Technical Report

Project Kitchen Occupation

TSBB11 HT 2013

Version 1.1

Status

Reviewed	–	2013–12–18
Approved		

2013–12–18

Project Kitchen Occupation

Bilder och Grafik CDIO, HT 2013
Department of Electrical Engineering (ISY), Linköping University

Participants

Name	Tag	Responsibilities	Phone	E-mail
Mattias Tiger	MT	Project manager	073–695 71 53	matti166@student.liu.se
Erik Fall	EF	Tracking	076–186 98 84	erifa226@student.liu.se
Gustav Häger	GH	System integration	070–649 03 97	gusha124@student.liu.se
Malin Rudin	MR	Evaluation	073–800 35 77	malru103@student.liu.se
Alexander Sjöholm	AS	User Interface	076–225 11 74	alesj050@student.liu.se
Martin Svensson	MS	Documentation	070–289 01 49	marsv106@student.liu.se
Nikolaus West	NW	Quality Control	073–698 92 60	nikwe491@student.liu.se

Homepage: densekitchen.bloip.se

Customer: Joakim Nejdeby, Linköping University, Origo 3154
Customer contact: 013–28 17 57, joakim.nejdeby@liu.se
Project supervisor: Fahad Khan, Linköping University, fahad.khan@liu.se
Examiner: Michael Felsberg, michael.felsberg@liu.se

Contents

List of Figures

List of Tables

Document history

Version	Date	Changes	Sign	Reviewed
1.0	2013–12–13	First release	MT	Everyone
1.1	2013–12–19	Minor changes, mainly regarding figure captions	MT	Everyone

1 Introduction

Linköping University has several student kitchens all over its campuses where students are given a possibility to warm their food. Critics claim that there are too few student kitchens and that the existing ones are usually overcrowded. That all kitchens are overcrowded at the same time has not been confirmed by sample inspections. One standing hypothesis is that students do not know where all the kitchens are, nor do they want to risk going to a kitchen in another building in case it is full as well.

The aim of this project is to develop a system that will provide the students with information regarding student kitchen usage. The system uses an computer vision approach, estimating the number of people currently using the kitchens.

This document, together with the Code Reference Manual and the User Manual constitutes the final documentation of the project and describes the current implementation together with some failed attempts and conclusions drawn from these. Performance results of the current implementation are presented in section 4 of this document.

2 System Overview

The system counts the people entering and leaving a room, presents a classification of the severity of potential queues to enter the room, and sends this information to a web server over a REST API. To be able to do this, a Microsoft Kinect is placed over each entrance to the room, and is connected to the computation device that runs the system software for that room. A debug/configuration GUI can be used to calibrate and configure the software. This produces configuration files that are used by the system. Further tuning and configuration can be done by editing the configuration files. A system overview can be seen in figure 2.1 below.

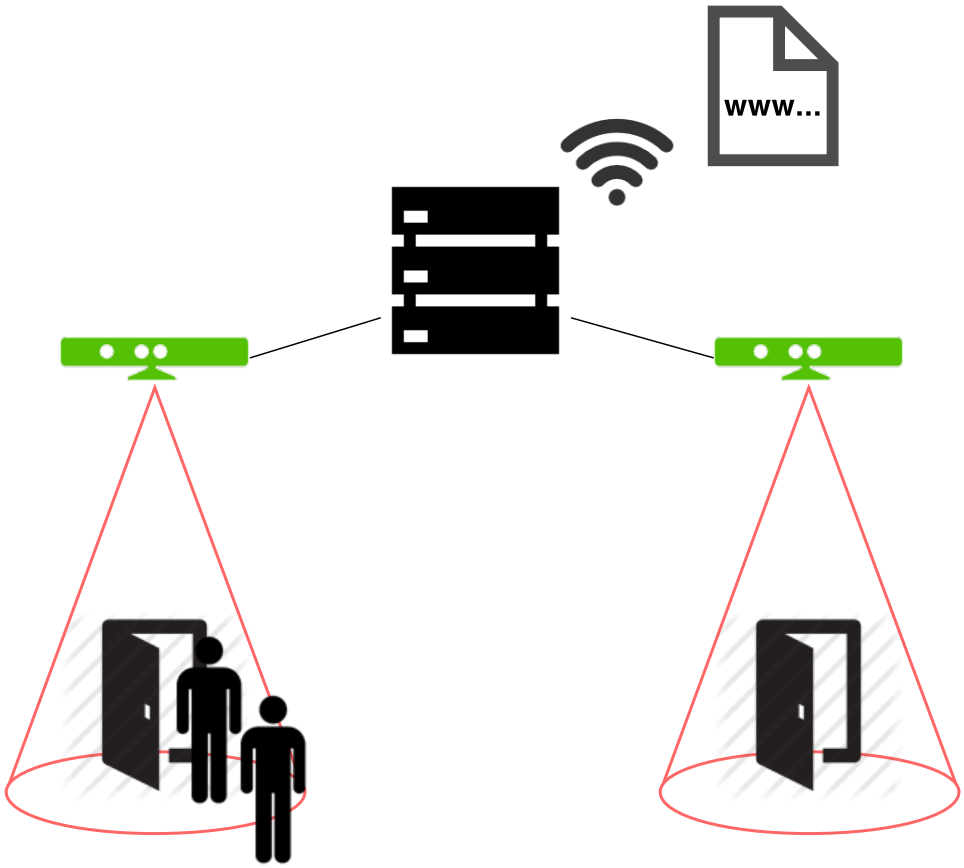


Figure 2.1: *System overview. One Microsoft Kinect (green objects) is placed over each entry to the room (doors). The Kinects are connected to a computation device (between the Kinects above), which communicates to a server over the Internet.*

2.1 Platform and hardware requirements

The system runs and has been tested on Windows, Mac OS X, and Linux. It has run at real time frame rates on a Linux virtual machine with 512 MB RAM but has not been tested on any embedded devices. It however should work given that the device has sufficient processing power. Care must be taken when considering embedded devices so that the USB module can handle the data volumes needed to stream depth images from the Kinect device at 30 fps.

2.2 Configuration

The whole system can be configured by editing the plain text YAML configuration files. Among other things, it is possible to set what image processing algorithms should be run, in what order to run them and the value of

most parameters. Many components in the Debug GUI, camera settings and important file paths are also set in the configuration files.

2.3 Modular software architecture

The software has a modular and easily extendable design, where each module is responsible for a specific domain of the system. The architecture of the system is that of a pipeline of algorithms, where data pertaining to the latest few frames is piped through each pipeline step by the main program. The order is specified in a configuration file and set in place during program initialization. The frame data consists of both raw sensor data and computed data from the different pipeline steps.

2.3.1 Network module

The network module manages all system input/output, i.e. sampling the sensors and posting output data on the web server specified by the configuration file. Currently, support exists for all OpenCV-compatible cameras as well as the Microsoft Kinect depth sensor. The module is designed to be as modular as possible, allowing for a relatively easy integration of new sensor types into the system. The network module also supports running several sensors in parallel.

2.3.2 Image processing module

The image processing module handles all processing that is done directly on any RGB or depth data. This includes detection of objects and object attributes as well as object tracking.

2.3.3 Statistics module

The statistics module handles computation and processing that is not done directly on RGB or depth data. This includes counting the number of entering and exiting objects, using object attributes to infer the presence of queues and all data aggregation.

2.3.4 Evaluation module

The evaluation module handles the evaluation of the algorithm pipeline by comparison between its results and the ground truth on labeled data sets.

2.4 Debugging and configuration GUI

The system includes a debugging and configuration GUI where the results from each of the process steps can be viewed in real time, and one step at a time. It can also be used to assist in tuning and calibrating the system, and is necessary for proper setup of a new sensor and room.

3 Current pipeline

This section describes the image processing pipeline that is delivered with the system and that has yielded the best performance so far. Some less successful attempts are described in appendices B, C, and D. An overview of the current pipeline can be seen in figure 3.1.

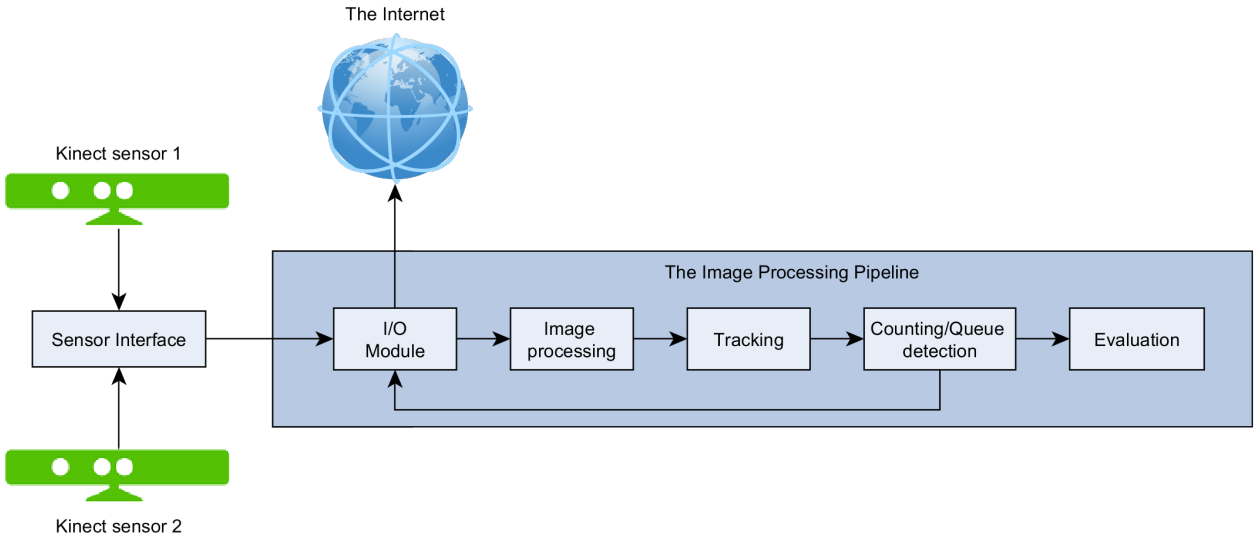


Figure 3.1: *An overview of the current pipeline.*

3.1 Sensors

After reading several papers on counting people using a single camera with top-down view, and implementing the methods described in said papers, the realization was made that, in order to be able to solve the problem described in section 1, some form of depth information would be necessary. Both stereo and Kinect-style sensors were considered. However due to time limitations and a customer desire to run the system on cheap hardware, the Microsoft Kinect sensor was chosen.

3.2 Image processing

The human segmentation is based on the assumption that human heads are distinguishable modes in the depth image and that people moving very close to each other seldom differ much more than a head in height. It is realized by a series of threshold and morphological operations, contour drawing and searches for local maxima. The output from some of the steps in the segmentation algorithm can be seen in figure 3.2.

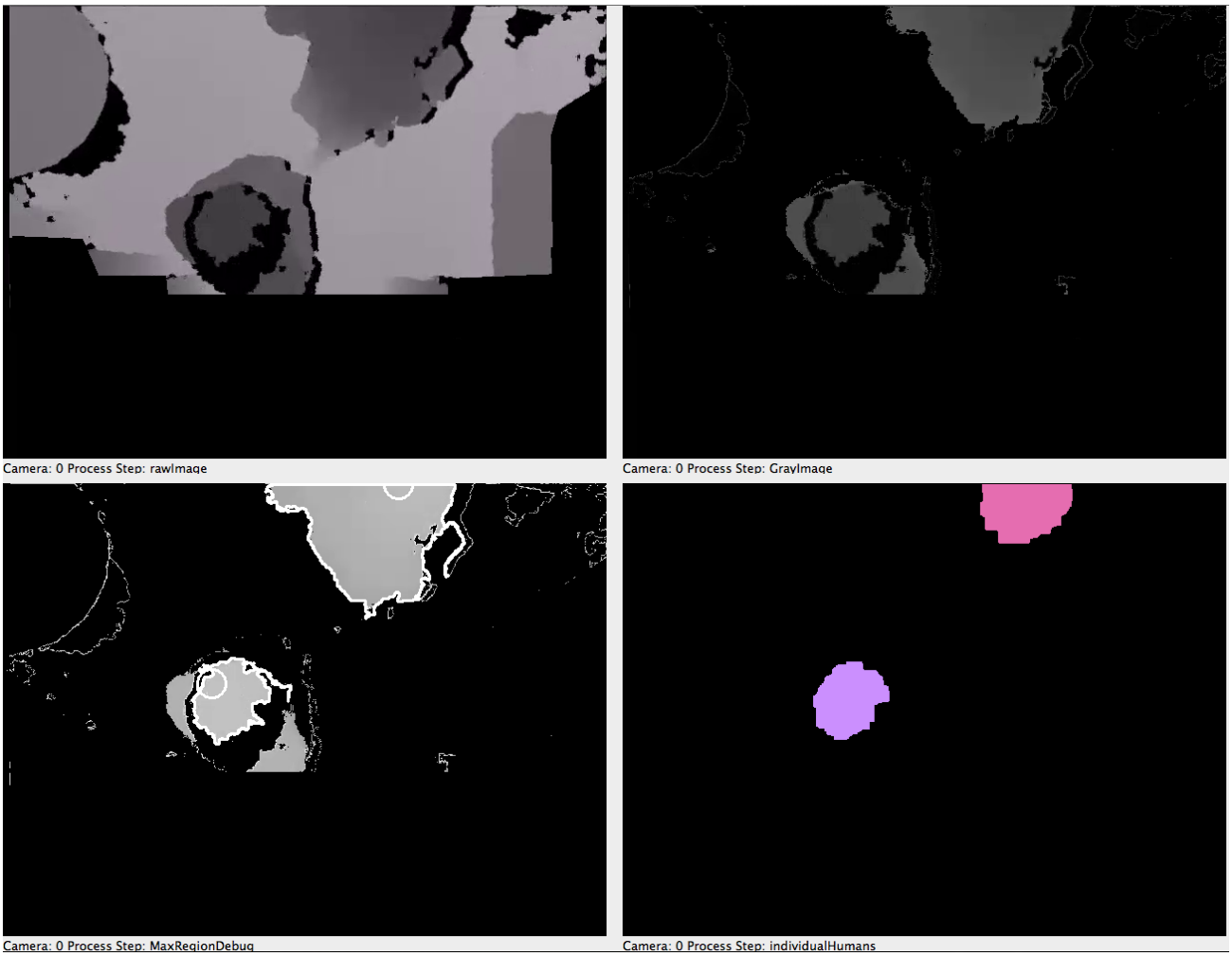


Figure 3.2: *Image processing steps. The figure shows the results from four of the processing steps in the image processing pipeline.*
Top left: Raw depth image.
Top right: Image after first threshold operation.
Bottom left: Contour drawing and local maxima detection.
Bottom right: Segmented human heads.

3.3 Tracking and counting

The tracking algorithm performs six different steps for each frame iteration. The tracker has a list objects found by earlier image processing steps and one list of objects passed from the previous frame. Tracking is done by pairing closest objects with each other, from previous frame to the current frame. Objects need to live for a predetermine configurable amount of frames before it will be counted. This prevent false flickering objects from generating false entries. Lost objects are also kept alive for some frames, this enables objects to be lost and found. Figure 3.3 illustrates how objects are seen as real objects, lost objects or potential objects, depending on their lifetime.

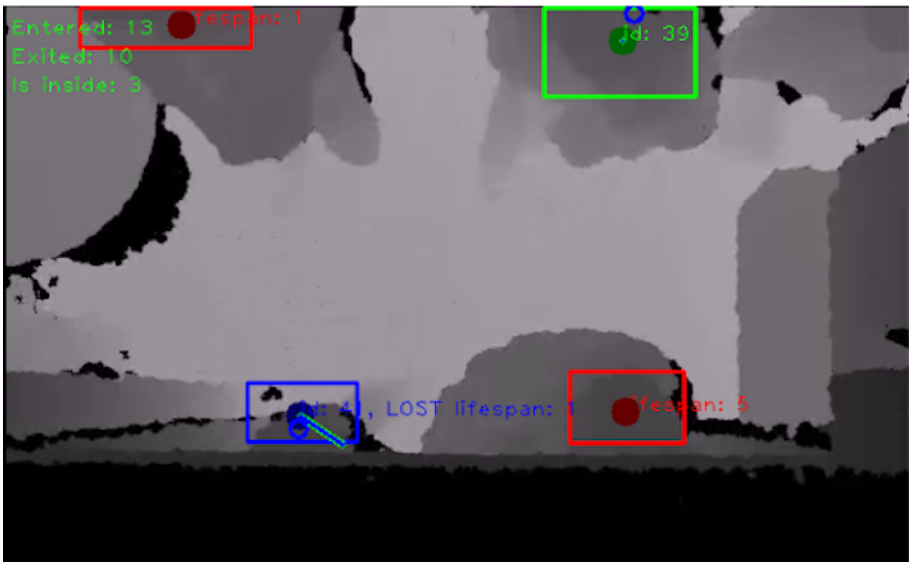


Figure 3.3: *Visualization of the tracking. The image shows one person (green box) who has lived long enough to be considered a person, two (red boxes) who has just been found and one (blue box) which is a person who has just been lost*

To register if objects enter or exit the room the objects has to fulfill some requirements. To be considered as entered, an object has to be found in a door area and pass three circle lines in addition to have existed a minimum of a set amount of frames. To be registered as exited an object has to have existed a minimum of a set amount of frames and disappear inside a door area, while also at least once passed the three lines. Examples of these circle lines and a door area can be seen in figure 3.4.

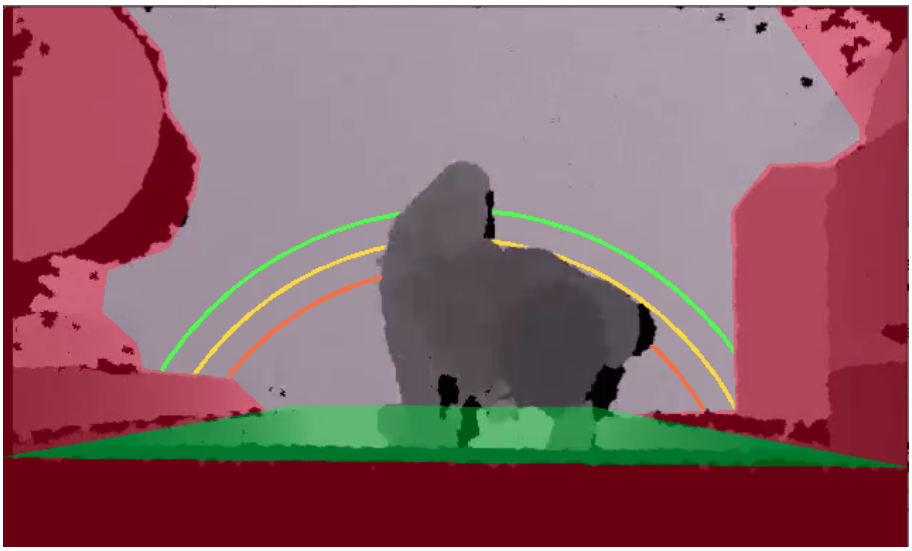
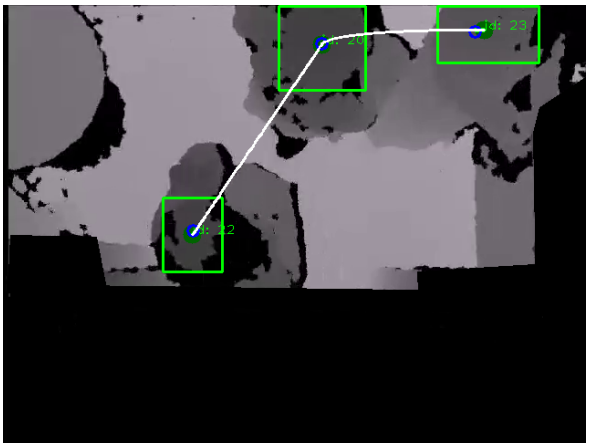


Figure 3.4: *Entry and exit counting. Three circles are used to decide when a person has entered. The green area is the door area and the red area is excluded by the system.*

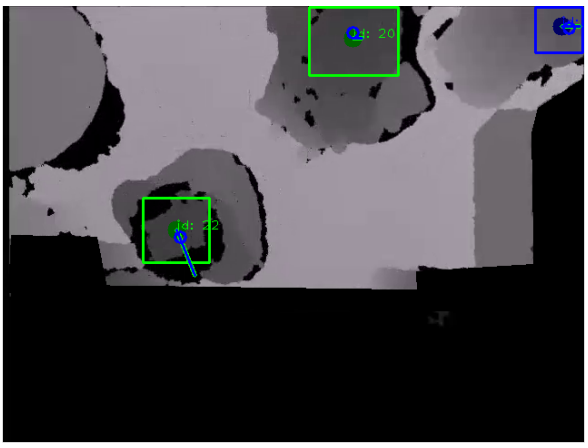
3.4 Queue severity estimation

Queues are detected by using the positions and velocities, which are calculated using a simple Kalman filter, of all visible persons. This information is used to draw splines between each of the visible persons. Two people are then considered to be in a queue if a short enough spline can be drawn between them. The result is that persons that stand still or move slowly along the same path are considered to be in a queue. Examples of this can be seen in figure 3.5a and 3.5b.

Queue severity is estimated by considering the proportion of frames with visible queues, and optionally the amount of people inside the room, over a set time interval.



(a) When persons are close together and moving slowly, they are connected by short splines and thus considered to be in a queue.



(b) Persons moving in different directions, with large velocity, are connected by longer splines, and are thus not considered to be in a queue.

Figure 3.5: *Detection of queues using splines. The images show the result of the queue detection steps in two different cases. Splines that are short enough are drawn to visualize detected queues.*

4 System Evaluation

In order to measure performance a performance metric needs to be defined, and test data and training data needs to be collected from several system use cases. The number of people entering and exiting is thoroughly evaluated according to the below described criteria. However, because the queue severity estimation is poorly defined, this part of the system has only been qualitatively evaluated by visual inspection of performance on the collected data sets.. The system also includes functionality to evaluate the tracking, which is not currently used but is further explained in appendix A.

4.1 Ground Truth

The evaluation needs access to ground truth data, which defines the best possible achievable counter output. Currently the ground truth files consist of arrays with values for each frame denoting how many people entered or exited in that specific frame. Since the focus of the system lies on maintaining a correct count over some time, the exact moment when a person exits the room is deemphasized.

4.2 Evaluation Metric

Reading several papers on the subject of people counting it became clear that no standard for measuring people counting performance exists. In the absence of such a standard the below measurement methods were chosen since they are thought to provide an accuracy measurement of the outputs the customer is most likely to be interested in (total number of people using the room, and total number of people currently in the room). What was decided to be most desirable and important to measure was the number of people entering, leaving, and the error in the room occupancy estimation as a function of the total number of people that have entered or left the room. The equations for the three different metrics are defined below, where the subscript *Est* denotes the value generated by the system and GT the true value that was read from the ground truth data.

$$A_{in} = 1 - \left| \frac{\sum_{frames} in_{Est} - \sum_{frames} in_{GT}}{\sum_{frames} in_{GT}} \right| \tag{4.2.1}$$

$$A_{out} = 1 - \left| \frac{\sum_{frames} out_{Est} - \sum_{frames} out_{GT}}{\sum_{frames} out_{GT}} \right| \tag{4.2.2}$$

$$A_{bias} = \frac{(\sum_{frames} in_{Est} - \sum_{frames} out_{Est}) - (\sum_{frames} in_{GT} - \sum_{frames} out_{GT})}{\sum_{frames} in_{GT} + out_{GT}} \tag{4.2.3}$$

5 Results

The system results were obtained by evaluating against three recorded sequences from three different kitchens. The sequences were labeled manually using a self-developed labeling program.

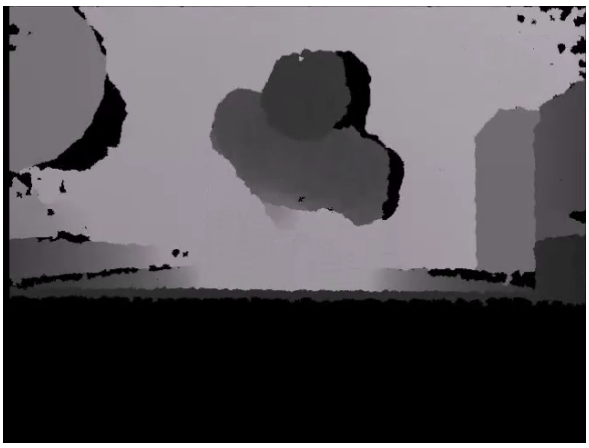
5.1 Evaluation Scores

In table 5.1 below the video clips chosen for the evaluation are presented, along with the achieved accuracy measurements. The first 5 minutes of the data set R-Kitchen was used when tuning and developing the system. The data set B25-Kitchen show how robust our system is, since the system wasn't calibrated for the specific room and the camera was slightly miss-placed. The majority of the errors was confirmed to depend on these two factors by ocular inspection.

Sequence Name	Total entered (GT)	A_{in}	Total exited (GT)	A_{out}	length
R-Kitchen	108 (108) people	100 %	101 (104) people	97 %	32 min
U-Kitchen	123 (122) people	99 %	134 (135) people	99 %	31 min
B25-Kitchen	131 (141) people	93 %	82 (91) people	90 %	30 min

Table 5.1: *Counting performance according to the evaluation metric as described in section 4.*

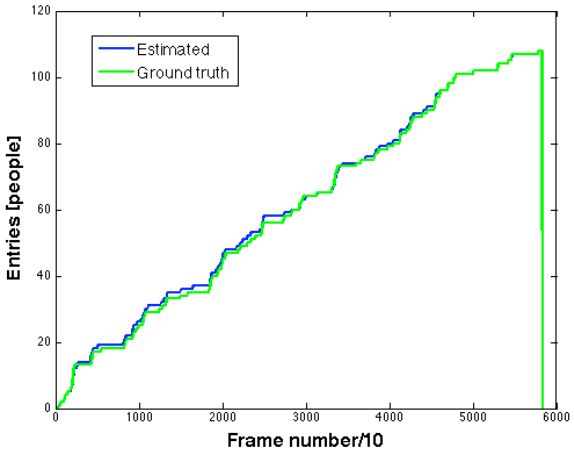
Plots of ground truth comparison and accuracy for the evaluation sequences can be found in figures 5.1 to 5.9.



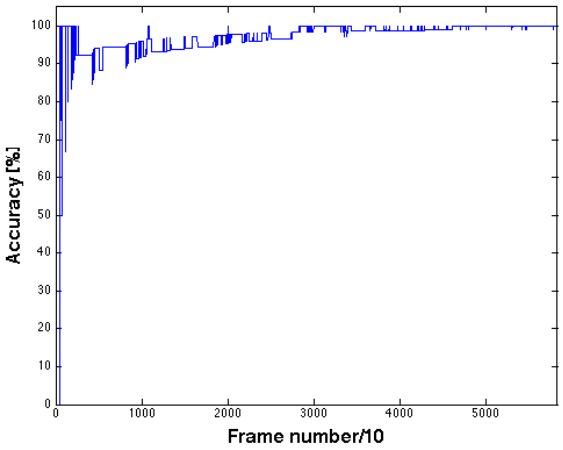
(a) Example depth image from the R-kitchen data set

(b) An other, more crowded example depth image

Figure 5.1: *Example depth images from the R-kitchen data set*

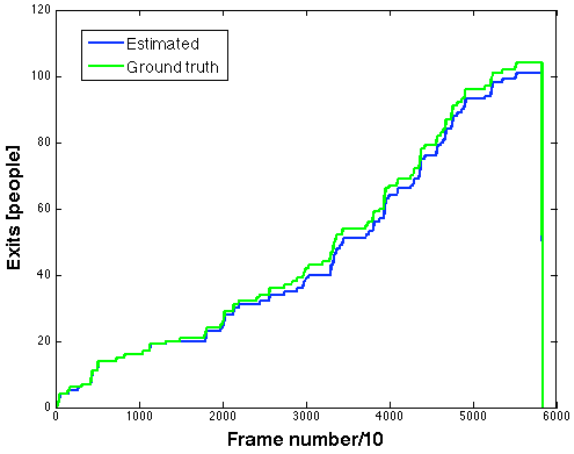


(a) Measured entries and ground truth

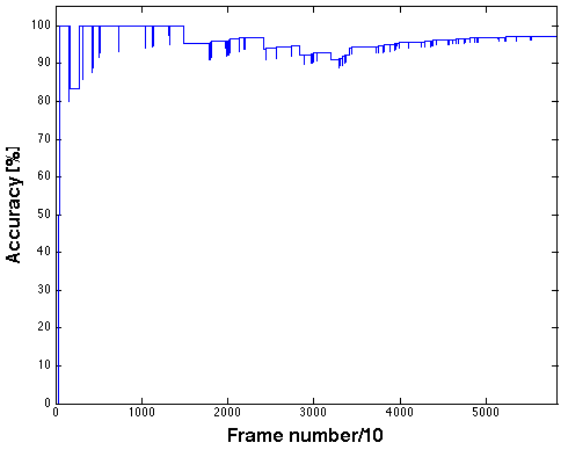


(b) Accuracy

Figure 5.2: *R-Kitchen data. Plot of measured entries, ground truth and accuracy*

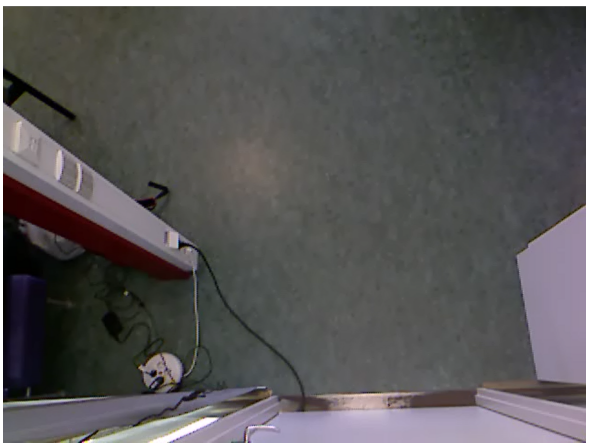


(a) Measured exits and ground truth

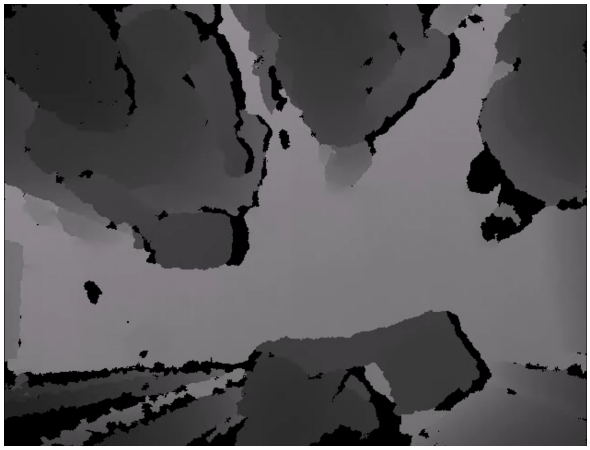


(b) Accuracy

Figure 5.3: *R-Kitchen data. Plot of measured exits, ground truth and accuracy*

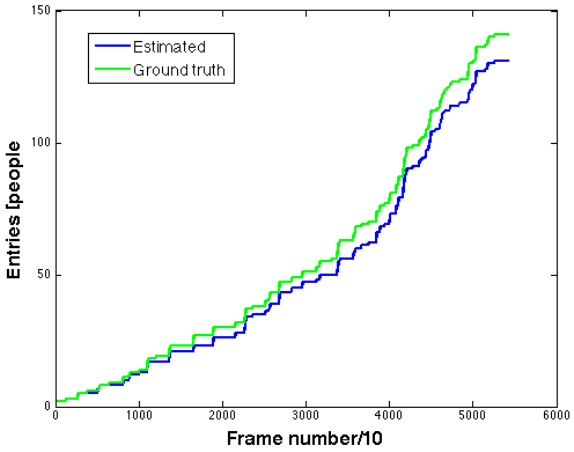


(a) An example color image taken from the B25-kitchen data set

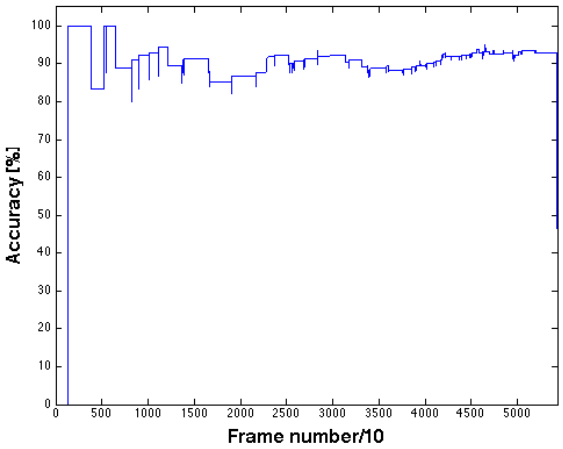


(b) An example depth image

Figure 5.4: *Example color and depth images from the B25-kithcen data set*

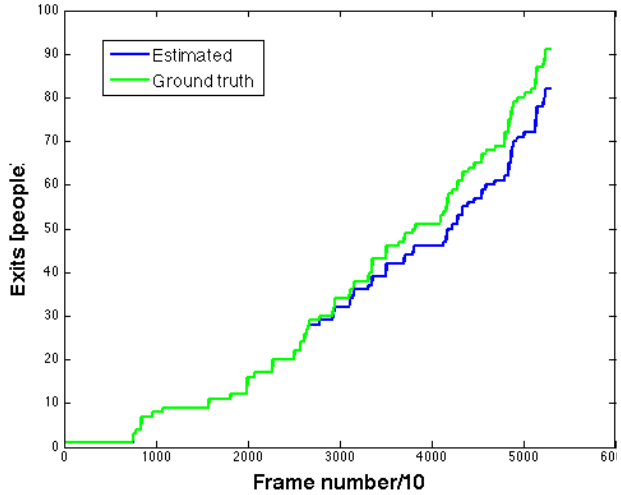


(a) Measured entries and ground truth

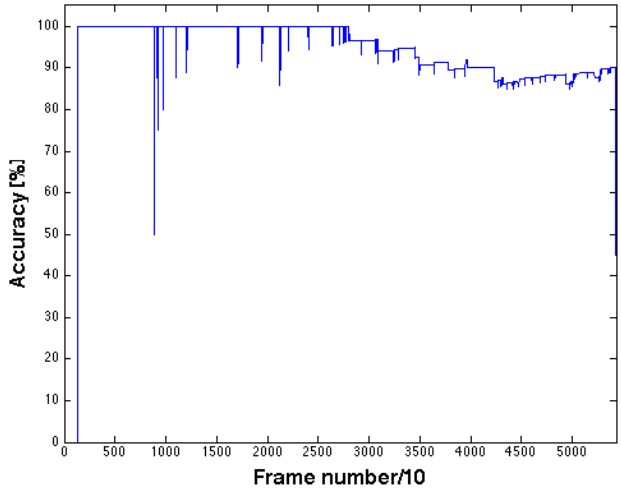


(b) Accuracy

Figure 5.5: *B25-Kitchen data. Plot of measured entries, ground truth and accuracy*



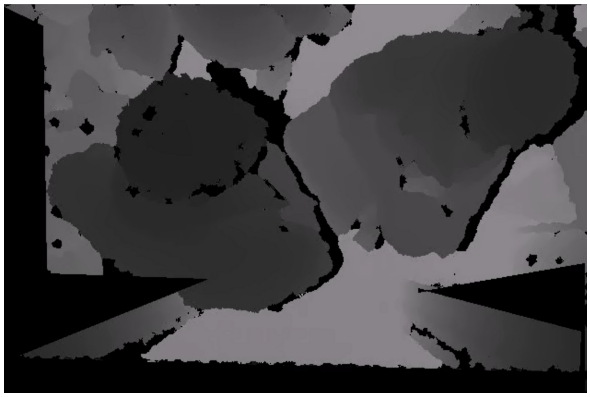
(a) Measured exits and ground truth



(b) Accuracy

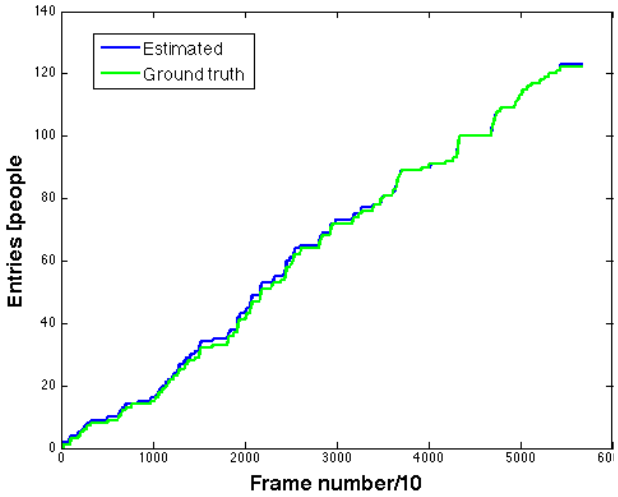
Figure 5.6: *B25-Kitchen data. Plot of measured exits, ground truth and accuracy*

(a) An example color image taken from the U-kitchen data set

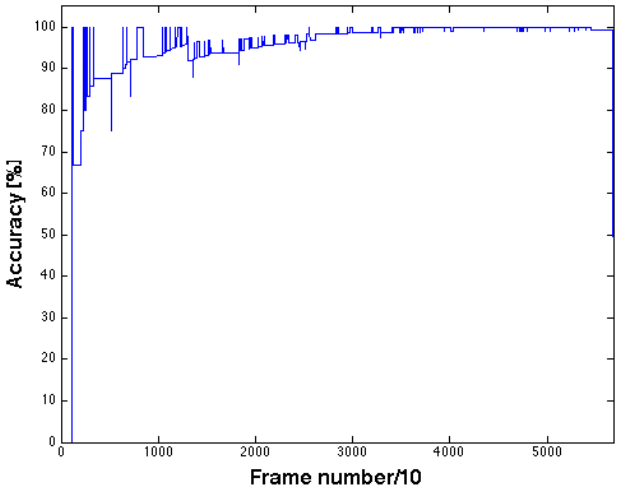


(b) An example depth image

Figure 5.7: *Example color and depth images from the U-kitchen data set*

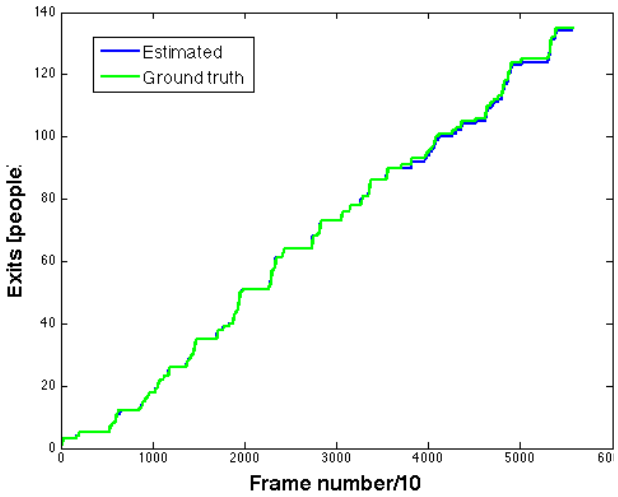


(a) Measured entries and ground truth

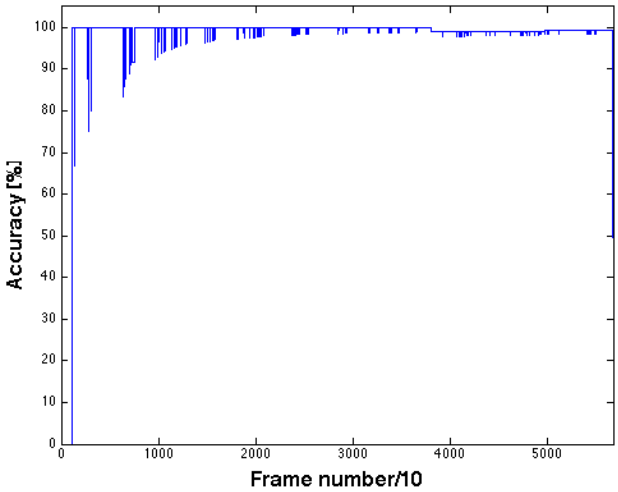


(b) Accuracy

Figure 5.8: *U-Kitchen data. Plot of measured entries, ground truth and accuracy*



(a) Measured exits and ground truth

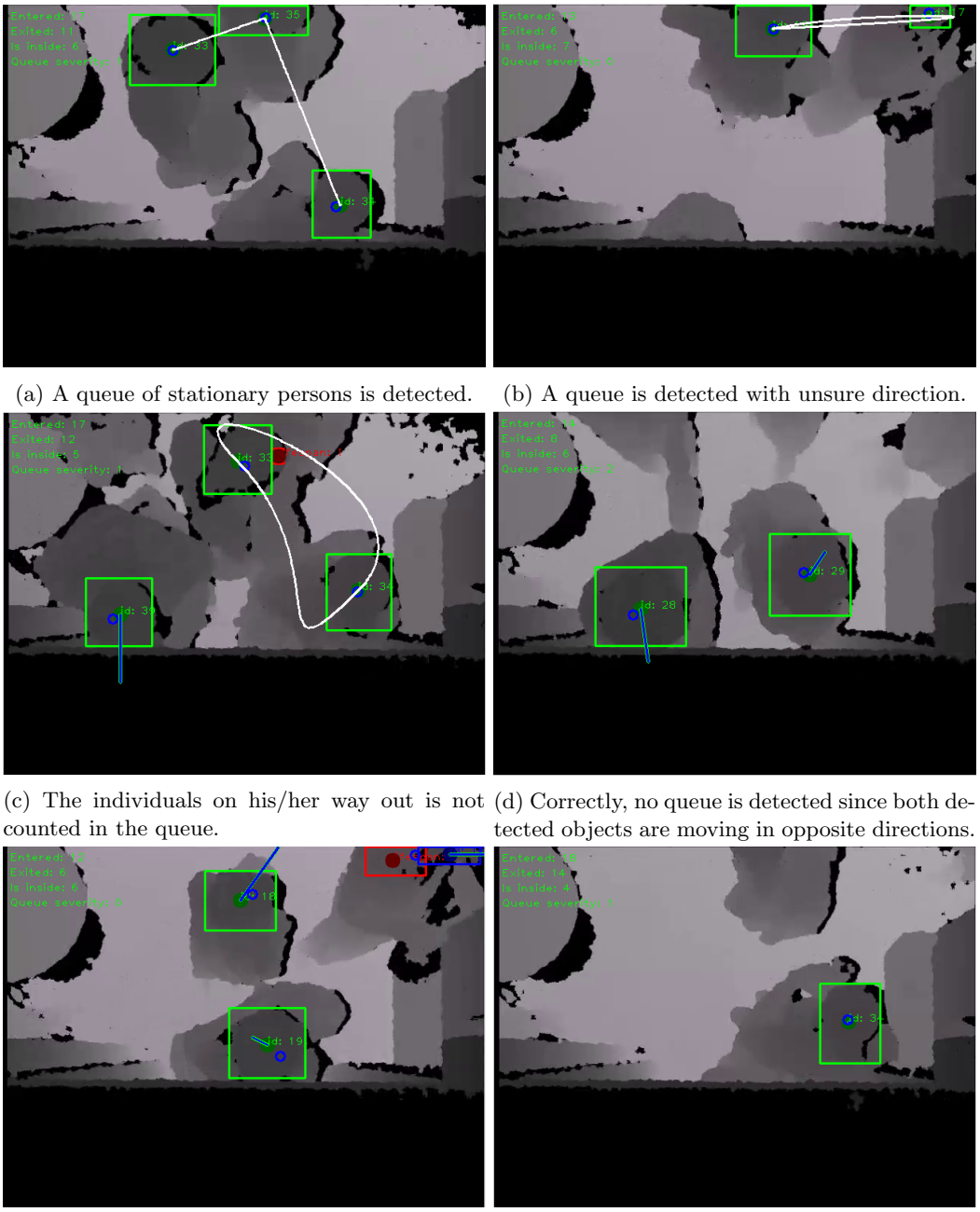


(b) Accuracy

Figure 5.9: *U-Kitchen data. Plot of measured exits, ground truth and accuracy*

5.2 Queue Detection

Below in figure 5.10, several screenshots from the queue detection steps are shown. Several difficult cases are shown to be handled gracefully. The main difficulty is when the small field of view gives situations where a queue is present but only one queuing person is detected at one time. However, this is usually handled well by the queue severity estimation, as the thresholds for the proportion of frames with detected queues can be calibrated to account for these results, given that several queuing persons are in view at simultaneously sometimes.



(a) A queue of stationary persons is detected.

(b) A queue is detected with unsure direction.

(c) The individuals on his/her way out is not counted in the queue.

(d) Correctly, no queue is detected since both detected objects are moving in opposite directions.

(e) Correctly, no queue is detected since both detected objects are moving too fast.

(f) Incorrectly, no queue is detected due to that the next person's head is not in view.

Figure 5.10: *Depth data images from the R-Kitchen, with green boxes showing detected objects,blue dots showing object centers, blue lines from the centers showing estimated velocities, and white curves showing the fitted Beziér splines.*

5.3 Discussion

The performance of the R-Kitchen and U-Kitchen sets are very good. The R-Kitchen set is a sequence of 30 minutes, where only the first 5 was used for tuning. The system performed worse on the B25-Kitchen sequence, but that is probably explained by the fact that the sensor was a bit misplaced, making parts of the upper door area on one side to be outside of the field of view, more dynamic placement of count-checkpoint-lines might have helped as well. The reason no more data sets were used was simply that no more time was available.

The queue detection and severity estimation in general worked as expected but performed worse for more sparse queues, in combination with a small field of view.

6 Conclusions

The most prominent conclusion that can be drawn from this project is that the problem of counting people passing through a doorway using a single camera is much harder than what it may seem at first glance. Another is the fact that spending time on building a good software architecture increases the possibility to handle fast changes of project circumstances, as were the case here. Because of the high quality of the software architecture, we were able to change approach quickly and achieve good results in a short period of time.

6.1 Future work

As with most projects the possible improvements are many. An improvement that would most likely yield a notable increase in performance would probably be to replace the current Microsoft Kinect with a similar sensor that has better depth accuracy.

The biggest obstacle to more stable performance and to be able to give more reliable and interesting estimates of the queue severity is the small field of view afforded by the Microsoft Kinect. A solution to this could be to calculate depth from several cameras if the disparity map can be calculated fast enough on cheap hardware. Another approach might be to have one Kinect on each side of each entrance so that a larger field of view would be obtained without significant interference between the two sensors.

A Additional functionality - Tracker accuracy evaluation.

Previous experience of group members on implementing object tracking in video sequences sparked the idea of incorporating a method for evaluating object trackers into the system evaluation. The main reason for this was that an implementation in C++ using OpenCV was already available from a previous computer vision project. The evaluation metric available was the MOTA/MOTP system defined by Bernardin & Stiefelhagen in (2008) [3].

Initially the evaluation system promised good results as the implementation easily fit into the project architecture. The first evaluation was performed on short RGB test data files. Even though the implementation went smoothly, obtaining ground truth data was a whole other matter. No suitable (top-down view of people passing a doorway) clips with pre-created ground truth data were found, which meant that ground truth data would have to be labeled manually. The ground truth data was on the format specified by the CAVIAR project [4]. A program for labeling sequences frame by frame is available on the CAVIAR homepage. Because the tracker evaluation depends on accurate object positions it soon became clear that labeling enough ground truth data by stepping through sequences frame by frame to provide a feasible basis for evaluation would be far to time consuming. This is especially noticeable when one considers the fact that the actual goal of the project is to simply count people passing by, and not to track them with perfect accuracy.

The tracking evaluator is included with the final software because it works well and might be useful if the software is to be used for an other application, for example people tracking outdoors or other open spaces.

B An unsuccessful attempt - people detection using background-foreground segmentation.

A very simple solution was tried early in the project. It consisted of a Mixture of Gaussian model [1] used for segmenting the moving foreground from the background as a binary mask. After the segmentation, all connected moving objects were found and tracked using a simple tracker. This approach was developed and performed well for tracking people moving relatively fast, but worse or not at all for people moving slow and standing still. This approached suffered from problems such as being unable to handle common occlusions when people stand and move too close to each other, especially when doing so from entering to leaving the field of view. To detect individual persons in such situations required more advanced methods, some of which were tried, but never really tuned or good enough to perform well enough. Further improvements on this approach were therefore never implemented. The approach also required quite some morphological operations to connect loosely connected components of humans, as seen in figure B.1, or very restrictive limits on minimum object area.

Using a more sophisticated tracker with prediction and which remembered persons even as they became a part of the background, most problems except severe occlusions could possibly been coped with. Another solution would have been to identify humans and disallow the background model from learning those region, eliminating the disappearances of tracked persons.

The algorithm described in [1] was chosen since it was the most recently developed background subtractor in OpenCV [6], and also the best performing one according to the OpenCV reference manual.

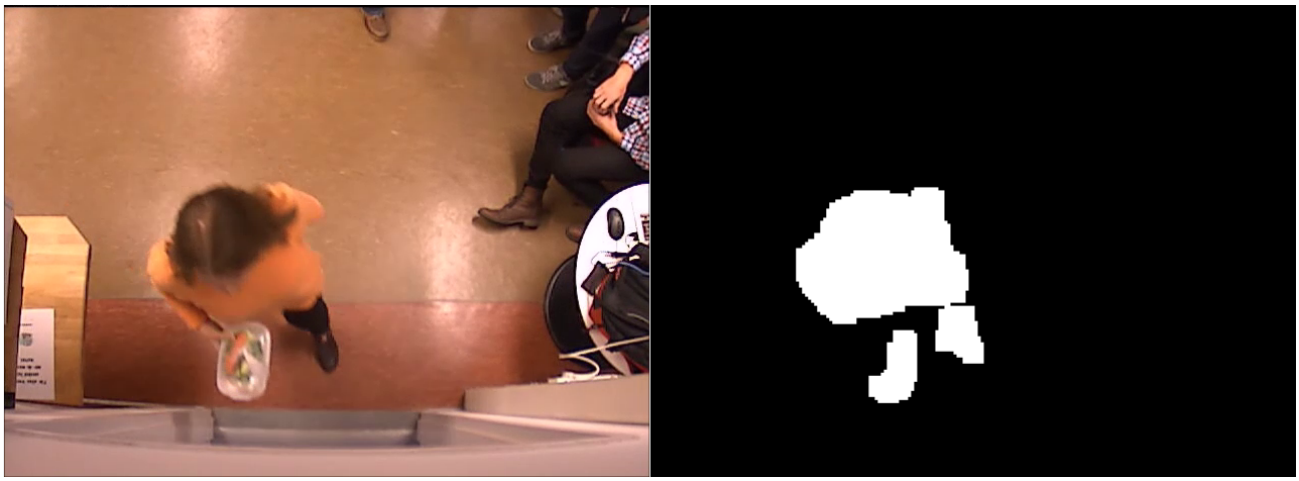


Figure B.1: *The left image shows a moving person who is detected by the background subtractor. The right image shows the binary image generated from the left image, illustrating the scattered parts separated due to none-uniform motion.*

C An unsuccessful attempt - head detection by means of circle detection.

In [1] Gardel et.al. a method of head detection from above based on circle detection is proposed. In their approach circles, i.e. heads, are detected by first performing canny edge detection on each image frame and then performing a form of hough voting to detect circles. The hough voting is done by combining the results of a series of different filters designed to give high responses at circle and ellipse centers. We implemented their approach and experimented with many different variations of filter sizes and canny variables, with both types of circle filters described in the paper. We also tried using background subtraction on the canny image, using a mixture of Gaussian-based foreground segmentation [2] of the raw image, which gave slightly better results in simple situations. We were however not able to get any usable results for anything but the simplest cases, and therefore gave up on this approach after many fruitless attempts at tuning the system. Outputs from some of the different steps of cases where the approach was both successful and unsuccessful can be seen in figure C.1 and C.2 below.

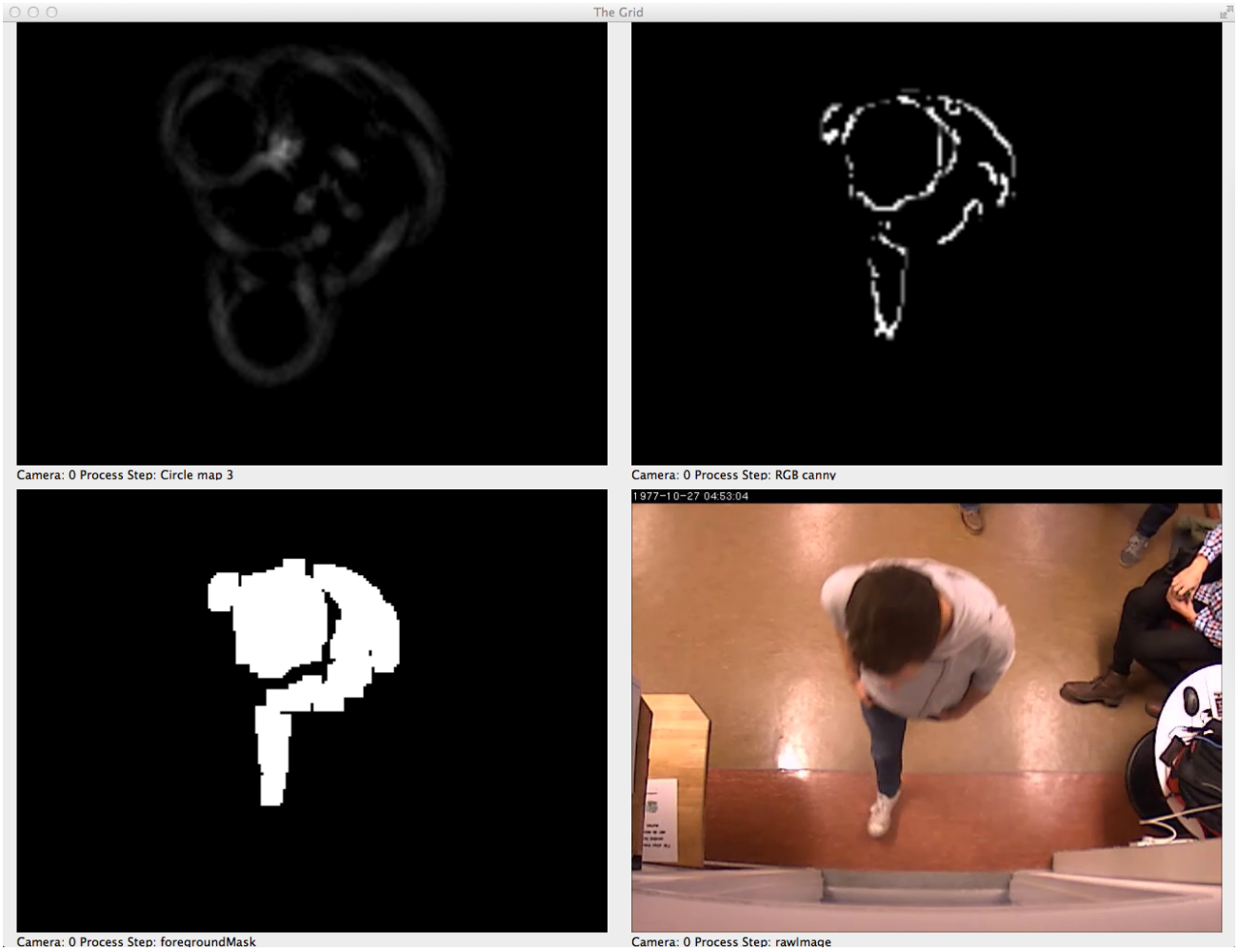


Figure C.1: *Successful case. Results from four processing steps in head detection by means of circle detection.*
Top left: output from one of the circle filters.
Top right: output from the canny edge detection after background subtraction.
Bottom left: foreground mask.
Bottom right: raw image.

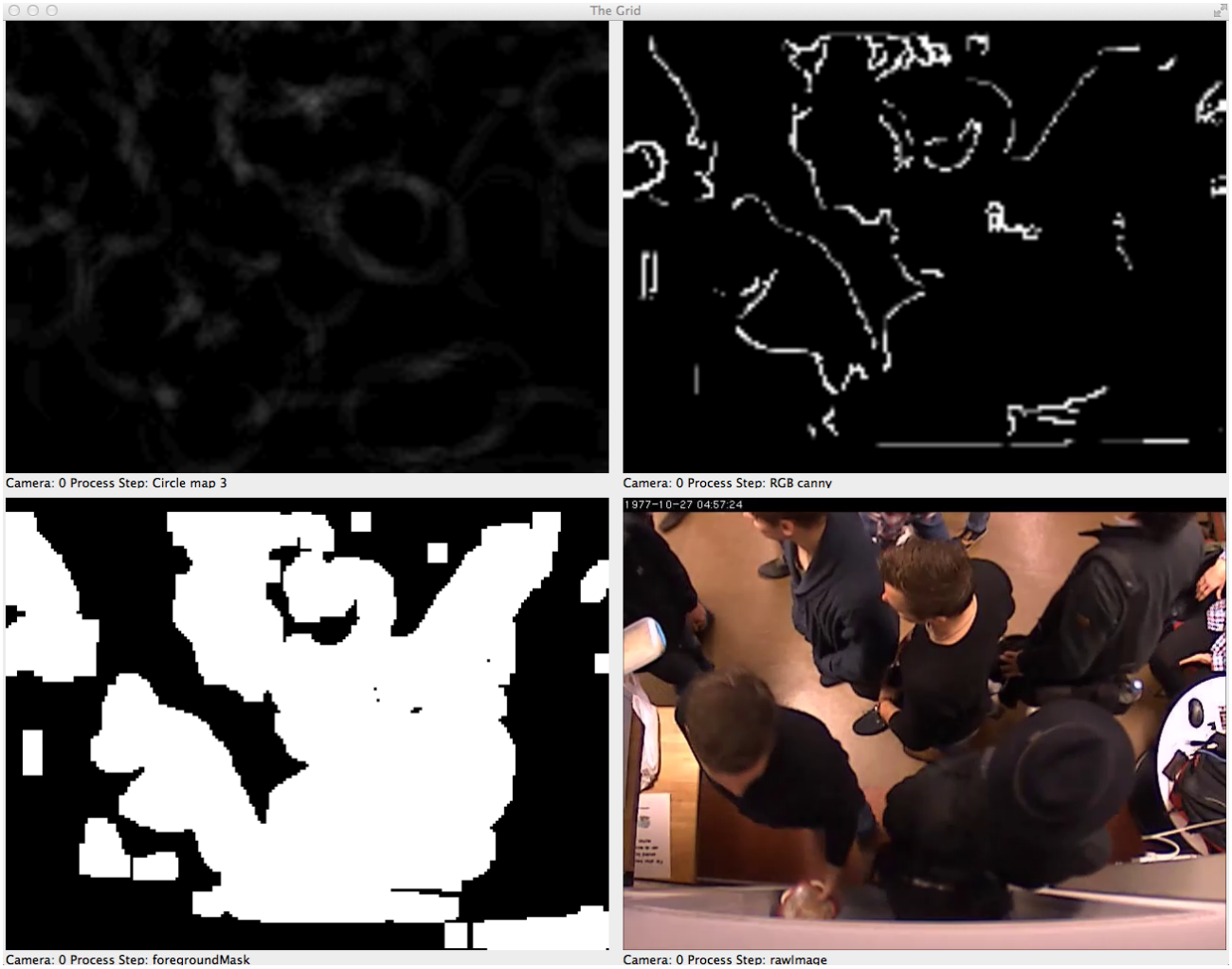


Figure C.2: *Unsuccessful case. Results from four processing steps in head detection by means of circle detection.*
Top left: output from one of the circle filters.
Top right: output from the canny edge detection after background subtraction.
Bottom left: foreground mask.
Bottom right: raw image.

D Explored possibility - Depth image from stereo.

There are different ways in which one can obtain depth information. The easiest is perhaps to use a Kinect sensor which does all the job for you. However, this sensor is not compatible with the initially intended platform, namely the Raspberry Pi. An alternative to the Kinect is to use stereo vision. The semi-global stereo block matching algorithm proposed in [5] was tested with partly successful results. The problem is its speed. Each frame takes approximately 300 ms to process which is too much for this application. This could be remedied to some extend by sacrificing image quality. The computation time can be reduced to 50 ms but the result is harder to use in later process steps. This algorithm could have been greatly sped up by a GPU implementation, but this approach was abandoned in favor for the Kinect which is both faster and easier to use. The results can be seen in figure D.1 below.

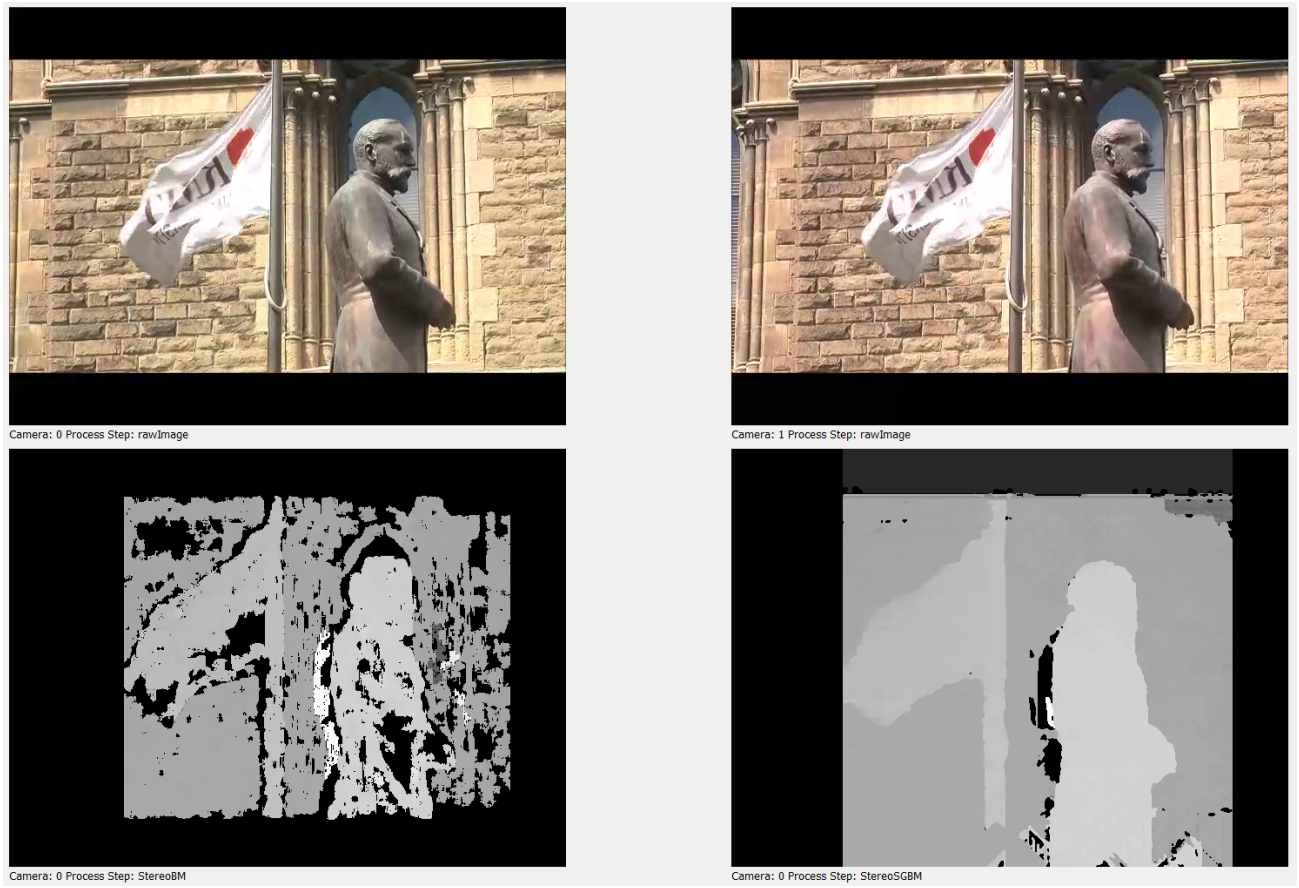


Figure D.1: *Evaluation of stereo block matching for depth calculations.*
Top: Left and right image of stereo sequence.
Bottom left: Faster approach with lower quality.
Bottom right: Slower approach with good quality.

References

[1] Gardel, A., Bravo, I., Jimenez, P., Lazaro, J.L. & Torquemada, A.
"*Statistical Background Models with Shadow Detection for Video Based Tracking*,"
Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on?? Page: 1-6.

[2] Zivkovic, Z. & Heijden, F.
"*Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction*,"
Pattern recognition letters, Vol. 27, No. 7. (2006), pp. 773-780.

[3] Bernardin, K. & Stiefelhagen, R (2008)
"*Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics*,"
Interactive Systems Lab, Institut für Theoretische Informatik,
Universität Karlsruhe, 76131 Karlsruhe, Germany

[4] "*CAVIAR: Context Aware Vision using Image-based Active Recognition*,"
EC Funded CAVIAR project/IST 2001 37540
http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

[5] Hirschmüller, H (2008)
"*Stereo Processing by Semiglobal Matching and Mutual Information*,"
IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 30(2) pp. 328-341.

[6] OpenCV *Open source computer vision*
http://docs.opencv.org/
Accessed on 2013-12-13