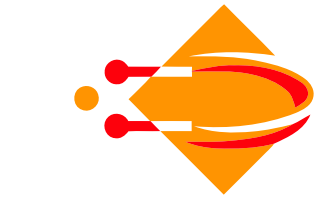
# Occupancy Analyzer

Global Software Development Project

**Christoffer Wilken Pagaard**
**Jacob Romme Rasmussen**
**Theresa Brandt von Fackh**
**Tomas Miseikis**



IT University
of Copenhagen

15. December 2013

**Abstract**

Write abstract text...

# Contents

# 1 Introduction

## 1.1 Context

Smart use of energy resources is an ongoing topic these days. The reduction of expenses is mostly the biggest driving factor for companies. But also the debate around climate change brings new legislation to reduce the waste of energy resources, whose production is damaging to the environment and future generations. The IT University of Copenhagen (ITU) has an interest in producing an occupancy model for commercial buildings, like the ITU building, to detect where energy ressources are needed and where they can be saved. Energy resources are needed for e.g. lighting and heat-regulating systems, which are relevant for occupants in a commercial building. With the detected occupancy data the ITU can predict occupancy and develop concepts for a smart use of energy resources in commercial buildings.

The Strathmore University in Kenya has also an interest in building up an occupancy model, but mainly for surveillance reasons. Surveillance can be used for several purposes like traffic monitoring, public safety and facilities surveillance. An IT-based surveillance system can automatically analyse the scene without the use of human ressources. By analysing the scene the detection of occupancy is a major part. Moreover a real-time prediction model on top of the occupancy data can be used to prevent criminal activity by triggering alarms or other surveillance systems.

Currently there is no existing infrastructure to build up an occupancy model in the Strathmore University or the ITU building. Both universities want a solution for an occupancy analyzer based on Raspberry Pis due to the minimal consumption of computational and monetary ressources. Furthermore, Starthmore University requests for an Android application, which represents a live-feed of the occupants in a monitored room.

A group of students from both universities have to collaborate to come up with a solution for an occupancy analyzer, which can satisfy the needs of both university interests. Ideally a product should be developed, which can be adapted to fit the needs of one or the other university. Furthermore, a collaboration project is mandatory for the student group from ITU, in which they have to face the challenges of global collaboration, navigate compromises and come up with a solution.

This report contains the product result, design of the product, details of the project work and the learning outcomes, which were achieved in the project with the globally distributed team from the perspective of the ITU students. The project team consists of international students located in Nairobi, Kenya (East African Time) and Copenhagen, Denmark (Central European Time).

## 1.2 Problem

The content of this project is to build an occupancy analyzer, which detects people in a room or corridor and predicts their movement. The occupancy analyzer has to be based on Raspberry Pis, which comes with computational restrictions, and webcams. A solution for the right architecture and programming languages has to be found, which can deal with those limited ressources.

Due to the usage of webcams, a visual detection of people has to be made. Analyzing images by detecting people - which are moving objects and not part of the room - is one major challenge to face. Visual conditions of the image can change as a result of daylight. For example dynamic lighting and moving shadows should not influence the detection of people. To best capture occupants and the requirement to represent the occupants on an Android application, the webcam positions becomes important. The differentiation between multiple people, as well as between people and the setting of the room, is important for the quality of the detection. Only if reliable data about occupancy exists, can the data be used to construct a reliable prediction model usable for future concepts and projects.

The question of how the collected data can be used, has to be considered. Building prediction models, which relies on historical data and real-time data, is another requirement, which the project dealed with. Decisions on what kind of prediction for a meaningful application - like the one mentioned in paragraph 2 in section 1.1 - and how the data will be stored and processed have to be made.

Besides the design and implementation of an occupnacy anaylzer, another task is the collaboration of students from two different located universities. Cultural differences, difference in time, spatial distance and locally related influences have to be overcome. Different perspectives have to be combined and compromises have to be made.

## 1.3 Related Work

The concept of an occupancy analyzer, which uses visual recognition of occupants, is already handled in several papers and projects.

NREL IPOS Project:

## 1.4 Approach

## 1.5 Report Structure

# 2 | Analysis

# 3 Design

## 3.1 System Overview

Figure 3.1 displays the different components in the system and how they relate and communicate with each other.

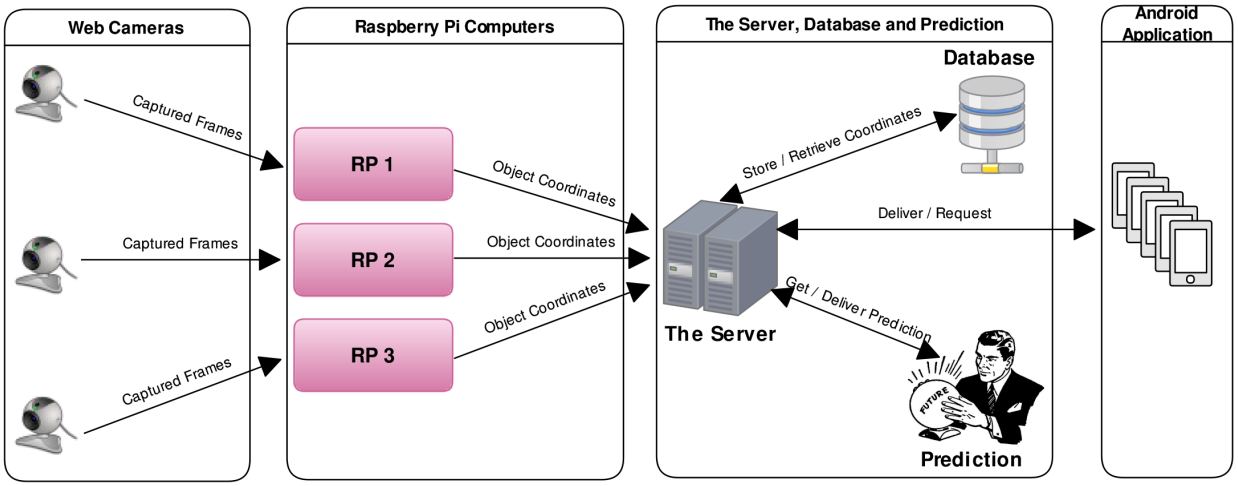



Figure 3.1: Occupancy Analyzer System Overview

As we can see, the system primarily consists of 4 components:

1. Web cameras,
2. Raspberry Pi computers,
3. The server,
4. and the Android application.

All of these components are discussed further on in the respective chapters.

## 3.2 Web Cameras

The purpose of the web cameras is simply to surveillance the area they have been placed in and forward the captured frames to Raspberry Pi computers for further processing, as shown in Figure 3.1. The cameras can be placed in a room, corridor, atrium or any other similar place in or outside of the building, where the services offered are required. Cameras can be placed either directly above the observed area (Figure 3.2) or in the corner of it, as illustrated in Figure 3.3. Naturally, a camera placed above the observed area would give better results, since this increases its field of view and makes it easier to correctly detect and distinguish between multiple people walking side by side. Furthermore, for the best results one must also take many different factors into account, such as the distance between the camera and monitored area, environmental conditions of the area the camera is placed in, lighting conditions.

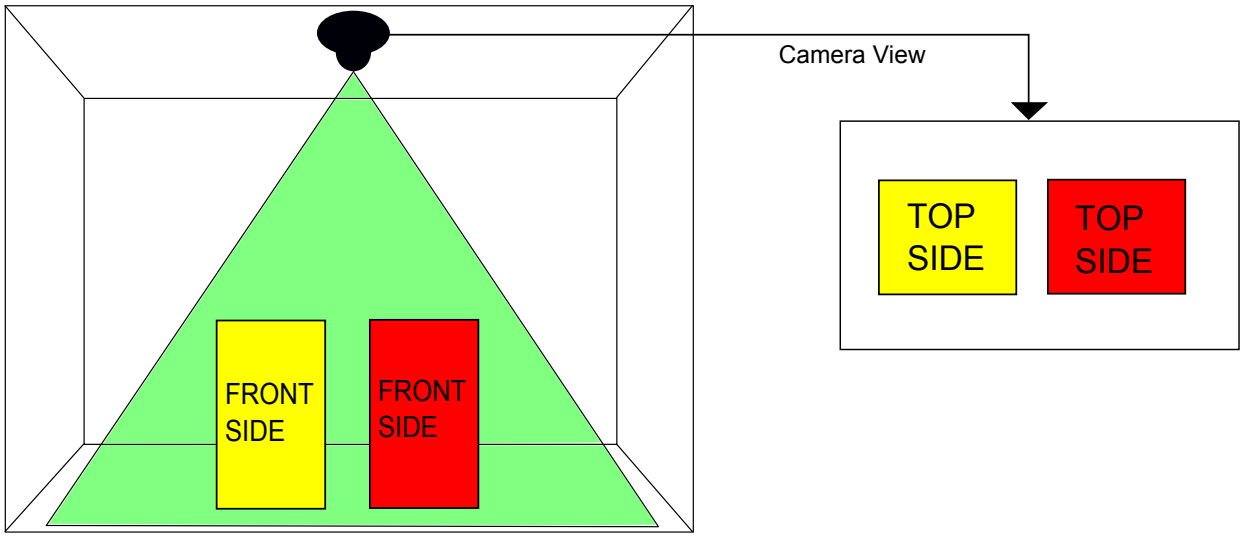



Figure 3.2: Camera top placement

## 3.3 Raspberry Pi Computers

The next component in the system architecture is Raspberry Pi computers. These computers have at least one web camera attached to them, and are responsible for processing the frames captured by these cameras. The main goal of processing these frames is to try to detect people in the monitored area and determine their position in that area. There are many different challenges in object detection, as well as various concepts and techniques that can be used to achieve this. We discuss these in the next sections.

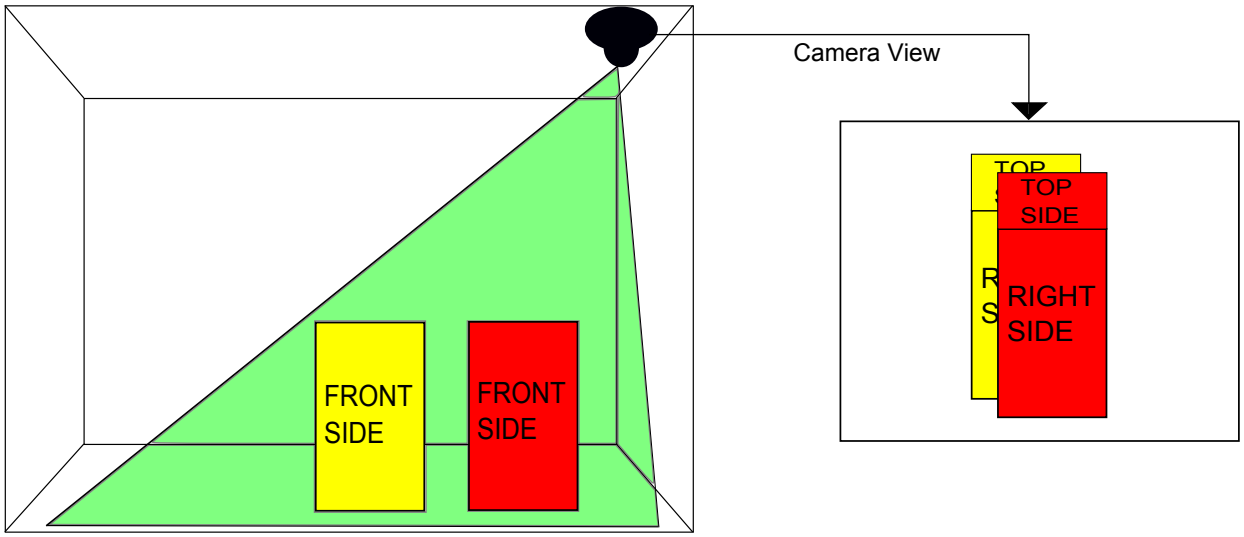



Figure 3.3: Camera corner placement

### 3.3.1 Object Extraction

To detect and extract objects, or in our case, people and their movement, we need to apply several motion detection techniques on the frames we are receiving from the web camera. First of all, to detect changes in some monitored area, we naturally need to have at minimum two images, which we must compare to see what changes occurred. We will try to look into two different approaches in doing this, a simple one, and one that is a bit more complicated and sophisticated, but much more adaptive and flexible.

#### 3.3.1.1 Background Subtraction Approach

A simple approach - called Background Subtraction - would be to have a static background image of observed area that was taken prior to the analysis and did not have any people in it. Then, one can simply detect changes and movement in the area by subtracting the static background image from every newly taken image of the monitored area [1]. The difference between the two images would then allow us to see if any changes happened, since after subtraction the resulting image would either be almost totally black (Figure 3.4) - meaning no one walked passed the observed area - or the image would have some resulting bright contours of detected differences (Figure 3.5).

After doing some research and experimenting with background subtraction technique, one will quickly discover that there are multiple weaknesses to it.

- First of all, if the initial background image is always static and never changes, this technique will fail in environments where lighting is dynamic. This is perfectly illus-

Figure 3.4: Background Subtraction with no background changes

Figure 3.5: Background Subtraction with background change

trated in Figure 3.6. We can see that the lighting is much darker in the second image, possibly because the light was turned off in the monitored room, thus after subtracting our static background image from this image, the resulting image is simply a lighter version of the two images, and not the intended black image. The reason why this happens is because the original brighter background image has a much higher intensity, thus the average value of it's pixels are higher than the pixel values of the newly taken darker image. Naturally, this is a big problem, because now even if a person moves through the monitored area (Figure 3.7), he or she will not be as easily extractable as in Figure 3.5.

Figure 3.6: Background Subtraction with background lighting change

- Another problem with background subtraction approach surfaces when a camera is placed inside an area which has objects that constantly change their original position

Figure 3.7: Background Subtraction with background lighting change and lego figure appearing

(chairs, tables, appliances, etc.) by being moved, even small changes in object's location will spoil the resulting image after subtraction. As we can see in Figure 3.8, the object is displayed twice in the resulting image, even thought we were not even interested in it, making it much harder to detect actual people moving in the area. From now on, the resulting image after subtraction will always be corrupt unless the object is placed back in its original position.

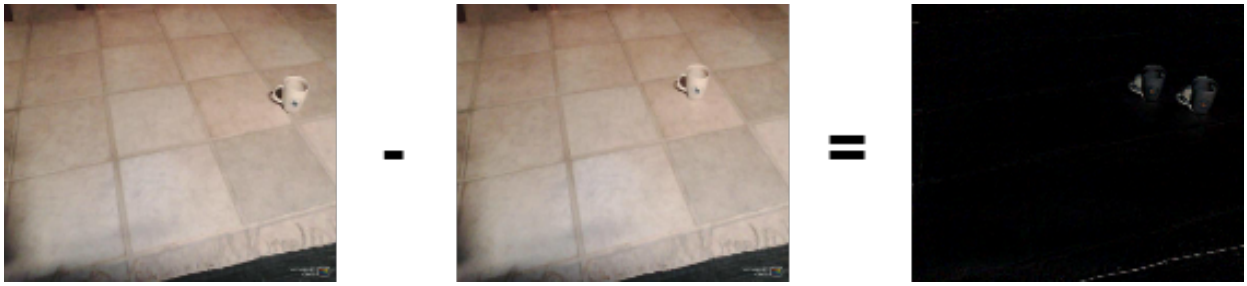


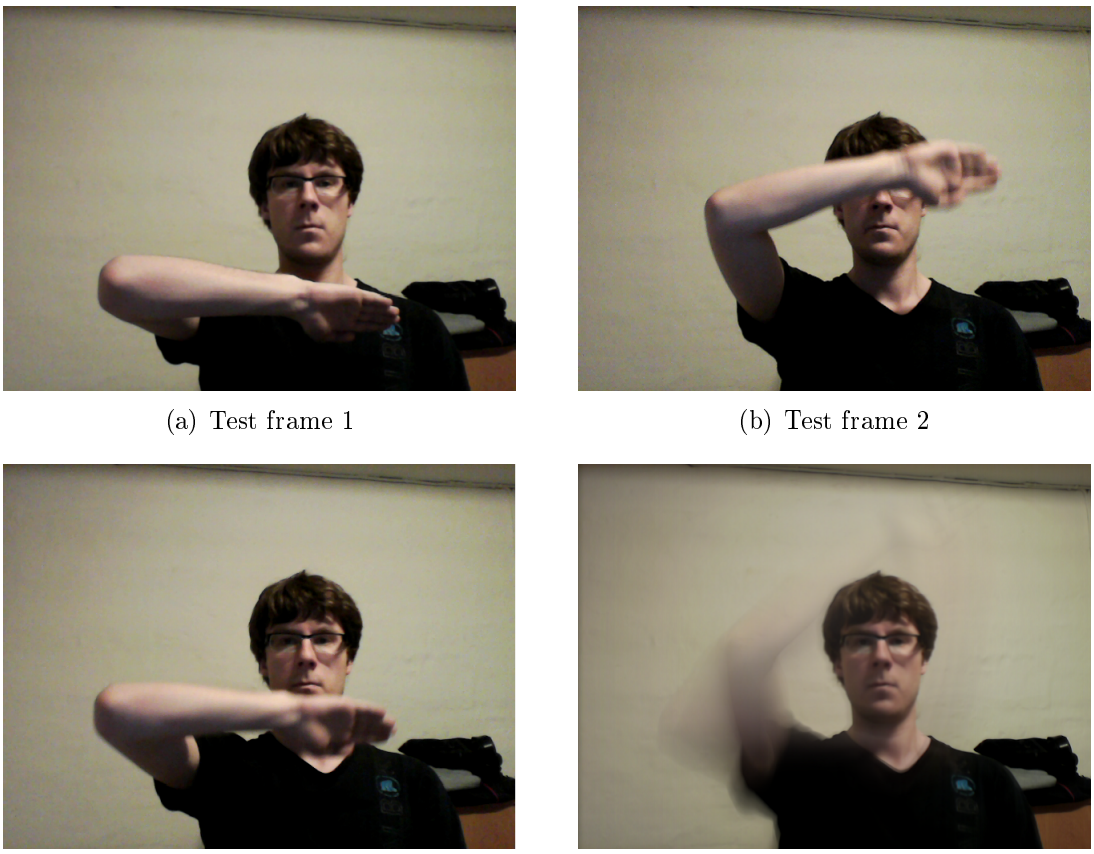
Figure 3.8: Background Subtraction with object changing it's location

In conclusion, we can see that simple background subtraction approach can work well in static environments, however it falls short in dynamic spaces. Naturally, these mentioned drawbacks of background subtraction approach need to be handled for object detection to work well, which creates additional challenges when implementing the system.

#### 3.3.1.2 Running Average Approach

A much better approach for movement detection is using a running average method. In this technique we do not need to rely on a static background image of the monitored area taken prior to analysis. Instead, we try to find a new "approximate" background image by interpreting any changes in the background as noise and blurring them out. This is accomplished by taking a train sequence of multiple previously captured frames and performing arithmetic

averaging on that sequence [2]. This exact approach is illustrated in Figure 3.9. As we can see, hand motion moving up and down (Figures 3.9(a), 3.9(b) and 3.9(c)) gets blurred out when applying running average method, thus producing an approximate background image (Figure 3.9(d)) that can be used for subtraction of the test frames from it. This method of object extraction works very well and has a lot of flexibility. It can adapt to environmental changes in the monitored area, thus eliminating most of the weaknesses that the background subtraction approach has. For these reasons, the running average approach was chosen for our design.



(a) Test frame 1

(b) Test frame 2

(c) Test frame 3

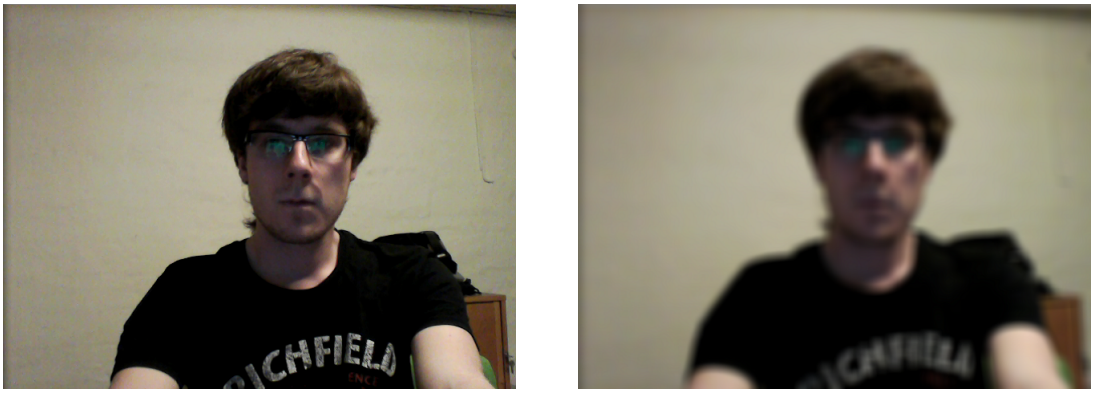
(d) Running Average frame

Figure 3.9: Example of Running Average technique

### 3.3.2 Object Detection

Now that we can extract objects using running average technique, we need to be able to actually find them in the resulting image we get after we perform subtraction. For this, we need to apply several key techniques in image processing.

- After we capture the initial frame of the monitored area, it will often contain noise and small details that we are not interested in. To deal with this, we must first apply blur or smoothing filter. In blurring technique we calculate weighted averages of areas of pixels in a source image by passing through it [3], which helps to reduce image noise and detail, as shown in Figure 3.10.



(a) Original frame (b) Frame after blur is applied

Figure 3.10: Blur example

- After we remove the initial noise, we can perform subtraction using running average approach (described in Section 3.3.1.2). After subtraction we will either get a nearly totally black image, meaning no motion occurred, or an image where some colors stand out, meaning some motion has occurred.

- In either case, for further processing of the taken frame, we can choose to convert it to a grayscale image. The reason for this is that the original RGB image we get has three channels, while grayscale image has only one [4], thus it is easier to work with. This procedure is illustrated in Figures 3.11(a), 3.11(b) and 3.11(c).

- For further processing of the image, we apply threshold technique, which converts the image to black and white and removes some more unwanted details and noise [5]. After threshold is applied we get the image shown in Figure 3.11(d).

- Moreover we want to expand the interesting parts of the image and contract smaller pieces, which can be considered noise and managed to slip through, even after we performed thresholding. To do so, we use two fundamental operations in morphological image processing, that is, dilation and erosion. Dilation allows us to expand the shapes contained in the image [6], whereas erosion simply shrinks shapes [7], so that bright regions surrounded by dark regions shrink in size, and dark regions surrounded by bright regions grow in size. When we apply dilation and erosion we get the image shown in Figure 3.11(e).

- Now, to project the detected area onto the original frame, we simply use bounding box technique, which gives us the coordinates of the rectangular border that fully covers the extracted white silhouette [8] that we got in 3.11(e). Then we use these coordinates to draw a simple rectangle, as well as mark it's middle position by a red circle, as illustrated in Figure 3.11(f).

In conclusion, by applying the steps discussed in this section, we can fairly accurately detect people and their movement in the monitored area.

### 3.3.3 Object Differentiation

There will naturally be cases when multiple people will walk through the monitored area and will be captured by the cameras, therefore we must have a way to differentiate between them. This task becomes rather difficult if people are very close to each other, since they will simply be interpreted as one person. However, as long as people are far enough from each other, the task becomes significantly easier. There are multiple ways of differentiating between objects.

One of them is to simply look at the objects histogram, which gives a graphical representation of the intensity distribution of pixels [9]. Since people are usually dressed in different color clothes, we can simply calculate a histogram for every detected person and remember it. Now, every time we receive a new frame and detect a person in it, we go through our previously saved histograms and check whether any of them are the same as our newly detect persons histogram. If there is such histogram, we interpret the person we detected in our new frame as the same person we detected in the last, otherwise, we conclude that we have not detected this person before, thus save his histogram for future reference. The biggest weakness of this approach is that a person's clothes might have different colors from the front and back. Therefore, his histogram calculated while he is facing the camera might be rather different than the histogram of when his back was towards the camera. For this reason, if the person decides to turn around midway, he might be interpreted as a new person - never seen before by the camera - when in fact his frontal or back histogram was already saved.

Another approach of differentiating between multiple people, and in fact the approach we used in our design, is to simply use the whole frame as a coordinate system and remember the last coordinate of every single detected person. Now, similarly to histogram approach, whenever we detect a new person in the frame, we simply look throughout previously saved coordinates, and if we find that this new person's coordinates is relatively close to some previously saved person's coordinates, we simply interpret him as the same person we detected a second or or few seconds ago, otherwise we see him as a new person. Naturally, we must regularly clear our previously saved coordinates, so that newly arrived person would not simply, by taking a similar path, be interpreted as a person who is no longer in the monitored area. This approach of differentiating between people is is illustrated in Figure

TBA.**TODO: add some figures with 2 or more people walking down the atrium and being differentiated.**

## 3.4 The Server

**TODO**

## 3.5 Prediction

In our scenario where occupants enter and exit the room constantly, typically within seconds or minutes of each other, it is somewhat difficult to find the immediate use of predicting an occupant's future actions. However, our partner team from Kenya expressed a need for such a system because they have had issues with thefts. Incorporating such a feature to their surveillance systems could warn a guard whenever suspicious activity might occur. A suspicious activity could involve a person moving too close to a given exit - possibly leading to a restricted area. In order to calculate a somewhat reliable prediction, we need to capture and store data about the behaviour of previous occupants. When the data set grows, the prediction gradually becomes smarter and generally more precise. This section covers two different approaches while focusing on our chosen solution.

### 3.5.1 Approaches

In order to satisfy the prediction requirement, various existing approaches have been considered. This section concerns the approaches deemed most appropriate for our situation, namely the *k-nearest neighbors algorithm* and an existing prediction model, the hidden Markov model. The former assigns a given object to a group depending on the $k$ nearest objects. To apply this to our scenario we imagined previously observed occupant positions being separated into groups dependant upon the chosen exit. Given any position we would analyse the $k$ neighboring previous positions of distinct occupants and predict the exit to be that of the majority. Figure 3.12 depicts an example where the center dot is the current position and the squares and triangles are previous occupant positions who chose separate exits. The circle with the solid line resembles a situation where $k=3$ which indicates that the occupant at the center is most likely to take the same exit as the occupants previously positioned at the triangle positions. The circle with the dashed line resembles a situation where $k=5$. This yields a different output where the occupant is most likely to choose the same exit as the occupants previously residing at the positions of the squares. The approach has proven to be inefficient on larger data sets[10], which can be averted by performing clever data reduction. This, however, deemed the approach out of the scope of this project.

Figure 3.13 depicts the different parameters a hidden Markov model works with. Each x is a state and each y is a possible observation, where a is a state transition probability and b is an output probability. When translating this to our scenario we need to define what exactly a state and an observation is and how to calculate the different probabilities. Since the ultimate goal is to predict what exit an occupant is going to take, a possible observation could be that an occupant exits a room at a given location. A state could depict a room area where a transition to another room area would have a state transition probability a. Given an occupant resides in any room area, b is the probability that the person will exit at a given location. After some investigation on the matter, there exists several hidden Markov model libraries, one of which will be discussed here, namely *Jahmm*. Essentially, random probability values are assigned to a and b. Gradually, these values are updated to reflect actual observed actions of occupants. Over time, the probabilities are modified and represent how the average occupant is moving given a current location, resulting in more accurate predictions. However, the investigated library leaves no opportunity to add custom rule sets to the prediction. (DISCUSS: This might be wrong) Imagine a scenario where an even number of occupants move from one exit to another in both directions, evenly split. At any location along the path the probabilities to each exit remain equal, since equally many previous occupants have taken each exit given the current location. Additionally, the predictions will be incredibly unreliable in the beginning until a large enough dataset has been collected. In an attempt to partially avoid these issues, or at least produce some more reliable results with lesser data sets, we wanted to establish rule sets and integrate those into the calculations. Thus, we chose to design our own prediction model, heavily inspired by the hidden Markov model, since the original interpretation of states and probabilities as depicted in figure 3.13 is maintained. The custom model is explained in detail in section 3.5.3.

### 3.5.2 Gathering statistical data

In addition to storing data about each occupant we also store data about how occupied each section of the recorded image is. The process is simple; we split the image of a room into sections, referred to as cells, and each time an occupant enters a cell the stored activity for the given cell is increased. Each room has a different and independent set of cells. This data tells us which sections of the room are most occupied. We use the activity data to perform predictions about an occupant's future actions using a custom prediction model.

### 3.5.3 Custom prediction model based on HMM

We have chosen to build our own prediction model heavily inspired by the hidden Markov model. Each cell in the image is a state where the state transition probability is the likelihood of an occupant moving to an adjacent cell and the output probability is the likelihood of an

occupant going to an exit given any current cell. Unlike a regular hidden Markov model, we do not store probability values individually for each state, but rather do the necessary calculations each time a prediction is requested by using the stored activity values of each cell. Additionally, our custom model allows us to take several custom factors into account during calculations. These factors work as a rule set for likely or unlikely occupant actions:

- An occupant entering a room from a given exit is less likely to exit the room at the given exit.
- An occupant is more likely to continue moving in his general direction and least likely to return to his previously visited cell.
- An occupant is more likely to move to the adjacent cell with the highest amount of previous activity.
- The likelihood of an occupant exiting at a given exit (unless it also serves as the occupant's entrance) is inversely proportional to the direct distance to the given exit, producing a magnet-like effect.

These factors have an influence on a final value of a cell or exit, which is used when calculating each probability. The sum of the probabilities of an occupant moving to each individual exit given a current cell is 100. The cell with the highest probability will be the predicted cell.

Figure 3.14 shows the custom model translated to a scenario. Each cell contains an identification number, an activity count and a value denoting whether or not the cell is an exit cell. The occupant entered the room at exit $y_3$ and the current state is $x_{14}$. $a_{14-13}$ denotes the state transition probability of the occupant moving from state $x_{14}$ to state $x_{13}$. $b_{14-1}$ denotes the output probability of the occupant ultimately choosing exit $y_1$. Taking our custom rule set into account, the occupant is least likely to exit at $y_3$ and most likely to continue to $x_{13}$. Moving to $x_9$ would increase the probability of the occupant exiting at $y_1$ ($b_{9-1}$) significantly.

## 3.6 Android Application

**TODO**

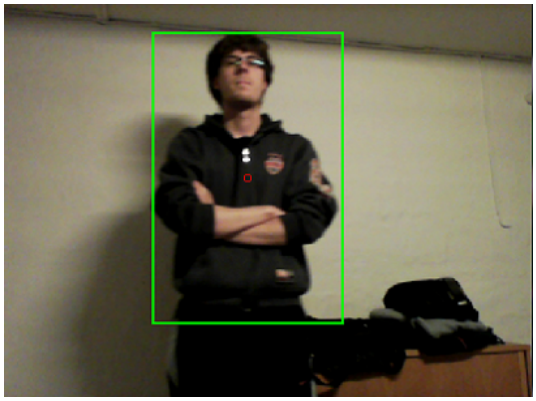
(a) Original frame

(b) After subtraction

(c) After grayscale filter is applied

(d) After threshold is applied

(e) After dilation and erosion



(f) After bounding box is drawn

Figure 3.11: Object detection example

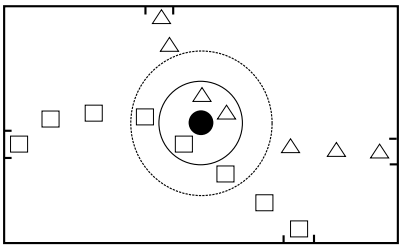


Figure 3.12: An illustration of the *k-nearest neighbors algorithm.*

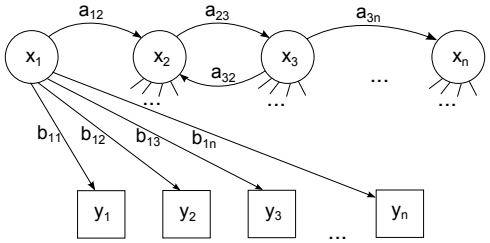



Figure 3.13: An illustration of the hidden Markov model.

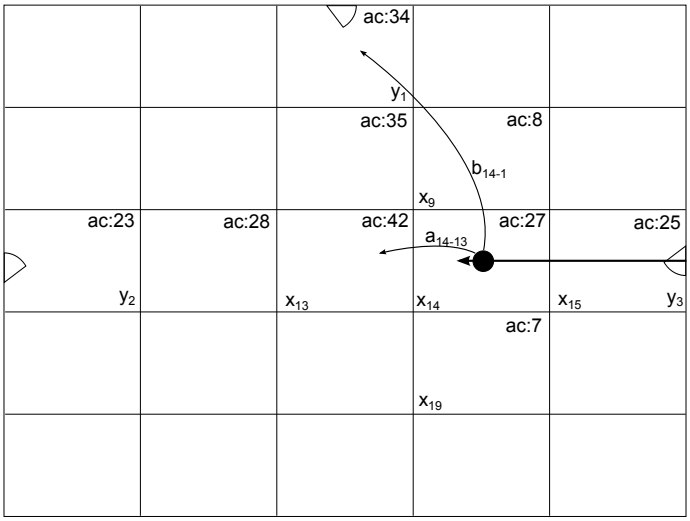



Figure 3.14: Applying the custom model to a scenario.

# 4 Implementation

## 4.1 Image Processing

### 4.1.1 Library

As non of the group members had any previous experience with image processing, it was quite unrealistic to try to implement all of the image processing techniques needed for this project, and at the same time make them optimized enough, so it could work well and fast on Raspberry Pi computers with limited resources. For this reason the group decided to use an existing image processing library to ease the implementation process. After some research and experiments, we decided go with one of the most popular and well documented libraries called OpenCV [1]. OpenCV is an open source computer vision and machine learning software library that has more than 2500 optimized algorithms for image processing. Furthermore, it provides interfaces to multiple popular programming languages, including C++, C, Python and Java.

### 4.1.2 Programming Language

For determining which programming language to use for image processing on Raspberry Pi computers, skill and preference document was created, where both, ITU and Strathmore University, teams indicated their skill and preference for various programming languages. The two languages that stood out the most were Java and Python, thus to choose one of them, we decided benchmark [2] them in order to compare their performance. The results are presented in the next section.

[1] OpenCV homepage: `http://opencv.org/`.

[2] Code used for benchmarks: Java - `http://itu.dk/people/tmis/javatest/`, Python - `http://itu.dk/people/tmis/pytest/`

#### 4.1.2.1 Performance Comparison Between Java and Python

Since the project dealt with a real-time vision application that had to process large amount of frames, we were interested in how fast Java and Python can perform different image processing algorithms. Benchmarks were performed on a laptop [3] and a Raspberry Pi computer [4] to give some perspective on how much faster a modern laptop is compared to a Raspberry Pi computer.

- At first we tried to benchmark how fast can Java and Python perform a simple matrix multiplication of two $300 \times 300$ size matrices. The results are illustrated in Figure 4.1. We can clearly see that Java was way faster than Python in this benchmark. It took Java less than half of a second to perform the multiplication of two $300 \times 300$ size matrices on a laptop, whereas it took more than 3 seconds to do the same in Python (Figure 4.1(a)). Multiplication on a Raspberry Pi computer (Figure 4.1(b)) was naturally much slower than on a laptop. Java was again much faster than it's counterpart by dealing with the task in less than 23 seconds, whereas Python was very close to hitting 2 minute mark to accomplish the same task.

  In conclusion, as it was expected Java convincingly won the first benchmark.

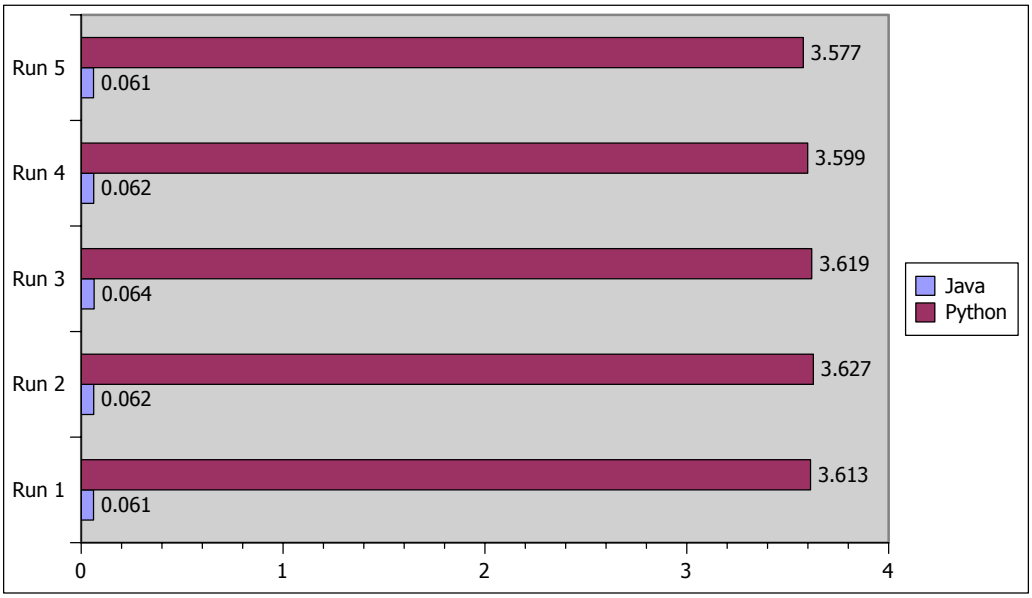
- The second benchmark involved testing how fast Java and Python can perform different image processing algorithms. For this benchmark we used the OpenCV library that we introduced earlier. OpenCV has Java and Python interfaces and all the computations are performed on the native level [5], hence we have some overhead that is equal to a cost of one or several API calls. Consequently, the purpose of this benchmark was to see, which language - Java or Python - has less overhead and can perform API calls faster.

  As we can see in Figure 4.2, the difference between Java and Python in this benchmark was rather small on both, laptop 4.2(a) and Raspberry Pi computer 4.2(b). This was rather surprising at first, since Java had a big edge in the first benchmark. One can speculate that Python's interface to OpenCV library has a better optimization than Java's, thus the outcome. In general, the difference was only in terms of a few milliseconds, however, it was a good enough reason for us to choose Python's interface to OpenCV library for image processing part in this project.

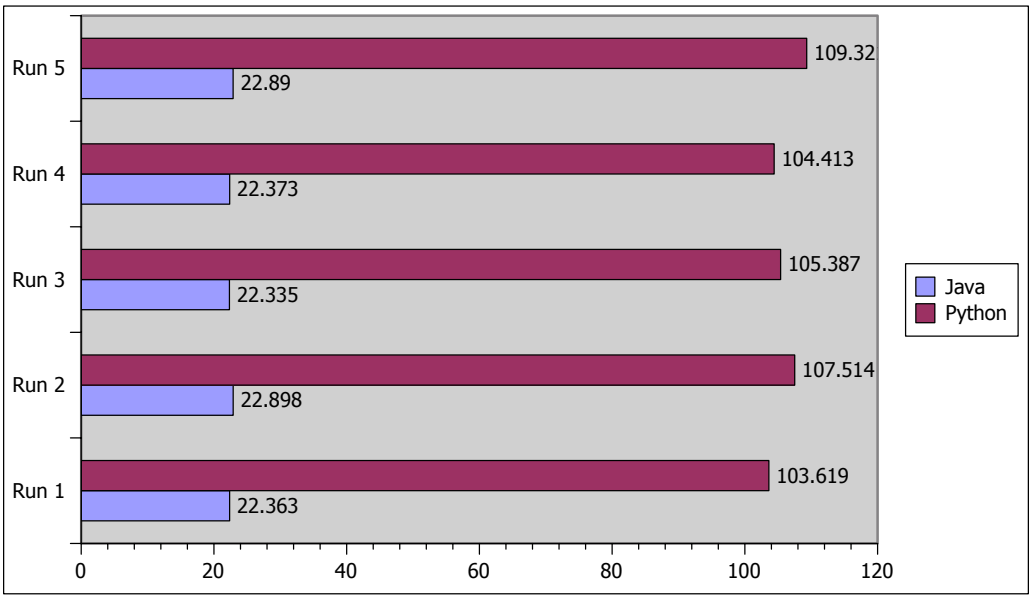
[3] Toshiba Satellite L855 Laptop (Intel Core i7-3630QM 2.40 GHZ, 4GB DDR3 1600MHz, Radeon HD7670M 2GB, Windows 7 OS).

[4] Raspberry Pi (ARM1176JZF-S 700MHz, 512 MB memory, Broadcom VideoCore IV graphics, Linux Raspbian OS).

[5] In C++.

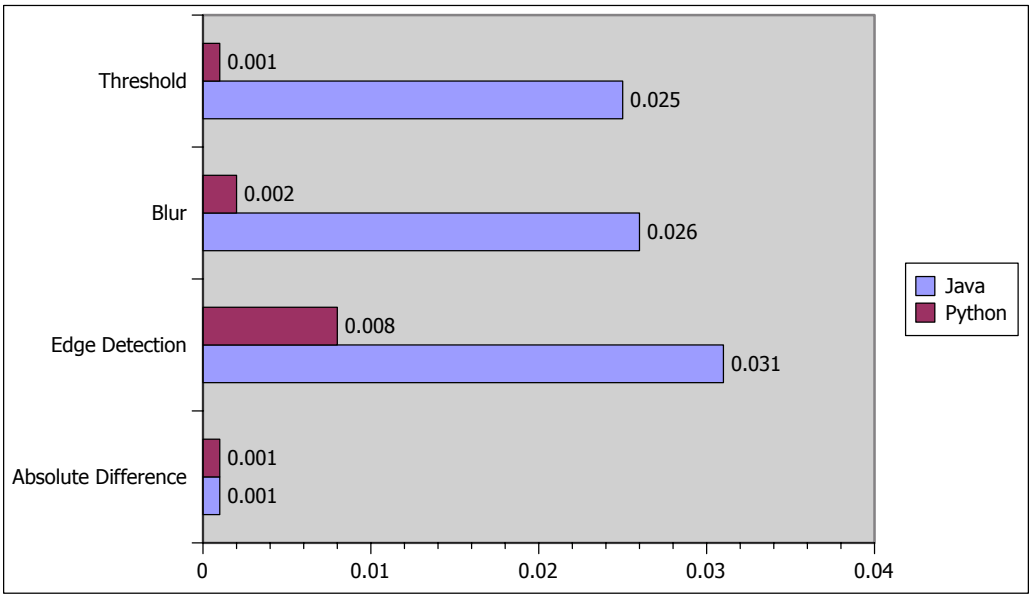



(a) Time (in seconds) needed to multiply two $300 \times 300$ size matrices in Java and Python on a laptop

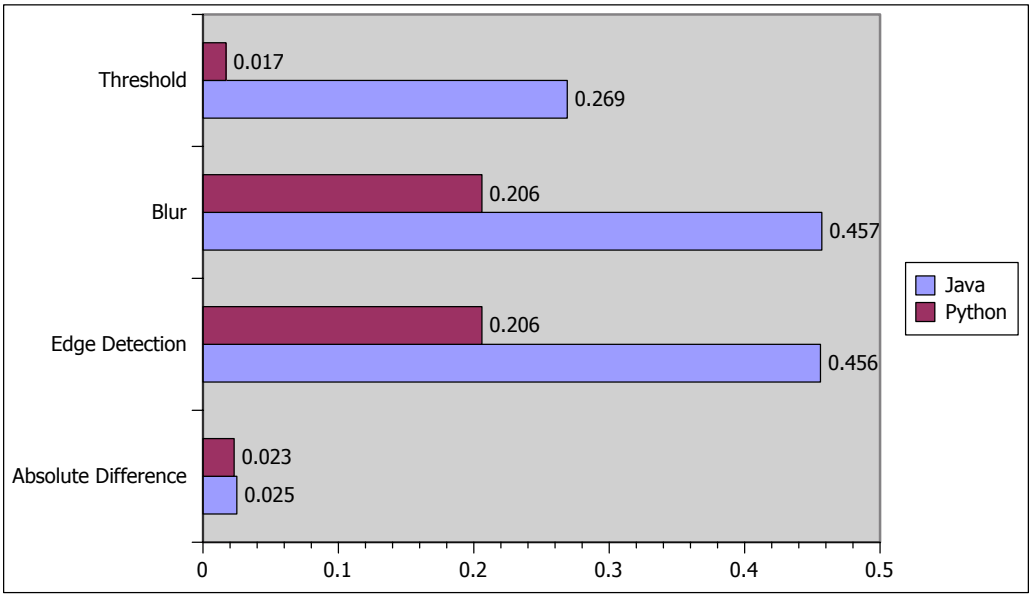



(b) Time (in seconds) needed to multiply two $300 \times 300$ size matrices in Java and Python on a Raspberry Pi computer

Figure 4.1: Benchmark of matrix multiplication of two $300 \times 300$ size matrices




(a) Time (in seconds) needed to process images in Java and Python on a laptop




(b) Time (in seconds) needed to process images in Java and Python on a Raspberry Pi computer

Figure 4.2: Benchmark of image processing

# 5 Evaluation

## 5.1 Verification

## 5.2 Benchmarks

# 6 Collaboration

This chapter deals with the collaboration and process in the project. We introduce the project team, the initial situation and present the chosen project management concept. The procedures and tools, which were used in the project, are declared and evaluated. We also list the difficulties, which occurred during the project and within the global collaboration. Furthermore we reflect the process and collaboration and explain our learning outcomes.

## 6.1 Introduction of the Project team

The project team consists of two groups of students. One group is from the Strathmore University located in Nairobi (Kenya). The other group is from the IT University (ITU) in Copenhagen (Denmark). The project team agreed on to name the two groups "Team Kenya" for the student group from Strathmore University and "Team ITU" for the student group from ITU. This helped to address each group in meeting reports, emails and conversations.

In the beginning Team Kenya consists of three members, which are all in the Masters programme "Telecommunication and Innovation" of the Strathmore University. The members are all from Kenya. The official languages in Kenya are English and Swahili. The members of Team Kenya met each other the first time on the 1.10.2013 (–> Global Meeting Report 1.10.2013). One member had to leave the project in the last third of the project due to workload of other projects and obligations (–> Global Meeting Report 26.11.2013).

Team ITU started with four members, which are all in the Masters programme "Software Development and Technology (Software Engineering)" of the ITU. The members are from three different countries: Lithuania, Germany and Denmark. Although there are minor differences between the nationalities, which could have an influence on the team work, we will not go into this, because it is out of scope for this report. The communication language within Team ITU is English, which is not the mother tongue of any of the members. The members of Team ITU met each other the first time on the 27.8.2013. One member left the project after two weeks, because he changed to another project and project team. Each member of the project team created a member profile to introduce themselves, which are attached in the –> Appendix xx.

## 6.2 Initial situation

The students of Team ITU have to complete the project under the course "Global Software Development Project", which is mandatory for their masters programme. The requirements and deadlines for the project are given in the course base from ITU (–> Link to the course base) and by the advisor of the project. The course is rated with 15 ECTS points, which corresponds to approximately 20 hours per week per student. The students of Team ITU have to hand in this report as a mandatory requirement.

There are no mandatory requirements or deadlines, which the Kenyan students have to achieve, except that they have to create a documentation for their advisor to prove the progress of the project. Team Kenya agreed on to go with the deadlines from Team ITU (–> Global Meeting Report 24.11.2013).

## 6.3 Projectmanagement Method

- Classic approach
- SCRUM method
- Our method (Structure)
- Weekly meetings internal, global, with advisor
- Time recording
- Splitting up assignmnts
- Plans

## 6.4 Collaboration tools

- Communication
- File sharing tools
- Time recording

## 6.5 Project Team and Organisation

- Project team members
- Skills
- Preferences
- Roles
- Splitting of the technical parts

## 6.6 Timeline

- Overview of the phases, milestones, deadlines, other exams/hand-ins
- Ressource planning
- Time-Recording (Toggl)

## 6.7 Collaboration Issues

- Illusion of the project work and project team
- Failure to comply with the assignments
- Communication
- Lack of skills
- Other exams/hand-ins
- Differing requirements
- Attendence of meetings
- Equipment

## 6.8 Hypothetical Scenarios

- Assignments
- Communication
- Requirements
- Organisation by the universities (Requirments, clarification, )

# 7 | Discussion

# 8 Conclusion

# 9 References


[1] Massimo Piccardi. 2004. *Background subtraction techniques: a review*, p. 1.

[2] Mohamad Hoseyn Sigari, Naser Mozayani and Hamid Reza Pourreza. February 2008. *Fuzzy Running Average and Fuzzy Background Subtraction: Concepts and Application*, p. 1.

[3] Herman Cheng, Zhicong Huang and Mark Kumimoto. Spring 2006. *Final Project Report - Image Processing Techniques*, pp. 2-4.

[4] Matt Siber. Fall 2005. *Resolution and File Size*, p. 1.

[5] Brian Lovell *Thresholding Techniques*. Spring 2003.

[6] R. Fisher, S. Perkins, A. Walker and E. Wolfart. 2003. *Dilation*.

[7] R. Fisher, S. Perkins, A. Walker and E. Wolfart. 2003. *Erosion*.

[8] Dr. Nicolas Pronost. *Introduction to Image Processing*, p. 9.

[9] Ahmed Elgammal. Spring 2008. *Histograms of Digital Images*.

[10] Bhatia, N., 2010. *Survey of Nearest Neighbor Techniques*, p. 303. `http://arxiv.org/ftp/arxiv/papers/1007/1007.0085.pdf`