Technical Report

Project Kitchen Occupation

TSBB11 HT 2013

Version 1.0

Status

Reviewed	–	2013–12–13
Approved		

2013–12–13

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	–	076–186 98 84	erifa226@student.liu.se
Gustav Häger	GH	System integration	070–649 03 97	gusha124@student.liu.se
Malin Rudin	MR	–	073–800 35 77	malru103@student.liu.se
Alexander Sjöholm	AS	–	076–225 11 74	alesj050@student.liu.se
Martin Svensson	MS	Documentation	070–289 01 49	marsv106@student.liu.se
Nikolaus West	NW	Testing	073–698 92 60	nikwe491@student.liu.se

Homepage: TBA

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
0.1	2013–12–13	Initial draft	MS	MT

1 Introduction

Linköping University has several student kitchens all over its campuses where students are provided with the ability to warm food. Critics claim that there are too few student kitchens and that the existing ones usually are 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 that they want to risk going to a kitchen in another building in case that it is full as well.

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

This docment, together with the code reference manual (appendix **??**) 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.

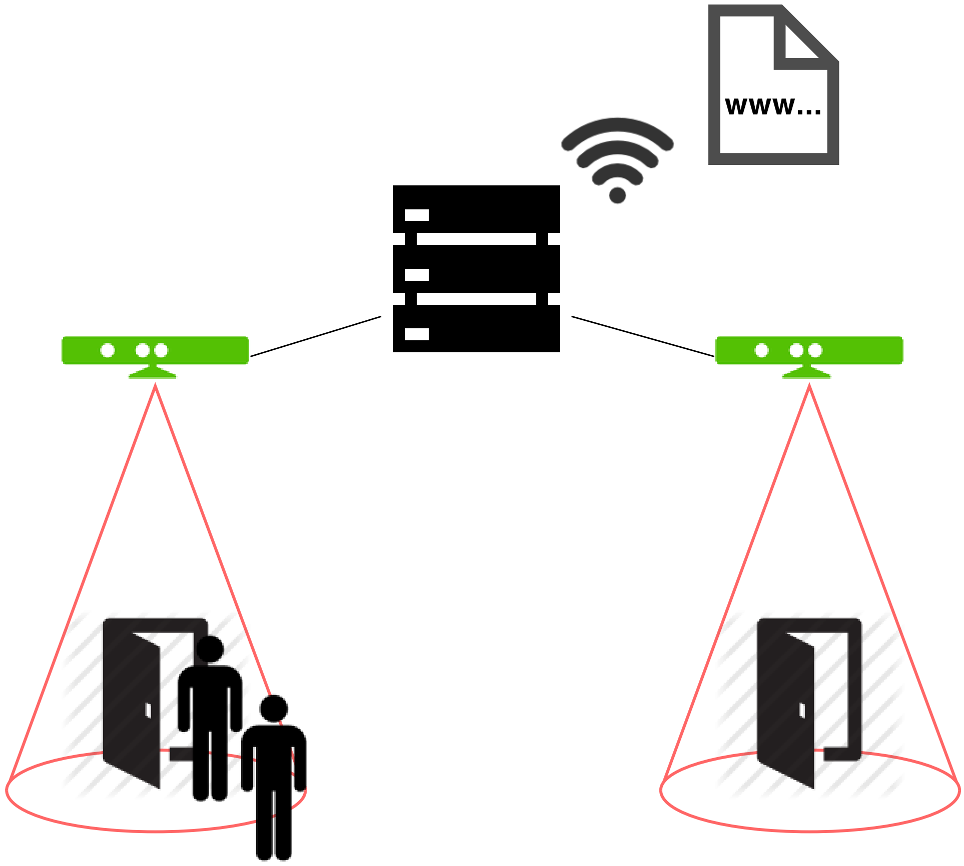


Figure 2.1: *System overview.*

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 as low as —– insert the smallest processing power tested on VM —–. It has not been tested on any embedded devices but 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 both RGB and 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 initialisation. 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 labelled 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 set up of a new sensor/room.

3 Current pipeline

This section describes the image processing pipleine that is delvired with the system and that has yielded the best performance so far.

3.1 Sensors

After reading several papers on single camera top-down view people counting 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 the human heads are distinguishable modes in the depth image and that people moving very close to each other seldom differ more than a head in height.

Only distances far above ground is considered which naturally segment between tall objects, providing a similar situation as acquired in B but with more distinguishable objects and without the problem of losing still objects. How well people are distinguished can be improved by ignoring even closer depths, but with the risk of missing shorter persons. An additional trick is done in order to detect at least some people still occluded by standing to close together. This is done by a second segmentation... (not done)

3.3 Tracking and counting

The tracking algorithm performs six different steps for each frame iteration. The tracker has a list with objects from the last frame and a list with current objects found by earlier image processing steps. The tracker pairs closest objects with each other from previous frame to the current frame.

To register if objects enters and exits the room the objects has to fulfill some requirements. To be considered as entered, an object has to be created for the first time in the set door area and pass three circle lines and by that be elevated to a real object. To be registered as exited an object has to be a real object and disappear inside the door area, while also at least once passed the three lines.

3.4 Queue detection

Information about: how do i detect queue?

4 System Evaluation

In order to measure performance some form of performance metric need to be defined, and test data and training data needed to be collected from several system use cases.

4.1 Ground Truth

The evaluation needs access to some sort of ground truth which defines the best possible achievable counter output. Currently the ground truth files consist of arrays with each 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 of less importance.

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

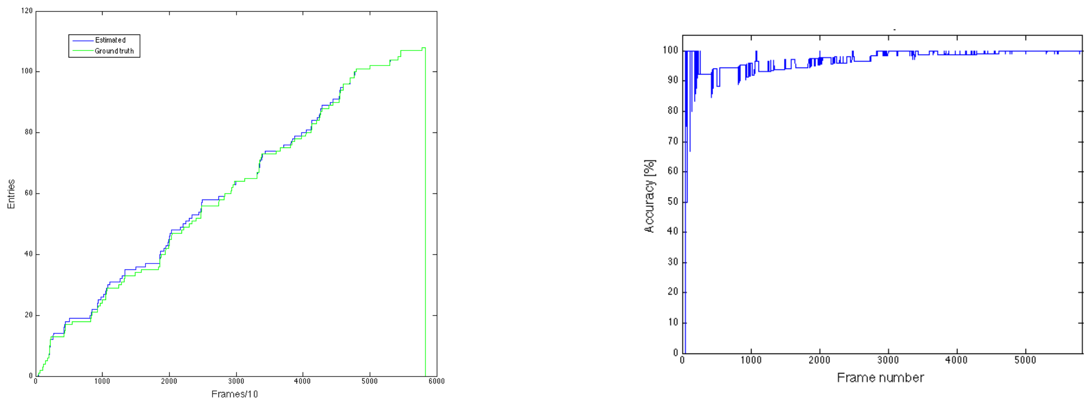
In this section the results of the project is summarized and the evaluation results are presented.

5.1 Evaluation Scores

In table 5.1 below the video clips chosen for the evaluation are presented, along with the achieved accuracy measurements. Training data set is a sequence of 30 minutes, where the first 5 minutes was used when training and developing the system . Evaluation data 1 is a sequence of 30 minutes.

Sequence Name	Total entered (GT)	A_{in}	Total exited (GT)	A_{out}	A_{bias}
Training data	108 (108)	101 (104)	0.99	0.97	0.99
Evaluation data 1	122 (141)	0.87	77 (91)	0.85	0.98

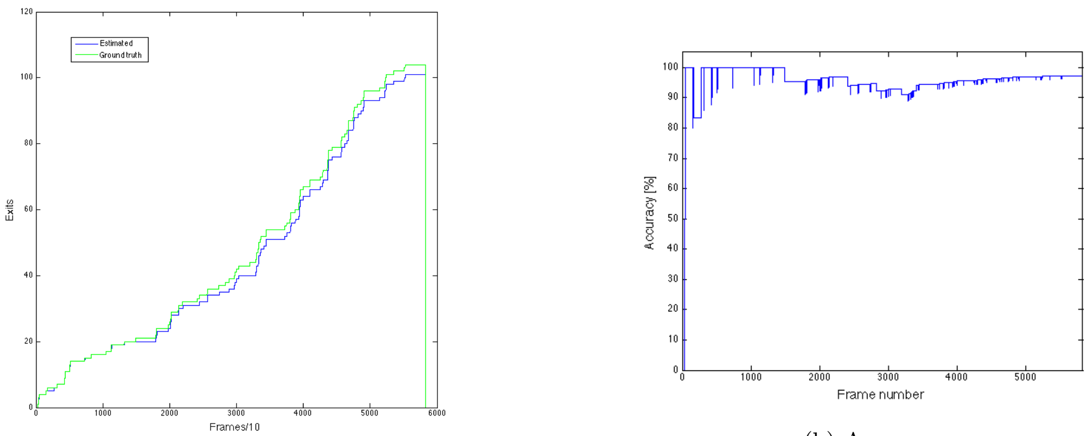
Table 5.1: *Counting performance according to the evaluation metric as described in section 4.*



(a) Measured entries and ground truth

(b) Accuracy

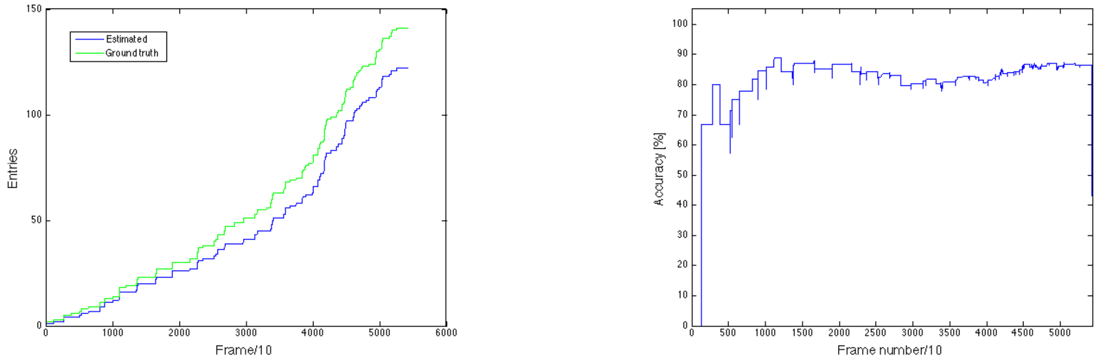
Figure 5.1: *Training data. Plot of measured entries, ground truth and accuracy*



(a) Measured exits and ground truth

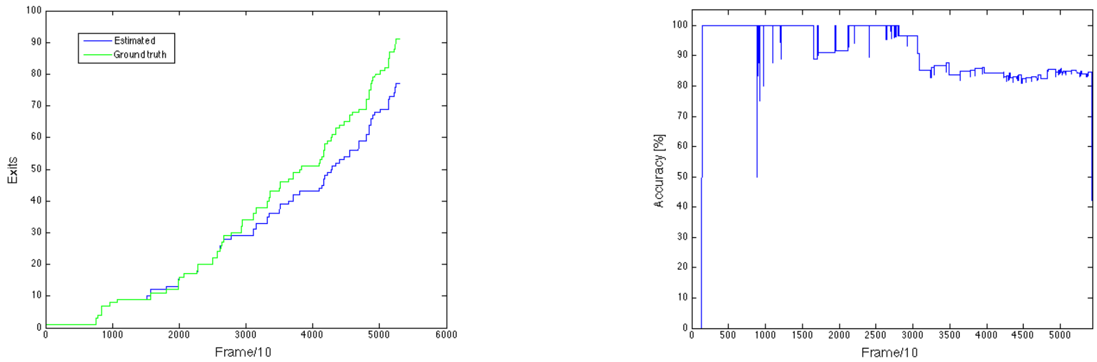
(b) Accuracy

Figure 5.2: *Training data. Plot of measured exits, ground truth and accuracy*



(a) Measured entries and ground truth (b) Accuracy

Figure 5.3: *Evaluation data 1. Plot of measured entries, ground truth and accuracy*



(a) Measured exits and ground truth (b) Accuracy

Figure 5.4: *Evaluation data 1. Plot of measured exits, ground truth and accuracy*

5.2 Discussion

The performance of the training data set is very good. This is a sequence of 30 minutes, where only the first 5 was used in development. Evaluation data 1 performed worse, but that can be explained. The camera of this sequence was palced badly and didn't cover the entry area completely.

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 much harder than what it may seem at first glance. An other 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.

Need brainstorming here.

6.1 Improvement suggestions

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

also needs more text here.

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 [9]. 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 subtraction.

A very simple solution was tried first in the project. It consisting of a Mixture of Gaussian model[1] used for segmenting the moving foreground from the background as a binary mask. Then all connected moving objects were found and tracked using a simple tracker.g of the moving objects. This approach was developed and performed good 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 be 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 segment persons in such situations required more advanced methods which we tried but never managed to work 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 area.

Using a more sophisticated tracker with prediction and which remembered still, persons even as they became a part of the background, most problems except the 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 recent developed background subtractor in OpenCV, and also the best performing one according to the OpenCV reference manual.

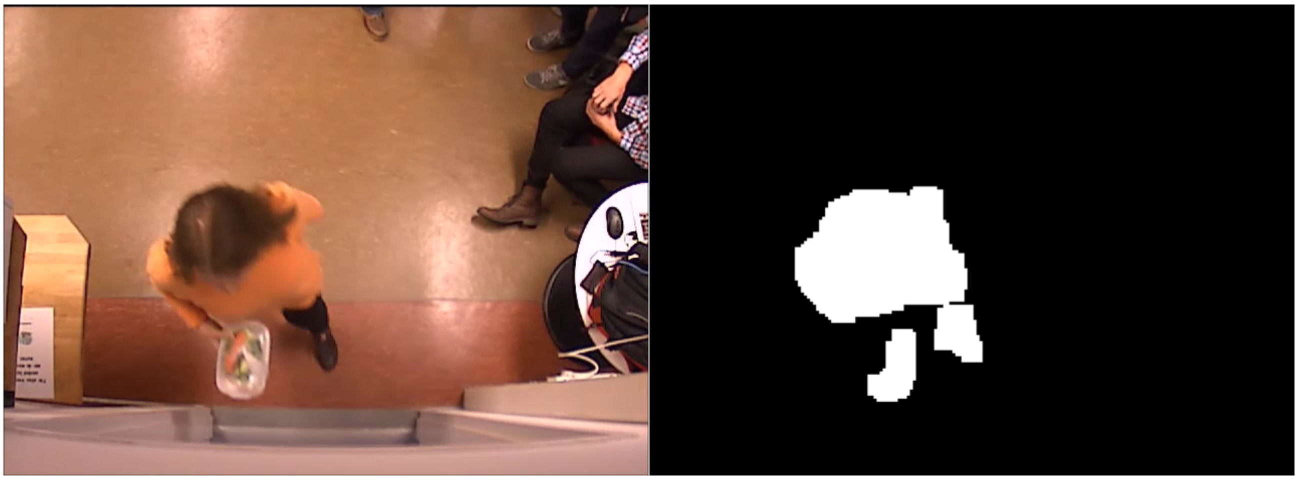


Figure B.1: *The left image shows a moving person who is detected by the background subtractor. The right image shows the binary image generated from the left image, illustrating the scattered parts separated due to none-uniform motion.*

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.

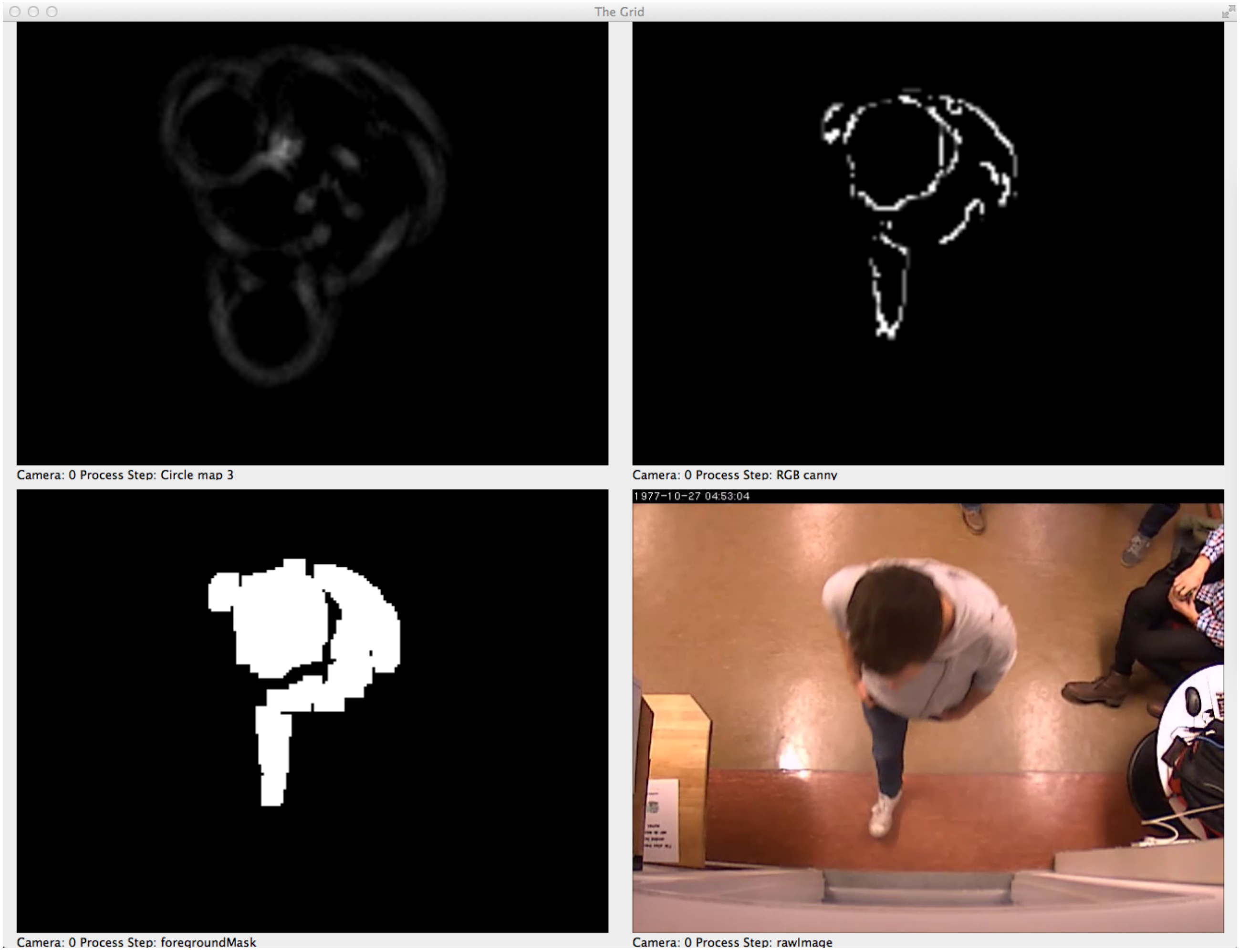


Figure C.1:
Top left: output from one of the circle filters.
Top right: output from the canny edge detection after background subtraction.
Bottom left: foreground mask.
Bottom right: raw image.

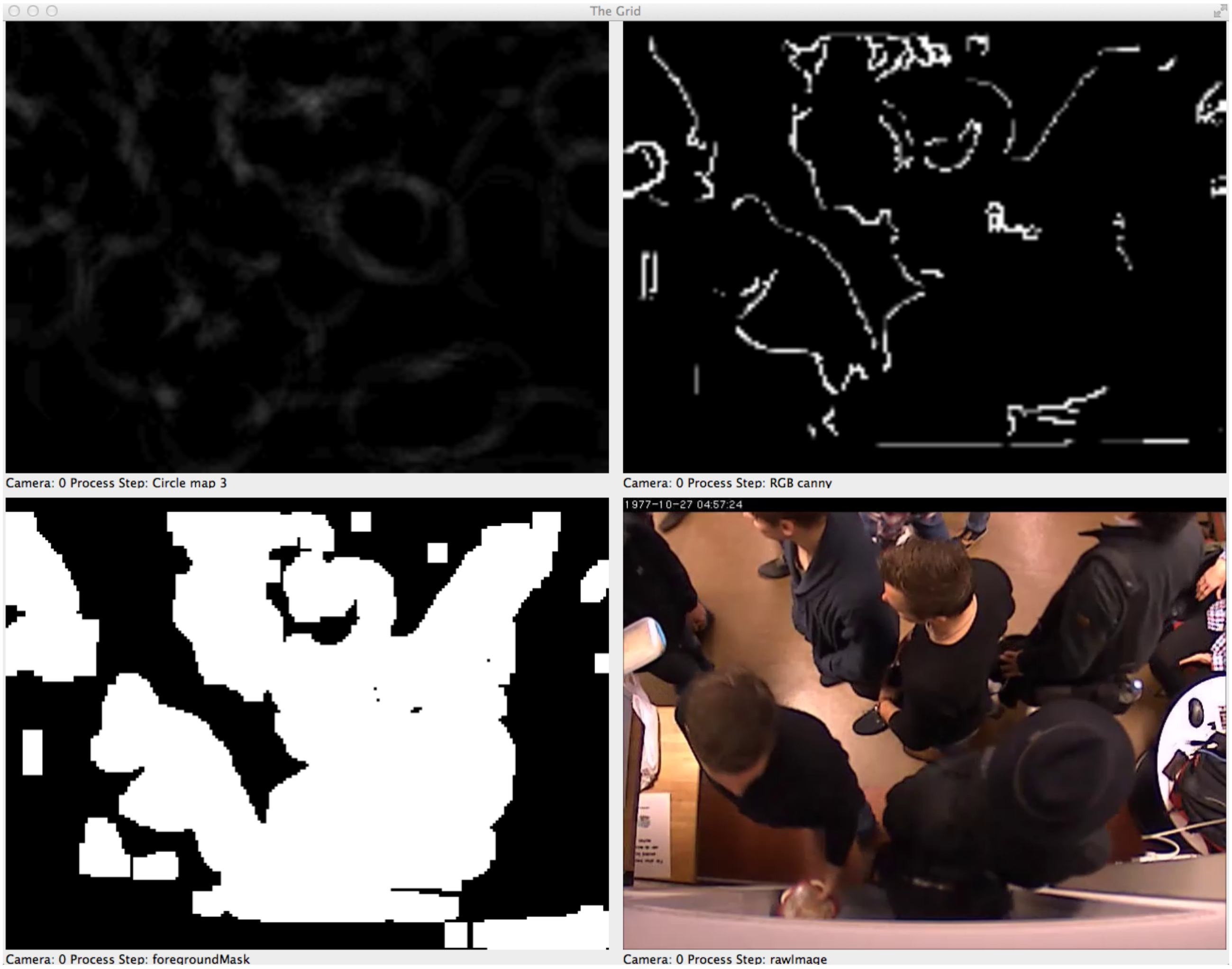


Figure C.2:
Top left: output from one of the circle filters.
Top right: output from the canny edge detection after background subtraction.
Bottom left: foreground mask.
Bottom right: raw image.

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 speed. Each frame takes approximately 300 ms to process which is too much for our application. This could be remedied to some extend by sacrificing quality. The computation time can be reduced to 50 ms but the result is harder to use in later process steps. This algorithm could be greatly speed up by a GPU implementation, but this approach was abandoned in favor for the kinect. The results can be seen in figure D.1 below.

Figure D.1:
Top: Left and right image of stereo sequence.
Bottom left: Faster approach with lower quality.
Bottom right: Slower approach with good enough 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.

EXAMPLE REFERENCES ONLY, REMOVE BEFORE HANDING IN

[6] Sonka, M., Hlavac, V. & Boyle, R. *Image Processing, Analysis, and Machine Vision.*
Toronto: Thompson Learning, cop. 2008, 3rd ed., ISBN 0495244384.

[7] Wood, J. (2007) "*Statistical Background Models with Shadow Detection for Video Based Tracking*,"
Master thesis, Linköping University, Department of Electrical Engineering.

[8] Gustafsson, F., Ljung, L. & Millnert, M. *Signal Processing.*
Studentlitteratur, Lund, Sweden, 2011, 1st ed., ISBN 978–91–44–05835–1.

[9] Hirschmüller, H (2008)
"*CAVIAR: Context Aware Vision using Image-based Active Recognition*,"
IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 30(2) pp. 328-341.