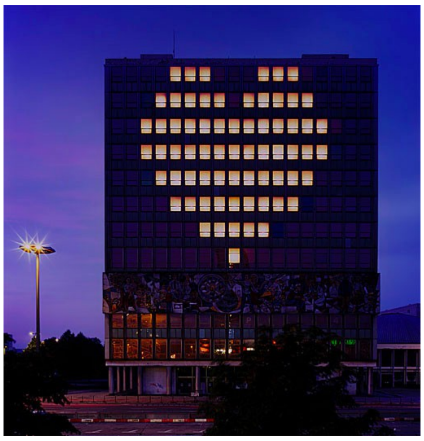
Adaptive Building Intelligence

Personalized blind control based on handcrafted adaptive controller

Andreas Fenkart
<afenkart@ini.phys.ethz.ch>



Advisors
Tobi Delbrueck, Institute of Neuroinformatics, ETH/University Zurich

uni | eth | zürich
Institute of Neuroinformatics
University and ETH Zurich
http://www.ini.ethz.ch

Abstract

This thesis describes a prototype blind control system for office buildings, including computer software that serves as an additional personal presence detector by sensing mouse and keyboard activity. This software allows users to control lights and blinds at their fingertips. The interactions are personalized and form the basis of our analysis, consisting of logged data from ten users that used a blind more than 20 times within a period of three months.

From observing collected data, we find that user interaction is very sparse. Hence we believe that no general-purpose algorithm of machine learning will converge to a state of user satisfaction. Our premise is that a human-designed controller that incorporates our understanding of building control and that contains just a few parameters that are adaptive to user interaction, will be able to predict user preferences and hence lead to a reduction of user interactions.

We suggest that the two main patterns of user interaction that are governing blind control are glare avoidance from direct light and control of indoor luminance. We outline how to divide the interactions between these two patterns by using hierarchical clustering and how to apply regression by Gaussian processes to relate blind height to outside irradiance and sunniness.

Currently we have strong evidence that most logged interactions are due to glare avoidance from direct light. The remaining interactions are further subdivided in neglectable and non-neglectable outside irradiance. In case of non-neglectable irradiance we are able to predict blind height of three out of five users whose office is never occluded by outside obstacles and four out of 10 users including offices with times of occlusion. Prediction errors for these users are less than 18% on average and maximally up to 54% as computed by a test-set/training-set of equal size.

The presented algorithms operate on abstractions such as sun-position and sunniness. Implementation is built on top of the ABI/ABLE multi-agent platform. Unfortunately software bugs combined with problems with the blind hardware tampered the evaluation to prove the usefulness under real conditions. Nonetheless they are built on a clean state-of-the-art and flexible framework ready for rapid prototyping by MATLAB generated C-functions.

Contents

1 Acknowledgements **5**

I Introduction **6**

2 Introduction **7**

3 Related work **8**

4 Setup **10**

5 Underlying assumptions about users **11**

II Method **12**

6 Presence detection **13**
- 6.1 Current solution . . . 15
- 6.2 Future extension . . . 15

7 Controllers **20**
- 7.1 Light controller . . . 21
  - 7.1.1 Current controllers . . . 21
  - 7.1.2 Future Extension . . . 22
- 7.2 Blind controller . . . 24
  - 7.2.1 Glare avoidance/Shadowline clustering . . . 25
  - 7.2.2 Luminance control . . . 25
  - 7.2.3 Evaluation . . . 26
  - 7.2.4 Experiment . . . 26

8 Preprocessing **29**
- 8.1 Shadow-line computation . . . 30
- 8.2 Irradiance/Illuminance . . . 34
  - 8.2.1 Which sensors to use? . . . 34
  - 8.2.2 Pseudo-linearity of irradiance and illuminance . . . 34
  - 8.2.3 Accessing sensor data . . . 36
  - 8.2.4 Astronomical prediction . . . 36
- 8.3 Sunniness . . . 39
  - 8.3.1 Heuristic . . . 39
  - 8.3.2 Model . . . 40
  - 8.3.3 Stability of weather . . . 42

III Software Architecture **43**

9 Framework **44**
- 9.1 ABI/ABLE framework . . . 44
- 9.2 PC-Bus . . . 45
- 9.3 Virtual Bus . . . 45

10 Multi agent system **47**
10.1 Presence . . . 47
10.1.1 UserPresenceAgent . . . 47
10.1.2 ClusterPresenceAgent . . . 48
10.2 Preprocessing . . . 48
10.2.1 IrradianceAgent . . . 48
10.2.2 ShadowAgent . . . 48
10.3 Controllers . . . 50

11 Description of MATLAB functions **57**
11.1 Blind control . . . 57
11.2 Irradiance interpolation . . . 58
11.3 Gaussian Processes . . . 60

IV Results **62**

12 Irradiance **63**

13 Sunniness **65**

14 Data-logging results **66**
14.1 Example User . . . 66
14.2 Remaining Users . . . 72
14.3 Prediction errors . . . 77
14.3.1 Luminance control pattern . . . 77
14.3.2 Discussion . . . 79

15 Experiments with the blind controller **80**
15.1 Building description . . . 80
15.2 User parameter evaluation . . . 82
15.3 User interaction evaluation . . . 84

16 Conclusions **86**
16.1 Future Work . . . 87

V Appendix **88**

A Gaussian Processes **89**

B MATLAB to Java **92**

C Description of weather station **95**

1 Acknowledgements

I would like to thank Ueli Rutishauser, Alain Schaefer, Jonas Trindler and Raphael Zwiker for their work on the ABI framework and providing me with such a good starting point. Especially I would like to thank Raphael Zwiker for his advice on software architecture and the patient discussions. Without him none of our tempting ideas would have made it into a functional device. I would like to thank Tobi Delbruck for his advice and moral support. It was his ingenious yet simple idea of using computers as additional presence devices and it was him who came up with the idea of representing user interactions in continuous space. I would like to thank Niki Shraudolph for sharing his experience in machine learning and giving valuable hints on how to seize the problem. Thank goes to Thomas Ott for giving me an insight on clustering and what algorithms might be useful for our problems. Daniel Kiper for explanation about biological vision, Josef Joller who came up with the idea of using wireless motes for presence detection and providing samples of them for evaluation. Shai Litvak for bringing up controversial ideas which pointed me into new directions when I got stuck on a problem. Thank goes to the supply service of the university that enabled us to control the building devices and showed excellent support in case problems with the devices occurred. I also would like thank those that revised my thesis and made it more readable. These are Yann Baud, Daniel Rubin, Ueli Rutishauser, Samuel Zahnd and most people already mentionned before.

Part I
Introduction

In this part we will motivate the problem and relate our approach to the work of others.

2 Introduction

Although there are some historical examples of building automation, such as Egyptian water clocks or Roman heating systems, computerized building automation is quite young. Most readers might remember the times when office blinds were operated by a mechanical crank the same that are still in use at home. Our school sensitized people to lower the blinds during the night because they provide additional isolation hence saving lots of energy. It just worked half-way, when turning back on the way home, counting the number of lowered blind, at least half of the blinds were always up. Today all newly built office buildings have automated blind and lighting systems mainly due to the fact of saving energy in unoccupied rooms. But there are more aspects than just energy-efficiency as illustrated in figure 1.

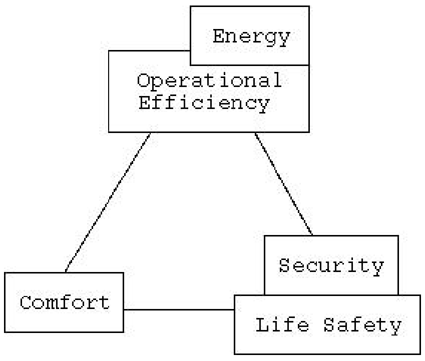


Figure 1: Conflicting aims in automated building control

Energy stands for shutting off unused building devices. But also using blinds as passive cooling or heating devices, dependant on whether sun irradiance should be kept outside or heat kept inside.

Operational efficiency Because these systems increase the complexity they add an additional overhead in maintenance. At the same time building automation with its central control possibility also provides a benefit, e. g. currently all outside doors of the building have to be manually locked by a security person, central building control would not only automate this task but also detect which doors are not closed properly.

Security Burglar or espionage protection, but also protecting building devices from extreme weather conditions, such storm wind.

Life Safety related to security but with the aim of protecting inhabitants. e. g. fire alarms.

Comfort What you and me are doing most when interacting with lights, blinds, HVAC[1], ...

Energy and comfort are often in conflict with each other except in case of unoccupancy because nobody needs to be made comfortable then. Guillemin[1]

[1]Heating, Ventilation and Air Conditioning

built a controlling system that optimized energy during room occupancy. He showed that such strategies can save up to 26% of energy used for heating and lighting. He did not separate lighting from heating, so the savings due to lighting alone were not evaluated.

It is our intention to continue along this path but to focus our attention on what users desire building devices to behave like. Our interest is to investigate how users interact with building devices and to find the underlying motivations or patterns of interaction. Currently we focus on blind devices, though we could also control light devices (see also section 4 below). But the problem turned out to be larger than we thought at first so we had restrict ourselves. We chose blinds because blind position does affect interactions with light, but the state of light does not affect interactions with blinds. At least this is our assumption which is backed on simple tests (see section 7.1 on page 21) that suggested making a relatively good performing light controller seems easy while making a real good one requires a thorough understanding of the impact of blind position on inside illumination.

We think that if self adaptive building control is ever going to be commercialized then because of the energy savings that can be achieved with it. In this context understanding blind interactions is a useful task itself because blinds are passive cooling and heating devices a large amount of energy can be saved by choosing energy efficient blind positions to remove load from HVAC system or use solar gains as an additional heating source. But such techniques are prohibited unless the range of user comfortable positions are well known. If we fail to predict this comfort range accurately enough the user will reject the system or will be annoyed by having to correct the systems choice regularly. This last note applies also to the study of Guillemin[1] where the cumulative number of user interactions with blind devices did not converge during the time of the experiment but kept increasing, which is an indicator that the system did not predict users preference accurately enough.

3 Related work

Not much research on user interaction with lighting and blinds is available. Foster et al.[6] recorded facades of office buildings on video tape. They used a self-developed indexing system to relate blind position to the current weather situation. They showed that the average blind occlusion level was almost always 50% during dismal winter days, and 60% during the sunniest days in spring. They came up with the following reasons why people might not raise the blinds more often:

1. Ergonomically, blinds may not be easy to control and therefore building occupants will only alter them when exposed to extreme environmental discomfort.

2. The layout of a particular office and its visual display terminals may mean that under most conditions glare[2] occurs if the blinds are not fully down and slats closed.

[2]In this paper glare is meant as lighting that obscures reading computer screens. It probably includes direct sun light that causes reflection on the screen, but also diffuse light that leads to too high contrast between screen and its surround.

3. Occupants of buildings may always require certain blinds to be down and shut for privacy.
4. Individuals may not feel empowered to alter the blinds within their open plan office as they may feel afraid of upsetting their colleagues.

The study concluded that the justification of having big windows to take profit from natural daylight is vaporized because blinds are mostly closed all the time. But for us it is interesting to see that lower blind positions seem more likely accepted by the users. So this could be kind of compromise policy in case of conflicting user interests in such a case it seems to be most promising to go with the lower value.

Most studies in building control were written by the architectural and natural environment community, such as Foster et al.[6], Guillemin[1], Lee et al.[2] and Vine et al.[3]. The primary focus was on energy-efficiency while user comfort was a subsidiary constraint in finding the optimal device states. User adaptation was only of concern as to reduce the rejection of the automated system.

Lee[2] used two identical test rooms stuffed with sensors[3]. In one room they simulated user behavior by setting the blind[4] to one of three predefined positions throughout the day while keeping the *workplace illuminance* at a minimum level of 510lx but not constraining it above. In the second room they deployed a blind controller that blocked direct sun and maintained a daylight level in the design range of 540-700lx. Because their system was designed to block direct sunlight, the cooling load throughout the year was rendered fairly insensitive to changes in solar radiation levels, given a certain temperature. The normalization to temperature is important because blinds can only block irradiance but not the heat exchange through the window. They showed that for the climate of Oakland California a reduction of up to 30% in daily cooling load could be achieved.

In a related study Vine et al[3] evaluated a similar controller by a sample of 14 users. The users were '*. . . reading, writing or doing computer-related work . . .*' during a 3 hour period of '*. . . predominantly clear and sunny weather . . .*'. Evaluation was done by questionnaires, that showed that most users($>$50%) were satisfied with the overall (natural) lighting conditions but about 30% felt the light levels were too dim. When asked what they liked best about the system some mentionned the maximal use of sunlight; what most people disliked was not having '*. . . enough direct control over amount of natural light into the room . . .*'.

Guillemin[1] uses fuzzy rules to incorporate expert knowledge. Adaptation to user wishes is provided through tuning the fuzzy membership functions by a genetic algorithm. The main aim of the system is to enforce energy-efficiency also in case of user presence. It was running for 9 months in the LESO building at the EPFL Lausanne. Guillemin showed that energy savings of up to 26% compared to a manual system could be achieved while the system was only rejected by 5% of the users.

Hagras et al[7] focuses primarily on user-comfort. Similarly to the aforementioned study they use fuzzy logic, but user adaptation is done through automatic

[3]15 illuminance sensors; 10 on working height, 2 vertical and 1 horizontal on the ceiling, 2 pointing to the window

[4]Only the slat angle could be controlled, the blind covered the entire vertical height of the window and was not retractable.

generation of rules. They use a supervised learning algorithm that gets its input by interaction with the user. They call this interaction a *machine-user dialog* but do not describe it explicitly. It seems that the user has to set the controller into learning state then either through a computer interface or by operating with building devices the user generates a training set. The optimization is done through genetic algorithms. It is not mentionned how personal presence detection is done, but they outline how they generate a rule-base for each user.

Rutishauser et al[4], Trindler and Zwiker[5] took a similar approach. The difference is that they use an unsupervised learning algorithm that does not require a special user interface. The training of the controller is done through normal user interactions with building devices. The latter study additionally uses a short and long-term memory for resolving conflicting user interactions.

The controllers from Trindler and Zwiker are running in a concurrent testing phase during the development of our controllers. An additional testing phase was necessary, because of software bugs discovered in their controllers, hence the results they presented in their thesis are not showing optimal performance. But we suppose that some problems might still remain; To make their general purpose algorithm converge faster they generate more user input by assuming device states that are not overriden by users as a silent agreement that users are satisified with the current device state. This assumption is probably not true because users are in general indifferent to device states and only interact when they are really disturbed, so there is a qualitative difference between non-action and action. Although the test is still running we think that the controller does very little because most of the time it learns that the current device states are correct.

4 Setup

The environment is the Institute of Neuroinformatics (INI) on the Irchel campus of the university of Zurich. The G floor of the institute has an installed Local Operating Network(LON)[8] building automation network that was donated to INI by the university building automation services. This network allows control of lights and blinds and provides access to wall switches, PIR(passive infrared) presence sensors, and both inside and outside[5] illuminance sensors.

A full featured air-conditioning(HVAC) is not installed as the climate of Zurich[6] is too mild for such devices to be useful. There is a ventilation system for fresh air but can not be controlled from LON. The same applies to the heating system that can only be controlled mechanically by a knob.

The offices are on the north, east and west faces of the building. The south side forms the entrance and public area. See figure 49 on page 80 for illustration. The offices have large window openings that can be covered by a Venetian blind system. Currently there is a problem with logging the slat angle because the motor system is too simple to set the slat angle consistently. Each blind can be operated independently but some were configured to be operated in groups of two, see figure 44 on page 72. In general the light can only be turned on or off while some conference rooms also have dimmers installed.

[5]Actually irradiance sensors, but see 8.2 on page 34 for conversion
[6]Zurich: 47°North, 8°East

The unique property of the setup is a computer software called *pc-bus* that allows personal presence detection of users by sensing mouse and keyboard interactions. It helps to overcome three problems of traditional passive infrared sensors namely i) PIR sensors have a very limited range that is further decreased by furniture, ii) they sense only movement and not presence, iii) they are not personalized. Combined with the fact that the cost of such a client consists mainly in software development and maintenance makes them very competitive with expensive PIR sensors.

Additionally the pc-bus client enables users to set blind and light position at their finger tips. Most users find this more comfortable, because they do not have to get up and go physically to wall-switches. The emphasize is that it will increase the number of interactions with building devices. All these interactions can be assigned to an individual user because clients are authenticated/authorized based on a login name.

Earlier projects developed a sophisticated agent-oriented framework Rutishauser et al[4], Trindler and Zwiker[5]. The low-level access to the building automation bus was abstracted by a Java interface, advertising devices through a whitepage-lookup mechanism. If an agent needs to register for device updates it looks up the device from this whitepage-service, it will then receive a message whenever a new value is available. The updates are communicated through an asynchronous messaging service. The system proved its stability during several months and came with an evolved built-in mechanism for logging building-bus and pc-bus traffic.

5 Underlying assumptions about users

Some users explained that their prefereed blind position is depends also on their current mood and that environment conditions are only part of the constraints. Currently we do not investigate inner states of users but assume constant behavior in their interactions. The second assumption is that users are spatially constant in one room. This means that some users might work in more than one room, but in each of them they work always at the same place. This assumption is important when it comes to blind controlling because glare from direct light depends on geometrical conditions.

Part II
Method

This part describes the division of the problem into subproblems and describes each of them.

6 Presence detection

Even though we will focus on blind devices we need to touch the problem of presence detection to build a functional controlling system. Presence detection is one of the core services of the system because occupancy of the cluster is the arbiter deciding which set of controllers is currently active; see figure 2 for illustration. Controlling an unoccupied room is easier because the single aim of controlling is to save energy given that security is not a concern. This problem can be expressed physically and the resulting controller has the form of a non-adaptive state-machine. See Guillemin[1] for a discussion of energy flow through windows depending on blind position. Controlling the light is even simpler as it has to be off when nobody needs it.

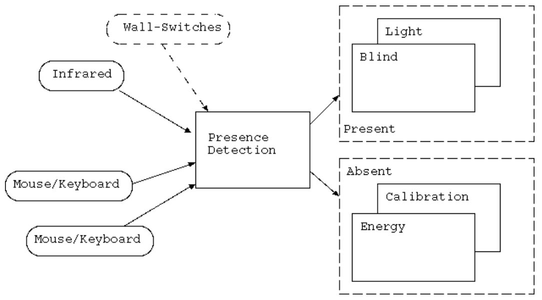


Figure 2: *Control arbiter* depending on whether a room or cluster is occupied a different set of controllers is active. They differ in complexity and their optimization goals.

In an occupied room the problem gets more difficult because different users likely have different preferences. In our approach we try to address the needs of each user individually. Based on their previous interactions we try to predict what they might want next. Hence for controlling it is a prerequisite to know which users are present because absent users are not considered when taking an action.

Presence detection can be qualified in terms of *prediction* and *reliability.* Controllers for different building devices require different qualities.

Prediction Prediction means forecasting users arrival or departure in a time-scale of minutes to hours. it becomes important when controlling devices with significant latency between sending a command until the desired effect is established. E. g. heating takes minutes to hours until a given room temperature is met, the same with less latency applies for cooling. In our case no such device is present or can be controlled. All devices that can be controlled in this study are of immediate effect and can be triggered on user detection, hence prediction is irrelevant.

Reliability A lack of reliability means declaring a user absent while he is present and vice versa. Reliability is important, because controllers only consider preferences of present users. So independent of how accurate the controller is in predicting user wishes, user satisfaction is not reached if users are erroneously assumed absent. To give a simple example: in single-person offices the energy controller will repeatedly turn off the light when the user is declared absent, hence the entire room assumed unoccupied. This might seem a trivial problem but throughout the experiment it was the main source of complaints.

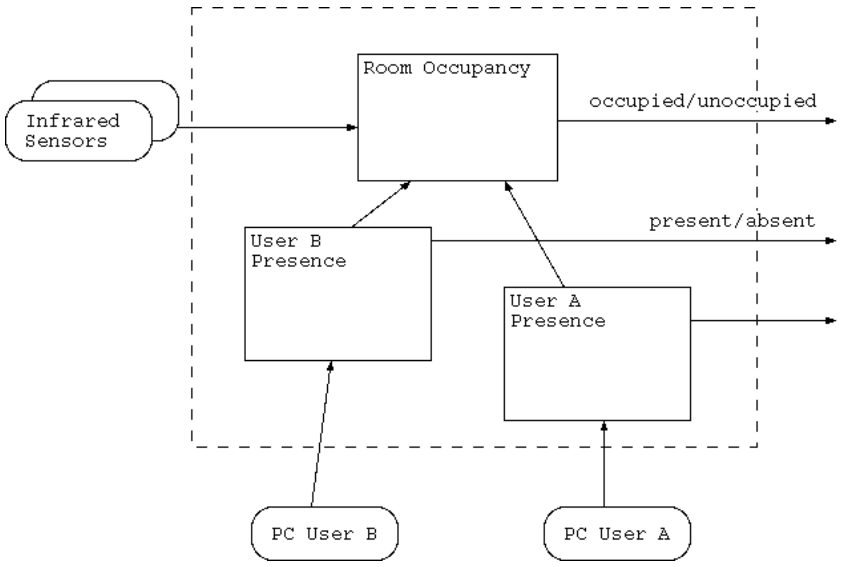


Figure 3: Hierarchy of presence signals. Entities outside the dashed square represent sensors. The arrows shows how sensors are mapped onto presence entities.

Figure 4: Infrared presence detector

Available sensors See figure 3 for illustration of the hierarchy between individual and cluster presence signals. The boxes outside the dashed square represent how physical sensors are mapped into the hierarchy. Currently the following sensors are available for presence detection.

- *Passive InfraRed detector*(PIR) illustrated on figure 4. Actually a bundle of several sensors (see also section 7.1 on page 21) mounted in every room. Emits signals on movement detection, illustrated by figure 6. Does non-personal presence detection.
- *Pc-bus* presence detector. Computer software that queries operating system for last mice or keyboard interactions. Does personal presence detection.

Both sensors have their weaknesses and strengths, neither provides a reliable personal presence detection. Genuinely individual users can only be sensed by the pc-bus detector. As an exception the PIR can be assigned to an individual user in single-person offices, but in general its signal is anonymous. The advantage of the PIR is that users do not have to run the client, it works also while doing non-computer related work or working with multiple computers when not all of them are running an instance of the client. Unfortunately the reliable range of the PIR is only 2-3m, outside of which it reports only large movements such as walking or waving hands. Furniture or other obstacles are further decreasing the detection range.

6.1 Current solution

As the focus of this work is to find structure in user interactions with blinds, we did not focus much on presence detection. Users participating in the controller evaluation were instructed to keep the client running all the time. See section 7.2 on page 24 of how users were sensitized to presence detection. User rejection of running the client all the time had two main reasons. First users forgot to start it or they wanted to save resources such as memory or space on their computer desktops. The second concern was intrusion into privacy, as the idea of surveillance is not pleasent at all. Due to the fact that this is a research project users did not insist much on the latter.

A conservative approach was taken to increase the reliability of detection. The propagation of absent signals was delayed by a large timeout. The default value was set to 20 minutes but was voluntarily increased when user requested it. As we did not focus on energy saving but solely on user satisfaction, this didn't bias our experiment.

6.2 Future extension

Bayesian approach Presence can be interpreted as a varying probability depending on the time of day denoted P_t, see figure 5. During the following discussion we will not talk of individual sensors, such as pc-bus or PIR, but of the detection system as a whole. The presence probability can be split as a conditional sum of reliable and unreliable detection by the detection system:

$$P_t(\text{present}) = P_t(\text{present}|\overline{\text{sensor}})P_t(\overline{\text{sensor}}) + P_t(\text{present}|\text{sensor})P_t(\text{sensor})$$

The notation $P_t(\overline{\text{sensor}})$ stands for *probability of no sensor signal at time t.* The term P_t from figure 5 is the short form for $P_t(\text{present})$. We will focus on the term $P_t(\text{present}|\overline{\text{sensor}})P(\overline{sensor})$ now that reflects non-detection of a present

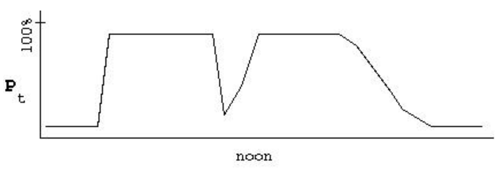


Figure 5: Hypothetical presence distribution of a user throughout a working day, based on typical measurements. The vertical axis represents presence probability in percent, the horizontal axis reflects daytime. The model says that users arrive in a small time-frame, probably due to public transport. During midday there is a dip of variable size. The presence signal slowly fades out probably depending on overtime.

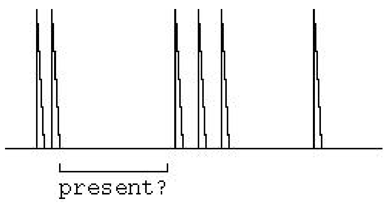


Figure 6: The PIR sensor reports detected movements. The line is flat if the absence of movement, detection is symbolized by peaks.

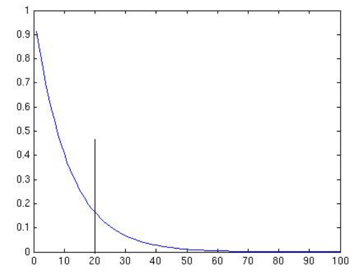


Figure 7: Decreasing presence likelyhood depending on time since last presence signal. The vertical bar represents a fixed timeout of 20 minutes.

user. By using Bayes law we flip the conditions of the conditional probability in this term.

$$P_t(present|\overline{sensor}) = \frac{P_t(\overline{sensor}|\text{present})P_t(\text{present})}{P_t(\overline{sensor}|\text{present})P_t(\text{present}) + P_t(\overline{sensor}|\overline{present})P_t(\overline{present})}$$

If we assume that no absent users are sensed present[7], which can be expressed as:

$$P_t(\overline{sensor}|\overline{present}) = 1$$

Then the previous expression can be simplified to

$$P_t(present|\overline{sensor}) = \frac{P_t(\overline{sensor}|present)P_t(present)}{(P_t(\overline{sensor}|present) - 1)P_t(present) + 1}$$

$P_t(present)$ is interpreted as the prior probability here, hence the remaining unknown is $P_t(\overline{sensor}|present)$, which reflects the unreliability of the detection system. This entity can not further be deduced but has to be measured somehow (see section 6.2 coming next).

The unreliability of the detection system is mainly influenced by the geometrical conditions such as the distance between PIR and the user's workplace and different working behaviors such as long periods of reading. There might also be a dependency on daytime schedule, like socializing in the morning or during coffee breaks. But let's assume reliability to be daytime invariant.

The detection reliability could be modeled as …

$$P_t(present|\overline{sensor}) = \frac{1}{e^{at}},$$

where a is constant and t is the time since the user was reported present for the last time, see figure 7.

Depending on where the user is located and the likelihood he is reading, the conditional probability given the last presence detection differs for each user. The conditional probability weighted by the prior distribution (see figure 5) results in different presence probabilities throughout the day. By comparing the probabilities to a given threshold the user is declared present or absent. The final delay until the user is declared absent will differ throughout the day. For example, at 9 o'clock AM the absence of a presence signal for 20 minutes will not result in declaring the user absent whereas at 9 PM it will.

Instead of computing the probability of presence and assuming absence as default value, we could compute the probability of absence and assume presence as the default state. In the previous example the declaration of present/absent would be flipped then. At 9 o'clock AM the absence of a presence signal for 20 minutes will result in being declared absent whereas at 9 PM it will not, because in the evening it is more uncertain that the user is actually absent.

Currently this probabilistic analysis is only theoretical because we just assume a fixed timeout, which can be interpreted as a flat prior and an identical

[7]Some mice have jitters, so some users were reported present 24h / day during several days.

presence likelihood for all users. To measure the unreliability factor of the detection system the *true* presence signal must be known or approximated. The next paragraph brings up new ideas how this could be done.

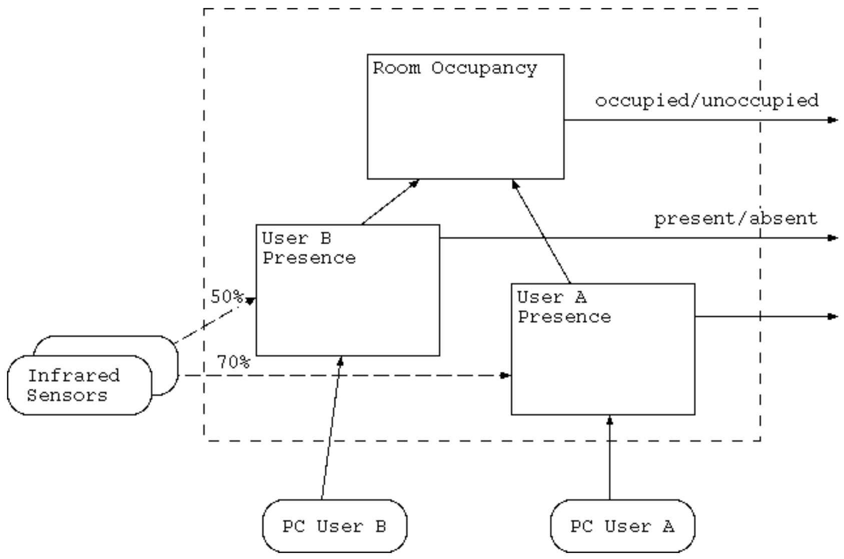


Figure 8: Attributing the PIR sensor to individual user. This is not done by hard-assignments as for pc-bus detectors but by soft-assignments. The probabilities do not necessarily sum up to 100% as the signal might be triggered by multiple users.

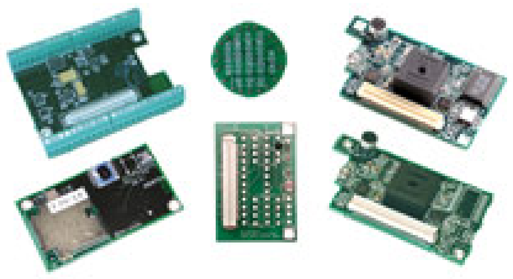


Figure 9: *Motes* little wireless devices, the middle top one has about the diameter of a coin.

Personalizing PIR detectors Currently we do not distribute the PIR signals among the users; they are used only for sensing occupancy of the entire cluster. Compare figure 3 to figure 8. But it is likely that some users trigger a presence signal more often than others. In rare cases such as single-person offices we can hard-assign the PIR signals to the single person in that room.

The idea is to extend this to multiple users, where the assignment PIR to user is not one-to-one.

By using supervised learning one might find the correlation between PIR signals and effective user presence. For a limited time each user would carry a *mote* with them, see figure 9. These are little wireless devices with a range of a few meters, hence if a base station is placed in the cluster it will report reliably which users are currently present. By using this data we could adjust the assignments between the PIR detector and users by supervised learning.

Though the PIR can be personalized, it can hardly be used to detect the arrival of a user. For example, if the person having the strongest assignment with the PIR becomes absent he could be resurrected by a PIR signal triggered by another user. So it seems to be difficult to personalize the PIR without a *true* personal detector, such as the pc-bus detector.

Additional sensors Despite these methods users located behind a bookshelf, hence invisible to the PIR detector while not working on their computers, will hardly be detected correctly without using additional sensors. Such sensors could be counting arrival/departure of persons, or sensing if movement is local or has a net direction.

7 Controllers

I was inspired by the section *Comprehensive Control System Description* in the thesis of Antoine Guillemin[1]. The value reported by the sensor might be accurate only very locally, the sensored entity might have completely different values at another position in the room. In the referenced section Guillemin models the distribution of each sensored entity in the room. Temperature for example is dependant on outside temperature, temperature of the the thermal mass[8], sun irradiance and heating. Hence there is no single temperature but a temperature distribution throughout the room. A drybulb sensor mounted on the ceiling is influenced by the proximity of nearby concrete, also it might get direct sun irradiance and the height also has its influence. Hence the relation between what the sensor measures and what that the user perceives does not have to be linear at all. The same applies to other entities such as illuminance, CO_2 concentration ...

From observing users interacting with blinds, we found that they are very definite in whether they want the blind to go up or down. Some of them invest a lot of time for finding the optimal position. The idea we got from this is that user interaction has a continuous function of mapping the environment into a tolerable blind setting. This observation also applies to user interactions with lights, except that in our they are not continuous because they can only be turned on or off.

A number of studies, (e. g. related work in section 3 on page 8) used fuzzy logic in building controllers, probably mainly due to the fact that controllers could be designed in the arguing space of humans. In fact, this original rationale for the use of fuzzy logic was not employed by all studies because some of them automatically generated the rules. In case the rules were static and manually chosen by the developer, we found that this approach might also be misleading, which is outlined by the following example. On days with similar outside irradiance but different overcast conditions one could reason that users will prefer the lower blind position on the sunny days, whereas in fact most users chose the lower position on cloudy days. See section 14.1 on page 70 for a possible explanation.

Based on this observation we found that first we need to understand better what users really do, by observing the actions in relation to the environment. Our assumption was that on an unconscious level users are mapping environmental conditions perfectly onto blind positions. If this is true and the underlying system was known and not too complicated, we could calibrate a model for each user by its input and then use it to predict future preferences. The use of a strong model is crucial because the average user produces only about 20-30 data points per device per month. With such little user input general methods such as neural-networks are not practical as there is simply not enough data to train a more elaborate network without overfitting.
So the method chosen here is to construct the hidden structure by a human and then use simple algorithms to split the data accordingly and extract the few parameters that fully describe this structure.

[8]building walls

7.1 Light controller

The most simple light controller is defining a threshold so that the light is turned off if illuminance is above the threshold or turned on if it is below (see below for how inside illuminance is measured). This solution will cause light turning on/off all the time if the inside illuminance is jittering around the threshold. An additional problem is that controller's action itself is likely rejected by users independent of whether the effect of the action is appreciated or not. These problems can be solved by taking two thresholds which implement a hysteresis separating illuminance levels of on/off events. The light is turned on if illumination drops below the turn-on threshold, once turned on the illuminance has to raise above the turn-off threshold before the light is turned off again. The same applies to turning off the light; once illuminance exceeds the turn-off threshold illuminance has to drop considerably before the light is turned on again. This solution ensures i) that the light is not turned on/off repeatedly ii) the action is really necessary, minimizing user rejection.

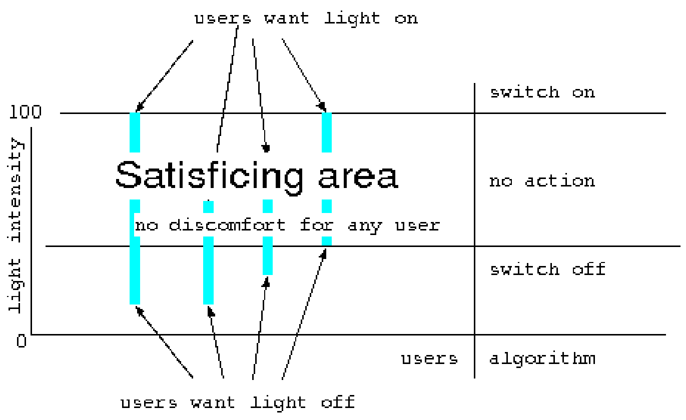


Figure 10: The vertical bars indicate the satisfaction area of an individual user. In this area the light might be on or off, the user doesn't care. The light should be turned off if illuminance goes above respectively turned on if it falls below the satisfaction area . The right part of the illustration denotes the actions of the controller. Combining all satisfaction area into a satisficing area we hope to resolve preference conflicts.

Different users have different needs, so the controller needs a policy how to resolve conflict situations. See figure 10 for illustration. The basic assumption is that users do not mind the light to be on but they care if the light is off in case they need it. This means that the satisficing area, the range of light values that all user can agree on, is defined by the maximum over all user turn-off and turn-on values. This way it is ensured that the light is never off when a user still needs it.

7.1.1 Current controllers

The HTS sensor (see figure 4 on page 14) contains a daylight measurement sensor. The sensor contains filters to discard light of fluorescent tubes [20] hence it is ideal for controlling lights that can only be turned on/off; if daylight value is below the turn-on threshold the light has to be turned on if it is above the turn-off threshold it has to be turned off.

The underlying assumptions is that what the user perceives is linearly correlated what the sensor reports. This is obviously not the case for users sitting next to the window, their perceived illuminance is different from the illuminance reported by the sensor because the latter measures a value more far from the window. We assume that next to window illuminance is better matched by outside illuminance ignoring blind position for a while. Indoor conditions such as artificial light can be neglected completely, because natural light is always a magnitude larger than artificial light. The trivial exception of this non-linearity of indoor sensor and perception is during night-time, because then no sun-light is present and both users and sensors report a daylight value of zero. So the non-linearity between user and sensor is only a problem for the turn-off threshold.

Ignoring users next to the window we built a first shot controller with two compile-time constants for upper and lower threshold. After evaluating the controller by ourselves we extended it by an adaptive approach: For each user interaction we collect the inside illuminance prior to interaction and whether the user turned the light on or off. By k-mean clustering we separate the user input into 2 clusters based on the illuminance values, so always one cluster contains user inputs at times of higher illuminance and the other of lower illuminances further called higher and lower cluster. From the higher cluster all turn-on events are discarded and from the remaining turn-off events the average was computed which is taken as the turn-off threshold. The same way the turn-on threshold was computed from the lower cluster by taking the average over the turn-on events.

There are two obvious problems with this implementation so far: i) There has to be found a solution for users sitting next to the window, when they turn on/off the light it might have little to do with the inside illuminance sensor. ii) Turn-on events in the high cluster should not simply be discarded. Maybe it never gets bright enough for users to turn off the light during their presence, hence their turn-off value is never increased. We should define a default turn-off level which is increment by a constant for each turn-on events. To prevent the turn-off level to increase to infinity the increments should be weighted by the time since issued, e. g. the increment per turn-on event should decay to zero as a function of time since issued.

7.1.2 Future Extension

The systematic error of the controller is the assumption that turning the light on/off only depends on inside illumination. During evaluation it turned out that this assumption is actually incorrect, because there were lot of contradicting user inputs (not shown). We should not forget that light usage is not always consistent with illuminance levels but depends also on user's activity such as reading requires more light than working on a computer. A discussion with Daniel Kiper brought some insight into another problematic; the receptors of the retina adapt to a wide range of illuminance so it is impossible for us to tell the absolute illuminance value like a luxmeter does, but we are very sensitive to changes in illuminance. The latter is probably best known when the we enter a dark room when for a moment we can't see till our receptors adapt to night vision.

The sensitivity to illuminance changes is likely to be one of the reasons for light controlling, which was explained to me the following way: If half of the the users's visual field is very bright whereas the other is in relative dark our receptors can not adapt to an average value. A naive thought was that part of the retina will adapt to the bright level and the other to the dark but this is impossible as our eyes are making saccades without our conciousness. So our receptors can never adapt and our receptors are always maximally stimulated. Such an environment must be very irritating for our eyes.

For our light controller it means that knowing the inside illuminance is not enough. In case the blind is in a high position and the area near the window is enlightened by sunlight whereas more remote areas of the room are in relative darkness, we assume that a user will turn on the light to average illumaninance in his surround. This although the average illuminance in the room might be even higher than at times where the user turned the light off. Alternatively of turning the light on the user could actually also lower the blind but turning on the light is probably more comfortable because faster. Another problem might arise if the computer screen is very bright while the surround is in complete dark. Mostly users will turn the light on then although Daniel Kiper mentionned that it would be enough to lower the monitors illuminance. Literally he said that it it is never necessary to turn the light on while working on a computer screen because it is always possible to find a monitor configuration that is acceptable to users.

Because visual conditions depends highly on blind position for users sitting next to windows we decided to delay the development of the light controller until we understand the impact of blind positions on inside illuminance better. Unfortunately up to now we did not have the time continue this approach, hopefully future works will benefit from our thoughts.

7.2 Blind controller

Glare avoidance from direct light is one of the first patterns coming to mind when tuning blind position. By glare we refer to stray light which worsens rather than improves vision. When the sun is travelling along a path close to the horizon, which is typical for sunny winter days, sun rays fall deep into the room glaring users sitting next to the window. In such cases the user usually puts the blind in a position so that the glare zone comes at most as close to his working place as a certain limit which will be called *shadow-line* throughout this discussion. If the glare zone comes closer the user is annoyed and sets the blind so that the sun light falls behind this shadow-line and will not cross it during the near future, e. g. the next hour. Summarizing the assumptions: a user will interact with the *shadow* on their desk, and the height of the blind is just the instrument to do so. There is a user which is the most annoyed and presumably it's the one closest to the window. Hence the distance from the window forms the natural priority queue of user wishes.

But glare avoidance is not the only pattern of interaction, because there is also user interaction when the sun is on the other side of the building and no glare can take place. From biological vision (Kiper[21]) it is known that eyes adapt to a wide range of illuminance whereas they have low tolerance for contrast. Hence when working on a computer screen with a constant background luminance, it is very fatiguing if the surround is too bright. Additionally as in glare avoidance which deals only with the direct part of illuminance, the same *glare* effect could happen solely due to the diffuse component of illuminance. For example, on heavily overcast days glare can occur while looking into the cloudy sky. But no direct illuminance is present because the clouds spread the light in all directions, producing only diffuse illuminance. In both cases it's assumed that users will lower the blind to bring illuminance in their surround into an acceptable range.

Hence we assume two different behaviors, which leads to the problem of splitting the data among these patterns.

1. glare avoidance of direct light
2. luminance control

The user input consists of blind height and the values of the environment sensors at the time of interaction, see preprocessing section on page 29. We assume that the user input resulting from glare avoidance is consistent, given a particular set of conditions. On cloudy days no clear shadow is produced, so sunniness is a prerequisite for glare; see section 8 on page 29 for computation of these entities. We assume that the shadow-line is a single value, a constant not varying with daytime or irradiance, because it is determined by the position of the user, his desk and his computer. This should result in a single dense cluster in shadow-line/sunniness space. Hence it seems promising to cluster the user input into a glare avoidance cluster and then assume the remaining points to be involved in luminance control.

Obstacles dropping shadows on users desktops will have additional influence, because there is no need to use the blind as an additional shading device during these periods. Such obstacles can be split into outdoor, e. g. trees, other buildings, or inside obstacles such as furniture or the left and right termination of the window. Such obstacles could be detected by a comparison of outside and inside illuminance values.

These ideas are not new as Guillemin[1] also separates the problem of blind controlling into glare and illuminance control. The new idea is to compute the shadow-line explicitly instead of using elevation and sun azimuth. The latter methods uses raw elevation/azimuth as input and let the controller do the mapping onto shadow-line. But this function is not invertible because the shadow-line maps onto different combinations of elevation and azimuth, hence more points are needed for learning.

On the other hand he used a setup of illuminance sensors spread throughout the room to directly compute a notion of *contrast*. Hence luminance control was performed by keeping inside contrast within a limiting contrast range. Because we do not have such a setup available (nor does an average normal automated building) this is not possible, hence a regression approach of blind height versus environment values seems feasible for a first shot.

If the dimming factors, which stands for the impact of blind height onto inside illuminance, of each blind was known, the effective inside daylight illuminance could be estimated from blind position and outside illuminance. In the estimated inside illuminance the blind position and outside illuminance is already extracted, hence no more need of regression to correlate the blind height to outside illuminance. Instead a simple *satisficing* range (see page 21) with an upper and lower natural illuminance threshold could be used to describe the users preferences. While this might work for people in the rear side of a room it will not work for people sitting next to the window because they are more sensitive to outside illuminance. The indirection via the dimming factor will make it likely to be too inaccurate to be useful for them. Nonetheless not everybody has a window seat, and because fitting blind height versus environment conditions can be computationally very expensive a satisficing range solution becomes interesting.

7.2.1 Glare avoidance/Shadowline clustering

As outlined above we expect to find a dense cluster in shadowline/sunniness space, because we assume that users puts a blind such that the shadow it produces falls onto similar positions and this presumably on sunny days because on cloudy days no direct illuminance is present. See sections 8.1 8.3 for sunniness and shadow computation. Natural clusters are found by the method of hierarchical clustering[22]. As a very dense cluster is searched, the inconsistency threshold is set to a very low value which cuts the dendrogram[9] into lots of little clusters, in the worst case consisting of individual points. Among the created clusters the largest cluster is chosen to be the result of glare avoidance. No bias is applied to prefer only clusters on sunny days. The shadow line is computed by taking the mean of all those points in the glare avoidance cluster.

7.2.2 Luminance control

For luminance control all user interactions assumed in glare avoidance are removed. Usually an equal number of interactions is left. Their representation in

[9]See figure 40 on page 67 for illustration

in sunniness/shadow-line space was either such that i) they were not part of the dense cluster or ii)have a negative shadow-line because the sun was not on the side of the building at the time of user interaction. From these user inputs a new training set is formed with irradiance and sunniness as input and the blind height as the target dimension.

Regression is performed by Gaussian processes, see appendix A for more details about the method. The maximization of the Gaussian processes was done by conjugate gradient method, which might be stuck in local minima. So ten runs are performed and the hyperparameters with the highest likelihood given the data are chosen. All computations are performed on normalized data with unit variance.

7.2.3 Evaluation

The controller was improved during several iterations where it was exclusively tested and evaluated by ourselves and comrades in the same office. After it reached sufficient stability it was also deployed to regular office users, which were as familiar with the subject as they were concerned about the blinds position.

In the meanwhile the rest of the buildings G floor was controlled by a competing fuzzy controller[5]. This controller was deployed on a server, which will be called *deployment* server from now on. It was logging the building-bus and pc-bus traffic during the period from the March 4, 2004 to until July 11, 2004.

These collected user inputs were used to build and verify the user interaction model. This data will be used to demonstrate the fitness of the model. The likelyhood of luminance reduction will be shown graphically be contour plots. Performance of different regression models are compared based on their mean-square-error which is computed between the measured height and the evaluated height at the same input conditions (irradiance/sunniness).

For glare avoidance a dense cluster is predicted. Because there is no built-in bias that glare avoidance has to occur on sunny weather, the sunniness coordinate of the cluster center, is an additional indication whether our assumptions are correct. To make this explicit: the method is not restricted to find a glare avoidance cluster only during sunny weather. It could happen that glare avoidance cluster is found on overcast weather were no direct illuminantion is present, hence no clear shadow is produced. The method does not prevent this. Hence, if glare avoidance clusters are in general found on sunny days, it is an additional indication that our assumptions were correct.

7.2.4 Experiment

Unfortunately up to now only very few user interactions are available, which means that almost any model will fit these few points. Partly this is an inherent problem in controlling building devices, because average users are very tolerant to settings and usually only interact in case of extreme discomfort. So in an uncontrolled room there is typically less than one interaction per day, user and device. Hence a more proactive way in gathering user input was chosen.

Users are asked if they agree that a new controlling agent can be applied to their room. First users are sensitized to reliable presence detection by deploying a *blind measurement* agent that is only active when the cluster is unoccupied (all users absent). It was explained to the users that this agent is necessary to compute the impact of the blind position on inside illuminance and that it is only active while their cluster is unoccupied. It was made clear that anything happening while users are present is an error and and should be reported.

The main purpose of the measurement agent is to measure the correlation between inside illuminance, blind position and outside irradiance. The measurement is done by lowering all blinds except one, the *probe* blind, which was set to a position out of ten possible positions. The position of the probe blind changed every 12 minutes until it visited all ten positions. Once a cycle finished a new blind was chosen. This way the impact of blind position on inside illuminance, called *dimming factor* can be estimated.

The by-product of the agent is that the blind is left in a random position when users come back. Hopefully intolerable to them, so they have to interact with the blind and set it to their currently preferred position producing a new data point. Due to the fact that the agent does something useful that can be explained quickly, user rejection was minimized.

This setup was running for at least two days to make users familiar with the new situation and emphasize that there is a clear distinction between the present and absent case. After this period the comfort controllers were deployed trained by the inputs gathered during the previous days. Users immediately perceived the difference because the blind moved when they entered an unoccupied room.

To prevent that users feel at the mercy of the system, the controller contains a locking mechanism that disables it on every user input. Disabling means that the controller is not allowed to override a position a user has set from his pc-bus client. The lock is personalized and as soon as the user is absent it is removed. In addition during the experiment the lock got also removed automatically after $2\ \frac{1}{2}$ hours. It's assumed that after this period the environment has changed considerably, e. g. the sun left plane of facade. Hence the blind position chosen last is no longer up-to-date and probably needs an update. For the users this means that after $2\ \frac{1}{2}$ hours since their last interaction the controller computes a new blind position and possibly acts on it. It was verified that users that prefer a constant blind height independant of outside conditions are not annoyed by the adaptivity of the controller. This is ensured because the regression of luminance reduction has a low prediction error in case of constant blind height, see results on page 77. So the luminance reduction pattern puts an upper constraint on the blind position, because the conflict algorithm choses the lowest estimated position of all users and all patterns.

Care has to be taken that the overriding mechanism of the controller is not too annoying, otherwise it feels educational to users and they might soon stop interacting when they feel it has no sense. So there is a trade-off in collecting more data and being annoying to users. Our worst case was that user had to interact at most four times during an 8-10hours working day plus the times when the measurement agent was setting the blind to an unpleasant position and the comfort controller couldn't predict the users preferred position. Hence we assume that users did not feel annoyed by the possibility that positions they set could be overridden from time to time.

The performance of the controller can be measured by the number of interactions. If the cumulative number of interactions stabilizes at a given value, the model seems to converge and the users to be mostly satisfied. Otherwise if the number of interactions keeps increasing linearly the model was unable to predict the users preferences when sensing them present. See results in section

15.3 on page 84.

8 Preprocessing

To be able to describe the controller operations on abstractions such as sunniness and sun-position, these values have to be computed from existing sensor inputs. In summary the reasons for preprocessing inputs are:

Interpolation The sensor data is not synchronized with user input, the existing values have to be interpolated at intermediate timestamps.

Virtual data such as sun position have, aside a clock, no physical sensor.

Abstractions such as sunniness that describe the overcast condition of the atmosphere. Because no sensor is available that can see and count clouds this value is estimated from simpler sensors.

8.1 Shadow-line computation

As outlined in section 7.2 on page 24 we want to directly compute the border between glare and shadow zone called *shadow-line*. The indirection using elevation and azimuth has the disadvantage that several sun elevations and azimuths map onto the same shadow-line. Because we assume that a user's preferred shadow-line is constant and does not vary with daytime or season, we can reduce the number of inputs necessary for training. This abstraction is crucial in a system of sparse user interactions.

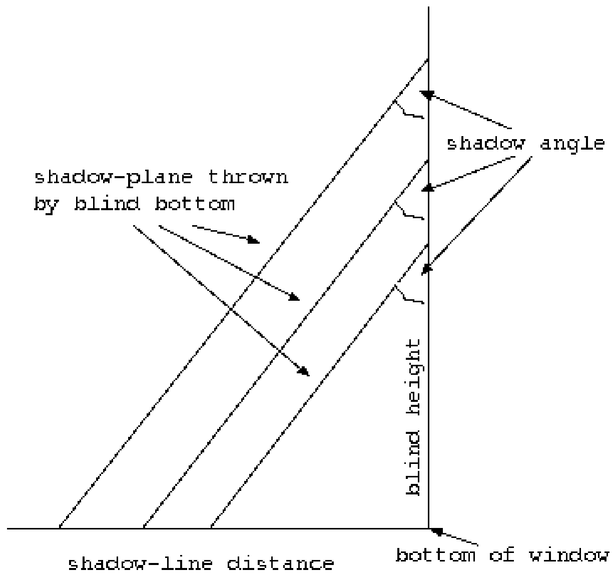


Figure 11: Shadow line: Distance of the shadow border to the window bottom, measured in blind ticks.

We define the shadow-line as the distance between the window bottom and the intersection of shadow plane and a horizontal plane at bottom window height. See also figure 11 for illustration. Because not all windows have the same height (see figure 44 on page 72) we do not measure the shadow-line in centimeters but express it as a fraction of the window size. This is sufficient for our purpose because it relates linearly to any glare-zone border at any height the user might care about.

Calculation The computation becomes trivial once the blind height and the angle between shadow and window plane is known, see below:

$$dist = tan(\angle_{ceiling}) \times blind_{height}$$

Because the blind height can be retrieved from the low-level blind-motor controller the main problem is to compute the smallest angle between shadow and window plane. The calculation, which requires some vector geometry, can be divided into three steps:

1. Calculating the sun position
2. Translation into facade coordinates
3. Computing the shadow angle

1. Sun Position Publically available libraries provide calculation of sun position based on latitude and longitude of the observer and time. See Geotools[11] for a Java or Roy[13] for a MATLAB implementation. For experimenting there is a website Cornwall et al.[12] which is fun to play with. A description of the calculation itself can be found in Meeus[14] or Guillemin[1].

Figure 12 shows the sun position in cartesian and polar coordinate space. Table 1 lists the definitions of the polar coordinates elevation and azimuth.

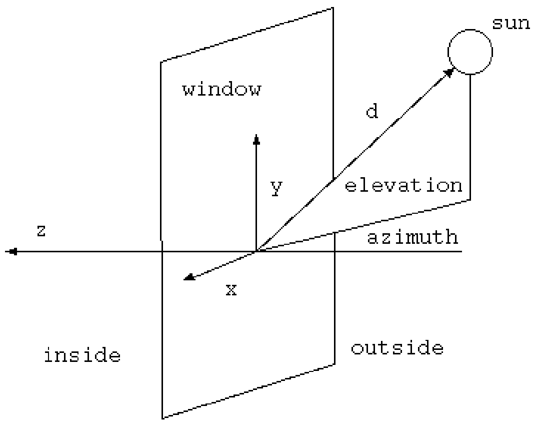


Figure 12: Cartesian room coordinate system

h:	$-\frac{\pi}{2} \leq h \leq \frac{\pi}{2}$	Solar elevation, the angle between the sun direction and its projection onto the horizontal plane. Positive when the sun is above the horizon.
a:	$-\pi \leq a_r \leq \pi$	Azimuth, angle between the direction of the sun projected on the horizontal plane and true north. Clockwise positive towards east.

Table 1: Sun position in the polar coordinate system of the observer

2. Translation into facade coordinates The direction of the sun is rotated into the orientation of the facade by the following formula. See table 2 for a declaration of the variables.

$$a_r = a - a_0$$

To apply common vector geometry the sun position is translated from polar to cartesian room coordinates. Care must be taken to make the room coordinate

a_0	The angle between facade normal and true north, positive values mean that the facade heads eastward.
a_r	The azimuth relative to the facade normal, clockwise positive. An angle of zero means that the sun is perpendicular to the window. A positive value means the sun is to the right of the observer inside the room.

Table 2: Rotation into facade local polar-coordinate system

y-coordinate	is bijective with elevation as the sine is unique in the range $-\frac{\pi}{2} \leq h \leq \frac{\pi}{2}$
x-,z-coordinate	the combination of sine and cosine makes sure that the azimuth gets mapped on the correct quadrant.

Table 3: Unique translation from polar to cartesian coordinates

system a right-handed-system (see figure 12). The three defining unit vectors of the room coordinate system are $\vec{x}$ along the window bottom pointing to the right as seen from inside the room, $\vec{y}$ along the window height and $\vec{z}$ perpendicular to the others pointing into the room.

The translation of the sun position from polar → cartesian is done by the following equations. The translation is uniquely invertible as shown in table 3.

$$\begin{aligned} x &= cos(h) \times sin(a_r) \\ y &= sin(h) \\ z &= -cos(h) \times cos(a_r) \end{aligned}$$

3. Computing the shadow angle This is the smallest angle between the shadow plane thrown from the blind bottom and the window plane; see figure 13 for illustration. The normal onto the shadow plane is computed by a vector cross product.

$$\vec{n}_{shadow-plane} = \vec{d}_{sun} \times \vec{x}_{unit}$$

Where $\vec{d}$ points to the sun, $\vec{x}$ is the unit vector and $\vec{n}$ is the shadow-plane normal. Whether the sun is in the plane of facade can be seen by the z-coordinate of the sun position vector $\vec{d}$. If $z < 0$ the sun is in the plane of the facade in which case the multiplication order of the cross-product ensures that the normal always points down to the floor [10].

In figure 13 there are two 90°angles of importance; one between shadow plane and its normal and the second between the y- and z-coordinate. As β and α sum up to 90°and α and γ as well, β must be equal to γ. So the ceiling shadow β can be found by the following formula.

$$\beta = \gamma = arctan(\frac{n_y}{n_z})$$

[10]The direction of the resulting vector of a vector cross-product can be found by the *right-hand-rule*. Spread thumb, pointer and middle finger so the are perpendicular to each other. Then the thumb points along the first vector, the pointer along second and middle finger points into the direction of the product vector.

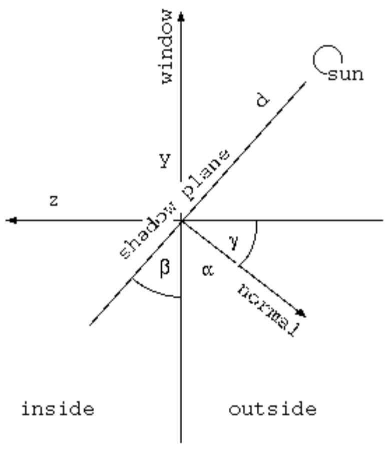


Figure 13: Blind bottom shadow: β is the angle of interest. In the center of the coordinates is the bottom of the blind. The x-axis is not drawn as it is perpendicular to the paper, pointing towards the reader.

Where n_y and n_z are the coordinates of the shadow normal. The n_x coordinate is zero as the shadow normal is perpendicular to the x-axis.

8.2 Irradiance/Illuminance

Most of the user interactions with lights or blinds are directly or indirectly related to illuminance and irradiance either as glare, diffuse outside illuminance, inside illuminance or heat due to irradiance. Hence knowing illuminance/irradiance at times of user interactions is crucial for the analysis. While going from analysis to prediction, it might be useful to foresee the trend of illuminance/irradiance. If these entities keep increasing the blind could be set so that illuminance will be tolerable until the next blind update is scheduled. Unfortunately these quantities can exhibit considerable noise and not as many sensors are available as we probably wish to have. Summarizing, this section deals with

- Interpolating measured values at times of user interaction.
- Predicting values in the near future, in time scales of minutes or hours

8.2.1 Which sensors to use?

There are irradiance sensors on the roof for east, south and west sides of the building[11]. These sensors are mounted vertically parallel to the corresponding facade of the building. Additionally there is one illuminance sensor which is horizontal. Horizontal means that its sensitive area is parallel to the ground and therefore measures the illuminance falling onto horizontal planes.
For our purpose the vertical sensors are more interesting as shown in [17]. This paper says that there is a strong correlation between perceived inside illuminance and vertical diffuse outside illuminance. The problem we have is that there are vertical mounted sensors but they measure irradiance, not illuminance. So we have to convert between these two entities.

8.2.2 Pseudo-linearity of irradiance and illuminance

The relation between irradiance and illuminance is not linear. The difference between irradiance and illuminance is related to the difference between *radiometry* and *photometry.* Radiometry includes the entire optical radiance spectrum 3×10^{11} - 3×10^{16} *Hz*, while photometry is limited to the visible spectrum as defined by the response of the eye *360 to 830 nanometers*, see figure 14. The following definitions are taken from Palmer[18].

> **Irradiance**(a.k.a. flux density) ... SI derived unit and is measured in $\frac{W}{m^2}$. Irradiance is power per unit area incident from all directions in a hemisphere onto a surface that coincides with the base of that hemisphere. ... The symbol for irradiance is **E**. Irradiance is the derivative of power with respect to area, $\frac{d\Phi}{dA}$. The integral of irradiance over area is power.
>
> **Lux** (unit of luminous flux density, or illuminance). **Illuminance** is a SI derived unit which denotes luminous flux density. The unit has a special name, the *lux*, ... $\frac{lm}{m^2}$. the symbol is E_v. Most light meters

[11]North has no sensor and its irradiance has to be estimated by the side where the sun is currently absent. E. g. either east or west.

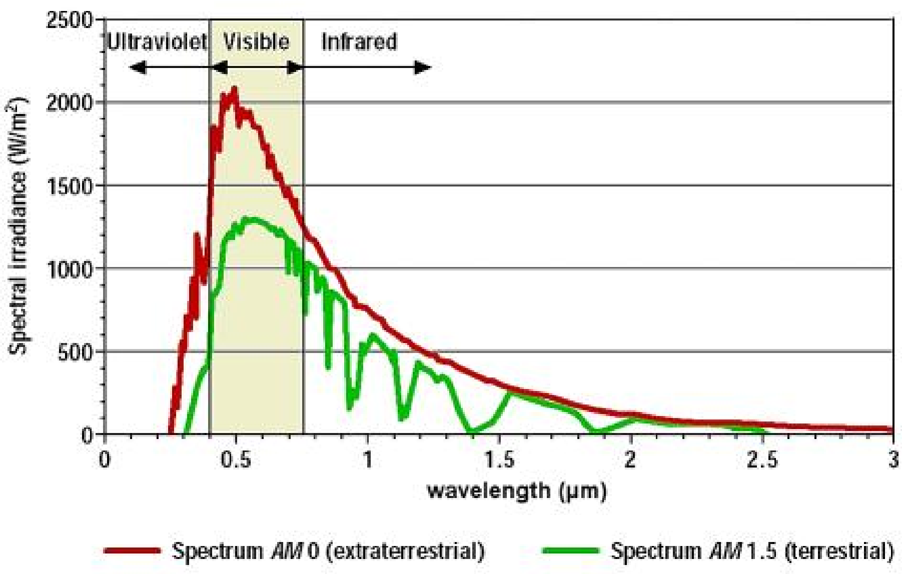


Figure 14: The upper curve shows spectral irradiance in space. The lower curve illustrates spectral irradiance on earth filtered by the atmosphere. Print permitted by Volker Quaschning[19]

> measure this quantity, as it is of great importance in illumination engineering. The IESNA lighting Handbook has some sixteen pages of recommended illuminances for various activities and locales, ranging from morgues to museums. Typical values range from 100 000 lx for direct sunlight to 20-50 lx for hospital corridors at night.

We assume that the shape of the spectral distribution does not vary based on time, season, or cloudiness. Further we assume that an overcast sky or the length of the path the light passes through the atmosphere filters the irradiance uniformly but does not change its shape considerably. See figures 14 15 for illustration. So we assume linear relation between irradiance and illuminance.

$$illuminance = irradiance \times const$$

Unfortunately this assumption is incorrect, because depending on daytime the shape of the spectrum is changing. For example if irradiance is equal at midday and at sunset the illuminance at midday is higher because at sunset more irradiance in the visible range is filtered out by the atmosphere. We were also made aware of that the irradiance sensors do not compute the integral $\int_0^\infty E_\lambda d\lambda$ but more likely are silicon detectors with an overlying filter that flattens the response in the silicon range of sensitivity. Hence they are not sensitive over the entire irradiance range but apply an estimation of irradiance based on a subrange. This knowledge might be used to build a better estimation of illuminance out of the irradiance sensors by restoring the underlying response of the silicon detectors and then use a different filter to estimate illuminance.

Unfortunately we did not take profit from this and used the simple pseudo-linear behavior outlined above during our analysis.

8.2.3 Accessing sensor data

The irradiance sensors are not directly attached to the *LON* building automation network that we have direct access to. Their values are routed through a gateway that copies value updates from the neighboring network. This mechanism is likely repeated by an upper level campus-wide network. Relevant to us is that we cannot request sensor data but have to wait until new values drip in. Usually this results in about 80 values per sensor between 4:00 AM to 8:00 PM, in an average interval of 12 minutes.

This might seem an unnecessary detailed discussion but once the user takes an action we would like to know the irradiance at that time accurately, because we assume it is the irradiance that leads the user to an interaction. By taking a possibly outdated measurement that even might be an outlier will not ease the analysis.

In case we could address the sensor directly we would probably poll it a medium interval like every other minute and then average over the 5 minutes before and after user input. This could be implemented by using an alternative connection like a TCP/IP connection to a computer that has direct access to the sensor in the weather station. But currently we have to live with the data provided, hence we have to interpolate the sensor values at times of user input.

8.2.4 Astronomical prediction

In a first shot we felt tempted to compute the irradiance out of the sun position because the sun is the underlying reason for the irradiance. The intention was to derive a function that encodes times of significant changes in daily irradiance such as the time of the peak value, sunrise/sunset or entering/departure of the sun in the face of the sensor. Prediction is done by assuming equally cloudy sky for a short interval, and to fit the function to the sensor values obtained during that interval. This fitted curve could then be used to predict the irradiance in the near future. Whole day predictions are not feasible because the state of the atmosphere might change depending on clouds appearing or disappearing.

In short this is the idea of using a prior that encodes our general knowledge about how we think irradiance changing during the day. The problem is that the sun position is not the only factor influencing the measured irradiance but that the distance the light travels through the atmosphere plays an equally important role see figure 15 for illustration.

In space, the irradiance is maximal when the sun is perpendicular to the sensor plane. More generally the irradiance is maximal if the angle between perpendicular and sun position is smallest[16]. Applied to our case where we have east, south and west sensors, this would imply that the sun has to travel to an azimuth angle of approximately 90°[12]. Given that the sun needs 24h to travel 360°the prediction would say that it takes about 6 hours between the maximal irradiance of east and south[13]. Taking a look at figure 16 one can see

[12]This is a simplified as the sun travels along a path that is parallel to the equator and not to the horizon. So azimuth and sun *declination* map one-to-one only on the earth poles.

[13]The same simplification as above, but the dimensions are okay

that peaks are much closer together than six hours.

Hence we conclude that we need to include also the distance the light travels through the atmosphere to make this approach usable. Unfortunately we didn't have time for an additional develop and test iteration, so prediction is currently not very reliable. As an alternative to building a physical prior we could do the same by a machine learning technique such as principal component analysis(PCA). If take the last e. g. twenty days and train the model with it, we expect the component with the biggest eigenvalue to have a similar shape to what we our handrafted function would look like.

Interpolation with Gaussian processes Initially we tried to interpolate with simple polynomials, but this lead to some problems. Assume we want to interpolate an intermediate value. Global fits are unthinkable unless we do not know the underlying curve very well or use polynomials with very high degrees and introduce the problem of overfitting the data. Local fitting in turn has the problem that on unstable days the sensor values are quite noisy as they vary between the extremes cloudy/sunny. To average and subtract the noise we need to take enough data. But given the rate of sensory inputs[14], taking more data points, quickly extends the time basis of the interpolation into hours which introduces the problems of estimating the times of irradiance peak or the sun leaving the plane of the sensor.

Gaussian processes brought us relief from choosing the polynomial with the right degree. They are non-parametric smoothing devices (see appendix A) yet simple in their use as they only involve matrix manipulations. They can be interpreted as a neural-network with infinitely many hidden units and therefore having infinite VC-dimension, while having a low tendency to overfit the data.

We use them to interpolate the irradiance at times of user interaction to train the users preference model. This interpolation is done at the end of the day when all sensor values are known. So this is strict interpolation and not prediction because we know the sensor values before and after the time of interest. See section 12 on page 63 for a selection of fitted curves.

[14]About 80 points between 4:00 to 20:00

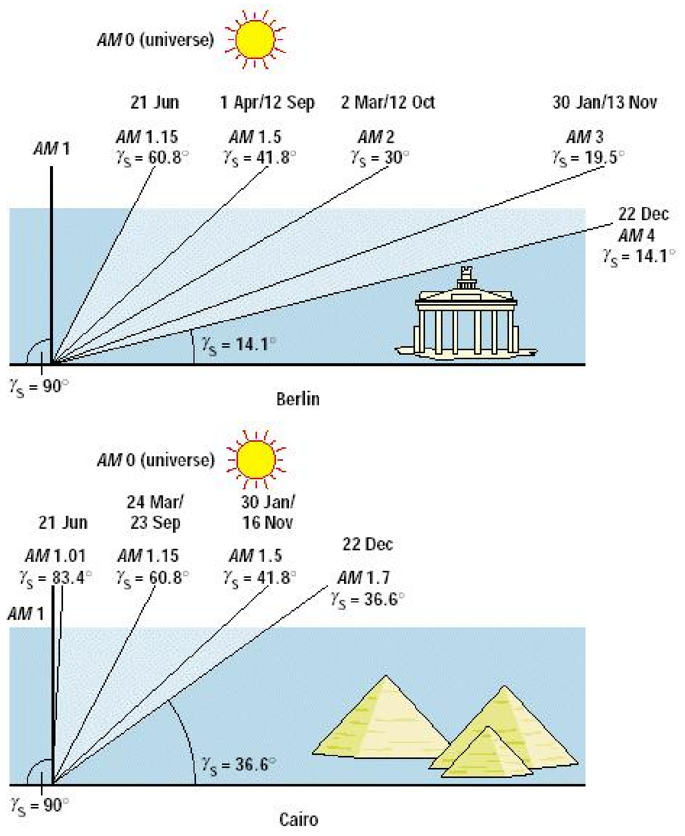


Figure 15: If the sun is perpendicular to the earth's surface, sunlight only has to pass through the air mass (AM) of the atmosphere once. Therefore, this state is called AM 1. In all other cases, the route of the solar radiation through the atmosphere is longer. This way depends on the sun's height. AM 2 indicates that the path of the sunlight through the atmosphere is twice AM 1. This is the case if the sun is 30 above the horizon (S = 30). In general, the definition of the air mass is AM = 1/sin(S). The figure shows the variation of the air mass during the year for Berlin and Cairo at solar noon - that is, the time during a day with the highest sun elevation, which depends on longitude, latitude and date. It is obvious that in Cairo the air mass is always lower than in Berlin. Printed by the permission of Volker Quaschning[19]

Figure 16: Plots of east, west and south irradiance values during a sunny day. The y-axis represents watts but there are some problems with the scaling of the sensors.

8.3 Sunniness

We define sunniness as a value between 0 and 1 expressing whether the sky is currently overcast or open, whereas 0 means a completely overcast sky and 1 denotes an open sky. We will now describe two methods for estimating the sunniness.

8.3.1 Heuristic

We do not have access to sensors that natively reports the current sky condition. So we have to estimate this value from other sensors. The underlying idea is illustrated in figure 17 Its essence is to compare the direct component of irradiance to the global irradiance[15] because in space all irradiance is virtually parallel, hence the diffuse component is created by the clouds and particles in the atmosphere. If the direct component of irradiance is by a magnitude larger than the diffuse component it's likely to be sunny.

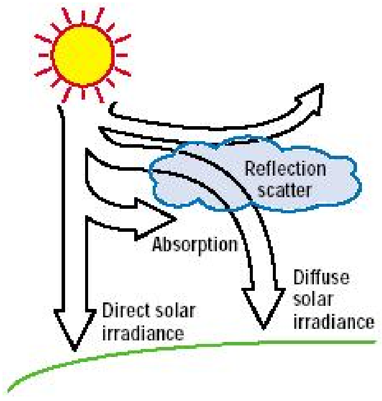


Figure 17: Outside the atmosphere, an therefore not subject to its influence, solar irradiance has only a direct component - all solar irradiance is virtually parallel. Atmospheric particles reflect or scatter sunlight. Only part of the extraterrestrial beam reaches the earth's surface directly. The scattered part that reach the earth together with ground reflectance form the diffuse irradiance. Print permitted by Volker Quaschning[19]

By comparing figures 18 and 19 the idea is that comparing sensor values from east and west sensor might tell us whether it is currently cloudy or not, see figure 20 for illustration. Because the sun can either be in the west or east but not both at the same time, there is always one sensor that measures only the diffuse component of irradiance. The estimation of the direct component is simply taken to be the difference between the east and west sensor or in general the difference between the highest and lowest sensor value as shown in the formula below.

$$sunniness = \frac{abs(max(east, south, west) - min(east, west))}{max(east, south, west)}$$

[15]global irradiance = direct + diffuse irradiance

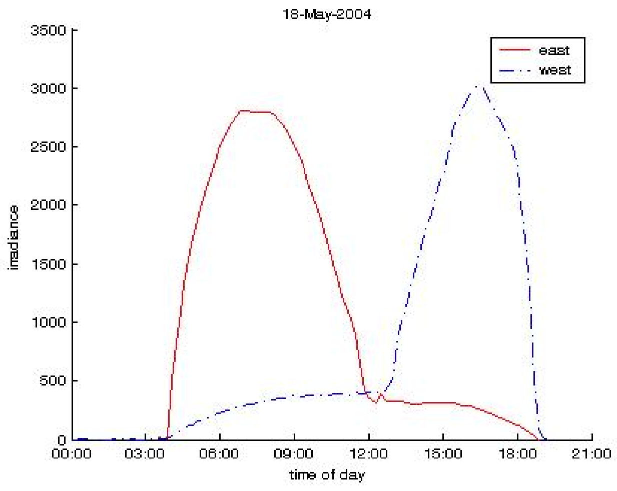


Figure 18: Data from vertical irradiance sensors, presumably from a sunny day.

There are several problems with this approach. First the direct component should be estimated more carefully but aside that there are technical problems as well: The irradiance sensors are not synchronized, so we have to interpolate for a common timestamp, hence we operate on interpolated data, which introduces another level of indirection. Another problem is that we assume both sides of the building to be identical with respect to reflections, e. g. a builidng with a lake on one side and a parking lot on the other, will have different diffuse components on each face of the building. The diffuse components on the lake-side will be larger as it sums up all reflections by the water. Also in case of some scenarios our sunniness fraction might be incorrect such as if a cloud covers only the hemisphere of the direct irradiance measuring sensor. In the worst case our sunniness fraction will indicate cloudy weather whereas the sky is perfectly blue on the other side of the building.

8.3.2 Model

The above method is just a heuristic and may not reliably report sunniness. Mainly it is due to a problem with the current sensors i) they report the global irradiance and not just its diffuse or direct component ii) they are fixed and do not track the sun. Because the diffuse irradiance is by definition not dependant on sensor orientation, we assume that estimating it from the sensor where the sun is absent is sufficiently reliable for our purpose. More problematic is that the heuristic method does not account that the direct component is dependant on sensor orientation; see figure 21. Following equation shows how to compute the direct irradiance from east south or west sensors.

$$irradiance_{direct} = \frac{irradiance_{east|south|west} - irradiance_{diffuse}}{cos(\Delta_{east|south|west})}$$

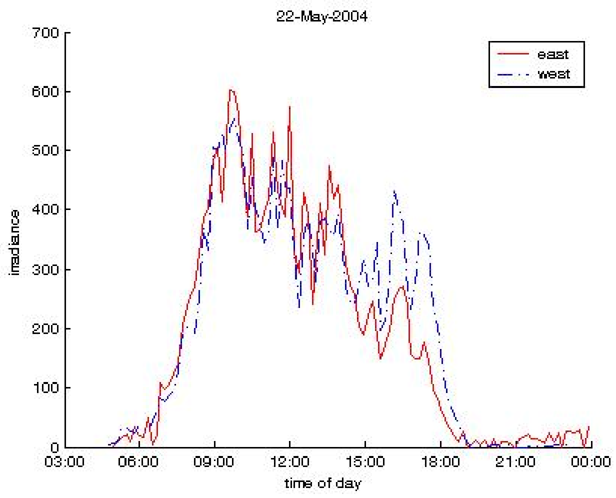


Figure 19: Data from vertical irradiance sensors, presumably from a cloudy day.

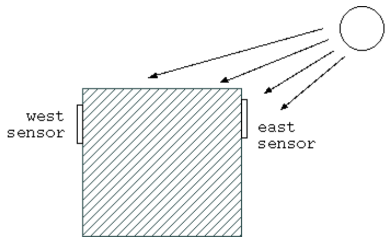


Figure 20: Illustration of vertical sensors attached to the east and west side of the building. The direct components of the sunlight cannot reach both sensors at the same time. One sensor measures only the diffuse part of global irradiance.

Whereas Δ is the angle between sun direction and the normal on the sensor plane and can be computed by the spherical cosine theorem, see Walser[15]. Remember that calculating the direct component only make sense if the sun is in the face of the building which means Δ to be smaller than 90°.

$$cos\Delta = cos(azimuth_{facade} - azimuth_{sun}) \times cos(elevation_{sun})$$

In analogy to the heuristic method, the sunniness becomes

$$sunniness(t) = \frac{irradiance_{direct}(t)}{irradiance_{diffuse}(t) + irradiance_{direct}(t)}$$

Because the measured irradiance is affected by the path the light travels through the atmosphere (see figure 15 on page 38) the sunniness fraction can still vary even on completely stable days. So again as in the case of illuminance prediction (see page 36) atmospheric effects should be removed. But even when ignoring these atmospheric effects the model is not stable enough to be useful.

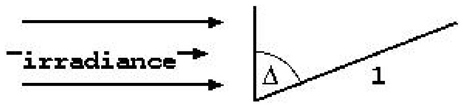


Figure 21: If the sensor is not perpendicular to the net direction of irradiance, it measures only a fraction of the irradiance per square-meter. Δ is the angle between sun direction and the normal on the sensor plane.

While it improves the situation for the dips between east and south, and south and west, the division by $cos(\Delta)$ increases the irradiance by an order of magnitude when Δ gets close to 90°. Although the heuristic has some problems it is also simpler to compute and more stable, hence it was used as input to the blind controller. See section 13 on page 65 for results.

8.3.3 Stability of weather

It might be interesting to know if the day is stable or not. In case of a stable day blinds and lights can be operated more closely along the predicted values. For unstable days settings have to be done more conservatively. E. g. if the sun is hidden by a cloud it is more favorable to turn on the light, and use more energy than to raise the blinds and risk annoyed users when the cloud passes and the sun rays fall back on the users desk.

Part III
Software Architecture

This part gives an overview of the underlying ABI framework, the design of our controllers and a description of MATLAB functions used for training and analysis.

9 Framework

This section describes the building access framework and other preliminary work.

9.1 ABI/ABLE framework

Agent Building and Learning Environment(ABLE) ABLE is a multi-agent-framework implementing the FIPA(Foundation for Intelligent Physical Agents) and JAS(Java Agent Services) standard. It provides agent lifecycle and directory service among other services, but these are the most relevant for our purpose.

Adaptive Building Intelligence(ABI) The ABI framework is built on top of ABLE and provides the following enhancements[16]

- Abstraction of building bus access.
- Structure management service
- Message distribution service

Building bus access Please refer to Trindler and Zwiker[5] for complete documentation.

Structure management service This service publishes available building devices in a hierarchical way. It can be thought of as a white-list directory advertising different mailing-lists. All agents that deal with building devices have to contact this service to lookup which devices are available.

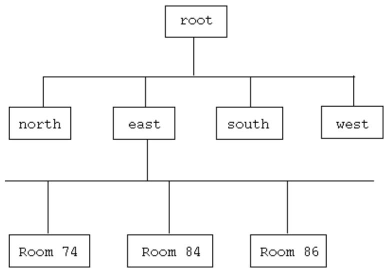


Figure 22: Devices are attributed to the cluster reflecting their spatial relevance. A cluster in turn contains all devices directly attributed or contained within a subcluster, so the root cluster contains all devices, the west cluster only those relevant for the west facade. Devices having a very limited range are attributed to a room clusters or to a leaf clusters(not shown) below the room clusters level.

The structure is strictly hierarchical, see figure 22. Each building is broken into different facade orientations and on each facade there are different rooms.

[16]Additionally it provides a genetic fuzzy controller, which is not part of the core framework, we are focusing here

Throughout this section the word cluster will be used also for room and facade. Usually a room cluster contains additional subclusters that can be seen as microcosmos, largely independent from each other. Although you will encounter the word *dynamic structure change* throughout the discussion of the architecture, the hierarchy of clusters is stable, unlikely to change at system runtime.

The structure is configured by the administrator such that effectors, driven by controllers, are inside a single cluster. Typically on each leaf cluster a controller will be deployed. This forms a way of configuring the system, by prearranging sets of devices, making extraction at runtime easier. Devices required by multiple clusters such as presence detectors, are passed up until the level where they are available to all subclusters. Presence detectors for example are shared among different control clusters. Vertical irradiance sensors are shared by all control clusters on the facade of the building. The horizontal illuminance sensor is added to the root cluster as it cannot be attributed to a single facade.

Message distribution service Figuratively this is the mailling-list server. The topics are the devices advertised by the structure service. Other topics are structure changes of clusters and purely abstract topics such as topic create/destroy. Once a new value has to be distributed it is sent to the message distribution agent which sends it asynchrously to all registered agents. Please see Trindler & Zwiker[5] for a more complete description.

9.2 PC-Bus

The pc-bus mimics the behavior of the building bus but uses the ethernet network for communication. Currently it is used to implement a software client for presence detection and enabling users to set building devices. But it could also be used to add prototype sensors to the ABI framework. The API comes with extensive package descriptions, see Delbruck & Fenkart[9].

9.3 Virtual Bus

The virtual bus is only used to encapsulate computation of variables needed by multiple agents. The idea is to advertise the computed variables in the form of a virtual device by the ABI default publish/subscribe mechanism. To give a concrete example consider sun position or the presence signal at the room level. Both variables are not directly available through sensors but they are required by multiple agents, the former at facade level, the latter at the cluster level.
Alternative solutions would be

1. Building a parallel structure service used only for abstract values
2. Every agent computes the values independently.
3. Integrate abstract values into publish/subscribe service.

The first solution increases code complexity as such a service has to be implemented from scratch and likely will lead to an additional abstraction layer on top of the ABI framework. The second is inefficient in terms of performance, because an entity such as shadow angle is needed by all blind controllers running on a facade. The last solution provides transparent integration into existing

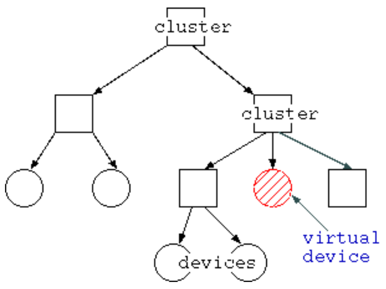


Figure 23: Like other devices a virtual device is attributed to a cluster so device lookup is completely transparent for agents subscribing for updates.

services, the drawback is that it requires adaptation of the ABI structure agent service, which might lead to incompabilities with other software based on the framework.

The framework enforces that only the BusAgent can add devices to the structure tree. We chose to keep this behavior but extent it so that other agents can mimic the behavior of the BusAgent.

- The agent must be able to register devices in the structure agent.
- Other agents must be able to lookup the busagent by device type.

The first problems is technical because currently only agents of type *BusAgent* are allowed to add devices to the structure and requests from other agents are bounced. This problem was fixed by a marker interface[17]. The second problem arises at startup when the agents request current values of the variables. Currently the reponsible bus agent is looked up by device type and a request message is sent to it. The solution chosen is that each virtual bus agent has to define an ABLE property *ABIConstants.JAS_BUS_TYPE*. The responsible bus agent can then be found by the underlying ABLE directory support, by the following code:

```
AgentDescription query = directoryService.createAgentDescription();
query.set(ABIConstants.JAS_BUS_TYPE, type);
AgentDescription[] busAgents = directoryService.search(query);
```

Distribution of new value updates is done through the *MessageDistribution-Agent*. Currently for each virtual device type a virtual bus agent is running, actually even multiple agents for the same virtual device type. This is a lack of design and should be fixed in a future release. The result is that when looking up a bus for a device type multiple buses are returned and requests have to be sent to all of them while only the one owning the device is responding.

[17]An empty interface, providing no functionality

10 Multi agent system

The software architecture is subdivided according to the problems outlined in the method part(page 12):

1. Accumulation of physical sensor data
2. Robust presence detection
3. Controlling effectors

Each of these problems is handled by an individual agent and this section will give an overview of the design of these agents.

10.1 Presence

The presence problem is split into personal presence detection and occupancy/unoccupancy of a cluster as outlined in the method (see page 13). Each problems is dealt by an individual agent.

10.1.1 UserPresenceAgent

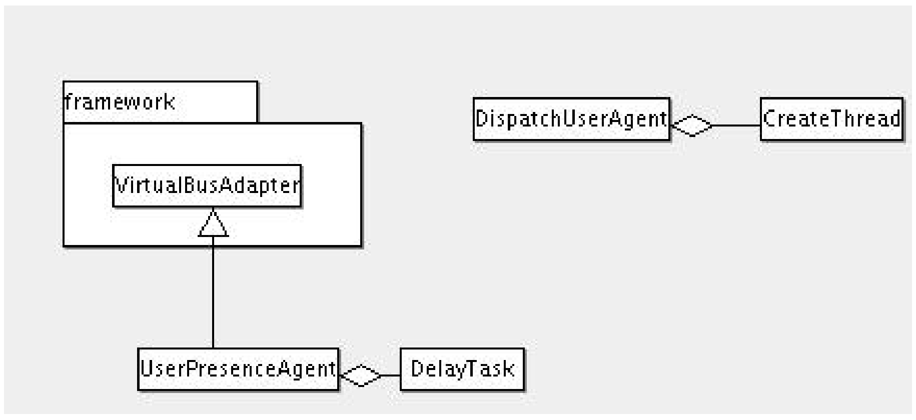


Figure 24: Class diagram of UserPresenceAgent

UserPresenceAgent is a virtual bus agent and adds a virtual device *feature.presence.user* to the cluster the user is controlling. For each user of the pc-bus client such an agent is started. The main purpose of the agent is to encapsulate the aggregation of the presence signal. Currently this agent is empty and it only forwards the values of the pc-bus detector, whereas the propagation of the absent signal is delayed by a fixed timeout. In the case of single-office it also listens to the PIR presence detector. See figure 24 for the agent class diagram.

No UserPresenceAgent is started at system startup. It is the task of the *DispatchUserAgent* to start necessary agents. On structure-change notifications it checks whether an added device is a pc-bus presence detector in which case is starts a new instance of the UserPresenceAgent.

10.1.2 ClusterPresenceAgent

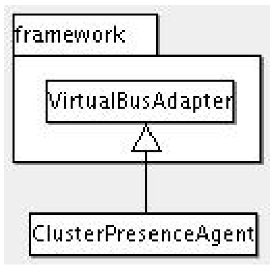


Figure 25: Class diagram of ClusterPresenceAgent

The ClusterPresenceAgent is a virtual bus agent too and adds a device *feature.presence.user* to the structure. The purpose of this agent is to *or*-combine the signals of the *feature.presence.user* signals and the PIR into a room occupancy signal. It also represents the presence device of the anonymous user. See figure 25 for a class diagram overview.

10.2 Preprocessing

Preprocessing takes care of the following computations.

- Online training of Gaussian Process fitting the irradiance values.
- Calculation of the sun position and shadow angles

10.2.1 IrradianceAgent

On each side of the building an irradiance agent is running. It collects the sensor values and trains the Gaussian processes for later evaluation at intermediate values, usually by the blind controllers. All sensor data and the most likely hyperparameters are then written into a file. These data are then used by the blind controller to build the training-set for learning the user parameters. See figure 26 for class diagram overview. The training of the Gaussian processes is implemented by MATLAB functions, which are accessed through the Java Native Interface, see appendix B for explanation.

10.2.2 ShadowAgent

entity	type
ceilingShadow	double
sidewallShadow	double
isSunOnCurrentSide	boolean
facadeAzimuth	double

Table 4: Value tuple of the shadow device

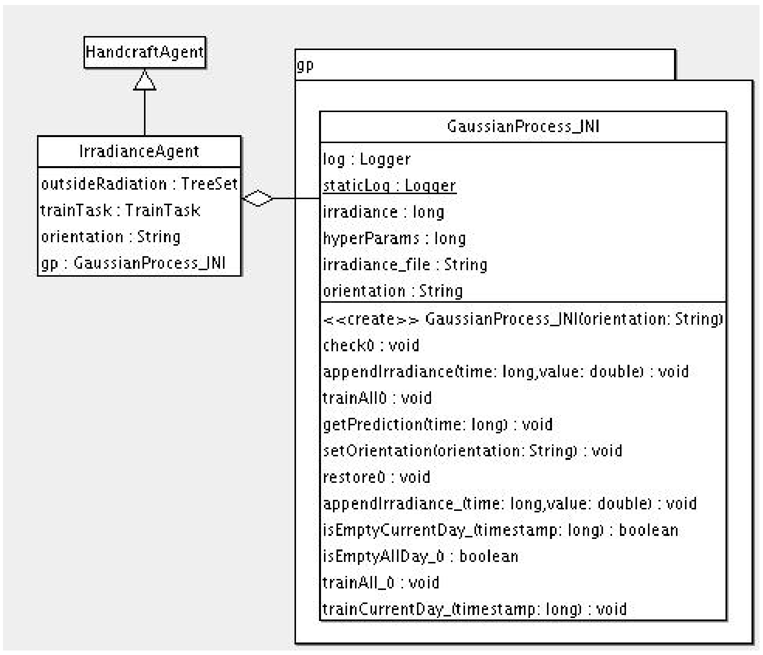

HandcraftAgent
gp
IrradianceAgent
outsideRadiation : TreeSet
trainTask : TrainTask
orientation : String
gp : GaussianProcess_JNI
GaussianProcess_JNI
log : Logger
staticLog : Logger
irradiance : long
hyperParams : long
irradiance_file : String
orientation : String
<<create>> GaussianProcess_JNI(orientation: String)
check() : void
appendIrradiance(time: long,value: double) : void
trainAll() : void
getPrediction(time: long) : void
setOrientation(orientation: String) : void
restore() : void
appendIrradiance_(time: long,value: double) : void
isEmptyCurrentDay_(timestamp: long) : boolean
isEmptyAllDay_() : boolean
trainAll_() : void
trainCurrentDay_(timestamp: long) : void

Figure 26: Class diagram of irradiance agent

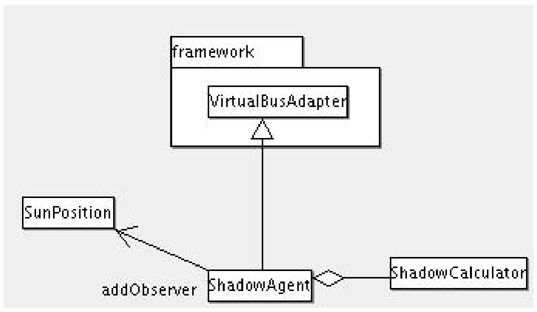


Figure 27: Class diagram of shadow agent

The ShadowAgent is another virtual bus and adds a virtual device *virtual.shadow* to the structure. On each facade of the building such an agent is running, because the shadow is identical for all devices on that side of the building. Usually blind controllers will subscribe for value updates of this device. See figure 27 for class diagram overview. The sun position is computed by an instance of *SunPosition*, which is a singleton instance, unique per Java Virtual Machine process. Every ShadowAgent subscribes itself as an observer of this instance. On *notifyObserver*-events the relevant entities are computed by the helper class *ShadowCalculator*. See table 4 for a list of distributed values.

10.3 Controllers

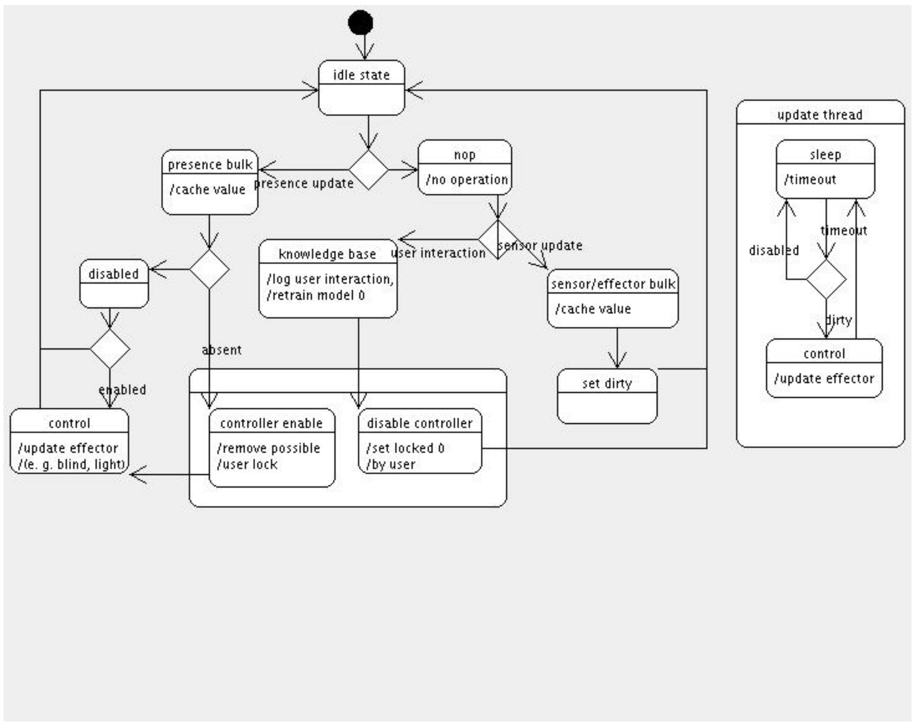


Figure 28: Control flow diagram

The controller is arranged around a central *ControlManager*. It keeps track of the current state as described by the flow diagram in figure 28. In general the controller is in the idle state until value updates are received.

Value updates can be divided into two major categories and three subcategories.

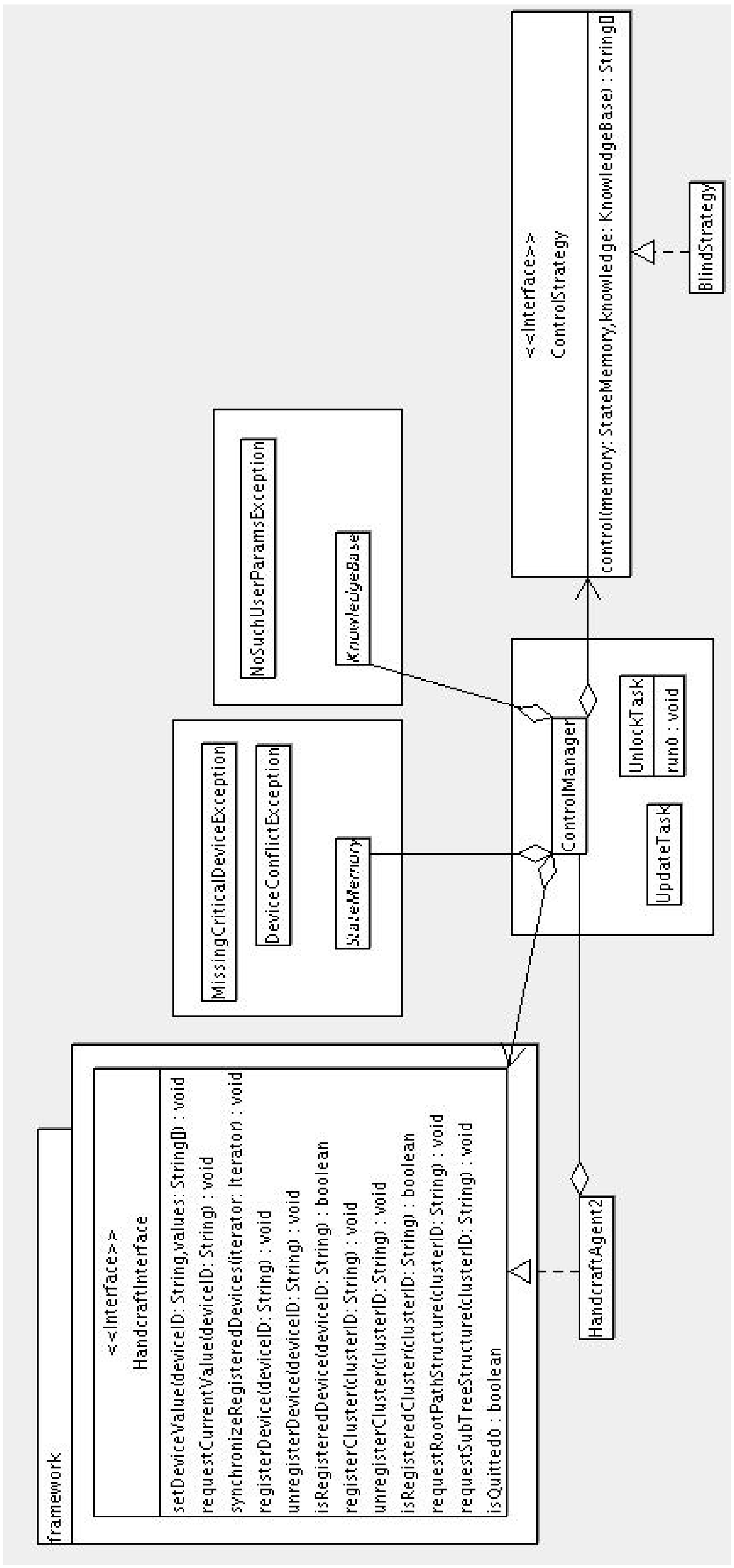

framework
<<Interface>>
HandcraftInterface
setDeviceValue(deviceID: String,values: String[]) : void
requestCurrentValue(deviceID: String) : void
synchronizeRegisteredDevices(iterator: Iterator) : void
registerDevice(deviceID: String) : void
unregisterDevice(deviceID: String) : void
isRegisteredDevice(deviceID: String) : boolean
registerCluster(clusterID: String) : void
unregisterCluster(clusterID: String) : void
isRegisteredCluster(clusterID: String) : boolean
requestRootPathStructure(clusterID: String) : void
requestSubTreeStructure(clusterID: String) : void
isQuitted() : boolean
HandcraftAgent2
MissingCriticalDeviceException
DeviceConflictException
StateMemory
NoSuchUserParamsException
KnowledgeBase
ControlManager
UpdateTask
UnlockTask
run() : void
<<Interface>>
ControlStrategy
control(memory: StateMemory,knowledge: KnowledgeBase) : String[]
BlindStrategy

Figure 29: ControlManager composite

- environment update
  - sensor (e. g. illumination) update
  - presence (e. g. PIR & pc-bus client) update
  - effector (e. g. blind; light) updates
- user interaction

The processing of environment updates means storing the values for later availability by the *ControlStrategy*. This is done by the *StateMemory*. The subcategories of environment updates (sensor, presence and effector updates) differ in the delay by which the effector position has to be reevaluated. Sensory values usually follow a trend, e. g. it is getting darker outside, which has no sharp threshold and does not require immediate reevaluation. Delayed updating is an easy way to summarize different sensor updates into a single effector update, hence reducing the total number of effector movements. Presence updates, on the other hand, especially those indicating a user becoming present, require immediate reaction in the hope to establish the users desired preferences.
User interactions which are the second main category of value updates after environment updates, are handled differently. Their immediate value has no meaning in controlling the effector, hence user interactions have to be generalized and accumulated for future prediction. This mechanism is encapsulated by the *KnowledgeBase*.

ControlAgent The mapping of previously described flow chart to java is described by figures 29 30 32. The controllers are subclassed from *ABIAgent*. The agent provides the living hull of the controller and its composite classes in the multi agent system. It is abstracted by the *HandcraftInterface*, which limits the vast number of functions of the *ABIAgent* to about a dozen[18] effectively required to enable controlling.

ControlManager The composite class *ControlManager* keeps track of the overall state of the controller. It builds a bridge between the controller subsystems and the multiagent framework. It ensures the control flow for message processing and periodically calls the *ControlStrategy* to reevaluate the effectors position. Value updates are first passed to the StateMemory and then to the KnowledgeBase. If the device source of the update is unknown to both of them, an *ABIException* is thrown and the update is bounced. In case of a valid update the controller enters the dirty state. Either immediately in case of presence or eventually in case of sensor update the controller calls the ControlStrategy to reevaluate the effector position.

On any user interaction the controller gets disabled, which is implemented as a lock variable in the user's *CtrlParametersProxy* (see KnowledgeBase below). Disabled means the ControlManager is not allowed to change the effector position. The lock hold by the user is removed once he becomes absent. In the experiment it gets also removed automatically after 2.5 hours which is checked by the UnlockTask, see method 7.2.4 on page 26.

[18]Not shown on the figure, basically send/request value, register/unregister for value updates and unpacking of structure-change notifications.

ControlStrategy This is an interface providing a single function *control*. The *BlindStrategy* is implemented as a flyweight pattern[10] that is completely stateless because the entire context is passed through the input parameters of the function. The context consists of the StateMemory and the KnowledgeBase. The return value of the function are the desired effector parameters in form of a string array, e. g. height and tilt value in the case of a blind effector.

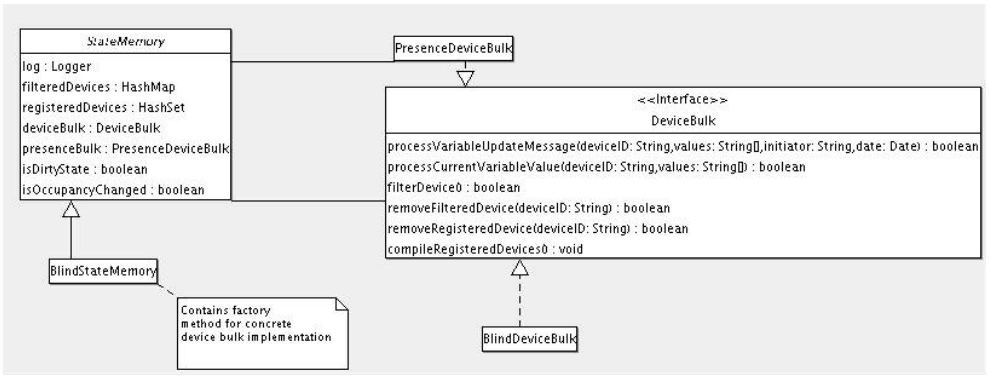

StateMemory
log : Logger
filteredDevices : HashMap
registeredDevices : HashSet
deviceBulk : DeviceBulk
presenceBulk : PresenceDeviceBulk
isDirtyState : boolean
isOccupancyChanged : boolean
PresenceDeviceBulk
<<Interface>>
DeviceBulk
processVariableUpdateMessage(deviceID: String,values: String[],initiator: String,date: Date) : boolean
processCurrentVariableValue(deviceID: String,values: String[]) : boolean
filterDevice() : boolean
removeFilteredDevice(deviceID: String) : boolean
removeRegisteredDevice(deviceID: String) : boolean
compileRegisteredDevices() : void
BlindStateMemory
Contains factory method for concrete device bulk implementation
BlindDeviceBulk

Figure 30: State memory class diagram

StateMemory The purpose of the abstract StateMemory is to select the devices involved in controlling the effector and to cache their most recent update message. The caching is done by updating a device-id/value pair in a *java.util.HashMap*. The main problem is to extract the devices involved in controlling and enforcing constraints between devices throughout structure changes. This problem is encapsulated by two device bunches: the presence device bunch and the concrete sensor device bunch which depends on the effector type.

To make an example of a constraint between devices, a personal blindswitch is useless without the corresponding presence device, because the controller doesn't know when to optimize for the specific user. Additionally at least one effector device is necessary for controlling, but in contrast having more than one shadow device will result in a contradictory shadow angle updates. Missing devices or conflicting devices will lead to termination of the controlling agent.

The question is why to get so fancy? It would be sufficient to register to all devices and then let the ControlStrategy decide which values to take. This would certainly solve the problem for the StateMemory but as the design of the ControlStrategy and sensor device bunch (DeviceBulk[19]) go hand-in-hand, it means just postponing the problem. There has to be a decision which devices are involved in evaluating the effector position and it's more economical to subscribe to them only.

The abstract StateMemory is based on the facade and template pattern[10]. The StateMemory implements the control flow to process structure-change messages. The effective processing is handled by two classes: *PresenceDeviceBulk*

[19]The word bulk is misleading, because the author was mistaken by the German word *Pulk* which is translated into English as the word *bunch*. So read bunch when you see bulk in the class names.

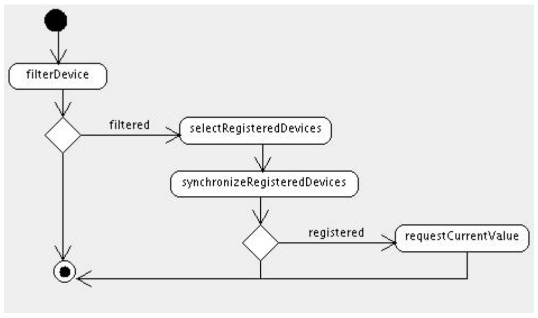


Figure 31: Processing of structure change message by the state memory.

and *DeviceBulk*. The presence bunch is always fix as knowing the presence of users is necessary for all controllers. It encapsulates the user management such as which users are able[20] to control the effector and which of them are actually present. The sensor bunch(DeviceBulk) must fit the effector type and the concrete type gets selected by an abstract factory method in the StateMemory. The concrete type of the StateMemory which is declared abstract is part of the configuration of the controller and gets set at agent startup(HandcraftAgent).

See figure 31 for an activity diagram illustrating the mechanism of registering for device updates. When a structure-change arrives the device type is checked for possible interest for controlling by the filterDevice method of the two device bunches. If either method indicates that the device is of possible interest the function *selectRegisteredDevices* of the sensor device bunch(DeviceBulk) gets called. The function assembles the list of interesting devices out of all filtered devices. As outlined previously it enforces a set of constraints on devices. The control manager then passes the assembled list to the HandcraftAgent which synchronizes it with its internal list of subscribed devices. If the device triggering the control flow is an interesting device its current value is requested. On agent startup when requesting the control cluster all devices can be filtered and subscribed in one batch and the list of registered devices has to be assembled only once.

The removal of a registered device works the opposite way. By calling *removeRegisteredDevice* the device gets removed from the list of registered devices. If a device constraint gets violated removeRegisteredDevice is called recursively for all devices part of the violated constraints until no more constraint is violated. Finally the device that triggered the unregistering of devices is also eliminated from the the list of filtered devices.

Hence the StateMemory manages the list of filtered and registered devices while the device bunches encapsulate the problem of processing individual devices. Two flags indicate whether an immediate or an eventual update is necessary depending on whether the sensor device bunch or presence device bunch has been updated since the last reevaluation of the effector position. The interface of the StateMemory is rather large as it proxies all important functions of the two device bunches.

[20]E. g. are running the pc-client

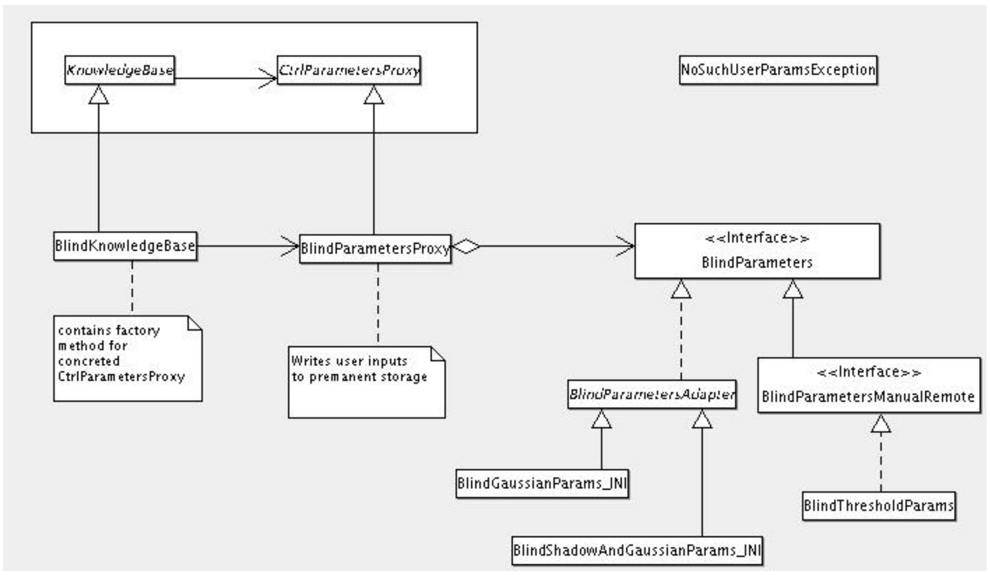


Figure 32: Knowledge Base

KnowledgeBase The purpose of the knowledge base is to store user inputs and to generalize them to predict user needs in the future. See figure 32 for class diagram overview. For each user an instance of a subclass of *CtrlParametersProxy* is created. The list of active users[21] is extracted from the state memory, which is accessible by the knowledge base. The proxy or bridge pattern is used to separate the static from volatile parts of the parameter implementation. The static part of the parameters involve logging of inputs on permanent storage and managing the user locks used to disable the controller. The actual algorithm was quite experimental and changed frequently during development. They were mostly implemented as matlab generated C-code[22], wrapped by a Java Native Interface(JNI). So the Java part of the algorithm implementation consists of transforming the user inputs and passing them to native functions. The actual parameter implementation is selected as a compile time constant, but a mechanism exists to change it at runtime via RMI.

The control-flow for processing user inputs is one-way. The user input is sent to a concrete subclass of CtrlParametersProxy by a method-call *addUserInput*. This proxy class logs the data on persistent storage and passes the values to the effective parameter implementation, which passes it along to the compiled matlab functions. On each user interaction the entire model gets retrained. In case of BlindShadowAndGaussianParams_JNI in the figure 32 the implementation is identical to the one used in simulation and visualization, see results 14 on page 66. It was designed as an offline learning algorithm, processing all user inputs in one batch, but due to the small number of interactions it is computationally feasible to use it as an online learning algorithm. There exists also an incremental algorithm as outlined in Mackay[29] but has not been implemented

[21]those involved in controlling, not necessarily present
[22]See appendix B

yet.

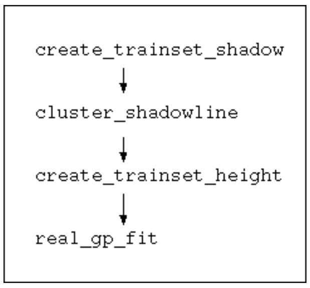


Figure 33: Control flow of function train_blind. The listed names reference other MATLAB functions described later.

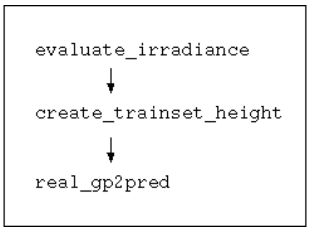


Figure 34: Control flow of function get_irritation_height. First the input tuple irradiance/timestamp has to be built and the trainig-set to be restored. Then the function *real_gp2pred* is called to evaluate the height at the requested timestamp.

11 Description of MATLAB functions

11.1 Blind control

Training Training of model parameters is done by function *train_blind2*, see description below.

```
SYNTAX: [ shadowline, MLfit, non_shadow_inputs ] =
                train_blind2( user_input, orientation, simulate )
Used to extract most likely model parameters, such as shadow line
and hyperparameters for Gaussian Process.  The irradiance data are
read from file system.

user_input          n x 3 array:  timestamp, height, tilt
orientation         facade orientation [ 'east' | 'west' ]
simulate            [ 0 | 1 ]

shadowline          distance from window base in blind height ticks
MLfit               hyperparameters for Gaussian processes
non_shadow_inputs   user_input not classified as glare avoidance
```

The control flow of the function is show in figure 33. For the detailed description of the purpose of this function see section 7.2 on page 24.

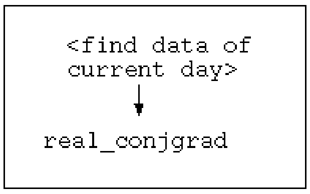


Figure 35: Common control flow of the functions train_irradiance_all and train_irradiance_1day.

Evaluation Because the shadow-line is extracted during training phase, evaluation reduces to interpolation of the height. See figure 34 for outline of control flow. First the second input dimension, irradiance, has to be interpolated. Because training and evaluating Gaussian processes is not very different the training set used in training has to be restored. Then the Gaussian processes can be evaluated at the requested timestamp/irradiance-tuple. See description of *get_irritation_height* below.

```
SYNTAX: height =
        get_irritation_height( non_shadow_inputs, M, eval_time, simulate)
Estimates the maximal height the user can stand at a given time.
The irradiance data are from the file system.

M                   hyperparameters found by train_blind[x]
non_shadow_inputs   training set
orientation         facade orientation [ 'east' | 'west' ]
eval_time           time of evaluation
simulate            [ 0 | 1]

height              assumed maximal tolerated height
```

11.2 Irradiance interpolation

There is a difference between the Gaussian processes used for blind height regression and those for irradiance interpolation. In the blind height problem the Gaussian processes are trained and evaluated by the same agent.

For irradiance interpolation the training is done by the *IrradianceAgent* while the evaluation is done through the blind controller *ControlAgent*. Hence the problem arises how the sensor data and the hyperparameters are passed between the different agents. Currently this is done through the file system because it is the most trivial solution. The IrradianceAgent see 10.2.1 on page 48 writes the sensor data and the most-likely hyperparameters into matlab file in a configurable directory on disk. For evaluation these values then just have to be reread from the file. The problem is that there might be a race-condition when the file is read while it gets overwritten by the training agent. While the problem has not been observed yet, it can not be excluded.

Training Two functions are available for training.

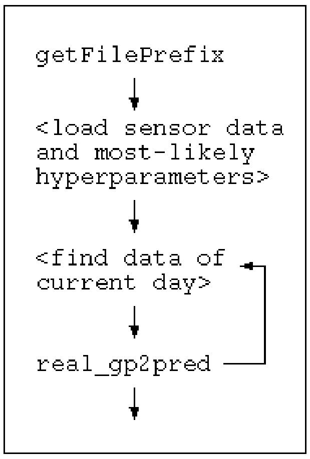

getFilePrefix
<load sensor data and most-likely hyperparameters>
<find data of current day>
real_gp2pred

Figure 36: Control flow of the function evaluate_irradiance.

- *train_irradiance_all* Batch processing of all sensor data.
- *train_irradiance_1day* Just training a single day.

The control flow of the two functions is identical, see figure 35, except that the former iterates over all days of available sensor data while the latter only trains the day indicated by a timestamp. See description of *train_irradiance_1day* below.

```
SYNTAX: ML = TRAIN_IRRADIANCE_1DAY(ML_in, irradiance, timestamp)
finds the most-likely(ML) hyperparameters of the Gaussian processes
for the irradiance values for a 1 day indicated by the timestamp

irradiance          irradiance sensor data
timestamp           timestamp indicating the day
ML_in               call-array of most-likely parameters for each day

ML_out              identical to ML_in except for ML_in<day>
```

Evaluation Evaluation of irradiance at a given time is done by the function *evaluate_irradiance.*

```
SYNTAX: rir = evaluate_irradiance(orientation, X, simulate)

Evaluates the Gaussian processes previously trained by train_irradiance_[1day|all]

orientation  facade orientation [ 'east' | 'west' ]
X            vector of timestamps where gp should be evaluated
simulate     [ 0 | 1]

rir          evaluated irradiance values
```

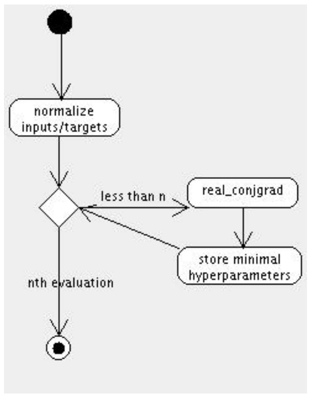


Figure 37: For training Gaussian processes.

See figure 36 for illustration of the control flow of the function. Function *getFilePrefix* returns the name of directory where the file with sensor data and hyperparameters are found. *evaluate_irradiance* then breaks all requests for evaluation into individual days and evaluates request of one day by a call to *real_gp2pred*. The last step is iterated until irradiance values for all requested days are processed.

11.3 Gaussian Processes

The matlab implementation for evaluation and training is taken from a course of David Mackay's whose implementation in turn is based on the C implementation listed at the end of the David Rasmussen[31] thesis. See appendix A for more details such as used covariance function.

Training Training in our case means finding the maximum-likelihood hyperparameters of the covariance matrix. This is done by conjugate gradient method. The normalization of the data and control flow is implemented by the function *real_gp_fit* see figure 37.

```
SYNTAX: M = REAL_GP_FIT(INPUT, TARGET, NFIG, SIMULATE)

Optimizes hyperparameters of Gaussian processes then times
by conjugate gradient method.  Returns the hyperparameters
with the lowest mean-square-error between target and
predictions at the input values.


see also REAL_CONJGRAD, REAL_GP2PRED, GP_ERR
```

As the space of the hyperparameters is non-linear the conjugate gradient might be stuck in a local minimum. We take a probabilistic approach by performing 10 training runs and while drawing starting values of the hyperparameters from a gaussian distribution. The final hyperparameters taken are those with the smallest mean-square-error between prediction and target values. This is actually an error as the the most likely parameters given the data, should be chosen and not the means-square-error between prediction and target values. Hopefully this does not decrease the perfomance too much.

Evaluation Similar to training, see appendix A. The difference is that the optimal hyperparameters are already known so the covariance matrix has to built only once. It would be more efficient to cache covariance matrix corresponding to the most-likely hyperparameters, but currently it gets rebuilt each time for evaluation, which leads to unnecessary computation.

Part IV
Results

This part will present results from preprocessing stages and evaluation of our blind interaction model.

12 Irradiance

This section shows interpolation of irradiance sensor values by Gaussian processes, see figure 38. The majority of the fits are surprisingly good. There are problems with unstable days where irradiance jumps between two extremes; then the interpolation takes the average whereas for blind control it might be more important to now the maximal value irradiance can take, because users probably care most about high irradiance values.

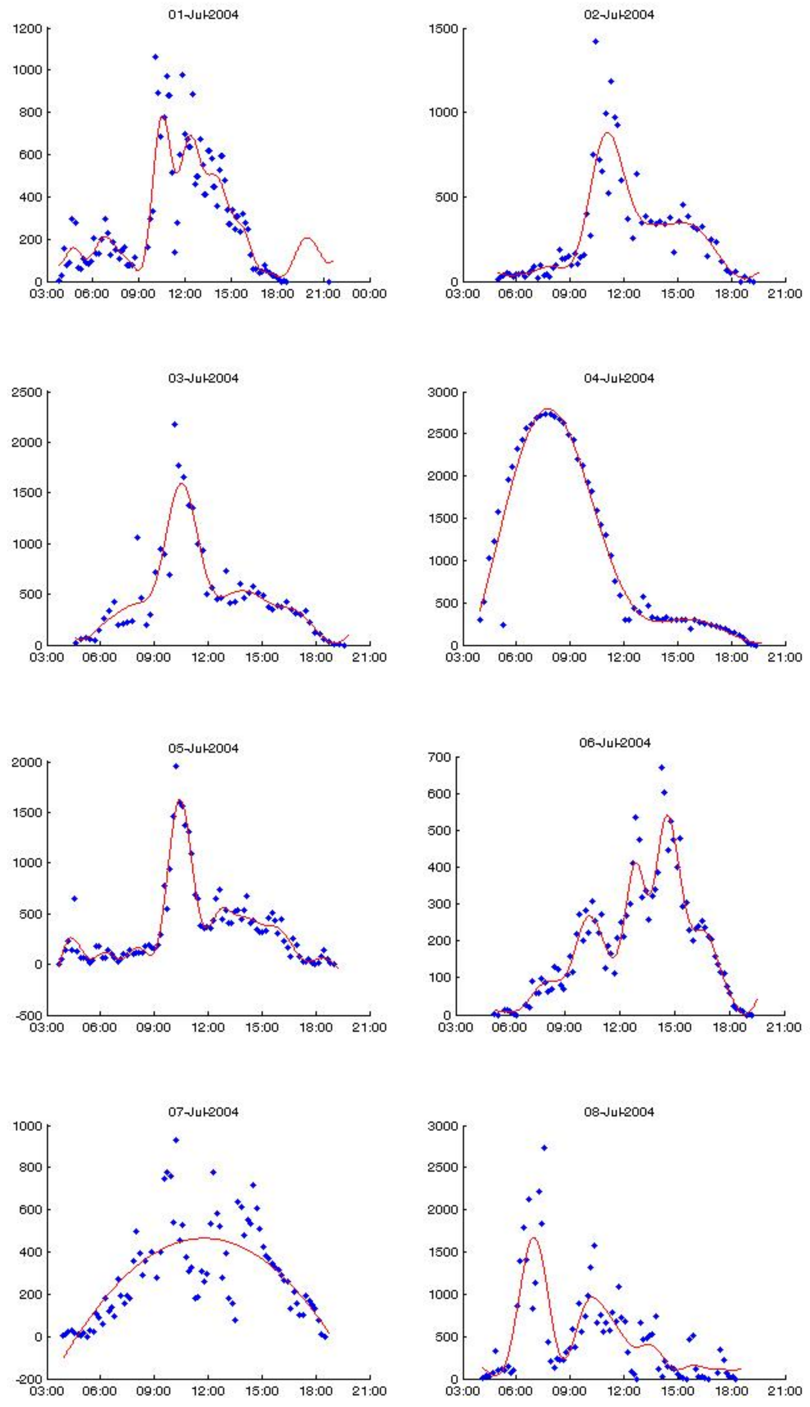


Figure 38: Irradiance interpolations of east sensor from first 8 days in July 2004. The y-axis is the irradiance in arbitrary scale and the x-axis is daytime. The (blue) points are real sensor measurement and the (red) line are values interpolated by Gaussian processes.

13 Sunniness

In this section we will discuss the performance of the heuristic for estimating the sunniness. The discussion is based on a plot of irradiance versus sunniness fraction, see figure 39.

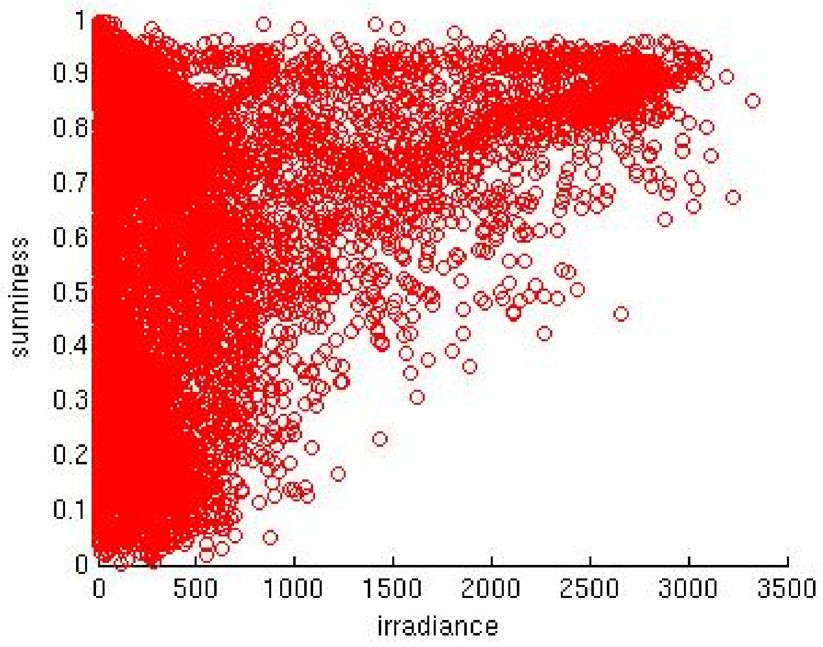


Figure 39: Plots of sunniness versus irradiance. The values are computed by iterating through all measured values of the east irradiance sensor and computing the corresponding sunniness fraction.

All datapoints lay in a triangular area. For a low sunniness value which should indicate cloudy weather the corresponding irradiance value is also low. For high sunniness values a much larger range of irradiances occurs. It seems though that for all sunniness values low irradiance values are predominant and that some areas within the triangular area are less covered by datapoints, especially mid-range irradiances are are less frequent than high or low irradiance values.

But we have to remind that the plot is limited because it relates sunniness fraction to the east irradiance taking not into account whether the sun is currently present on the east side. This means that that a low irradiance value might be due to overcast sky or just because the sun is absent on the facade and the sensor measures only diffuse irradiance.

14 Data-logging results

Presented data were logged by the deployment server that had no blind controller running. The period of logging is from 4th of March till the 11th of July 2004. It has to be mentioned that the west side of the building is occluded by a nearby building which affects the date and the analysis because our model has no obstacle detection yet. Beside the limitation of our model the data from the west side is very interesting for shadow-line clustering, because the sunset is in the west, so supposable most users will operate the blinds during the afternoon, when the sun is coming from behind the neighboring building.

14.1 Example User

First we will demonstrate the method by a user where the model seems to fit particularly well. Later the data of the remaining users will be presented.

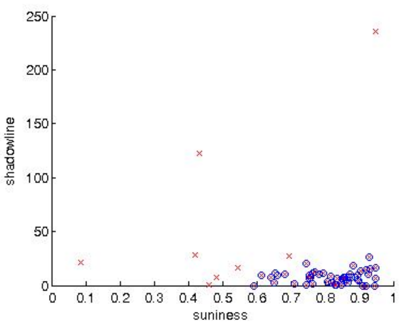
Shadow-line clustering This paragraph demonstrates shadow-line clustering. The method of hierarchical clustering is best illustrated by the **dendrogram** (see figure 40). The dendrogram is a tree where the leaves represent individual points. Points with the smallest distance in normalized input space are joined into clusters, symbolized by a crotch in the dendrogram. Each cluster is in turn treated like an individual point, so all points and clusters form bigger and bigger clusters until all points are in a single cluster symbolized by the root of the tree. The height of each crotch is relative to the average distance between points within the cluster, as discussed next.

The **inconsistency** compares the height of a link in a cluster hierarchy with the height of adjacent links below it. It can be seen as the fraction between the height of the crotch divided by the average of the height of its subcrotches. In the space of the input points it is the inter-cluster distance divided by the average intra-cluster distance. A low inconsistency means that the distance between the subclusters might be about the same as the distance between two arbitrary points within the same subcluster.

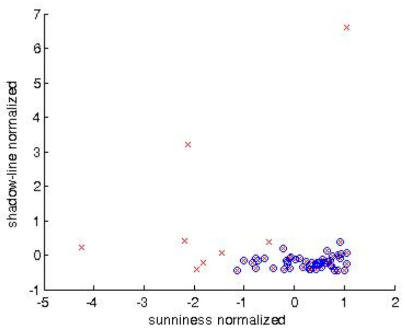
By defining an inconsistency threshold, which says that the inconsistency shall not exceed a given threshold, the dendogram breaks into several clusters, the smallest containing only an individual point. Currently the threshold is set to 0.35 by trial and error. From the generated clusters the biggest cluster is chosen, see a) and b) in figure 40

Most of the points have about the same shadow-line and a high sunniness value. They all seem to form a single dense cluster. Many other points are not present, but points with a negative shadow-distance are not shown on the plot. A negative shadow-line occurs when the sun is on the other side of the building and the shadow thrown from the blind bottom would point out of the window.

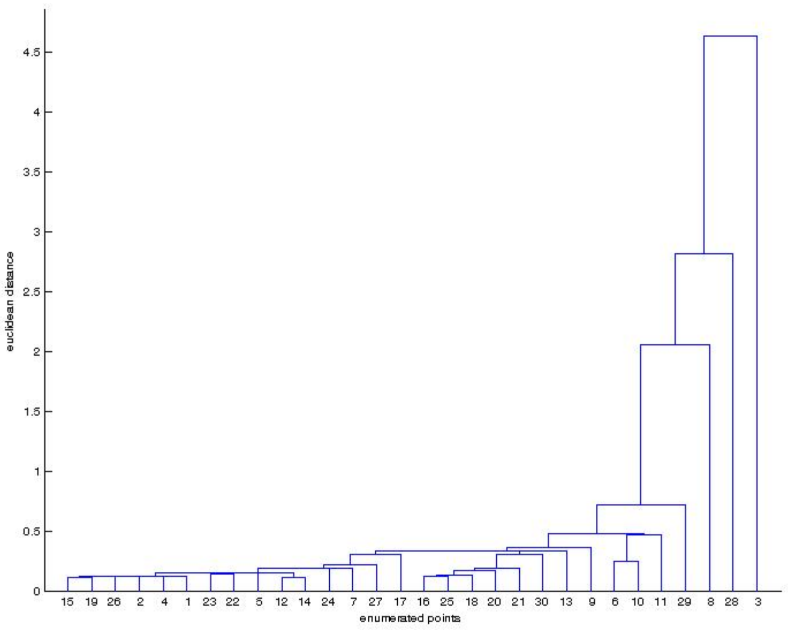
Although we expected this to happen it is surprising that the glare avoidance cluster is very dense and most points are part of it. Figure 41 plots all interactions of the glare avoidance cluster in blind height versus daytime space. The interactions do not always happen at same daytime and the resulting blind height is mostly different. This makes our assumption of a glare avoidance pattern even stronger, because all these interactions map onto a similar shadow-line value which is unlikely a random effect.



(a) user input



(b) normalized user input



(c) dendrogram

Figure 40: *User1:* The top-left plot shows the user inputs in shadow-line/sunniness space. A large shadow-line means that the shadow enters deep into the room, whereas a small shadow-line means the shadow is close to the window. Low sunniness means it is cloudy. Each (red) dot represents a user interaction, but only inputs with a positive shadow-line are plotted. Points with (blue) circles around them are part of the largest and densest cluster as identified by hierarchical clustering and are assumed as interaction from glare avoidance. The top-right plot shows the same data after normalization (variance=1). The bottom plot shows the dendogram which illustrates the hierarchical clustering algorithm (see text for explanation). The clustering algorithm operates on the normalized data.

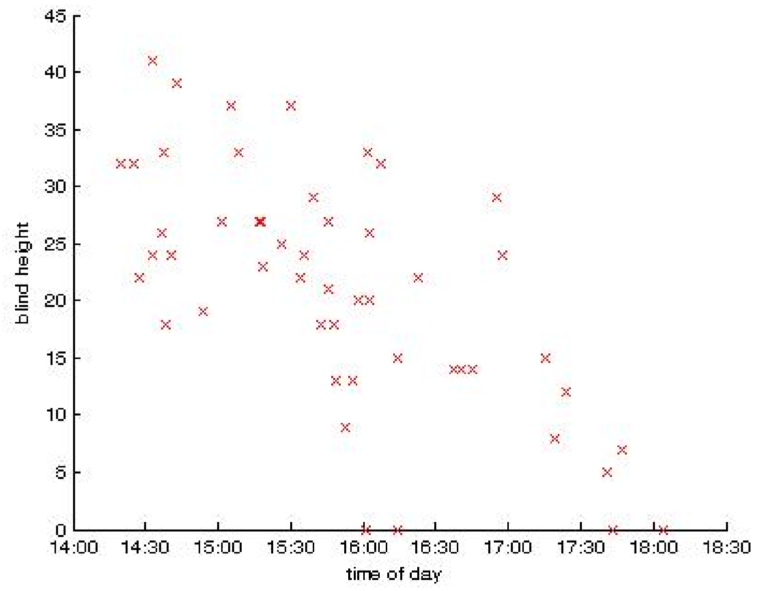


Figure 41: The plot shows user interaction assumed as glare avoidance. The x-axis is daytime and the y-axis is blind height

Illuminance control This section demonstrates illuminance control. The input for illuminance control consists of the remaining interactions not assumed as glare avoidance, these are all points not part of the biggest cluster in figure 40 plus all user inputs with negative shadow-line because the sun was on the other side of the building at the time of interaction. Figure 42 shows these remaining points by a contour plot in sunniness/irradiance/blind-height space.

The input points lie within a triangular frame in the sunniness-irradiance plane. The corners of the triangle are low sunniness/low irradiance, high sunniness/low irradiance and high sunniness/high irradiance. Points at low sunniness/high irradiance do not occur. This matches the intuition that high irradiance requires an uncluttered atmosphere that is not reflecting and absorbing the irradiance. For extremely low irradiance values(<350 arbitrary scale) the user wants the blind almost completely up on cloudy days, whereas not much can be said about sunny days at low irradiances because no interactions are present then. This might be because on sunny days low irradiance values are very rare or may occur just early in the morning and and late evening when users are absent.

For irradiance values above, ca. 400 arbitrary scale, the user input is quite consistent. The overall tendency is to lower the blind with increasing irradiance values. At times of equal irradiance the blind is lower on cloudy days. There are three local extrema contradicting this tendency in the area of high sunniness and medium irradiance. Two of the extrema are maxima where one of them has 60% blind opening, which is double as much as what neighboring points suggest. The remaining minima deviates from the trend only by a value of 20%

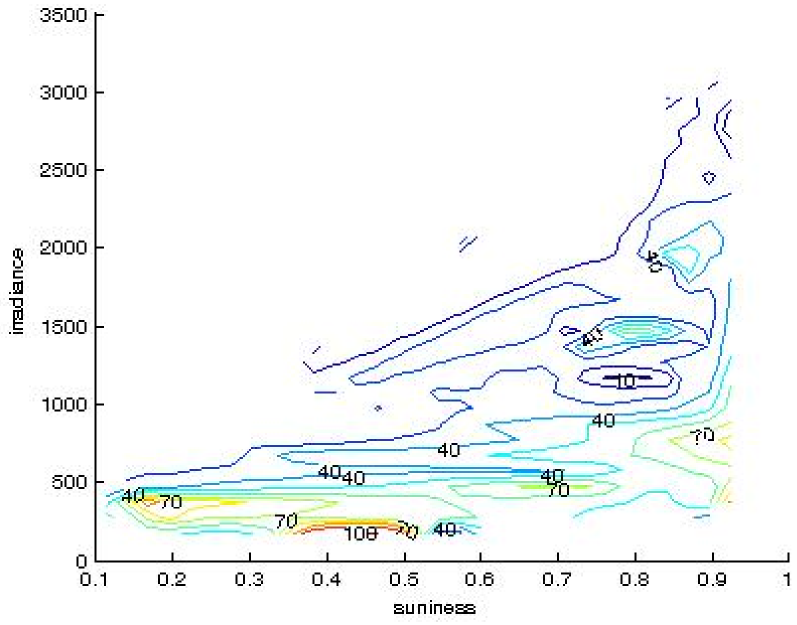


Figure 42: *User1:* The x-axis represents sunniness and the y-axis the outside irradiance in arbitrary scale. The z-axis which represents the blind height is shown by contours. The plot was generated by the MATLAB method griddata, which interpolates the user input with equidistant steps of 10% of blind opening. The irradiance scale is arbitrary as the sensor is not correctly calibrated. The maximal blind height is 100% but has been over heighten to 120% for regression so that a prediction of 100% can still be reached even in the presence of outliers or exceptional interaction.

of blind opening.

After showing the results to the user, he gave an interesting hint about why he chooses lower blind positions for low sunniness values. He mentioned that the diffuse light on cloudy days is brighter because the clouds spread the light into many directions, so more light ends up in the observers eye. For the same irradiance but on a clear day the amount of light that enters the eye is much lower as it gets only reflected by the surround such as grass or concrete. So the latter case might be less fatiguing when working near the window.

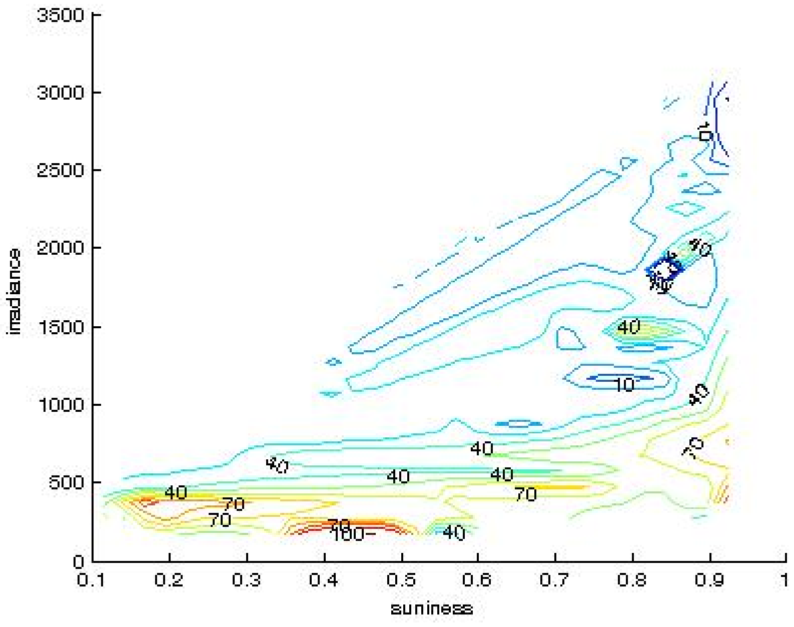


Figure 43: *User1:* The axis of the figure have the same meaning as in figure 42. But no clustering and elimination of assumed shadow-line cluster was performed.

Figure 43 shows the method of illuminance reduction without prior removal of user interactions assumed in direct glare avoidance. Comparing it with figure 42 outlines that the removal of the glare avoidance cluster does not change the underlying structure of the data fundamentally because the two plots look almost identical. By comparing the extrema at medium irradiance and high sunniness we can see that the plot without prior removal of shadow-line cluster contains one more minima which differs from the average of adjacent interactions in figure 42 by 20% window opening. Table 5 shows the mean-square-error of the normalized data with and without discarding interactions assumed in glare avoidance. The mean is about the same in both cases. Throwing mathematical accuracy to the wind by taking the square root from the mean square error the estimated blind position is at the average about 50% off from what the user desired at time of interaction. Which is to inaccurate to play it back to the user.

Fitting function	with clustering	without clustering
Gaussian Process	0.25787	0.25063

Table 5: MSE User1: The mean-square-error computed between the height set by the user and the evaluated height by the fitting function. The error is computed on the normalized training set (standard variance = 1).

Discussion user 1 Figure 40 shows that most interactions are part of a dense and big cluster in shadow-line/sunniness space, which we assume to be motivated by glare avoidance from direct light. This assumption is backed by figure 41 showing that the interactions part of the densest cluster are separated in blind height/daytime of interaction space.

In figure 42 the remaining interactions not assumed in glare avoidance were projected in blind height versus sunniness and outside irradiance space. The plot suggests that there is a tendency in user interaction with blinds that depends on irradiance and sunniness. But as we found by using Gaussian processes to perform regression of blind height versus irradiance and sunniness, this tendency is not very succinct as we discusse next.

The performance from applying Gaussian processes for regression in luminance reduction are quite frustrating, see table 5. Not only that the removal of interactions assumed as glare avoidance brings no improvement but also the estimated blind heights quite off from what the user requested at the time of interaction. Taking into account that we did not separate the data into training and test set which will likely decrease the results further it seems that current approach of simply applying Gaussian processes to *all* remaining points is not the way to go.

The bad performance of Gaussian processes is likely because the interactions not assumed in glare avoidance do not have the simple and smooth structure in sunniness/irradiance space that we hoped for. The data contains a couple of extrema and below an irradiance of 350 arbitrary scale the blind height is increasing quickly, breaking the relatively simple structure observed above 500 arbitrary scale. So we assume that the remaining points are produced by at least two patterns, because below an irradiance of < 350 arbitrary scale, the user seems to be doing something different than what he does beyond 400 arbitrary scale. We suggest that there is a threshold when the user starts to care about outside illuminance and that below this threshold the blind position does not depend neither on irradiance nor sunniness. This threshold seems to be mostly dependant on irradiance and only slightly on sunniness.

The aforementioned extrema could be due to artifacts in the training-set such as occlusion by the neighboring building, interactions due to glare avoidance that have not been removed by hierarchical clustering or they indicate even more patterns of interaction than we identified so far. Another simpler explanation is that these are a consequence of our weak preprocessing, recall that especially sunniness has its weaknesses because it is dependent on irradiance, but also irradiance itself is not perfect due to the interpolation at times of interaction, see section 8 on 29.

While talking to the user he requested additionally for splitting the blinds into its individual subparts. See figure 44 for an illustration of the problem. The user requested the blinds to be split as the sun rays fall on his desk mostly

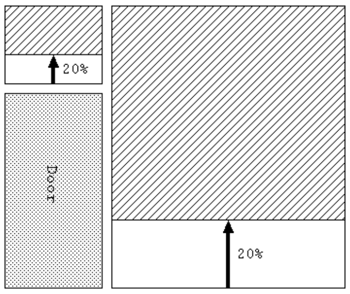


Figure 44: *Composite blinds:* The low-level blind controller has been configured to send the same command to both blinds. This commands encodes the relative position, e. g. 50% of the heigh88t of the window it covers. It does not encode the absolute distance e. g. in centimeters from the top ceiling. This means that if the small blind is set to an almost closed positions, the big will be almost closed as well.

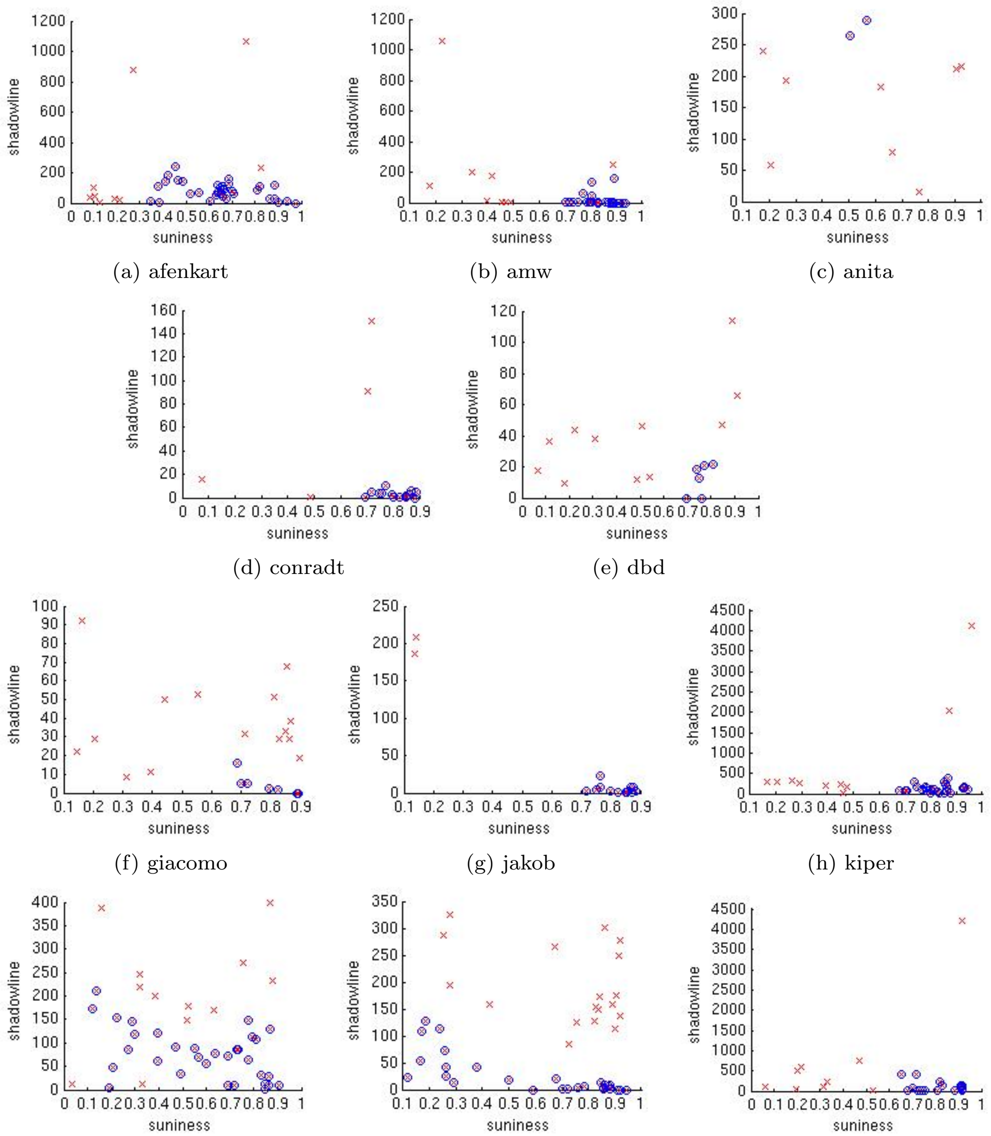
through the small blind. So for glare avoidance he is using the small blind while for illuminance reduction the big blind is used due to the larger area it covers.

The effect of this behavior on our model is probably a shift of the shadow-line as defined it on page 30. Recall that the shadow-line is measured relative to the size of the window. So small and big blind throw different shadow-borders on the floor or user's desk even if the computed shadow-line is identical. This because of the different size and different bottom height of the window they cover. Assuming that glare avoidance is done consequently through the small blind, then the shadow-line of the big blind gets overridden by the small blind, because it defines the minimum relative position. Effectively the composite blind makes the shadow-line of the big blind smaller because it could stay at a higher position resulting in a bigger shadow-line if operated independently. The shape of the glare avoidance cluster would not be affected because it would still be dense in case glare avoidance is done consequently through the small blind. So extraction of the glare avoidance cluster would not be a problem.

So as long as glare avoidance is done consequently through one blind a composite blind is not a problem. If glare avoidance alternates between the two blinds then we should see two clusters in shadow-line/sunniness space with its centers at similar sunniness but different shadow-line coordinate, which is not the case for this user.

14.2 Remaining Users

Figure 45 shows the results from shadow-line clustering for the remaining users. For most users a dense cluster is found and if so it is found exclusively on sunny days. An expected exception to this are users having an office on the north facade (c, e). On the north facade direct glare does not occur except very early in the morning or late evening hence if there is a largest and densest cluster it is not due to direct light avoidance and no prediction is made about its placement in shadow-line/sunniness space. This can be observed for user (c) where the cluster is not that large and dense and is does not occur on sunny days but not



(a) afenkart (b) amw (c) anita

(d) conradt (e) dbd

(f) giacomo (g) jakob (h) kiper

(i) mchr (j) pratap (k) rzwiker

Figure 45: These plots are identical to the leftmost plot in figure 40. Each dot represents a user inputs in shadow-line/sunniness space, but only inputs with a positive shadow-line are plotted. Inputs assumed as glare avoidance are marked by a circle. Users (c,e) are located on the north side of the building were no direct glare can occur.

for user (e) which exhibits an almost as dense cluster as users on the east and west facade with also a similar placement.

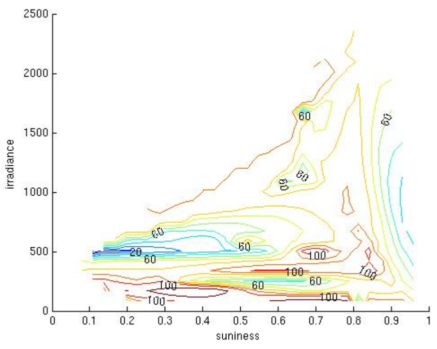
There seems to be a problem with the clustering algorithm for users (i), (j) and maybe (a) where the clusters are not very dense, probably due to the inconsistency threshold. All these users have generated lot of datapoints which cover a larger area in shadow-line/sunniness space. In these cases it seems the inconsistency threshold should be lowered because otherwise too many neighboring points are taken to be part of the densest cluster.

Figure 46 shows regression plots of users having their office on the east side of the building. The regression plots are similar to those of the the example user, again there seems a different behavior for low irradiance values than for high irradiance values. The exception is user (c) which seems to prefer a constant blind position independent of the outside conditions.

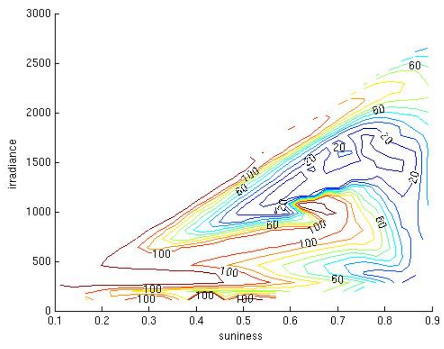
Figure 47 shows regression plots for users having their office on the north face of the building. The plots for this face of the building are not very similar to the one from the example user. For user (a) it can be said that too few data is present because the plot has a very small extent. User (b) used the blinds more frequently but the irradiance was never that high as for users on the east/west facade. There exist two extrema, one for low irradiance/large sunniness with a high blind height and the second for medium irradiance/medium sunniness with a low blind height. It seems that the user does not care about outside irradiance below a certain threshold which again matches the structure in the plot of the example user.

Figure 48 shows regression plots for users having their office on the west face of the building. User (d) is the example user discussed previously in the example section. The plots of users (a,b,c) differ from user (d) in that they exhibit extrema. Especially one extrema of user (a,b) at large irradiance values with large blind heights indicate that the neighboring building was putting the users office in shadow. We are quite sure about that, because so large irradiance values are not produced by diffuse irradiance only but also include direct irradiance, so nobody setting next to the window would raise the blind at these times unless its desk is already in shadow. We think that if we could remove all interactions that occured during occlusion by the neighboring building the plots of users a,b,c would be more similar to user than they already are. Notice that again there seems to be a line around an irradiance of 500 arbitrary scale separating the data into two sets of interaction.

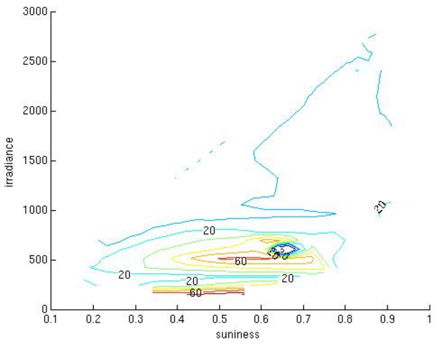
Summarizing, this data shows the difficulty of the problem; every user is different and not always consistent with even the face of the building.



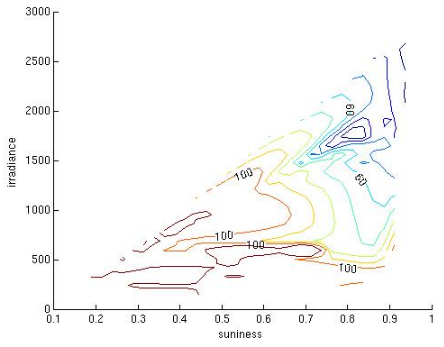
(a) afenkart



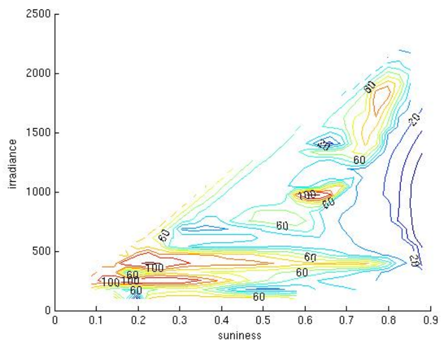
(b) rzwiker



(c) giacomo

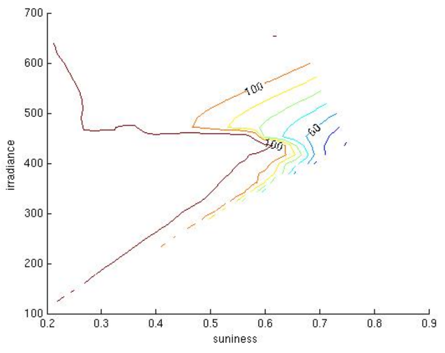


(d) kiper

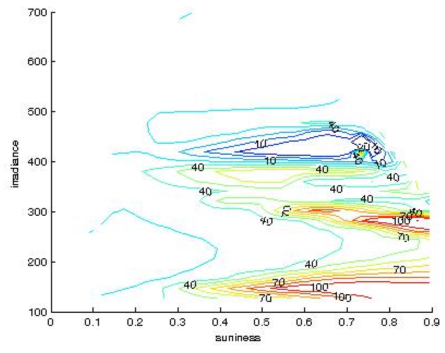


(e) mchr

Figure 46: Shows the data from users on the east facade after removing points attributed to glare avoidance. Except kiper all users have an unobstructed view. The x-axis represents sunniness and the y-axis the outside irradiance in arbitrary scale. The z-axis is represented by a contour and represents the blind height. The plot was generated by the matlab method *griddata*, which interpolates the user input at equidistant steps of 10% blind opening. The irradiance scale is arbitrary as the sensor is not correctly calibrated.

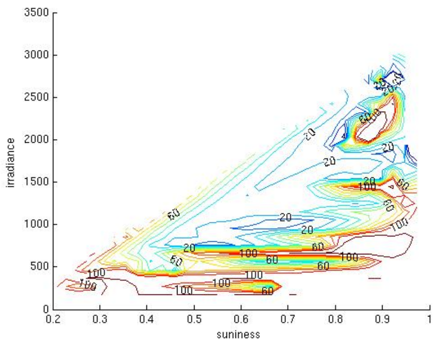


(a) anita

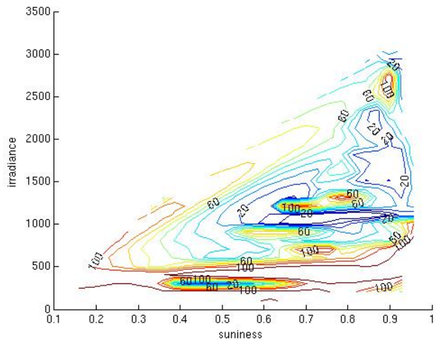


(b) dbd

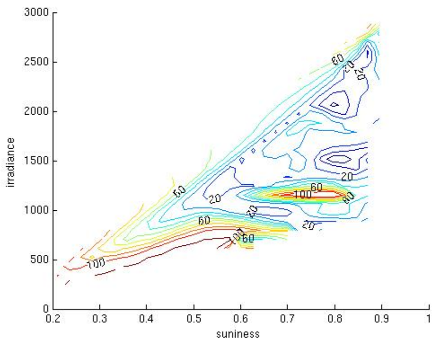
Figure 47: Blind height positions from users on the north facade. Axis labels are identical to those in figure 46



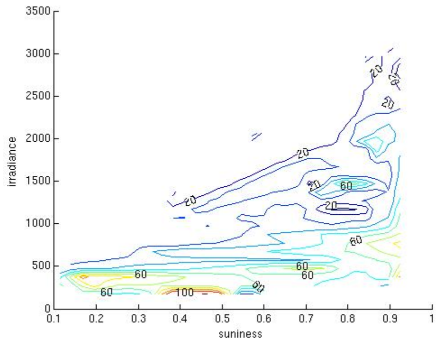
(a) amw



(b) conradt



(c) jakob



(d) pratap

Figure 48: Blind height positions from users on the west facade. Axis are the same as in figure 46 All users on the west facade don't see the sun directly until late noon. Depending on the location of their office it might occur between 2 PM to 5 PM.

14.3 Prediction errors

Unfortunately our method of applying Gaussian processes to the remaining interactions not assumed as glare avoidance from direct light, did not turn out to be useful. The assumption was that these interactions are motivated by controlling indoor light conditions or light flow through the window. Because the data exhibited a more complex structure (see sections 14.1 14.2), we find that this assumption is not true and that the problem is not that simple.

Up to now we stated implicitly the assumption, that all interactions can be assigned to exactly one pattern, e. g. in case of competing patterns such as glare avoidance from direct light and control indoor light conditions, it is always one that drives the user to set the final blind position. In our case for example the user will do glare avoidance in case of direct light, but if the surround is still too bright he will lower the blind even further until his desktop surround is dim enough. This assumption is important because it allows us to separate the data, so that the problem breaks into subproblems which can be treated independently.

During this section we will not go more into this, instead we try to isolate interactions motivated by luminance control and maybe try to identify more patterns of interaction. This section was written at the end of our work, so it has no effect on the controllers presented and the results collected with them. It is presented for completion and to give a hint in which direction we would be heading next.

From now on, when we talk about the user input we will always refer to interactions not assumed as motivated by glare avoidance. The attentive reader might indicate that removal of the glare avoidance interactions showed no improvement in term of mean-square-error(MSE) as outlined in the datalog section. We argue that the method we used to estimate the MSE was not suitable and that by visual comparison of the plots the data seemed to have simpler structure as can be seen for the example user where at least one extrema was eliminated completely (see section 14.1).

14.3.1 Luminance control pattern

During the discussion of logged user data, we frequently observed that there is a break in the structure of the data once the combination of irradiance and sunniness exceeds a given threshold. We do not know currently what drives user interactions below this threshold but in case above the threshold we assume the blind is used for luminance control.

In this section we try to evaluate our assumption by a trivial approach. We split the dataset into two sets by using a simple threshold that depends only on irrdiance and not on sunniness as observed in plots of several users. Further we do not use an adaptive threhold for each individual user but set it to a fixed irradiance value of 500 arbitrary scale[23].

To see if this brings improvement we apply Gaussian processes for regression of blind height to irradiance and sunniness. The error rates are computed by dividing the dataset into training- and testset of equal size (50% each). After

[23]Recall that the irradiance sensors are not calibrated correctly, as a rule of thumb divide by a factor of 3 to get the value in Watt.

training the Gaussian processes we evaluate the height for all points of the test-set and compute the absolute distance to what the user had chosen then, this difference we call error. Further evaluation is done by comparing the error rates with and without prior splitting the data set by a threshold of 500 arbitrary scale.

User	interactions	average error	maximal error
afenkart	24	17.6	40.3
amw	44	37.7	86.7
conradt	50	31.6	94.5
dbd	5	4.6	55.3
giacomo	18	8.0	17.7
jakob	32	32.6	86.4
kiper	13	15.4	39.3
mchr	20	25.0	70.7
rzwiker	10	45.6	78.1
pratap	51	14.4	53.2
tobi	8	41.6	67.2

Table 6: The table shows average and maximal error in percent. The error was measured by test and training-set of half size each. The interpolation was done by Gaussian processes. The input dimensions are global irradiance and sunniness; blind height is the target dimension. The dataset consists of interactions not assumed in glare avoidance and having an irradiance value of more than 500 arbitrary scale at time of interaction.

User	interactions	average error	maximal error
afenkart	46	26.0	102.6
amw	69	29.2	91.7
conradt	65	36.2	62.2
dbd	24	26.2	82.3
giacomo	26	12.9	29.9
jakob	35	35.0	93.1
kiper	21	16.1	35.1
mchr	50	25.4	65.0
rzwiker	25	41.6	95.5
pratap	79	21.5	90.6
tobi	12	37.5	62.5

Table 7: Similar to table 6 this table shows average and maximal prediction errors from the data-set consisting of all interactions not assumed in glare avoidance including those with an irradiance value below 500 arbitrary scale.

By comparing tables 6, 7 we find that in general that error rates are decreased by removing interactions at irradiancesof less than 500 arbitrary scales. Especially advantageous seems discarding low irradiance interactions for user pratap where the average error drops by 7% down to 14% and maximal error from 91% to 53%. This is especially exciting as he is also the user with the most interactions in both cases. The prediction errors for user giacomo are identical in both cases. An exception is user amw whose error rates are increased by splitting the data. But it has to be mentionned that this user sits on the west side and our algorithm does not have occlusion detection yet. The same problem tampers also the data of users conradt and jakob. Error rates from users with less than 10 interactions are not very expressive, because its more likely they all happened under same circumstances making prediction easier. By discarding interactions at times of low irradiance the data sets of each user has been reduced between 30% to 50%.

14.3.2 Discussion

By discarding interactions at low irradiances we reached prediction errors for two users (afenkart, pratap) around 15% which is probably accurate enough to satisfy the users if they are not very sensitive to blind position. Including user giacomo which is special in that he prefers a fixed blind height independant of outside conditions and user kiper that did not profit much from splitting we currently have 4 users that we probably could satisfy with our algorithm.

Unfortunately for the gros of the users we can't. Partly this is because they are users located on the west side of the building where occlusion detection is needed to know if the neighboring building is not throwing a shadow on the user's office. But this still has to be proven.

The method of choosing a hardcoded irradiance threshold is too simple as all users are different and probably need their own thresholds in addition that such a threshold is probably also dependant on sunniness and not irradiance only. Another issue to check is whether it makes a difference if the sun is present or absent on the face of the building at time of interaction. This probably makes a difference because if the sun is absent indoor reflections from desks next to the window are less likely to occur.

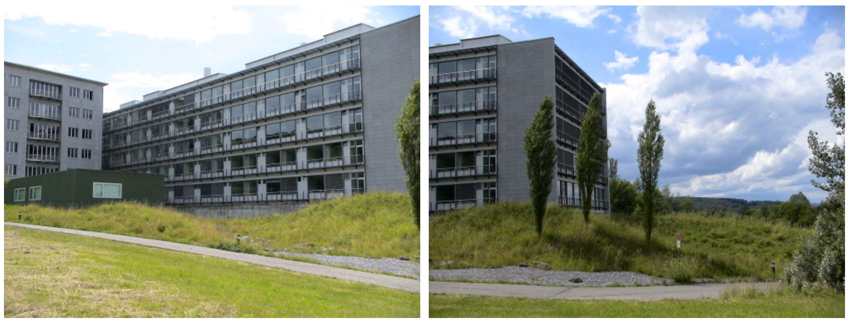
What users are doing when irradiance is very low has not been investigated in this section. If we assume that low irradiances are likely reached only during late evening, night or early morning, we suggest that around sunset/sunrise window reflections might play a dominant role in lowering blinds. During these times window reflections are more likely because most windows are mounted vertically, so for reflection to reach far the sun must be close to the horizon which means it is either sunset or sunrise

During night blind position depends probably on user's choice for privacy, because there is not much to see outside. Because the windows have no curtains users can be seen from far away when the room is illuminated in the night.

15 Experiments with the blind controller

This section will present the results gathered from experiment runs as described in section 7.2.4 on page 26. The data shown are taken in the period from the 25th June till the 12th of July.

15.1 Building description



(a) east (b) north

Figure 49: East and west side of the building. The G floor is the ground floor. On the picture of the east side the INI is in the right building. The green building in front of if does not project a shadow onto the office rooms because it is too small. The building to the left provides shading to nearby offices in the INI but not to those further away.

There are a couple of constraints in choosing suitable rooms for evaluating the blind controller.

- No obstructions such as other buildings or trees offering shadow.
- Offices with few people to prevent complicated conflict situtations.
- Not in use by a parallel running experiment.

Because the blind controller currently has no built-in obstacle detection, only offices were considered with an unobstructed prospect see figure 49. Because compromise finding is done by taking the minimum of all preferred positions, difficulties are to be expected when the regression erroneously predicts a low height for a user. Then the blind will go to that low position and nobody will know to whose preference it was. Hence ideally single-user offices and rooms with few persons are preferred.

Currently four rooms or subclusters fulfill these constraints: They are room 54 most-left and most-right subcluster, which are compartments of a larger room used as office space, see figure 50, then rooms 74 and 84 which are regular rooms in that they have doors to close them. In the controlled cluster between 1 to 4 persons are working. Due to technical difficulties only users of room 74 and 84 were participating in the experiment, see table 8.

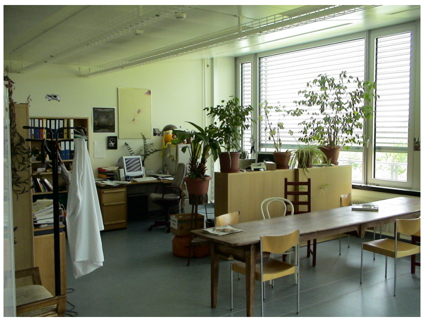


Figure 50: Compartments of *big-room* used as office space.

Room	Nr. Users	Users
Room 74	2	afenkart, rzwiker
Room 84	1	chicca

Table 8: Users involved in experiment

Protocol Unfortunately several problems occurred during the experiment.

1. Problems with the blinds.
2. Bug fixes in evaluation method
3. Glare avoidance even on days of overcast sky.

Problems with the blinds The blind motors are not designed to drive to an absolute position. The only positions that are detected by sensors is the top-most and down-most position. Any intermediate positions is computed out of the current position and the motor running time to drive to the desired position. To compute the running time the motor speed needs to be known and this speed is estimated by driving the blind from the top-most to the bottom-most position[24]. The problem with one blind was that the blind was unable get signals from the end-stopping sensors correctly, presumably due to a defective end-stopping sensors or erroneous communication between motor and motor-controller. But the problem is not yet isolated. The result was that the blind was completely unusable for our purpose and one of the controllers had to be turned off.

The second problem is that errors in the runtime estimation sum up if the blind is operated a few hundred times only in intermediate positions but never goes all up or all down. The errors come from the motor speed estimation which is inherently inaccurate. This problem has been solved by resetting position during the night. This is done by moving it to the top-most/down-most position a couple of times.

[24]And not vice versa

Still there are some unsolved problems as the top-most position is sometimes a bit under the mechanical limit. According to the vendor of the blind controller and the mechanics in charge of the blind motors this problem is unexplainable. The vendor of the blind controller says that if the blind is driving to the top- or down-most position the current is not broken until the end-stopping sensor gives a signal. According to the mechanic the end-stopping sensors work correctly because otherwise the motor would scream when the blind hits the mechanical limit. Summarizing, the blind shouldn't stop before reaching the mechanical limit, which it apparently does. This effect has been extreme in the chicca's room at least on 7^th of July, where the blind supposed the top-most position to be at 20% of maximal opening, so the results from this day had to be removed.

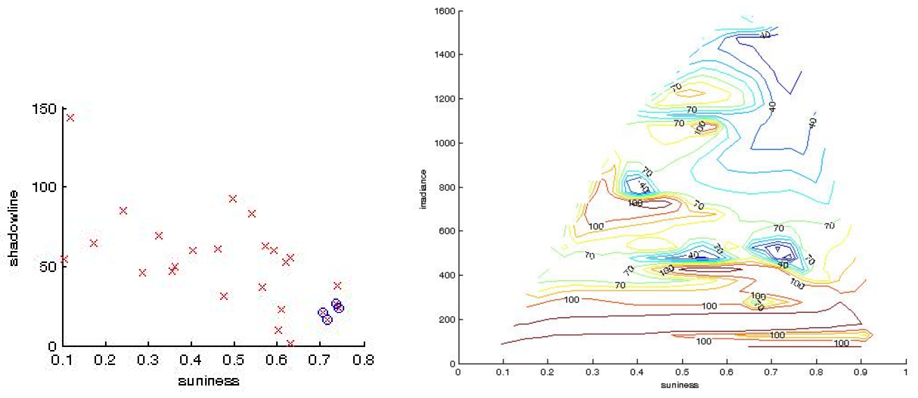
Bug fixes in interpolation method In the evaluation of the Gaussian process the coordinates for irradiance and sunniness were flipped, so evaluated blind positions were quite random. This was corrected on the 10^th of July and had the effect that the controller set the blind to an unwanted position probably resulting in more user interactions.

Glare avoidance even on days of overcast sky When the controller evaluates the effector position it should take into account that on overcast days no direct glare can occur. Currently this check is not done, because sunniness is not available from Java, which is not a technical problem just an oversight. Sunniness is correctly computed when training the user's parameters and estimating his shadow-line, but is not available to the ControlStrategy updating the blind position. The consequence of this bug is that the blind controller also does glare avoidance on cloudy days. Because it was cloudy on several days the users probably had to raise their blind manually, hence increasing the number of user interactions.

15.2 User parameter evaluation

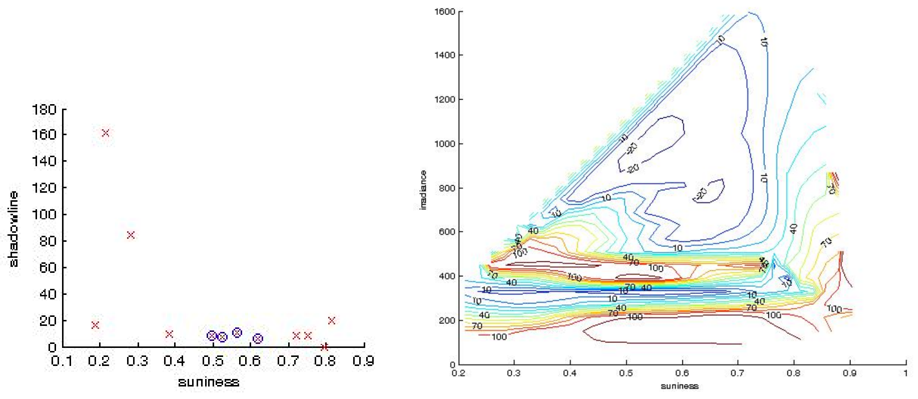
Due to the numerous problem outlined above the accuracy of the data is quite lost. For completeness we present the date here, keeping in mind that the results are suspect.

Figures 51 52 53 show the same plots as those in the data-logging section but this time the plots are grouped by users. Most interesting is the regression plot for user chicca as it matches exactly description of the example user in the data-logging section see page 70. The regression plots for user afenkart is quite different from plots of other users and also from results for the same user in the the data-logging section. It should be considered that this the author which is familiar with the working of the controller so he might do unusual interactions trying to make the model converge more quickly. Results from user rzwiker exhibit a different tendency than most other users, because he prefers a lower blind height on sunny days, whereas most other users usually chose the lower position on cloudy days. But this user is actually not sitting next to the window and can only see a small portion of the sky. So he probably is not bothered by diffuse light the same way as reported by the user described in the example section (see 14.1 on page 68).



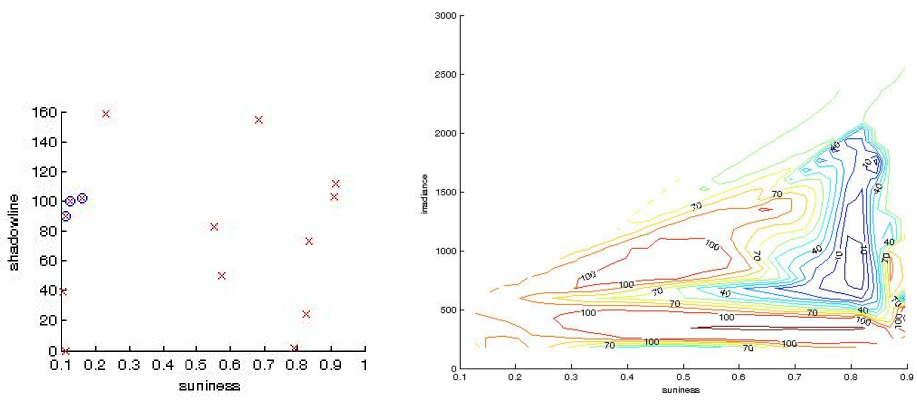
(a) clustering (b) regression

Figure 51: Results of user afenkart. Subplot (a) shows the results from glare avoidance cluster. Subplot (b) relates blind height to current outside irradiance/sunniness values at time of user input



(a) clustering (b) regression

Figure 52: Same plots as in figure 51 but with the data of user chicca.



(a) clustering (b) regression

Figure 53: Same plots as in figure 51 but with the data of user rzwiker.

15.3 User interaction evaluation

See figure 54 for plots of interactions per day versus day. Both rzwiker and afenkart are familiar with the algorithm and are probably not interacting with the blind as regular users do. So user chicca is currently the only unbiased user, which is discussed next.

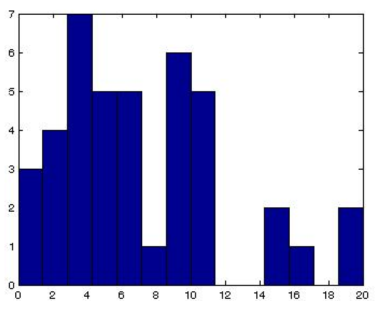
It general it seems that the controller did not reduce the number of user interactions. While for some days the user only interacts once with the blind which is half of what it was at the beginning of the experiments he also interacts with the blinds between 5 to 6 times on other days which is a triple as much as when the experiment started. This emphasizes that the controller is not really predicting user preferences accurately.

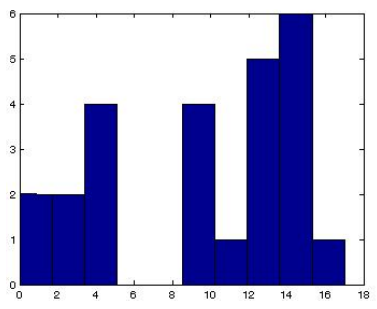
Table 9 shows average and maximal error in percent.

User	interactions	average error	maximal error
afenkart	55	17.7	36.2
chicca	17	25.7	58.8
rzwiker	25	42.6	97.3

Table 9: The table shows average and maximal error in percent. The error was measured by test and training-set of half size each. The interpolation was done by Gaussian processes. The input dimensions are global irradiance and sunniness; blind height is the target dimension. The user inputs are interactions not assumed in glare avoidance even including those at times of low irradiance.

By discarding all interactions that happened at an irradiance of less than 500 arbitrary scale, the following average and maximal errors are achieved.


(a) afenkart


(b) chicca

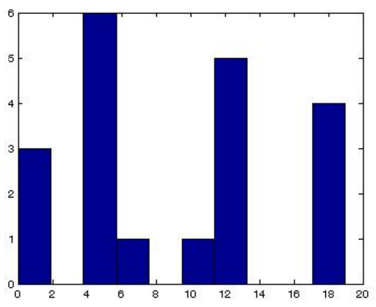

(c) rzwiker

Figure 54: The plots show the number of interactions per day. The x-axis denotes the number of the day since the start of the experiment. The y-axis shows the number of interactions on that day. The interactions have not been normalized by the amount of user presence.

User	interactions	average error	maximal error
afenkart	19	14.5	51.1
chicca	6	10.1	18.0
rzwiker	10	45.6	78.1

Table 10: The table shows average and maximal error in percent. The error was measured by test and training-set of half size each. The interpolation was done by Gaussian processes. The input dimensions are global irradiance and sunniness; blind height is the target dimension. The user inputs are interactions not assumed in glare avoidance and having an irradiance value of more than 500 arbitrary scale at time of interaction.

16 Conclusions

The main achievement of this work is to show that glare avoidance from direct light is a pattern of interaction with blind devices and to present a method based on hierarchical clustering that extracts the corresponding interactions. This method seems to work reliably, to the extent that the threshold used for cluster creation should be made adaptive. Another more biological result is that users sitting next to the window care most about diffuse irradiance not global irradiance because on days of equal irradiance the blinds are set to lower positions on overcast days. In contrast users having their desktop further away from the window seem not to care about diffuse irradiance likely because they see only a small portion of the sky.

Our assumption that all remaining points are motivated by controlling inside illuminance and trying to apply Gaussian processes to fit them was naive. It seems that luminance control is a pattern of interaction but only when outside illuminance exceeds some threshold. Because it does not explain all interactions not assumed as glare avoidance, there must be other reasons for interaction we are not aware of yet. When looking only at those interactions not assumed as glare avoidance, that happened when outside irradiance was higher than a given threshold, we calculated an average test-set error for predicting blind height of less than 20% for three out of five users(afenkart, giacomo, kiper) whose office is never occluded by outside obstacles. Prediction increases to four out of six, when including the unbiased experiment user. When including offices with times of occlusion, we can estimate four out of ten users with the same accuracy. The maximal error is less than 54%, which is still better than expected from pure guessing.

We believe this approach is worth to be continued. In summary it consists of isolating patterns of interaction, projecting the data into a subspace where interactions produced by one pattern can be isolated from other interactions by a simple method such as hierarchical clustering or even simpler linearly separated. Our believe that this is possible is based on the assumption that any interaction is driven by one main pattern, that serves a specific need of the user. This need in general depends on a physical entity, which should be different for any pair of patterns. So in most cases a subspace for easy separation of one type of interactions can be found by projecting the data into their most relevant physical subspace.

Probably the most sustainable technical step of this work is the further improvement of the ABI framework by making it fully dynamic and dividing the problem of adaptive building control into the subproblems: i) preprocessing sensor values ii) presence detection and iii) effector controlling. Each of them are dealt by an individual agent or a set of agents. Future work should be able to develop drop-in replacement for a subproblem by replacing the corresponding agent without having to touch other subproblems. Another major step we find is the integration of MATLAB into the framework by calling MATLAB generated C-functions out of Java. This leverages the implementation overhead for new control approaches and supports future developers with rapid prototyping.

16.1 Future Work

We are aware that the current work is unfinished. The way we estimate sunniness is an approximation that needs improvement. The same applies to access of outside sensors where access should be simplified by e. g. using a direct IP connection instead of routing updates through several field busses.

These two issues put our analysis on feet of clay, but even when improving these points prediction will likely improve only for those users where prediction is already quite good. Predictions with high error rates indicate that our model is too simple, so we assume that a more exhaustive search for interaction patterns is needed. Possible patterns we think of are window reflections of neighboring buildings at times of sunrise/sunset or a different user behavior in case the sun is present or absent on the building face due to a change in ambience. But maybe the simplest and also the most urgent, is to add detection of outside obstacles to the model which will in our case likely lower prediction errors for users on the west facade a lot.

As aforementionned our preprocessing of sunniness and irradiance is very weak and this drawback is always present when analyzing the data. Future work should revise and improve it either by better physical and astronomical models or better hardware sensors and better access to them. E. g. an irradiance sensor that tracks sun position for accurate measurement of direct irradiance will result in a more robust sunniness estimation. Another thing is revising the measurement of outside illuminance. The current assumption that irradiance and illuminance have a linear relationship is incorrect. The exact type of the sensors should be looked up to find a better way of estimating illuminance out of irradiance sensor values by using the physical structure of the sensors.

Another problem is that the blind motors and motor-controllers are not developed for the way we are using them. They are not designed as robot devices but as unreliable systems that are made reliable by users observing and correcting them. E. g. users interacting with wall-switches do not need to get the current blind position from the motor-controller, they watch the blind and they know it. Automatic controlling does not have this feedback available without additional sensors. Actually the current implementation of the motor-controller is not that far from what we need, because they already provide feedback about blind position. Unfortunately it turned out to be too unreliable supposable due to firmware bugs of the motor-controller. Another problem is the feedback from the tilt position, which is inherently unreliable due to the too simple mechanics of the blind itself. Future analysis should probably restrict the number of tilt position to fully transparent and fully closed and drop all intermediate positions. Although we risk that some users might be unsatisfied with this at least the feedback can be used for analysis then.

Part V
Appendix

A Gaussian Processes

You are encouraged to read the *Introduction to gaussian processes* by Mackay[29]. While skimming it over for first time it might look confusing but when reading it for a second time it becomes quite clear. Here I will just pick out a few key sentences and explain them with my own words. But first of all the definition of the variables.

$\mathbf{t}_N$	target values in the regression task
$\mathbf{X}_N$	corresponding tuples of input values

Background

> Neal showed that the properties of a neural network with one hidden layer [...] converge to those of a Gaussian process as the number of hidden neurons tends to infinity if standard 'weight decay' priors are assumed.

Overview

> The idea of Gaussian process modelling is, without parameterizing $y(x)$, to place a prior $P(y(x))$ directly on the space of functions.

Usually when doing inference an underlying function is assumed, e. g. student-t distribution or choosing a neural network with a given number of hidden units. In neural networks usually several models are trained and the model chosen is the simplest model with a low error on the test-set. Hence the selection of the model incorporates our prior knowledge of the shape of the function.
Gaussian processes are different as no underlying function is chosen, by working directly with the data.

> From the point of view of prediction at least, there are two objects of interest. The first is the conditional distribution $P(t_{N+1}|\mathbf{t}_N, \mathbf{X}_{N+1})$ [...]. The other object of interest, should we wish to compare one model with others, is the joint probability of all the observed data given the model, $P(t_n|\mathbf{X}_N)$, [...] Neither of these quantities makes any reference to the representation of the unknown function y(x). So at the end of the day, our choice of representation is irrelevant.

The defining property of a Gaussian process is

> The probability distribution of a function y($\mathbf{x}$) is a Gaussian process if for any finite selection of points $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ..., $\mathbf{x}^{(N)}$, the marginal density $P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)})$ is a Gaussian.

To make this explicit this is not assumed to be a single gaussian but a multivariate of Gaussian. The target values are assumed to differ by additive Gaussian noise of variance σ_ν^2 from the corresponding function value y_n. Hence $\mathbf{t}$ also has Gaussian prior distribution.

Okay that's nice but how does it work now? As the target values are assumed to be gaussian distributed, everything is controlled by the covariance function.

$$\mathbf{C} = \mathbf{Q} + \sigma_\nu^2 \mathbf{I}$$

Whereas the second term reflects the noise on top of the Gaussian process y(x). The matrix **Q** is the covariance matrix of y(x). This covariance matrix **Q** gets built by the covariance function ...

> ... the covariance is a function $C(x, x')$ which expresses the expected covariance between the value of the function y at the points **x** and **x'**.

So it's defining the stiffness between the output values. The covariance function used for irradiance and blind regression is shown by the following equation ...

$$C(\mathbf{x}, \mathbf{x}', \boldsymbol{\Theta}) = \Theta_1 \times \exp(-\frac{1}{2} \times \sum_{i=1..I} \frac{(x_i - x_i')^2}{(r_i)^2}) + \Theta_2$$

Whereas $\Theta = (\Theta_1, \Theta_2, \{r_i\})$ is called *hyperparameters.* The values r_i are called the length-scale. A very large length scale means that y is expected to be essentially a constant function of that input. Θ_1 defines the vertical scale of variations of a typical function. The Θ_2 hyperparameter allows the whole function to be offset away from zero. There is an additional hyperparameter Θ_3 that defines the amount of white noise.

> The only constraint on our choice of covariance function is that it must generate a non-negative-definite covariance matrix for any set of points $\{x_n\}_{n=1}^N$

This function chosen is said to be a stationary covariance function as it is translation invariant. There are other covariance functions and combinations thereof. Other aspects could be built-in such as a linear trend or periodicity. But this is not given in our case as we cannot assume more from irradiance or blind values except the result to be smooth.

Evaluation

$$P(t_{N+1}|\mathbf{t}_N) \propto exp\left[-\frac{1}{2}\left[\begin{array}{cc} \mathbf{t}_N & t_{N+1} \end{array}\right] \mathbf{C}_{N+1}^{-1} \left[\begin{array}{c} \mathbf{t}_N \\ t_{N+1} \end{array}\right]\right]$$

> Thus Gaussian Processes allow one to effectively implement a model with a number of basis functions H much larger than the number of data points N, with the computational requirements being only of order N^3.

Training The hyperparameters introduced by the covariance function are undetermined. We want to train them from the data similar to the weights of a neural network. As the covariance function is required to produce a non-negative matrix and due to the noise term the resulting covariance matrix is positive definite. This allows us to use gradient-based iterative methods such as *conjugate gradients*[30] for finding the most likely parameters. The problem is that the space of hyperparameters is non-linear and these methods find different solutions depending on the starting conditions. This problem was counteracted by performing 10 training runs with different starting conditions. Finally the hyperparameters with highest likelyhood given the data $P(\Theta|\mathbf{X}_N, \mathbf{t}_N)$ were chosen.

B MATLAB to Java

One might wonder why dedicating a chapter to this when there is perfect Java integration into MATLAB[25]. The problem is that we needed to go the other way; from Java to MATLAB, i. e. we wanted to use MATLAB functions from a Java program.

- There is a way to call MATLAB from Java via defining callbacks [23] but this is merely for GUI design. Specifically it means that the Java functions where called out of a MATLAB process in the first place, the callbacks are just used to pass back data to the calling function. To use it for our purpose there is still a need of RMI or Sockets to bridge to the ABLE multi-agent platform.

- Compiling matlab functions into C code then wrapping them by a Java Native Interface. By following the tutorial[25] it seemed like a pushbutton exercise.

- The next possibility would be to reimplement the algorithms in Java. There are some libraries, such as the colt library [24] that enable high-level matrix manipulation.

- MATLAB C-engine interface. From C a MATLAB is started as a second process by a call to the function *engOpen*[26]. Computation is performed, by sending strings with matlab commands to the engine. Parameters and results have to be wrapped/unwrapped into matrices. Finally this has to be done from C so there is the need of JNI.

For a final release of the controlling algorithms a pure Java implementation is first choice, both in terms of portability and performance. The first possibility with callback functions was discarded as the technology seemed not to be suitable for our purposes.

The second method, JNI/C was the method chosen. From previous experiences with JNI we thought this part of the problem could be solved. The compilation of MATLAB functions into standalone applications is described well by some tutorials and information on building libraries is available too. The attraction of this approach was the seemless integration of matlab into Java, as the matlab functions do not have to be changed. The generated C code depends on the Matlab Runtime Environment but does not require end-users to buy a license[27] Currently a flag indicates whether the purpose of the method run is simulation or deployment. This eliminated the need of parallel versions for development and deployment. Bug fixes discovered during simulation are automatically applied to the deployment system. Additionally this could be extremely beneficial for rapid prototyping of future algorithms.

Unfortunately a number of problems occurred while implementing the *glue* code, wrapping and unwrapping the arguments and results by matrices. So I

[25]MATLAB = matrix laboratory, which should emphasize that everything is a matrix(not an object).

[26]Could also be running on another computer

[27]For commercial distribution the current education license of the compiler is probably insufficient.

feel qualified to say, that it isn't actually that easy, so maybe Java callback functions might have been the solution after all.

But here is a list of problems one might encounter, when trying it this way.

- **The instruction at "0x0998a379" referenced memory at "0x0000034"** Probably you did not initialize static variables contained in the C-translated matlab functions. This problems has lots of other symptoms as well. But in general the translated code runs as a stand-alone application, but crashes when compiled as a library. Version 3.0 of the MATLAB C compiler(mcc) creates init/terminate functions which have common prefixes. e. g. *InitializeModule_gp_err*, *TerminateModule_gp_err* It has been found to be good practise to group matlab functions with similar functionality into separate libraries, then writing a C-file containing a pair of initializer/finalizer that get called on load unload of the library, see Wheeler[27]. Below a sample initializer functions.

```
void __attribute__
((constructor))gp2_init(void)
InitializeModule_gp_err();
InitializeModule_real_conjgrad();           // [1ex]
InitializeModule_real_gp2();
InitializeModule_real_gp2pred();
InitializeModule_real_gp_fit();
```

- **libmwlapack: load error: /usr/local/matlab/bin/glnx86/lapack.so: undefined symbol: xerbla_** Following was taken from a newsgroup [26].

  > . . . this module is linked to the appropriate shared library on the system containing the blas implementation. My understanding is that not all systems are smart enough to automatically dynamically link the blas library to the process when the module requiring is loaded . . .

  The problem has not been further investigated as a workaround was found by setting *LD_PRELOAD* variable. E. g. `export LD_PRELOAD` *your library* before starting the process. You also might want to set the `LD_LIBRARY_PATH` environment variable to point to the directory with your libraries.

- **Memory leaks** When you generate matrices, e. g. when wrapping input arguments for passing them to the corresponding matlab functions, you must make sure to also remove the matrix when it's no more needed. The most comfortable way is to use Matlab's automatic memory management, as described in the C Math Library manual [28].

- **Thread-safety** Most C functions such as *printf* are usually[28] not thread safe[29]. When calling such functions from a threaded environment, unpredictable behavior can occur up to crashes of the entire Java Virtual Machine(JVM). A simple workaround is to use a platform wide lock, to synchronize all calls from one JVM to native functions.

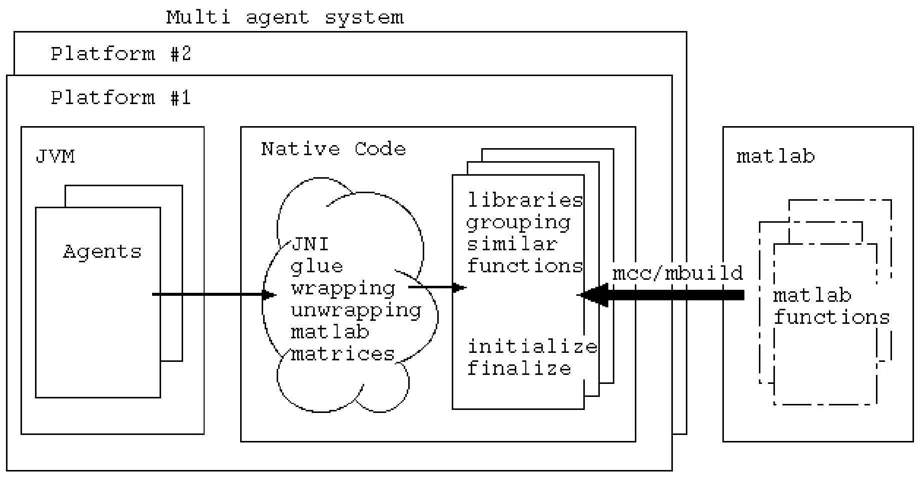


Figure 55: The window splits into two parts on the right the matlab world used for simulation and debugging and prototyping, the left side the multi-agent system consisting of multiple platform. From the matlab files shared libraries are built by the help of the *mcc/mbuild* compiler. The glue code abstracts the accessing of java-fields if necessary and wrapping of arguments and return values.

Figure 56: Illustration of how a matrix of e. g. user inputs or sensory data is made accessible from different native function calls. A java long field is casted to a matlab pointer.

See figure B for an overview. On the right side the matlab world used for simulation, debugging and prototyping On the left side the multi-agent system consists of multiple platforms. Each matlab file required by the control algorithm is translated into an identical C implementation by the help of the *mcc* compiler. Several files are then bundled into a library together with a pair of library constructor/destructor functions. The compilation to native code is done by the script *mbuild* which is wrapper for native compiler, e. g. gcc. The purpose of the glue code is reading/writing to fields in the calling Java class, wrapping/unwrapping of arguments and results and calling the according matlab functions in the previously built libraries.

See figure 56, describes how static MATLAB matrices are cached between different function calls. A java long field is abused by casting it as a matlab matrix pointer. This matrix can then be passed to the C-translated MATLAB functions.

[28]The C-library can be configured at compile time as being thread-safe

[29]e. g. they use global buffers not protected from access from different threads

C Description of weather station

This section will describe the irradiance sensors. Figure 57 shows all irradiance and illuminance sensors mounted in the weather station. The upper row of sensors, with transparent bulbs are illuminance sensors. The lower row of lengthy tubes are irradiance sensors. Near the mast we can see one illuminance sensor that is mounted upright horizontally.

Although there are vertical illuminance sensors we can only access the irradiance sensors. The reason why the illuminance sensors are not accessable from the LON bus is a technical. It was explained to us that, that it is complicated to route all sensor values to the building. First because of the restricted number of channels in the routing gateway and second it is very labour intensive to set up. Hence the upright illuminance sensor was installed which measures horizontal illuminance explicitly for building Y55.

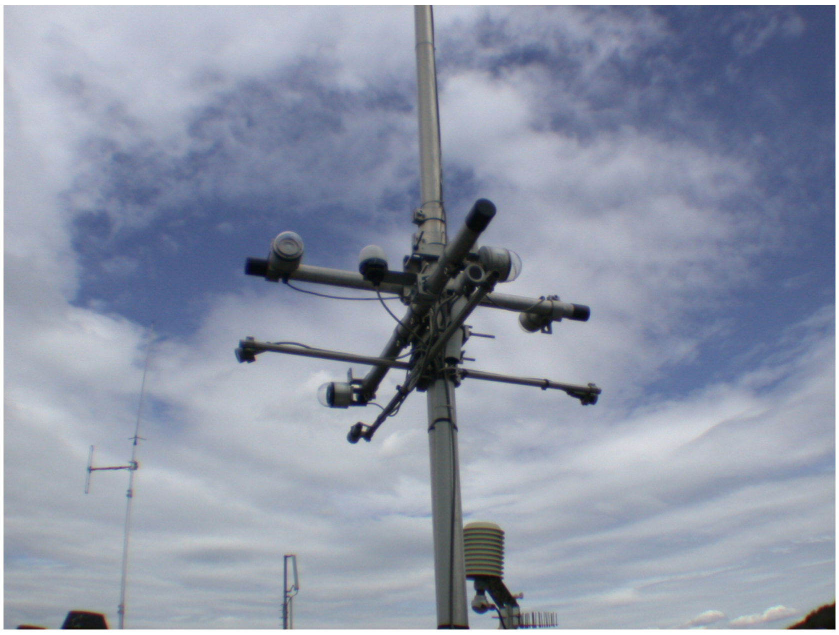


Figure 57: Antenna carrying vertical irradiance and illuminance sensors.

The weather station is located on top of the roof of a building Y13 which which is not far from the INI(Y55). The orientation of both buildings is identical, in that north is tilted by about 20 °from true north. See also Figure 58.

In Figure 59 you can see that for all irradiance sensors the view is almost unobstructed. Prospects of east and north direction have little reductions by either a nearby hill or the building itself, but this should not be a problem in case the sun is not close to the horizon.

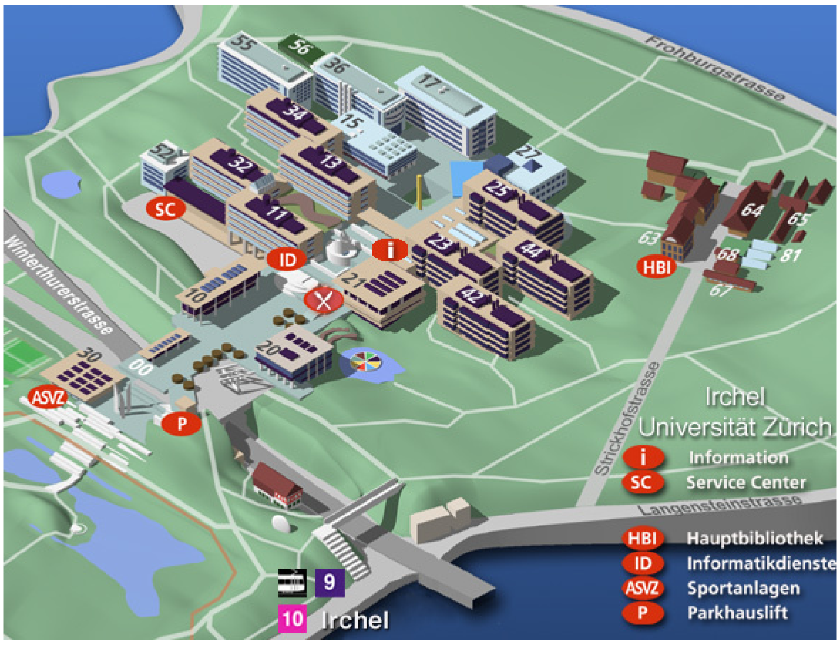
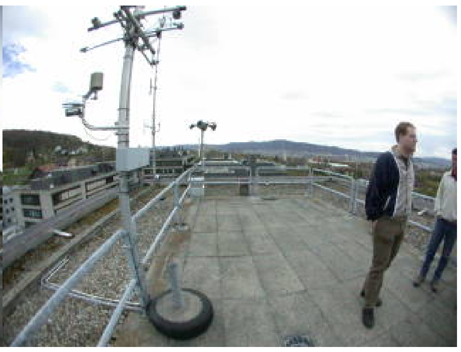
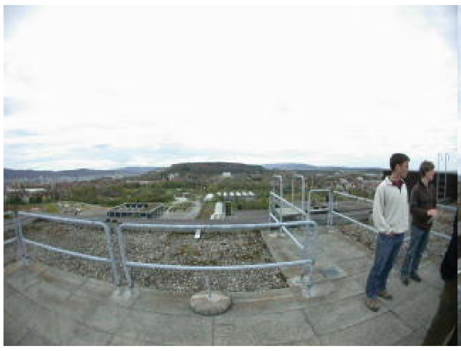


Figure 58: Orientation plan of the Irchel Campus. The INI is in building Y55, the weather station is on top of Y13, on the side of the center place.

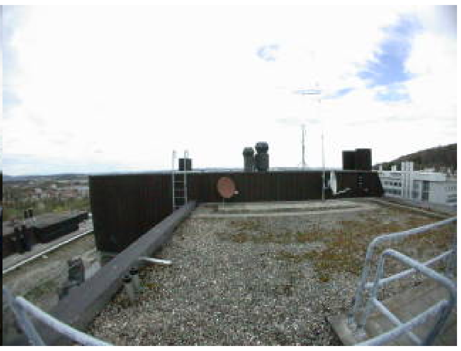
(a) east



(b) south



(c) west



(d) north

Figure 59: Prospect for each orientation of the irradiance sensors.

References

[1] Antoine Guillemin, *Using genetic algorithms to take into account user wishes in an advanced building control system* http://lesowww.epfl.ch/doctorants/guillemin/Thesis_2778.pdf PhD. Thesis (2003)

[2] E. S. Lee, D. L. DiBartolomeo, S. E. Selkowitz *Thermal and daylighting performance of an automated venetian blind and lighting system in a full-scale private office* Energy and Buildings 29 (1998) 47-63

[3] E. Vine, E. Lee, R. Clear, D. DiBartolomeo, S. Selkowitz *Office worker response to an automated venetian blind and electric lighting system: a pilot study*, Energy and Buildings 28, (1998) 205-218

[4] Ueli Rutishauser, Josef Joller, and Rodney Douglas *Control and learning of ambience by an intelligent building* IEEE Transactions On Systems, Man and Cybernetics: A, 20004 in Press

[5] Jonas Trindler, Raphael Zwiker *Adaptive Building Intelligence; Parallel fuzzy controlling and learning architecture, based on a temporary and long-term memory.* Diploma Thesis (2004)

[6] Michell Foster, Tadj Oreszczyn *Occupant control of passive systems: the use of Venetian blinds* Building and Environment 36 (2001) 149-155

[7] Hani Hagras, Victor Callaghan, Martin Colley, Graham Clarke *A hierarchical fuzzy-genetic multi-agent architecture for intelligent buildings online learning, adaptation and control* Information Sciences150 (2003) 33-57

[8] D. Snoonian *Smart buildings* IEEE Spectrum, vol. 40, no. 8, pp. 18-23 2003

[9] Tobi Delbruck, Andreas Fenkart *PC-bus API* http://www.ini.unizh.ch/ãfenkart/abi/

[10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides *Design Patterns: Elements of Reusable Object-Oriented Software* Addison-Wesley, 1995

[11] GeoTools http://geotools.sourceforge.net/gt2docs/api/org/geotools/-science/astro/SunRelativePosition.html

[12] Chris Cornwall, Aaron Horiuchi and Chris Lehman *Solar position calculator* http://www.srrb.noaa.gov/highlights/sunrise/azel.html

[13] Vincent Roy, *Sun position given observers time & location* http://www.mathworks.nl/matlabcentral/fileexchange/loadFile.do?-objectId=4605&objectType=file

[14] Jean Meeus, *Astronomical Algorithms* (Willmann-Bell, 1991)

[15] Hans Walser, *Sphaerische Trigonometrie, Berechnungen* p. 9 section 1.9

[16] Richard C. Jordan, Benjamin Y. H. Liu, *Applications of Solar Energy for Heating and Cooling of Buildings*American Society of Heating, Refrigerating and Air-Conditioning Engineers, 1977 (p. V13-V15)

[17] Extract paper indicating relationship between vertical diffuse illuminance and inside illuminance from guillemin thesis ...

[18] James M. Palmer *Radiometry and photometry FAQ* Optical Sciences Center, University of Arizona, Tucson

[19] Volker Quaschning *The Sun as an Energy Resou(r)*[30] *ce* (Renewable Energy World 05/2003 pp. 90-93)

[20] High Technology Systems AG *LON-Appliaktionsbeschrieb fuer Praesenzmelder ECO-IR 180LON / 360LON* http://www.hts.ch/deutsch/-pdfs/HTS1103008601.pdf

[21] Daniel C. Kiper *Biological and Computational Vision, Lecture 2* http://www.ini.unizh.ch/ kiper/ss04/lecture02_dk.pdf(p. 19-20)

[22] Matlab Statistics Toolbox, *Hierarchical Clustering* http://www.mathworks.com/access/helpdesk/help/toolbox/stats/-multiv15.html#61427

[23] Peter Webb *Integrating Java Components into MATLAB* http://www.mathworks.nl/company/newsletters/news_notes/win02/-patterns.html

[24] Wolfgang Hoschek *The Colt Distribution: Open Source Libraries for High Performance Scientific and Technical Computing in Java* http://nicewww.cern.ch/ hoschek/colt/

[25] Matlab Helpdesk *MATLAB Compiler* http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/

[26] Chris Lee *Dynamic loading question* http://www.red-bean.com/guile/guile/old/2481.html

[27] David A. Wheeler *Program Library HOWTO* http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Program-Library-HOWTO.html

[28] User's Guide *MATLAB C Math Library*, Mathworks Inc., Version 2.1

[29] David J. C. Mackay *Introduction to Gaussian processes*

[30] Jonathan Richard Shewchuk *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain* Edition $1\frac{1}{4}$ (1994)

[31] Carl Edward Rasmussen *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression* PhD thesis, graduate department of Computer Science, University of Toronto. 1996

[30] presumably typo in the title