is connected to the PC with a USB cable, and it communicates with the PC program using a virtual serial port.

[1]http://arduino.cc

[2]A servo is similar to a stepper motor in that its degree of rotation can be precisely controlled.

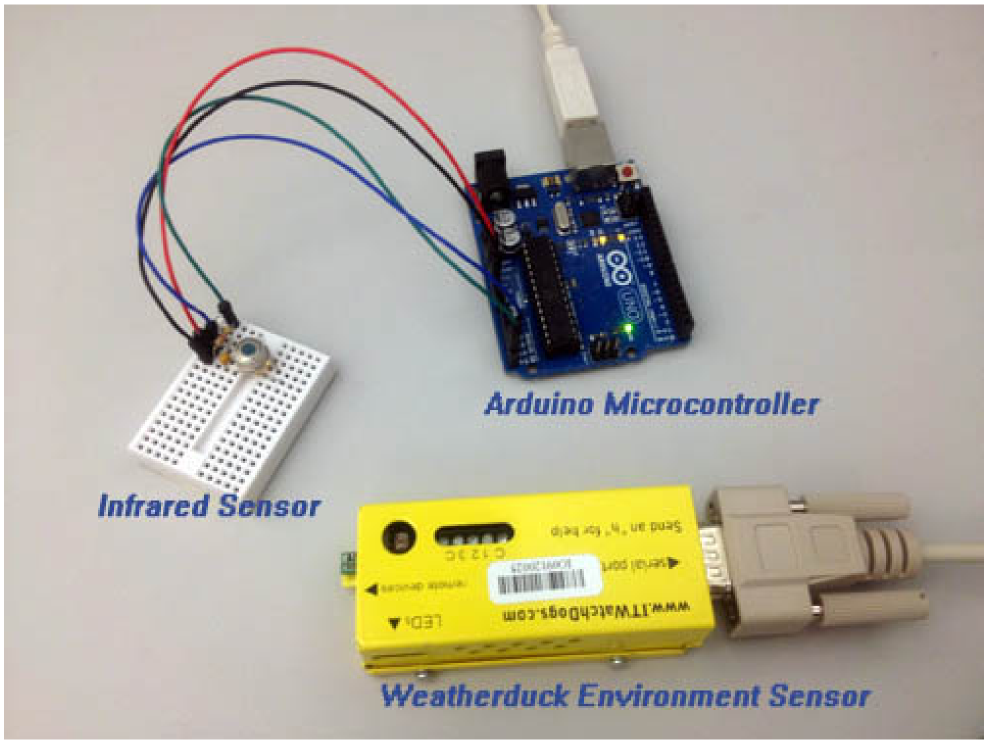

Figure 4.2: Infrared Thermometer and WeatherDuck Climate Monitor

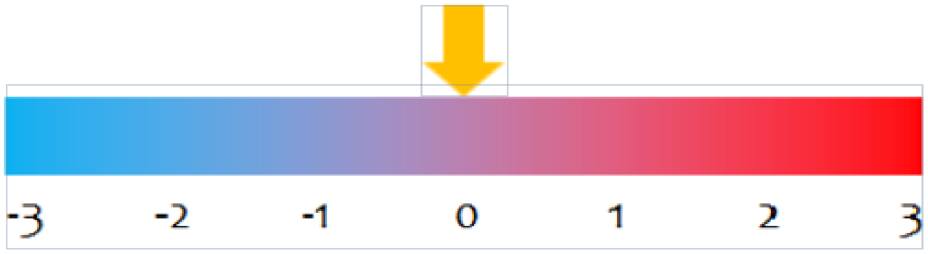

Figure 4.3: User interface for the user to vote on current thermal condition

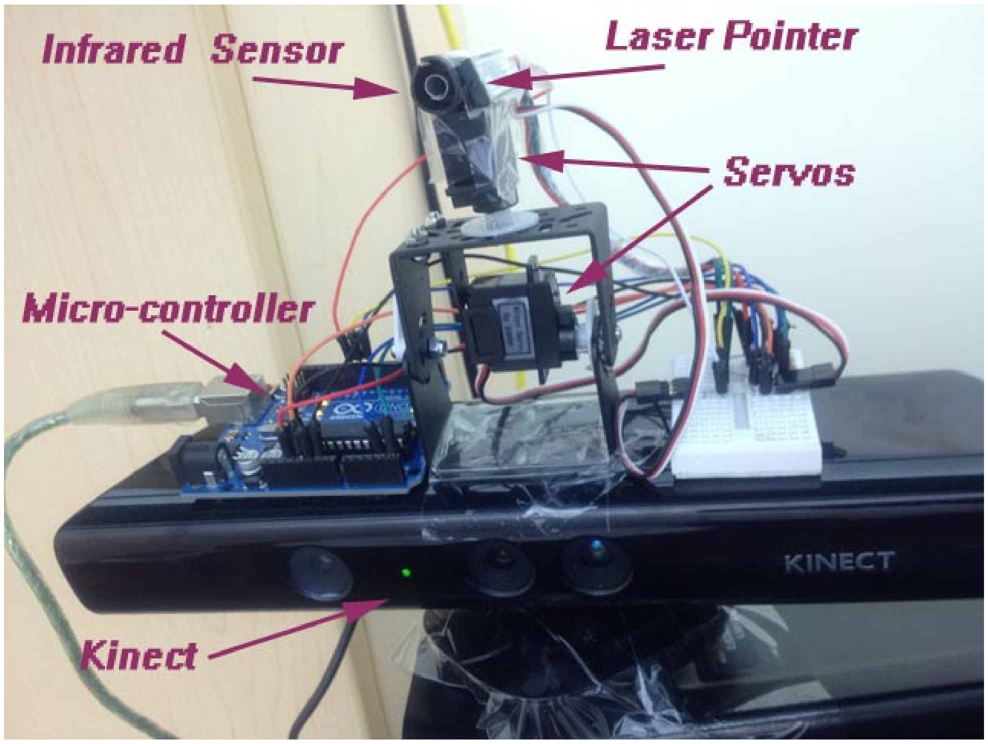

Figure 4.4: Kinect with infrared sensor mounted. The infrared thermal sensor and laser pointer are installed on top of the two servos, which can adjust the rotation angles of the infrared sensor and laser pointer. The micro-controller controls the laser pointer and servos. It also pulls infrared readings from the sensor.

When a worker enters the work space, the Kinect tracks the worker and sends a skeleton stream to the PC. The PC finds the location of the worker's body center and calculates the rotation angle of the servos. It then communicates with the micro-controller to adjust the angles of the two servos so that the infrared sensor is facing the body center. When the tracked worker is moving, the infrared sensor may not be actually facing towards the worker. Therefore, we introduce a 0.5 second measurement delay into the system. That is, the infrared sensor starts collecting data only when the worker has been standing still for at least 0.5s. The system then estimates the clothing insulation by the clothing surface temperature as described in §3.2.

4.2.3 Environment Sensor

SPOT senses environmental variables using the WeatherDuck Climate Monitor [3] (Figure 4.2 lower part), a low-cost sensor that monitors air temperature, humidity and air flow. It can detect air temperature from -10°C to 85°C and relative humidity from 0% - 100%. Its air flow sensor can detect wind speed from 0 to 100 CFM. It also measures the light and sound level of the room as side channels for occupancy detection. The WeatherDuck Climate Monitor is connected to the PC via a serial to USB converter.

4.3 Actuator

We use a remotely controlled heater to adjust the room temperature and maintain a constant comfort level in the office environment.

4.3.1 Heater

SPOT controls a SunBeam SLP3300-CN heater with a maximum power rating of approximately 1350W. The heater is a radiative heater, which does not contain a fan. Since the office room is usually a relatively small, a radiative heater is sufficient to heat the room quickly and evenly. The heater is a pure resistive load and thus we can cut-off the power at any time without affecting the life cycle of the heater.

[3] http://www.itwatchdogs.com

4.3.2 Z-Wave Wireless Controller

The heater is plugged on a Z-Wave smart energy switch, a power plug that is controlled over a Z-Wave wireless network. Z-Wave is specially designed for reliable, low-latency communication of small data packets, which is desirable for home appliance control. Z-Wave devices use command classes to achieve different tasks. In our research prototype, we use a DSC06106 Smart Energy Switch to control and sense the energy consumption of the heater. The Smart Energy Switch is controlled wirelessly by a Silicon Labs CP201s Z-Wave controller, which supports command classes to control the on/off state of a device and measure the energy consumption of that device.

4.4 System Implementation Details

We will discuss the details of implementing SPOT in this section. We first introduce the way to connect the circuits between the sensors, servos and Arduino Microcontroller in §4.4.1. We then describe the method to mount the sensors, servo racks and Arduino on top of the Kinect in §4.4.2. The sensors and servos are controlled by the Arduino, and we discuss the control logic in §4.4.3. To control the heater, we use Z-Wave communication protocol, which is discussed in §4.4.4. We use the Kinect the track the location of the office worker, which is discussed in §4.4.5. The details of the occupancy detection are introduced in §4.4.6.

4.4.1 Connect Sensors and Servos to Arduino Microcontroller

The Arduino Microcontroller controls three modules: the servo module, the IR sensor module and the laser module. The circuit diagram in Figure 4.5 shows the details of the connection. We color code the diagram for ease of understanding.

The servo module is mounted on top of Kinect and it controls the direction of the 5° IR sensor. We will discuss the way to mount the servos in §4.4.2. The module has two servos; one controls the horizontal movement and the other one controls the vertical movement. Each servo has three pins: the positive pin, the negative pin, and the PWM pin. PWM stands for Pulse Width Modulation, which creates pulse signals by turning on and off the power. The width of the pulse expresses the amount of time the power is on. By sending PWM signals, the Arduino microcontroller can precisely control the rotation angles of the servos. We connect the positive pins of both of the servos to the 5V pin on Arduino, and connect the negative pins to the GND pin. The PWM pins of the servos are connected to pin 8 and 9 on the Arduino as depicted in Figure 4.5. Both pin 8 and 9 are will send PWM signals to the servos to control the rotation angles.

The IR sensor module uses an I2C circuit to pull infrared values from the IR sensors. The I2C circuit allows the microcontroller to communicate with multiple devices using the same communication line. In SPOT, we use the MLX90614 sensor for infrared sensing. A 5° sensor is used for sensing the infrared emitted by human body and a 90° sensor is used for sensing the background infrared radiation. Each I2C device has a device ID. The default device ID for MLX90614 is 0x5A. In order for both of the IR sensors work at the same time, we changed the device ID of the 90° IR to 0x55.

The MLX90614 [10] has four pins: SCL, PWM, VDD and VSS. SCL is serial clock input for 2 wire communication protocol. We connect it directly to the analogue pin 5 (A5) on Arduino. SDA is the digital input/output pin for data communication between the devices and the Arduino. The SDA pin is connected to analogue pin 4 (A4). Both SCL and SDA lines require pull-up resistors and hence we put two 4.7kΩ resistors between them and the 3.3V pin on Arduino. The VDD pin is the external supply voltage and the VSS pin is the ground We connect them to the 3.3V voltage source and the GND pin respectively.

We install a laser pointer along with the 5° IR sensor for calibration. The positive pin of the laser pointer is connected to pin 10 and the negative pin of the laser pointer is connected to ground.

4.4.2 Mount the Sensors and Servo Racks onto Kinect

After setting up the circuits, we mount the servos and sensors on top of Kinect. We first install the two servos on the tilt bracket as shown in Figure 4.6.

Note that before we put the servos on the bracket, it is best to set the servo rotation angle to 90°. Assuming we have already connected the circuits as described in the previous section, we can do this by flushing a program into the Arduino board which simply changes the angles of both servos to 90°.

```
Servo servo_h;
Servo servo_v;
void setup()
{
  servo_v.attach(8);
  servo_h.attach(9);
  servo_v.write(90);
```

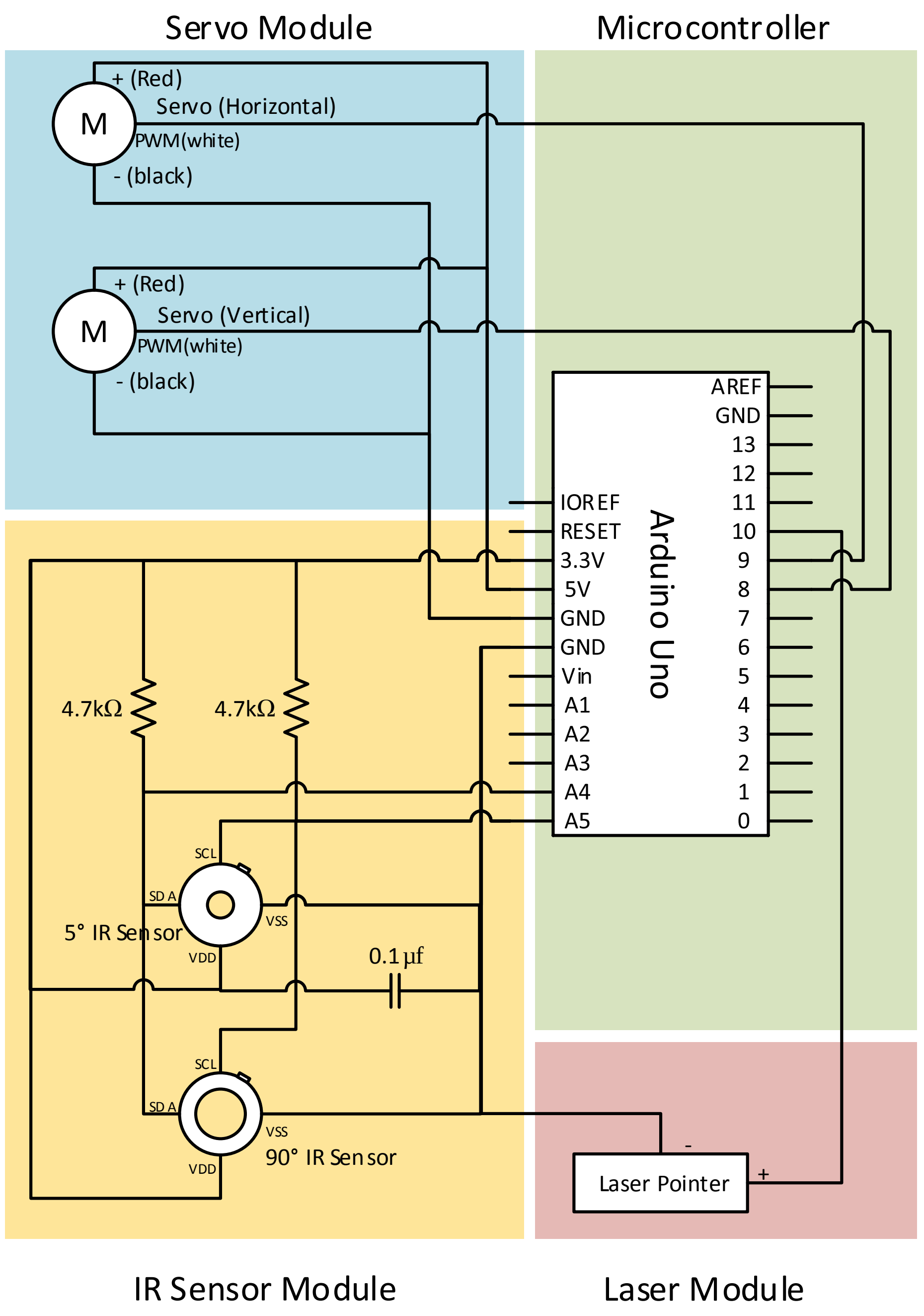

Figure 4.5: Circuit diagram of Arduino and components connected to it

```
  servo_h.write(90);
}

void loop()
{
}
```

We first use a self-tapping screw pan head to fix the small plastic disk onto the axis of the horizontal servo. We then use four screws to fix the horizontal servo onto to a bracket. Please make sure that the body of the servo is parallel to the rack, which is exactly the same as Figure 4.6-1.

In order to mount the vertical servo, we need to install servo bracket holder on the first piece of bracket we used in step 1. The servo bracket holder is fixed on the bracket using two M3*6 screws and M3 nuts as shown in Figure 4.6-2.

Figure 4.6-3 shows the way to mount the vertical servo on the second piece of bracket. Similar to the horizontal servo, we first fix the plastic disk on the servo and then use two screws to mount the horizontal servo on the bracket. The servo body should be parallel to the the bracket as shown in the figure.

Finally, we put the two pieces of brackets together as shown in Figure 4.6-4. Please note that different types of screws are used on different sides of the vertical axis. We also put two M3 flat washers between the two bracket to make sure that the brackets can rotate smoothly.

The brackets are then mounted on the top of the Kinect. The upper bracket (as shown in Figure 4.6-4) is fixed on the center of the Kinect using sticky tape. Now both brackets are upside down and we put the 5° infrared sensor and the laser pointer on the horizontal servo. We use a breadboard to connect the circuits. The Arduino microcontroller and the breadboard are both on the Kinect. After assembling all the parts, the system should be as shown in Figure 4.4.

4.4.3 Control Sensors and Servos Using Arduino

We introduce how to use Arduino to control the servos and read values from the sensors in this section. Part of the code are printed here for ease of explanation. There must exist two functions in an Arduino project: *setup()* and *loop()*. Function *setup()* is the initialization code for Arduino. It is executed when the Arduino is started or reset. After initialization, Arduino goes to the loop mode, in which it will execute the function *loop()* forever.

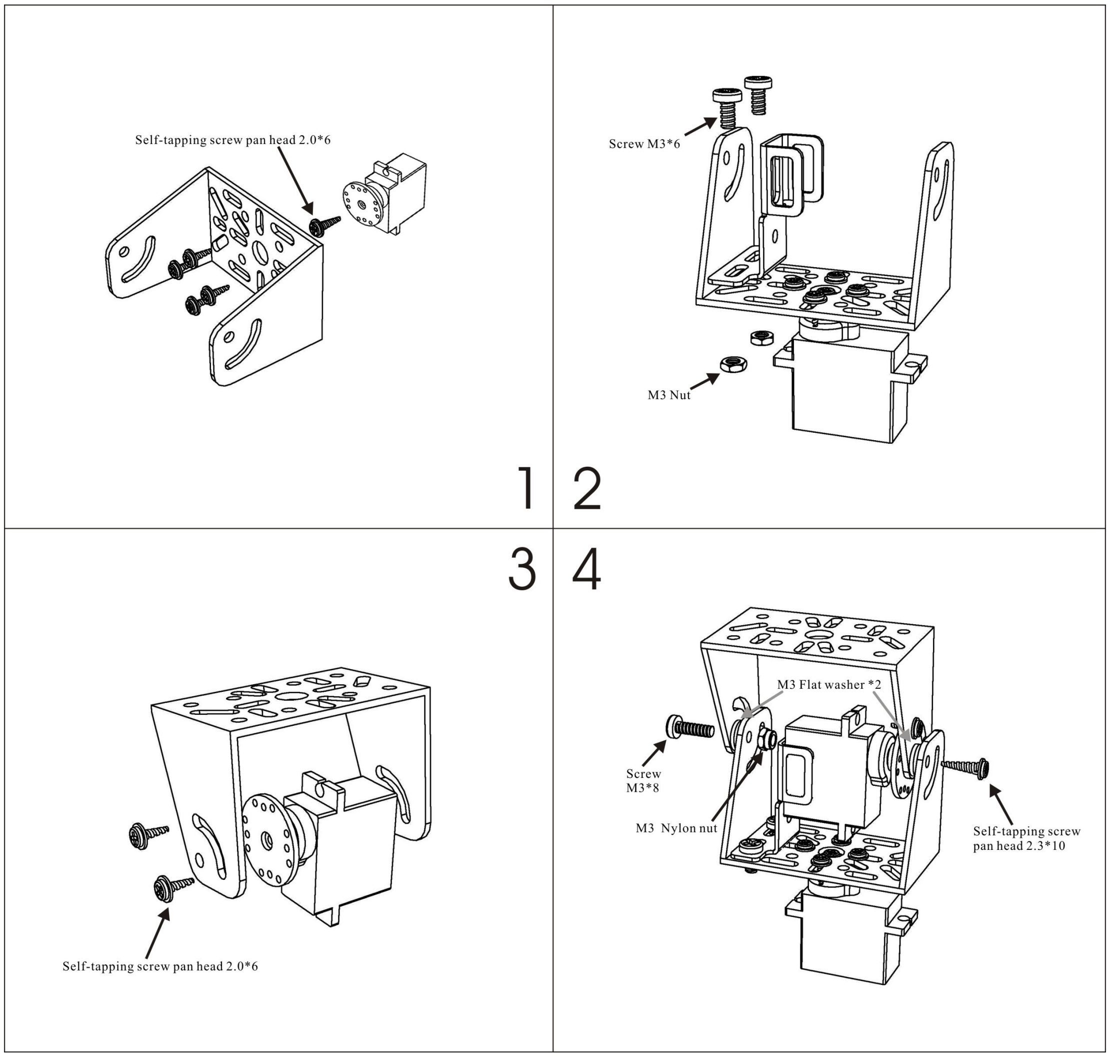
Self-tapping screw pan head 2.0*6
1
Screw M3*6
M3 Nut
2
3
Self-tapping screw pan head 2.0*6
4
M3 Flat washer *2
Screw M3*8
M3 Nylon nut
Self-tapping screw pan head 2.3*10

Figure 4.6: Assemble the servos onto the tilt bracket. Source of figure `https://www.sparkfun.com/products/10335`

```
1  #include <i2cmaster.h>
2  #include <Servo.h>
3  #include <avr/wdt.h>
4
5  void setup(){
6    Serial.begin(9600); //Initialize Serial Communication with the PC
7    i2c_init(); //Initialise the i2c bus
8    PORTC = (1 << PORTC4) | (1 << PORTC5);//Enable pullups
9
10   Servo servo_h;
11   Servo servo_v;
12   servo_v.attach(8); //Use pin 8 to control the vertical servo
13   servo_h.attach(9); //Use pin 9 to control the horizontal servo
14
15   pinMode(10,OUTPUT); //Use pin 10 to control the laser
16 }
17
18 void loop(){
19   wdt_enable(WDTO_4S);
20   loopcounter = (loopcounter + 1)%200;
21
22   if(loopcounter == 0){
23     Serial.print(”BD:”);
24     Serial.println(getTemp(0x5A));//Default I2C device ID is 5A, for body
         sensor
25     Serial.print(”BG:”);
26     Serial.println(getTemp(0x55));//We changed the I2C device ID of
27     //the background IR sensor to 55
28   }
29
30   //Code to read from Serial line (omitted)
31
32   servo_h.write(h); //Control the horizontal angle
33   servo_v.write(v); //Control the vertical angle
34   switchLaser(); //Control the state of laser pointer
35
36 }
```

We start Arduino by setting up a serial communication socket to the PC by calling the *Serial.begin(9600)* in line 6. The parameter 9600 is the baud rate of the serial line. To communicate with the infrared sensor, we must initialize the I2C bus by calling *i2c_init()*. Remember that we used pullup resisters in Figure 4.5 for both the IR sensors, we need to enable pullup in line 8. Pullup resistor brings the voltage up to the referencing point, which is 3.3V in our case. For I2C communication, pullup resistors are usually necessary.

To control the servos, we use the *Servo.h* library in line 2. We simply create two *Servo* objects and attach them to pin 8 and 9 on the Arduino for control in line 12 and 13. The last line in the *setup()* function is to declare the pin 10 to be an output pin. We use pin 10 to control the laser pointer.

In *loop()*, *wdt_enable(WDTO_4S)* is first executed to set the automatic restart time to 4 seconds. Function *wdt_enable()*is in the *avr/wdt.h* library, which is the watchdog timer handling library. The library is able to restart the system after a predefined timeout. We use this library as a fail safe. It resets the Arduino in the case when the sensors are not responding. If the sensors are not responding, the function *loop()* will be blocked. Since we have set automatic restart time to 4 seconds, Arduino will be reset after the 4 seconds timeout. In the case that the sensors are responding, function *wdt_enable(WDTO_4S)* will be called again and the restart time will be postponed. Therefore, Arduino will not be reset as long as the *loop()* keeps running normally.

The variable *loopcounter* in line 20 counts the number of times *loop()* is executed. Every time the function *loop()* executes 200 iterations, the counter resets to 0 and Arduino will fetch the temperature readings from the sensors (line 22 to 28). The reading will be sent back to the PC using serial communication. The function *getTemp()* is for fetching the data. We omitted the details of *getTemp()* here. The input parameter of *getTemp()* is the I2C device ID. Arduino first fetches data from the body IR sensor, whose device ID is 0x5A, the default device ID for MLX90614. Since we have changed the device ID of the background IR sensor to 0x55, we use it to get the background radiation temperature.

After fetching the temperature values and send them to the PC using serial line, Arduino read commands from the serial communication line. It parses the commands and control the angles of the horizontal and vertical servos (line 32 and 33). In line 34, it switches the laser pointer when necessary.

4.4.4 Communicate with Z-Wave Devices

The heater is controlled by a DSC06106-ZWUS Z-Wave Smart Energy Switch. The Z-Wave switch can control the on/off state and query the current power usage remotely. The Z-Wave switch is managed by a Aeotec Z-Stick S2 controller, which is plugged into a USB port on the PC. We discuss how to control the Z-Wave switch programatically by SPOT. Since Z-Wave is not an open-source protocol, there are no authoritative documentation for programming. We infer the protocol by monitoring the serial communication of a Z-Wave management program. The completeness of our implementation is not guaranteed. However, we do not find any problem in practice when using the current implementation.

When the Z-Wave controller is plugged in, SPOT first establishes a serial connection with it by executing the following code:

```
static SerialPort Init()
{
  sp = new SerialPort();
  //We omit the details of serial port configuration here
  sp.Open();
  receiverThread = new Thread(new System.Threading.ThreadStart(
     ReceiveMessage));
  receiverThread.Start();
  return sp;
}
```

After opening the serial port, we create a thread for receiving the message from the Z-Wave controller. The code for receiving a message is the following:

```
private static void ReceiveMessage()
{
  while (sp.IsOpen == true)
  {
    int bytesToRead = sp.BytesToRead;
    if ((bytesToRead != 0) & (sp.IsOpen == true))
    {
      byte[] message = new byte[bytesToRead];
      sp.Read(message, 0, bytesToRead);
      MessageReceived(message);
      if (sendACK)
      {
        SendACKMessage();
      }
      sendACK = true;
    }
  }
}
```

The process keeps reading messages from the serial port. Once a message is received, it is passed to the *MessageReceived()* function for processing. For most of the cases, an ACK message is required when the Z-Wave controller receives a message. The only exception is when the received message is also a ACK, which happens after the controller sending a message to the switch. So, we call *SendACKMessage()* to reply an ACK message when *sendACK* is *true*.

SPOT calls the following function to control the on/off state of the switch:

```
private static void SendSetOnOff(bool on, int nodeid)
```

```
{
  byte[] msg = new byte[] { 0x01, 0x0a, 0x00, 0x13, (byte)nodeid, 0x03, 0x20
      , 0x01, (on ? 0xff : 0x00), 0x25, seqNumber, 0x00 };
  msg[msg.Length - 1] = GenerateChecksum(msg);
  SendMessage(msg);
}
```

The byte sequence is the command to control the state of *nodeid*'th switch. A checksum is necessary at the end of the message to guarantee the integrity of the packet. We generate the checksum by calling the function *GenerateChecksum()*.

To query the current power of the switch (or the heater), the following function is called.

```
public static double GetSensorMultiLevel(int nodeID)
{
  lock (getWaiting)
  {
    getWaiting.Reset();
    SendSensorGetMultiLevel(nodeID);
    if (getWaiting.WaitOne(1000))
    {
      return getSensorMultiLevelValue;
    }
    else
    {
      return -1;
    }
  }
}
```

The function first locks an event object *getWaiting* to gain exclusive access to the Z-Wave controller. The event object *getWaiting* is reset to the default state. We then send a message to the switch to query the power by calling *SendSensorGetMultiLevel()*. After sending the message, the thread waits 1000 ms for the *getWaiting* event to be fired. If a message is received by the *ReceiveMessage* function, it calls the *MessageReceived* function to process the message.

```
private static void MessageReceived(byte[] message)
{
  //A reply on the current power usage is received
  if (MatchByteArr(new byte[] { 0x01, 0x0E, 0x00, 0x04, 0x00, 0x03, 0x08, 0
      x31, 0x05, 0x04, 0x64, 0x00, 0x00, 0x0A, 0x5B, 0xFB }, message, new
      byte[] { 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x01, 0x01, 0x01, 0x00, 0
      x00, 0x00, 0x00, 0x00, 0x00, 0x00 }))
```

```
 5    {
 6      double value = 0;
 7      for (int i = 11; i <= 14; i++)
 8      {
 9        value = value * 16 * 16 + (int)message[i];
10      }
11      getSensorMultiLevelValue = value / 1000.0;
12      getWaiting.Set();
13    }
14    //Omitted code for processing other known message types
15  }
```

In line 4, the function first matches the received message with the known message types. If the message is a reply message on the current power usage, it analyses the content of the message to get the current power usage (line 6 to 10). In line 11, the value is stored in a static variable *getSensorMultiLevelValue*. Once the value is stored, it fires the *getWaiting* event by calling *getWaiting.Set()* in line 12. Since the event is fired, the *WaitOne()* function in the line 7 of *GetSensorMultiLevel* will no longer be blocked. It returns the switch power value *getSensorMultiLevelValue* (line 9 in *getSensorMultiLevel()*). If there is any error, the *WaitOne()* function gets a time out and -1 is return to indicate the error (line 13 in *getSensorMultiLevel()*).

4.4.5 Track Users with Kinect

We use Kinect to track the location of the users. Kinect is able to track up to 6 users and extract skeleton points from 2 of them. We will discuss the details of Kinect programming in this section.

We start by creating a Windows Presentation Foundation (WPF) Window Project in Visual Studio. We initialize Kinect when the main window is loaded. Following is the partial code for initialization.

```
 1  private void Window_Loaded(object sender, RoutedEventArgs e)
 2  {
 3    //Initialize other components, omitted here
 4
 5    if (KinectSensor.KinectSensors.Count > 0)
 6    {
 7      sensor = KinectSensor.KinectSensors[0];
 8
 9      if (sensor.Status == KinectStatus.Connected)
10      {
        sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);  11
        sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);     12
        sensor.SkeletonStream.Enable();                                         13
        sensor.AllFramesReady += new EventHandler <AllFramesReadyEventArgs>(    14
            SensorAllFramesReady);
        sensor.Start();                                                         15
                                                                                16
        colorpixels = new byte[sensor.ColorStream.FramePixelDataLength];        17
        wbmp = new WriteableBitmap(sensor.ColorStream.FrameWidth, sensor.       18
            ColorStream.FrameHeight, 96, 96, PixelFormats.Bgr32, null);
      }                                                                         19
    }                                                                           20
}                                                                               21
```

We first initialize other components, such as serial communication modules with sensors. In line 7, we select the sensor to be the first Kinect attached on the system. In line 11 to 13, we enable three streams: the color image stream, depth image stream, and skeleton stream. Kinect uses a event based programming model. When image frames are ready, it fires a event called *AllFramesReady*. We added an event handler in line 14 to process the event. After that, Kinect is ready to start (line 15). To display the color image stream on screen, we initialize two objects in line 17 and 18. The *colorpixels* array temporarily stores the color image stream as a bitmap, and we display it on the screen using the *WriteableBitmap* object *wbmp*.

The image streams are handled by the function *SensorAllFramesReady*, which is printed here.

```
void SensorAllFramesReady(object sender, AllFramesReadyEventArgs e)
{
  if (!sensor.IsRunning)
    return;

  using (ColorImageFrame cf = e.OpenColorImageFrame())
  {
    if (cf != null)
    {
      if (!Helper.loadReady)
      {
        Helper.loadReady = true;
        Helper.loadingReadyEvent.Set();
      }

      cf.CopyPixelDataTo(colorpixels);
      wbmp.WritePixels(new Int32Rect(0, 0, cf.Width, cf.Height), colorpixels
          , cf.Width * cf.BytesPerPixel, 0);
      image3.Source = wbmp;                                                    18
    }                                                                          19
  }                                                                            20
                                                                               21
  firstSkeleton = GetFirstSkeleton(e);                                         22
  if (firstSkeleton == null)                                                   23
  {                                                                            24
    return;                                                                    25
  }                                                                            26
                                                                               27
  if (firstSkeleton.Joints[Arduino.pointingJoint].TrackingState ==             28
      JointTrackingState.Tracked)
  {                                                                            29
    ColorImagePoint colorPoint = sensor.CoordinateMapper.                      30
        MapSkeletonPointToColorPoint(firstSkeleton.Joints[Arduino.
        pointingJoint].Position, ColorImageFormat.RgbResolution640x480Fps30);
    int h = (int)((colorPoint.X - 640.0 / 2) / (640.0 / 2) * (57.0 / 2));      31
    int v = (int)(-(colorPoint.Y - 480.0 / 2) / (480.0 / 2) * (43.0 / 2));     32
    Arduino.SetAngle(h, v);                                                    33
  }                                                                            34
  //Other code, omitted                                                        35
}                                                                              36
```

In line 6, the **using** statement is used for creating the *ColorImageFrame* object *cf*. Note that *cf* is a disposable object, which means some of its member elements will not be automatically garbage collected unless we call the *Dispose()* function explicitly. An alternative way to release memory is by calling the **using** statement. The object *cf* will be automatically disposed at the end of the **using** block (line 20). In line 10 to 14, the *loadingReadyEvent* is fired the first time the frames are ready. The event indicates the Kinect is already working and other components could start working as well. In line 16 to 18, we copy the color image data to the temporary variable colorpixels and display the image on the screen.

The code then tries to process the skeleton information if the first tracked skeleton is not null (line 22 to 26). If the target skeleton joint is actually tracked by Kinect(line 28), we use the *CoordinateMapper* to map the 3D point to a 2D point on the screen. The Kinect camera has a 57° horizontal view angle and 43° vertical view angle. Using this information, we calculate the horizontal and vertical angles of the tracking skeleton joint (line 31 and 32), and change the direction of the infrared sensor using Arduino (line 33).

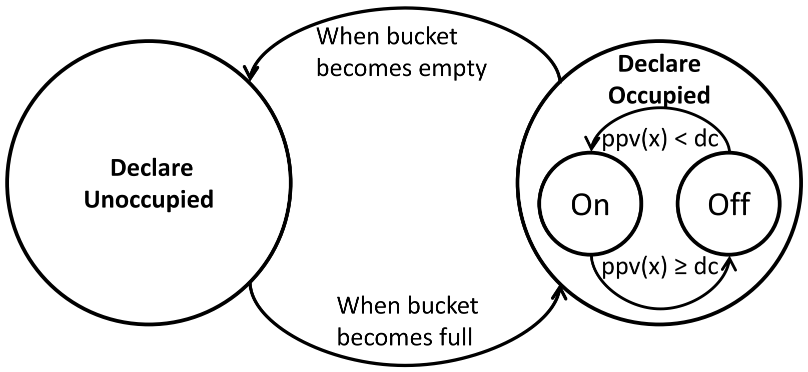

Figure 4.7: Heater control logic showing the leaky-bucket based low-pass filter. An office room will be declared as occupied only if it is occupied for a certain fraction of the past few minutes.

4.4.6 Detect Room Occupancy

SPOT detects room occupancy using the Microsoft Kinect. Recall that the Kinect APIs allows the controller to obtain near-real-time skeleton tracking. When there is a skeleton tracked by Kinect, SPOT considers the room as occupied. However, it does not turn the heater on immediately after it detects a worker to deal with transient occupancy of the work space. Instead, we have implemented a leaky-bucket based low-pass filter that turns on the heater only if the work space has been occupied for a sufficiently long fraction of the prior few minutes.

Specifically, the system has a virtual leaky bucket of size 5 units. The bucket is initially empty. At the end of each minute, if the Kinect sensor reports that the work space was occupied in the past minute, one unit of "water" is added to the leaky bucket, up to a maximum bucket size of 5 units; otherwise, one unit is subtracted. When the "water" in the leaky bucket reaches 5 units, the work space is declared to be occupied. Conversely, when the "water" in the bucket reaches 0, the work space is declared to be unoccupied. When SPOT thinks that the work space is occupied, it evaluates the current $ppv(\mathbf{x})$ value and compares it to the desired comfort set point dc. At the beginning of each minute, if $ppv(\mathbf{x}) < dc$, the heater is turned on until the PPV value reaches dc; otherwise the heater is turned off. The detailed heater control logic is demonstrated in Figure 4.7.

Chapter 5

Evaluation

We discuss the evaluation of our system in this section. Since the evaluation period was in winter, we validated our design by implementing a heating control system. Our ideas, however, are applicable to a cooling system, where we would replace the heater with a small personal fan.

To evaluate the effectiveness of our system, we deployed the prototype to a $11.9m^2$ office room at the University of Waterloo. The office room is usually occupied from 8:30 AM to 5:30pm on weekdays. Note that the room is in a building that also has its own HVAC control system whose design goal is to maintain a constant temperature of 23°C throughout the day. Therefore, the PPV setpoint is chosen to correspond to a comfort level that is somewhat warmer than usual (corresponding to a worker who prefers warm working conditions), as a positive offset to this nominal base value.

5.1 Accuracy of Clothing Level Estimation

We first discuss the effectiveness of the clothing level estimation. Since the system is designed for indoor thermal control, we assume that the clothing level is between 0.7 (a shirt) and 1.3 (shirt, sweater and jacket), which are common in our office environment. In the training phase, 23 data points were collected as training data, with the clothing level ranging from 0.7 to 1.25. This allowed us to compute a linear regression to estimate the clothing level from the infrared sensor reading.

Subsequently, about 20 volunteers were selected to participate in a test of accuracy. For volunteers wearing a jacket, we first tested the clothing level estimation algorithm when

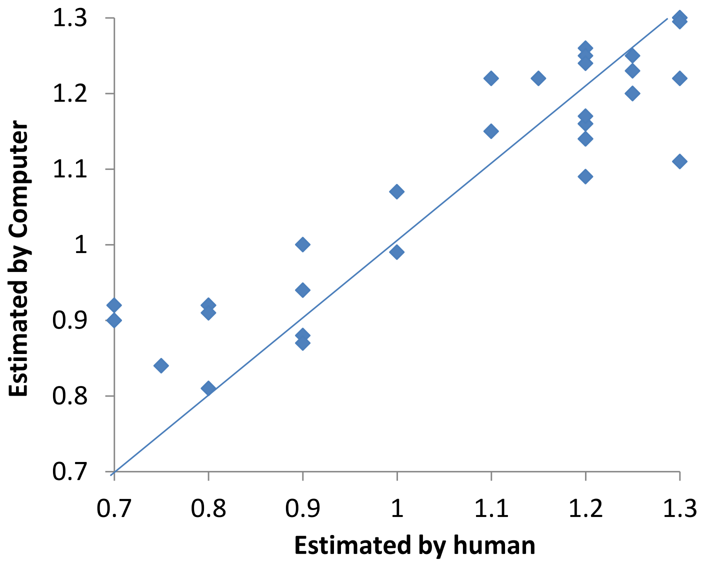

Figure 5.1: Clothing level estimated by human v.s. estimated by our algorithm. The RMSE of the estimation is 0.0919 and the Pearson correlation coefficient was 0.9201 indicating good linear correlation.

they were wearing their jackets. We then tested the clothing level after they took off their jackets. Therefore, we collected 35 testing data points in total.

To test our algorithm, the clothing level of each volunteer was first estimated by one of the authors using Table A.2. It was then evaluated using the estimation algorithm. The results are shown as a scatter plot in Figure 5.1. The root mean square error (RMSE) of the prediction was 0.0919 and the Pearson correlation coefficient was 0.9201 indicating good linear correlation.

Note that the infrared sensor we are using has a 5 degree detection angle. We found that when a subject was more than 2 meters away from the sensor, the clothing estimation result was inaccurate because of noise from background infrared radiation. Therefore, in a real deployment, we need to install the IR sensor no more than 2 meters from the worker. If this is an issue, for example in a large office, we advocate using sensors with a smaller detection angle.

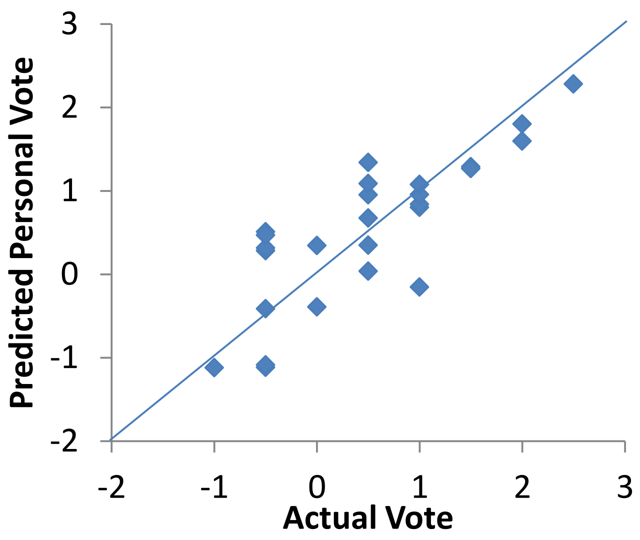

Figure 5.2: Actual personal vote v.s. predicted personal vote. The RMSE of the estimation is 0.5377 and the Pearson correlation coefficient was 0.8182 indicating good linear correlation.

5.2 Accuracy of PPV Estimation

This subsection discusses the accuracy of comfort level estimation using the PPV model. The evaluation was done in an actual office at the University of Waterloo for several days.

On the first day, the office owner gave votes to the system on the thermal environment to train the PPV model. Over the training period, 12 votes were collected as training data. We then tested the PPV model by comparing the predicted votes with 8 actual votes on the following days. The results are plotted in Figure 5.2. We found that the RMSE of PPV estimation was 0.5377 and the Pearson correlation coefficient was 0.8182 indicating good linear correlation.

5.3 Responsiveness of the Work Space to Thermal Control

This section discusses the responsiveness of the experimental workspace to thermal control using the radiant heater. To test responsiveness, we turned on the radiant heater at 4:20pm

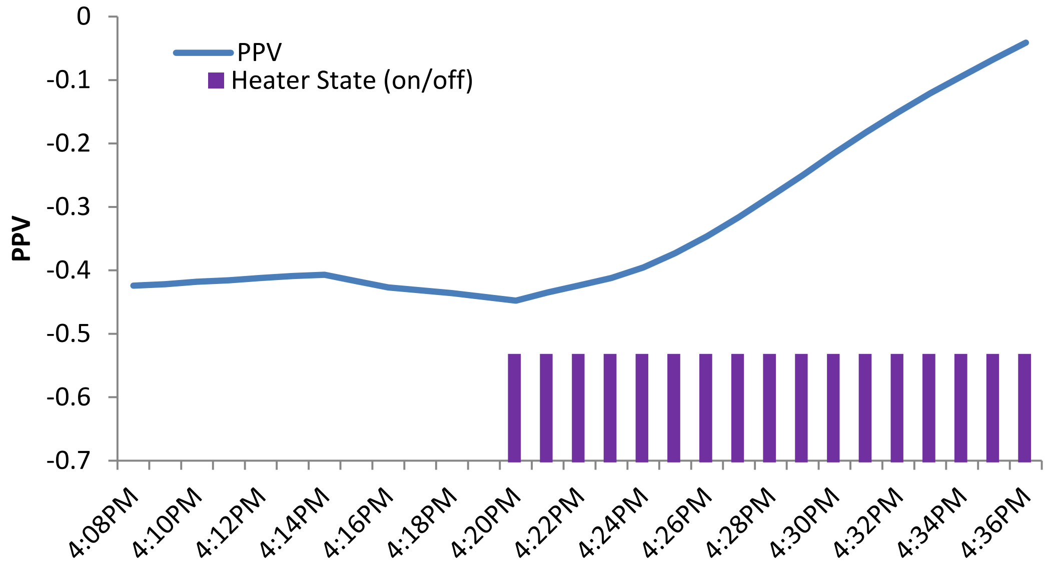

Figure 5.3: Room temperature and the heater state. The room temperature increased 1 degree Celsius after turning on the heater for 15 minutes.

and measured the PPV every minute. The result is plotted in Figure 5.3.

Before the heater was turned on at 4:20pm, the room temperature was maintained by the central HVAC system and the PPV was between -0.5 to -0.4. When the heater was turned on at 4:20pm, the room temperature as well as the PPV started to increase immediately, reaching the target PPV of 0 in 15 minutes at 4:35pm.

We conclude that it is feasible to use reactive control for personal thermal comfort without significantly reducing human comfort.

5.4 Thermal Comfort Over a Day

We now discuss the performance of SPOT over the course of a typical day. Our experimental setup was the same as for the other experiments. However, we required SPOT to maintain a PPV of 0. Figure 5.4 shows the results of this experiment, depicting room occupancy, PPV, and room temperature over the day.

Note that both the room temperature and PPV are relatively low before the worker entered the office at 10 AM in the morning, with the PPV of -1.5 being the comfort level corresponding to the temperature setpoint chosen by the central HVAC system. SPOT turned on the heater within five minutes of the worker's arrival and both the temperature and the PPV increased steadily to 0 over the next 45 minutes.

The PPV of the office was always maintained around 0 when the worker was in the office, with small excursions above zero when the HVAC heating system turned on from time to time. The brief change in occupancy just before 11 AM, of about 10 minutes, was too short to cause any appreciable change in PPV.

The worker left the office at 4pm for an hour. During that time, SPOT turned off the heater to save energy. This reduced the PPV to -0.5, but the PPV returned to 0 soon after the worker returned at 5pm. When the worker finally left at 5:30pm, the PPV declines, eventually reaching -1.5.

This demonstrates that SPOT can maintain the PPV at a chosen comfort value over the course of a day, despite the periodic activation of the central HVAC heating system and changes in office occupancy.

Note that, in this instance, PPV tracks room temperature quite closely. This is because there was little change in other environmental and personal factors, such as humidity and clothing level. In other circumstances, such as when the worker may put on or take off a jacket, SPOT would be able to maintain the comfort level by appropriately reducing the room temperature.

5.5 Trade-off between PPV and Energy Consumption

SPOT allows a building's energy consumption to be decreased in three ways.

- It allows the common areas of the building to be heated or cooled to a lesser degree than the ASHRAE standard of 23°C.
- It only heats or cools a work space when the worker is actually present.
- It allows the worker to choose a comfort level that is lower than 0, thus saving energy.

Here, we focus on the third element above.

To evaluate the possible amount of energy saving by lowering the PPV value, we measured the relationship between PPV and heater energy consumption[1]. We did this by

[1]This relationship is necessarily noisy because temperature is not the only determinant of PPV. Nevertheless, the trend is distinct.

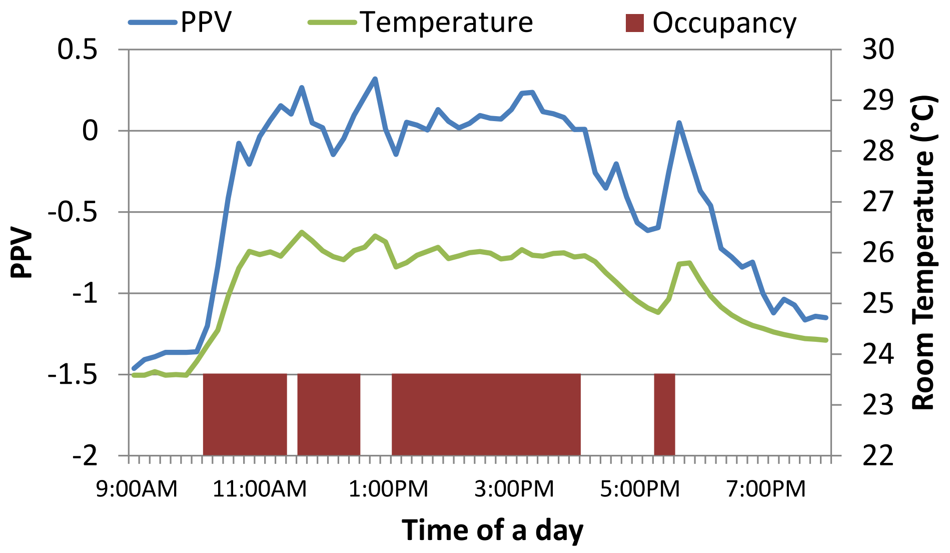

Figure 5.4: Room occupancy and PPV over a day. Each tick on the X axis is 10 minutes. For most of the time when the room is occupied, the PPV is maintained around 0, even through there are external disturbances from the central HVAC system.

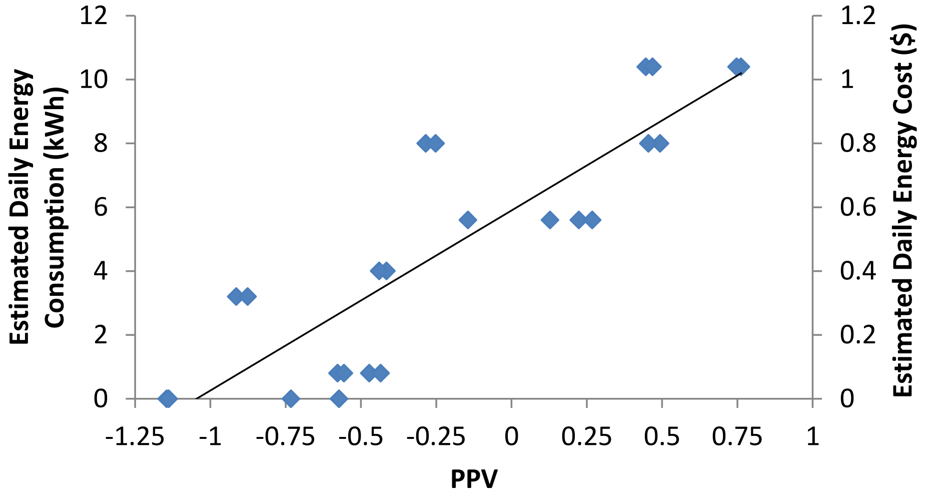

Figure 5.5: Relationship between daily energy consumption of the heater and the PPV. Maintaining a PPV of 0 consumes about 6 kWh electricity daily. By setting the target PPV to -0.25, we can save about 1.5 kWh electricity per day.

setting the heater power to different values and recording the PPVs when the room temperature had converged. When the heater was turned off, the room temperature was maintained by the centralized HVAC system at around 23 degrees, corresponding to PPV values between -0.5 and -1.24 depending on the central HVAC system's phase in its heating cycle. In contrast, when the radiant heater was set to its maximum power, the PPV was about 0.75 with the estimated power consumption per day was about 10.5 kWh.

Figure 5.5 shows this trade-off between PPV and the heater energy consumption in a day. We see that a reduction in PPV of 0.1, which is hardly noticeable by a human, results in the reduction in usage of 0.6 kWh of electricity in a day. This allows us to quantitatively select the trade-off between personal thermal comfort and heating energy consumption. For instance, an energy-aware office worker can set the target PPV value *dc* (as mentioned in §4.4.6) to -0.5 in order to save energy.

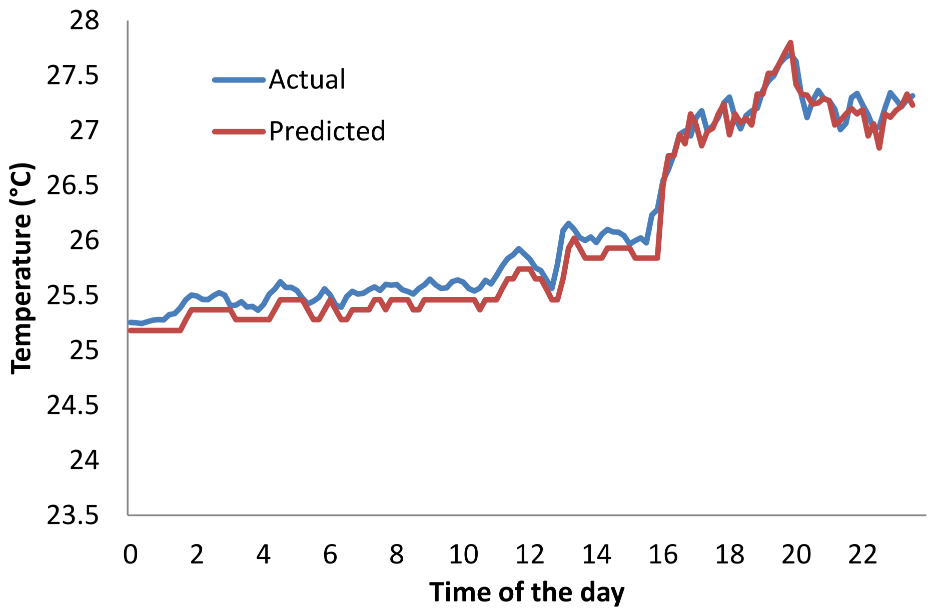

Figure 5.6: Actual temperature versus predicted temperature using LBMPC. The RMSE over a day is 0.1507°C.

5.6 Accuracy of LBMPC

We evaluate the accuracy of LBMPC in this section. Figure 5.6 shows the predicted temperature and actual temperature over a day. The horizontal axis is the time of the day and the vertical axis is the room temperature. During that day, we vary the heater power at different time of a day so that we can see the accuracy of prediction in diverse conditions.

The office room temperature is maintained at around 25.5°C in the morning. Even if there are some small temperature fluctuation, the LBMPC can predict the temperature accurately. Start from the noon, the temperature started to increase gradually. At about 16:00 on that day, we maximized the heater power and the room temperature changes dramatically. Given the huge fluctuation at the latter time of the day, LBMPC can still predict temperature accurately. The RMSE of the prediction over the day is only 0.1507°C.

5.7 Accuracy of Occupancy Prediction

We evaluate the performance of the occupancy prediction by collecting one week of data. Figure 5.7 compares the actual occupancy and predicted occupancy of 5 consecutive weekdays (weekend data is omitted as the room is always vacant). The false negative rate is relatively low, which is about 0.014. However, the false positive rate is relatively high, which is about 0.128.

Since the false negative rate is low, the office worker's thermal comfort is guaranteed. When he or she is in the office, SPOT guarantees the room temperature is at a comfortable range. We implement fail safe code for inaccurate estimation. When SPOT finds the actual occupancy is different from predicted occupancy, it recalculates the optimal control scheme using the true occupancy and updates the state of the heater.

5.8 Comparing Five Temperature Control Strategies

This section reports on a preliminary evaluation of five different temperature control schemes as well as the benefits of predictive over reactive control.

5.8.1 Temperature Control Schemes

We now describe the five different temperature control schemes that we implemented in this workspace.

- **Fixed Setpoint:** This control scheme has a fixed temperature setpoint of 25°C. If the measured temperature is lower than 25°C, the heater heats the room until measurements indicate that it has reached the setpoint.
- **Scheduled Setpoint:** This emulates the behaviour of a "Smart Thermostat": the controller maintains the room temperature at 25°C from 8 AM to 6 PM.
- **Reactive Temperature:** This control scheme starts to heat the room when occupancy is detected, and maintains a setpoint of 25°C only when the worker is present. To improve the robustness of the system, heating commences only after 5 minutes of continuous occupancy and stops when the workspace is vacant for 5 minutes. This reduces sensitivity to transient occupancy.

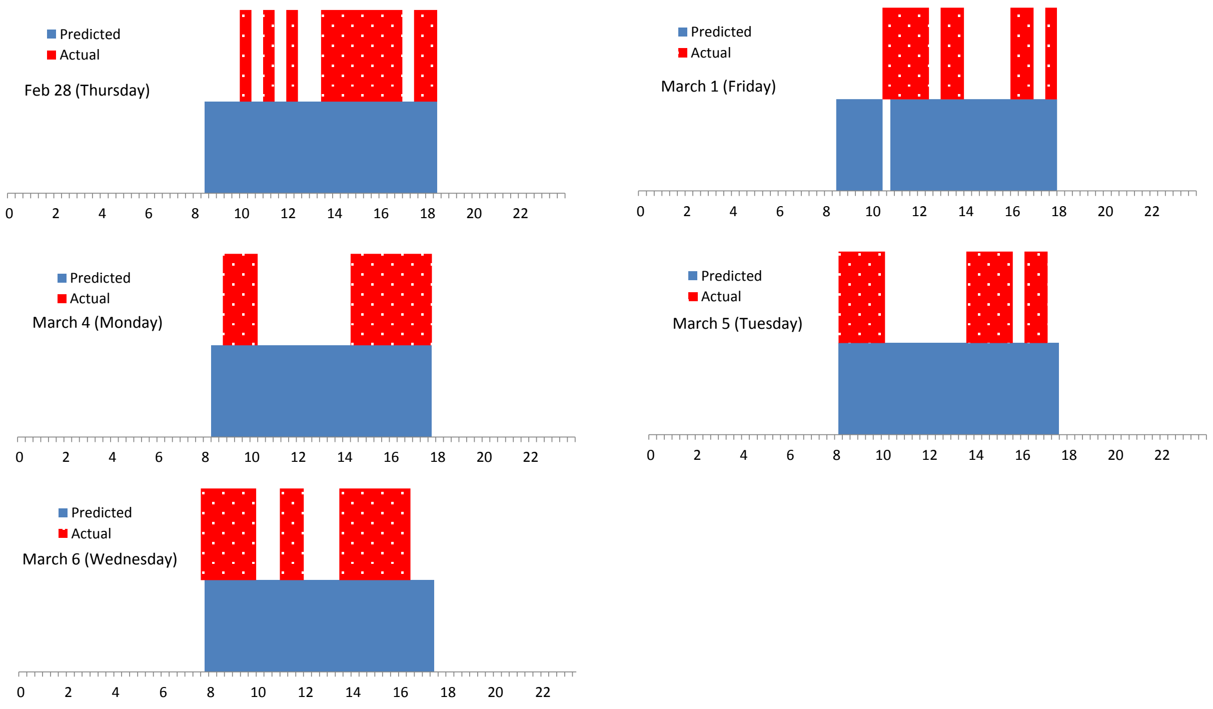

Figure 5.7: Predicted occupancy versus actual occupancy of five consecutive weekdays. The lower blocks represent predicted occupancy and the upper blocks are actual occupancy. The false negative rate is 0.01393 and the false positive rate is 0.1284

- **Reactive PPV:** Instead of maintaining a constant temperature as in reactive temperature control, reactive PPV control maintains personal comfort (PPV) at the category B thermal comfort environment [5] where $ppv \in [-0.5, 0.5]$.
- **Optimal Control** This scheme finds the best heating control sequence using LBMPC. To maximize worker comfort, by setting λ in Eq. 3.15 to 10000 and ϵ to 0.5.

5.8.2 Evaluation Methods

We attempted to run each control scheme for at least two days; the actual number of days for each scheme is shown in Table 5.1. For reactive temperature control, we obtained two days of data but later discovered that one day's data was not valid because one sensor had stopped working. Therefore the results for this scheme are not reliable. On the other hand, we have four and five days data respectively for reactive PPV and optimal control schemes, so the comparison of their relative performance is more reliable.

Control Scheme	Number of Days
Fixed	2
Scheduled	2
Reactive Temperature	1
Reactive PPV	4
Optimal	5

Table 5.1: Number of days tested for each control scheme

5.8.3 Evaluation Metrics

We propose three performance metrics. The **Average Daily Energy Consumption** is the total average energy consumed over a day. The **Average Absolute PPV** is the average absolute PPV value conditional on occupancy (because personal comfort only matters if the workspace is occupied). If $ppv(t)$ denotes the PPV value for time slot t during the day, the average absolute PPV is:

$$\frac{\sum_{t=1}^{T} ppv_t \delta_t}{\sum_{t=1}^{T} \delta_t} \tag{5.1}$$

where δ_t is the indicator of occupancy at t (eg. $\delta_t = 1$ for occupied, $\delta_t = 0$ otherwise). Consider a control scheme A that always maintains the PPV at -0.5 and an alternative control scheme B that maintains PPV at 0 for half of the time and -1 for the other half. Both schemes have the same average absolute PPV. However, a worker will feel much more comfortable under scheme A because a typical worker is comfortable in the PPV range of $[-0.5, 0.5]$. Thus, we define the **Average Discomfort** to quantify how uncomfortable a worker feels over a day. Specifically, we define the discomfort at timeslot t as:

$$d_t = max(|ppv_t| - 0.5, 0) \tag{5.2}$$

In other words, if the PPV at timeslot t is in $[-0.5, 0.5]$, the discomfort d_t is 0, otherwise, the discomfort is $|ppv_t| - 0.5$. We then calculate the average discomfort conditional to occupancy as in Eq. 5.1.

5.8.4 Evaluation Results

We compare the five temperature control schemes on their daily energy consumption, average absolute PPV, and average discomfort. Figure 5.8 and 5.9 shows the results.

The fixed temperature setpoint control consumes about 21.08kWh of electricity daily and the average absolute PPV and average discomfort is 0.42 and 0.07 respectively. Since the fixed temperature setpoint control always keeps the room temperature at 25°C, it guarantees the user comfort by consuming a lot of energy.

By setting a temperature control schedule, we can save about half of the energy consumption (because the heater is off at night). The scheduled setpoint control consumes 11.6kWh of energy and its average absolute PPV and average discomfort is 0.71 and 0.23 respectively. This method decreases energy consumption but has the potential to make the user feel uncomfortable.

We can further reduce energy consumption by applying reactive temperature control. The user leaves the office during working hours for lunch or meeting regularly. Reactive control turns the heater off on these occupancy gaps to save energy. Over a day, the system consumes 6.30kWh of electricity and its average absolute PPV and average discomfort is 0.86 and 0.36. Since the heater needs some time to heat the room to the desired setpoint, the user may feel uncomfortable during this time.

The reactive PPV control maintains a constant PPV level rather than a temperature setpoint, it can better capture the user comfort. By using reactive PPV control, we increase the user comfort with almost the same level of energy consumption. The daily energy

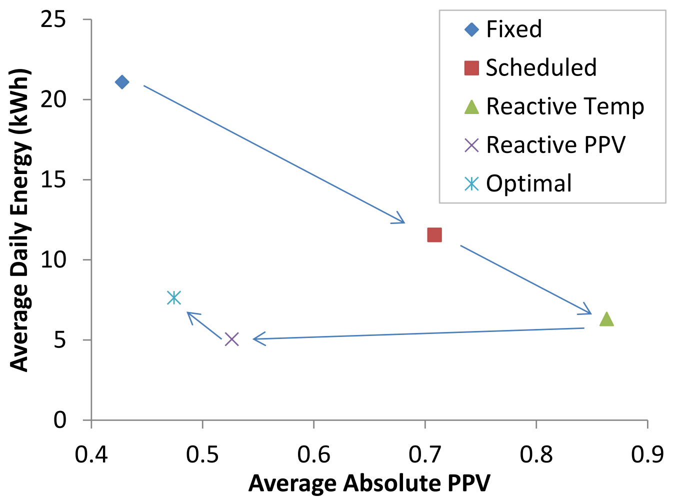

Figure 5.8: Average absolute PPV versus daily energy consumption

consumption of reactive PPV control is 5.04kWh and its average absolute PPV and average discomfort is 0.53 and 0.20.

Using optimal control, we can further increase the user comfort by using slightly more energy. Since we set the weight of user comfort λ in Eq. 3.15 to be a very large value so that user comfort is always guaranteed, optimal control has the lowest average discomfort. The results in Figure 5.8,5.9 reflects the parameter settings. In an average day, optimal control consumes about 7.62kWh of electricity and its average absolute PPV and average discomfort is 0.47 and 0.02.

We find that optimal control has the best tradeoff between energy consumption and user comfort. Although reactive temperature control scheme saves more energy by dynamically turning off the heater, it has the potential to reduce user comfort, which may deter user from adopting these control schemes. Figure 5.10 and 5.11 show how reactive PPV control and optimal control work. The lines are the PPV value, reflecting the temperature change. The bar represents the occupancy. For reactive control, the room starts to heat at around 8:40 AM when the occupant arrives. It requires more than half an hour to reach the target PPV value. In Figure 3.1, the room starts to heat at around 7:30 AM and when the occupant arrives at 9:00 AM, the room temperature is already in the comfort zone. We can conclude that users are more motivated to adopt optimal control for energy saving since it does not scarify thermal comfort.

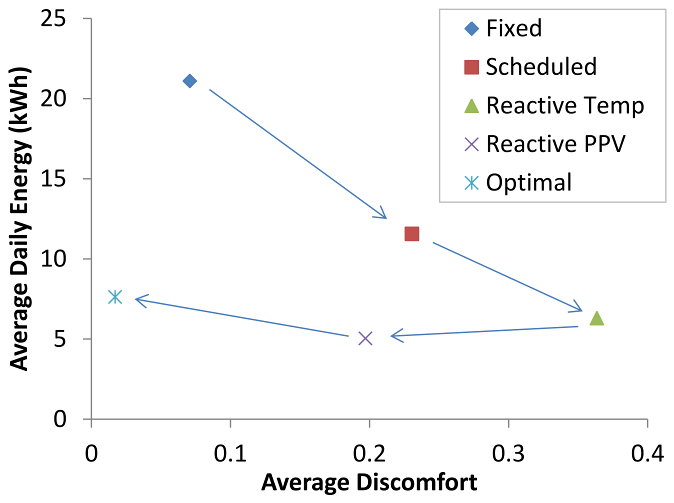

Figure 5.9: Average discomfort versus daily energy consumption

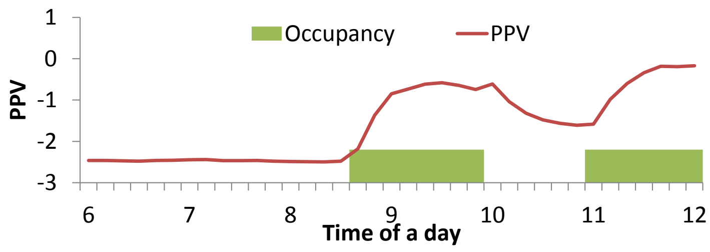

Figure 5.10: Reactive PPV control starts to heat the room when occupancy is detected

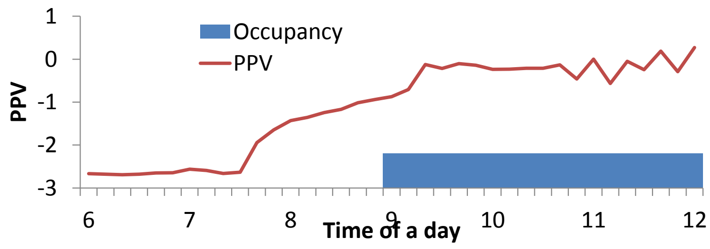

Figure 5.11: Optimal control starts to heat the room before the estimated arrival time

Chapter 6

Related Work

6.1 PMV Model

The PMV model is widely used for evaluating the performance of building temperature control systems. Yang et. al. [34] used PMV to reduce the energy consumption of a building's HVAC system. Since radiant temperature is a significant factor in PMV in hot and humid areas, air temperature and humidity control is not enough for cooling in summer. In their system, they control the air velocity as well in order to maintain PMV at the comfortable range. Aswani [14] et. al. have also used the PMV model in their building temperature control system. They use Learning-Based Model Predictive Control (LBMPC) to control the building HVAC system such that different zones of the building maintain PMVs close to 0. Our approach, instead, uses the personalized PPV model to achieve personalized thermal control.

The Thermovote [20] system allows workers in a building to vote on the current temperature. Instead of using PMV to predict users' feeling, they use the actual vote of the workers to adjust their comfort. However, the system requires the users to vote frequently, which is onerous. We avoid this problem by building personalized models for each individual. SPOT only requires votes during the training phase to calibrate the PPV model. Subsequently, the thermal preference of the user is used to control the HVAC system and no more voting is required.

6.2 Occupancy-based HVAC Control

There has been a considerable amount of work on improving the energy-efficiency of HVAC systems. For example, Aswani et. al. [14] use Learning-Based Model Predictive Control (LBMPC) to model and control HVAC systems in a large university building. They were able to save an average of 1.5MWh of electricity per day in their testbed. Fong et. al [22] used evolutionary programming (EP) to find the optimal HVAC setting, and apply this setting instead of the default one. They found that about 7% of energy could be saved by replacing the default HVAC settings by their optimized ones. Although sophisticated, these approaches control only the room temperature, rather than attempt to achieve a certain level of user comfort, as we do.

Turning off an HVAC system when no humans are present is an obvious technique to reduce energy use. It is also important to turn on heating in advance of human occupancy, because it can take tens of minutes to heat a cold building to tolerable levels. Most occupancy prediction methods use previously collected occupancy data. Lu et. al. [26] showed that by learning occupancy and sleep patterns, it is possible to save about 28% of energy use in a home environment. The PreHeat system [33] uses occupancy sensors to predict home occupancy patterns and automatically adjusts the HVAC temperature setpoint to save energy. If we consider the previous history of work space occupancy as a time series, and current time as a function of previous entries in the time series, we can learn this function as a Gaussian Process. This approach has been used in Erickson's and Rogers' papers [19, 32]. In another learning-based approach, home occupancy is modeled as a Markov chain and room occupancy is encoded as a state in the Markov model [18]. Mozer et. al. use a neural network and a lookup table to predict home occupancy in their Neuralthermostat project [28]. To build human interpretable model, Leephakpreeda [25] applied a grey model for occupancy prediction. Ardakanian et. al [15] uses sound and light level of a room to infer occupancy and applies POMDP for optimal HVAC control.

Most learning based models require relatively large amount of data in order to produce accurate predictions. Hence, to predict occupancy with limited historical occupancy data, there exists other approaches to employ some side channels to assist prediction. For example, Gupta et. al. [23] uses GPS sensors on mobile phones to estimate the arrival time of home owners and heat the house before they arrive.

HVAC control based on occupancy can be used in conjunction with our techniques to allow the HVAC controller to pre-heat or pre-cool a work space to achieve a target comfort level rather than a target temperature.

Chapter 7

Discussion

7.1 Extreme Sensing

The SPOT system, with its plethora of sensors, can be viewed as a somewhat extremal point in the space of HVAC control systems. We are keenly aware that our approach is hardware and compute intensive, and has a price point that may put it out of reach of most offices. Nevertheless, we believe that our approach is interesting for at least two reasons.

First, with the proliferation of sensing and compute systems, even high-end sensors such as the Microsoft Kinect will be much cheaper in the near future. Second, even if maintaining per-worker comfort is too expensive in terms of sensing, per-worker temperature control, a far more achievable goal, is cheap, effective, and well within reach in existing offices. We believe, therefore, that SPOT establishes an interesting data point in the thermal control design space.

7.2 Human Factors in Automation

Our discussion so far has assumed that the worker has little role to play in thermal control. In fact, workers themselves can be active participants in a thermal control system if they receive and act on energy-saving tips. For example, SPOT could, instead of turning on a heater, suggest to workers that they put on a jacket. This integration of humans into the control loop can be viewed as being unnecessarily intrusive. Nevertheless, we believe that, if properly presented to humans, such control actions can be both energy saving and marginally intrusive. We intend to explore this in future work.

7.3 Personalized Temperature Control

Thermal preferences vary from person to person. In a large building where temperature is centrally controlled, it is impossible to make everyone satisfied with the temperature. In order to make most people in a building feel comfortable, building operators usually take more aggressive temperature control measures. For example, temperatures are adjusted lower than necessary in summer and high than necessary in winter. While this could make less people complain about the temperature, it also wastes more energy.

In a environment that temperature is globally controller, there always exist people who are not satisfied. Personalized thermal control can solve this problem. Further, it brings new opportunities for energy-saving. An observation here is that we only need to preserve comfortable thermal environment at the places that people stay for the most of the time. For example, we can reduce the base temperature of a building to 20°C in winter, and keeps individual office rooms at the higher temperature based on the preference of the office owner. The temperature of common places like corridors are maintained by the central controller, have lower temperature than office rooms. However, since people rarely stay in the corridors for a long time, the lower temperature there will not make people feel uncomfortable.

Other common areas such as meeting rooms, however, should maintain a higher temperature as people may stay there for a long time. We can use the average PPV value or just the PMV value as the temperature control objective. The details of temperature control in meeting rooms are beyond the scope of this thesis.

7.4 Limitations

Our work has several inherent limitations that we discuss next.

Thermal isolation SPOT assumes the office worker's personal environment is relatively thermally isolated from others. This is true for office room environments. However, for open-plan offices where there is no thermal insulation between personal environments, this assumption does not hold. In this case, we can still apply the framework for temperature control. Instead of using the one office worker's PPV as the temperature control objective, we can use the median PPV of all workers in the same region.

Personalized work spaces We assume that most of the office workers stay in their own office for the majority of the time of a day. This assumption may not be valid in some situation. For example, office workers may go to a meeting room for a meeting. In this case, the personal work area assumption does not hold. However, a person's location in a building can be inferred by various techniques [16], [29]. With location information, we can identify the person and use the median PPV of all the people in the meeting room to control the temperature.

Cost Each SPOT system costs about $1,000. The detailed cost of each part is listed in Table 7.1. The major cost of SPOT is on the PC controller and the Kinect Sensor and we expect their prices will drop in the near future [9].

Item	**Model/Supplier**	**Quantity**	**Cost**
Dell PC	Inspiron 660s	1	$450
Kinect for Windows	Microsoft	1	$276
Z-Wave Switch	Aeon Labs Aeotec DSC06106-ZWUS	1	$59.95
Z-Wave Controller	Aeon Labs Aeotec Z-Stick-S2	1	$49.95
5° IR Sensor	MLX90614ESF-BCI-000-TU	1	$46.85
Heater	Sunbeam SLP 3300CN	1	$39.96
Microcontroller	Arduino Uno	1	$29.95
90° IR Sensor	MLX90614ESF-BAA	1	$19.95
Servo	Sparkfun	2	$8.95
Laser Pointer	Sparkfun	1	$7.95
Pan/Tilt Bracket	Sparkfun	1	$5.95
USB to RS232 Converter	Sparkfun	1	$5.95
Total			$1010.36

Table 7.1: Cost of SPOT

Calibration SPOT requires office worker vote on their thermal comfort when the system is first installed. This could be onerous by some workers. However, after the training/calibration stage, SPOT can work as an agent to control the temperature on behalf of the worker.

Validation Our experiment only validates that SPOT guarantees the user comfort. Since we do not have the access to the building temperature control system, we could not decrease the building level setpoint in winter. Therefore, we could not validate the hypothesis that SPOT can reduce the overall energy consumption of a building. However, since previous study [1] has shown that a two degree change of the temperature setpoint can result in a reduction of 37% home energy usage in summer, it is quite likely that SPOT can reduce the energy consumption of a building.

Environment SPOT is blind to windows that are open versus closed, to HVAC state, and user mobility. In our experiment, the office is controlled by a centralized heating system and the window is always closed. These factors may affect the effectiveness of SPOT in other environments.

Accuracy Due to various hardware limitations, SPOT gives inaccurate clothing level estimation in some cases. First, the infrared sensor is only accurate when the office worker is within 2.5 meters of the sensor. Second, Kinect sometimes detects people even when nobody is in the office. Third, when the office worker first enters the room, his or her cloth surface temperature is still affected by the outside air temperature. The clothing surface temperature converges to the correct value after 15 minutes.

Chapter 8

Conclusion

We have presented the design and implementation of SPOT, a smart personal thermal control system for buildings. SPOT introduced the personalized temperature control concept to the building thermal control system. Instead of keeping a universal temperature setpoint for all regions of a building, SPOT uses radiative heaters to provide a personalized temperature offset based on the preference of the office workers. SPOT is mainly based on three ideas.

First, we extend the PMV model to the PPV model, which enables SPOT to learn the user's personal thermal preference. SPOT automatically measures and estimates four environment variables (air temperature, humidity, air speed and radiant temperature) and two personal variables (clothing level and metabolic level). With these variables, instead of maintain a constant temperature, SPOT predicts the PPV value and adjust the temperature to maintain a constant PPV value such that the office worker's personal comfort is maintained.

Second, we design and implemented SPOT, which estimates the office worker's clothing level using a far infrared sensor. Assuming that people's skin surface temperature is maintained at a constant level, the infrared sensor measures the infrared attenuation by clothing and estimate the clothing level of the office worker. Based on our evaluation, under controlled environments, SPOT is relatively accurate on predicting user's clothing. The prediction accuracy of clothing estimation is only 0.091998.

Third, we use the LBMPC to learn the thermal environment parameters such as thermal capacity and thermal leakage rate. LBMPC predicts the control output given the control input of the system. We then use the optimal control framework to find the trade-off between energy-saving and user comfort. The optimal control framework and LBMPC works together to get the best temperature control schedule. For example, with occupancy prediction, SPOT turns the heater on prior to the arrival of the office worker. When the worker actually arrives, the temperature is maintained at the desired level.

We deployed SPOT in a real office environment and validated that it can maintain comfort over the course of a typical work day. Moreover, we have shown how SPOT allows a worker to trade off a reduction in comfort for saving energy. By changing the target PPV value, a energy-aware worker can save his or her energy consumption by slightly reducing personal comfort. We believe that our work demonstrates an interesting case study of how to maintain human comfort using extreme sensing. Finally, a limited version of our system, that only maintains personalized temperature offsets from a building-wide base setpoint, is not only easy to deploy, but is also likely to reduce overall building energy use.

APPENDICES

Appendix A

PMV Evaluation

The PMV [21] is computed as:

$$
\begin{aligned}
f_{pmv}(\mathbf{x}) = (0.303 \cdot exp(-0.036 \cdot M) + 0.028) \cdot \\
\left\{ \begin{array}{c}
(M - W) - 3.05 \cdot 10^{-3} \cdot (5733 - 6.99 \cdot (M - W) - p_a) \\
-0.42 \cdot ((M - W) - 58.15) - 1.7 \cdot 10^{-5} \cdot M \cdot (5867 - p_a) \\
-0.0014 \cdot M \cdot (34 - t_a) - 3.96 \cdot 10^{-8} \cdot f_{cl} \cdot ((t_{cl} + 273)^4 \\
-(\bar{t_r} + 273)^4) - f_{cl} \cdot h_c \cdot (t_{cl} - t_a)
\end{array} \right\}
\end{aligned} \tag{A.1}
$$

where t_{cl} is the clothing surface temperature, and W is the effective mechanical power which is 0 for most indoor activities.

Variable t_{cl} can be evaluated by:

$$
\begin{aligned}
t_{cl} = & 35.7 - 0.028 \cdot (M - W) - I_{cl} \cdot (3.96 \cdot 10^{-8} \cdot f_{cl} \cdot \\
& ((t_{cl} + 273)^4 - (\bar{t_r} + 273)^4) + f_{cl} \cdot h_c \cdot (t_{cl} - t_a))
\end{aligned} \tag{A.2}
$$

Variable h_c is the convective heat transfer coefficient, which is derived as

$$
h_c = \begin{cases} 2.38 \cdot |t_{cl} - t_a|^{0.25} & \text{if } 2.38 \cdot |t_{cl} - t_a|^{0.25} > 12.1 \cdot \sqrt{v_{ar}} \\ 12.1 \cdot \sqrt{v_{ar}} & \text{if } 2.38 \cdot |t_{cl} - t_a|^{0.25} < 12.1 \cdot \sqrt{v_{ar}} \end{cases} \tag{A.3}
$$

Variable f_{cl} is the clothing surface area factor, which is derived as:

$$
f_{cl} = \begin{cases} 1.00 + 1.290 I_{cl} & \text{if } I_{cl} \leq 0.078 m^2 \cdot K/W \\ 1.05 + 0.645 I_{cl} & \text{if } I_{cl} > 0.078 m^2 \cdot K/W \end{cases} \tag{A.4}
$$

In practice, the metabolic rate and the clothing insulation are first estimated by Table A.1 and Table A.2. Given the clothing insulation I_{cl}, we calculate the clothing surface temperature t_{cl} and the convective heat transfer coefficient h_c by iteratively applying Equation A.2 and A.3. Finally, by using Equation A.1 and A.4, we can estimated the Predicted Mean Vote.

Activity	**Metabolic Rate**	
	W/m^2	met
Reclining	46	0.8
Seated, relaxed	58	1.0
Sedentary activity	70	1.2
Standing, medium activity	93	1.6

Table A.1: Metabolic rates

Daily Wear Clothing	**Clothing Insulation (I_{cl})**	
	clo	$m^2 \cdot K/W$
Panties, T-shirt, shorts, light socks, sandals	0.30	0.050
Underpants, shirt with short sleeves, light trousers, light socks, shoes	0.50	0.080
Panties, petticoat, stockings, dress, shoes	0.70	0.105
Underwear, shirt, trousers, socks, shoes	0.70	0.110
Panties, shirt, trousers, jacket, socks, shoes	1.00	0.155
Panties, stockings, blouse, long skirt, jacket, shoes	1.10	0.170
Underwear with long sleeves and legs, shirt, trousers, V-neck sweater, jacket, socks, shoes	1.30	0.200
Underwear with short sleeves and legs, shirt, trousers, vest, jacket, coat, socks, shoes	1.50	0.230

Table A.2: Thermal insulation for different clothing level

References

[1] 2009 peaksaver Residential Air Conditioner Measurement and Verification Study. *Ontario Power Authority, May 2010.*

[2] ANSI/ASHRAE Standard 55-1992, Thermal Environmental Conditions for Human Occupancy. *Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers.*

[3] Arduino Software. http://arduino.cc/en/main/software.

[4] Energy Efficiency Fact Sheet: Heating, ventilation & Air Conditioning. *http://www.originenergy.com.au/files/SMEfs_HeatingAirCon.pdf*.

[5] Ergonomics of the Thermal Environment - Analytical Determination and Interpretation of Thermal Comfort using Calculation of the PMV and PPD Indices and Local Thermal Comfort Criteria. *ISO 7730:2005.*

[6] Gurobi Optimization. http://www.gurobi.com/.

[7] HVAC & Energy Systems. *http://canmetenergy.nrcan.gc.ca/buildings-communities/hvac-energy/908*.

[8] Kinect for Windows SDK. http://www.microsoft.com/en-us/kinectforwindows/.

[9] Microsoft's Surprise $40 Kinect Price Drop: Time to Buy One. http://techland.time.com/2012/08/22/microsofts-surprise-40-kinect-price-drop-time-to-buy-one/.

[10] MLX90614 family, Single and Dual Zone Infra Red Thermometer in TO-39. http://melexis.com/Asset/IR-sensor-thermometer-MLX90614-Datasheet-DownloadLink-5152.aspx.

[11] Nest, the learning thermostat. http://www.nest.com/.

[12] .NET Framework 4.0. http://msdn.microsoft.com/en-ca/library/w0x726c2(v=vs.100).aspx.

[13] Visual Studio 2010. http://msdn.microsoft.com/en-ca/library/dd831853(v=vs.100).aspx.

[14] Anil Aswani and Neal Master and Jay Taneja and Andrew Krioukov and David E. Culler and Claire Tomlin. Energy-Efficient Building HVAC Control Using Hybrid System LBMPC. *CoRR*, abs/1204.4717, 2012.

[15] O. Ardakanian and S. Keshav. Using Decision Making to Improve Energy Efficiency of Buildings. *ICAPS-10 POMDP Practitioners Workshop, May 2010.*

[16] P. Bahl and V.N. Padmanabhan. RADAR: An in-Building RF-based User Location and Tracking System. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784 vol.2.

[17] Richard de Dear and Gail Schiller Brager. Developing an Adaptive Model of Thermal Comfort and Preference. *UC Berkeley: Center for the Built Environment, 1998.*

[18] Varick L. Erickson and Alberto E. Cerpa. Occupancy based Demand Response HVAC Control Strategy. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 7–12, New York, NY, USA, 2010. ACM.

[19] Varick L. Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Amit Surana, Alberto E. Cerpa, Michael D. Sohn, and Satish Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, pages 19–24, New York, NY, USA, 2009. ACM.

[20] V.L. Erickson and A.E. Cerpa. Thermovote: Participatory Sensing for Efficient Building HVAC Conditioning. 2012.

[21] P. O. Fanger. Thermal comfort. *Analysis and Applications in Environmental Engineering, Danish Technical Press, Copenhagen, Denmark, 1970.*

[22] K.F. Fong, V.I. Hanby, and T.T. Chow. HVAC System Optimization for Energy Management by Evolutionary Programming. *Energy and Buildings*, 38(3):220 – 231, 2006.

[23] Manu Gupta, Stephen S. Intille, and Kent Larson. Adding GPS-Control to Traditional Thermostats: An Exploration of Potential Energy Savings and Design Challenges. In *Proceedings of the 7th International Conference on Pervasive Computing*, Pervasive '09, pages 95–114, Berlin, Heidelberg, 2009. Springer-Verlag.

[24] Byron W. Jones. Capabilities and Limitations of Thermal Models for Use in Thermal Comfort Standards. *Energy and Buildings*, 34(6):653 – 659, 2002. Special Issue on Thermal Comfort Standards.

[25] Thananchai Leephakpreeda. Adaptive Occupancy-based Lighting Control via Grey Prediction. *Building and Environment*, 40(7):881 – 886, 2005.

[26] Jiakang Lu, Tamim Sookoor, Vijay Srinivasan, Ge Gao, Brian Holben, John Stankovic, Eric Field, and Kamin Whitehouse. The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 211–224, New York, NY, USA, 2010. ACM.

[27] Geoffrey K Cook Mehdi Shahrestani, Runming Yao. Performance Characterisation of HVAC&R Systems for Building Energy Benchmark. *http://www.cibse.org/content/cibsesymposium2012/Poster026.pdf*.

[28] M.C. Mozer, L. Vidmar, R.H. Dodier, et al. The Neurothermostat: Predictive Optimal Control of Residential Heating Systems. *Advances in Neural Information Processing Systems*, pages 953–959, 1997.

[29] Lionel M Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P Patil. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless networks*, 10(6):701–710, 2004.

[30] J.F. Nicol and M.A. Humphreys. Adaptive Thermal Comfort and Sustainable Thermal Standards for Buildings. *Energy and Buildings*, 34(6):563 – 572, 2002. Special Issue on Thermal Comfort Standards.

[31] Luis Prez-Lombard, Jos Ortiz, and Christine Pout. A Review on Buildings Energy Consumption Information. *Energy and Buildings*, 40(3):394 – 398, 2008.

[32] Alex Rogers, Sasan Maleki, Siddhartha Ghosh, and Jennings Nicholas R. Adaptive Home Heating Control Through Gaussian Process Prediction and Mathematical Programming. In *Second International Workshop on Agent Technology for Energy Systems (ATES 2011)*, pages 71–78, May 2011. Event Dates: May 2011.

[33] James Scott, A.J. Bernheim Brush, John Krumm, Brian Meyers, Michael Hazas, Stephen Hodges, and Nicolas Villar. PreHeat: Controlling Home Heating Using Occupancy Prediction. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 281–290, New York, NY, USA, 2011. ACM.

[34] K.H. Yang and C.H. Su. An Approach to Building Energy Savings Using the PMV Index. *Building and Environment*, 32(1):25 – 30, 1997.