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

There exists many places in society where the degree of human occupancy and movement flow is desirable to know as basis for decision making. Such data answers if it is necessary to build more rooms and provides knowledge of actual user or consumer patterns. Example usages are measuring resource usage of public spaces, or which part of a store that attracts most people. It provides vast opportunities in resource management, marketing, sales and scheduling. There exist some plausible solutions to estimating the number of people at a location such as using cell phones or motion detectors, but this project aims at an image based approach with the possible benefits of being both cheaper and more robust.

1.1 Background

Today Linköping University has many places with similar functionality, e.g. student kitchens where students are provided with the ability to warm food brought with them. Linköping University has several such kitchens all over its campuses. Critics claim that there are too few student kitchens with microwave ovens 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 don't know where all the kitchens are nor that they want to risk going to a kitchen in another building in case that is full as well.

Linköping University has an ongoing project with the purpose of enabling the students to see the usage of some of the schools resources (e.g. group rooms) online. The aim of this project is to supply that system with data regarding the usage of student kitchens. It will provide all students with the ability of visualizing the crowdedness of each kitchen, thus providing them with the means of finding the closest, least occupied kitchen available.

1.2 About this document

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 5 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.

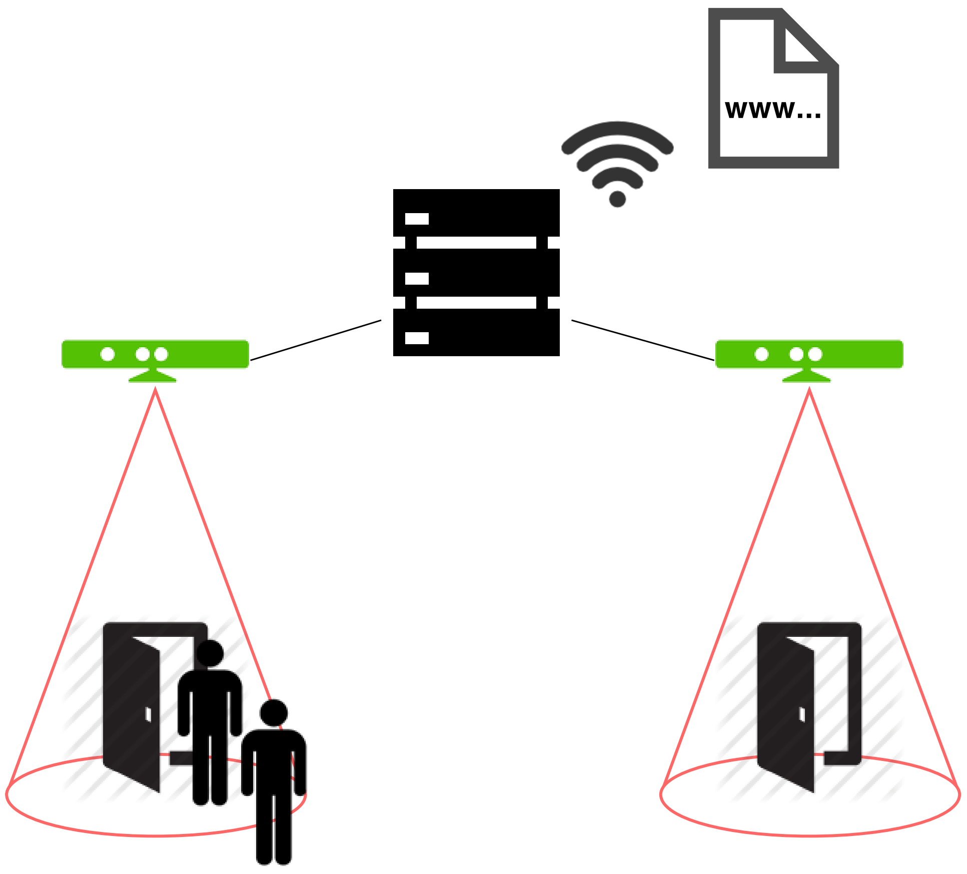

Figure 2.1: *System overview.*

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 design

The software has a modular and easily extendible design. The main components of the system are divided into different modules that can easily be swapped.

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 System Architecture

The system is designed to be as modular as possible, allowing for testing and implementation of a number of different algorithms and processing steps without having to spend a lot of time integrating these with the rest of the system. Tuning without having to re-compile the project was also a major goal that was achieved by using a set of configuration files that specify parameter values at startup.

Figure 3.1: *System overview.*

3.1 Configuration file system

The system configuration file contains all image processing pipeline parameters, including which image processing algorithms to run and in what order. together with the address of the current destination web service, where counting results are to be presented. There is also a GUI configuration file containing the Debug GUI as well as camera specific settings.

3.2 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.

3.3 Algorithm Interface

The algorithm interface simplifies the addition of new image processing or tracking modules to the system. Together with the config file, this allows a very flexible software system that could be used for virtually any computer vision problem. (Add more here)

3.4 Debug and Config GUI

Debug GUI features (Image and stuff) Configuration GUI management (more image and such)

4 Current pipeline

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

4.1 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.

4.2 Image processing

Information about the current image processing steps, thresholding etc etc.

4.3 Tracking and counting

Information about tracking (Kalman filter) and people counting (different area and such)

4.4 Queue detection

Information about: how do i detect queue?

5 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 different settings. A program for manually labeling these data sequences was also created and is delivered with the project.

5.1 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.

5.2 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 putputs 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.

$$A_{in} = 1 - |\frac{\sum_{frames} in_{Est} - \sum_{frames} in_{GT}}{\sum_{frames} in_{GT}}| \tag{5.2.1}$$

$$A_{out} = 1 - |\frac{\sum_{frames} out_{Est} - \sum_{frames} out_{GT}}{\sum_{frames} out_{GT}}| \tag{5.2.2}$$

$$A_{in-out} = 1 - |\frac{(\sum_{frames} in_{Est} - \sum_{frames} out_{Est}) - (\sum_{frames} in_{GT} - \sum_{frames} out_{GT})}{\sum_{frames} in_{GT} + out_{GT}}| \tag{5.2.3}$$

5.3 Tracker Evaluation

An initial thought was to use the MOTA/MOTP performance metrics as defined in [3] to evaluate the tracker part of the pipeline. This decision was made mainly because an implementation of these metrics was already available from a previous project. Initially some evaluation of the tracker was done using these methods. Since actual goal of the project was people counting, not tracking, in combination with the method of obtaining ground truth data for this evaluation being time consuming the tracker evaluation was scrapped.

6 Results

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

6.1 Evaluation Scores

In table 6.1 below the video clips chosen for the evaluation are presented, along with the achieved accuracy measurements.

Sequence Name	Total entered (GT)	A_{in}	Total exited (GT)	A_{out}	A_{in-out}
training data	0.99	0.99	0.99	0.99	0.99
evaluation data	0.99	0.99	0.99	0.99	0.99
other data	0.99	0.99	0.99	0.99	0.99
data stuff	0.99	0.99	0.99	0.99	0.99

Table 6.1: *Counting performance according to the evaluation metric as described in section 5.*

6.2 Discussion

Discussion stuff and more stuff.

7 Conclusions

The most prominent conclusion that can be drawn from this project 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.

7.1 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 An unsuccessful attempt - people detection using background subtraction.

Some group members had earlier experiences of tracking using background subtraction and therefore the first approach of the development was trying that method. Their approach was to generate a binary image from the raw image using [1] and after that use OpenCV's built in functions boundingRect in combination with own developed functions to separate e.g. blobs of humans. This approach was developed and performed good for tracking people moving relatively fast, see figure A.1, but worse or not at all for people moving slow and standing still. The tracker should be able to track a queue, which moves slowly or stands still, and in this regard the method using background subtraction ended up being too bad. The problem was that still or slow objects melted into the background and therefore disappeared from the binary image, different learning rates were tested but without good results. This is shown in figure A.2. The nature of the binary image makes it hard to differentiate people standing close to each other, because it then creates one united blob. 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.

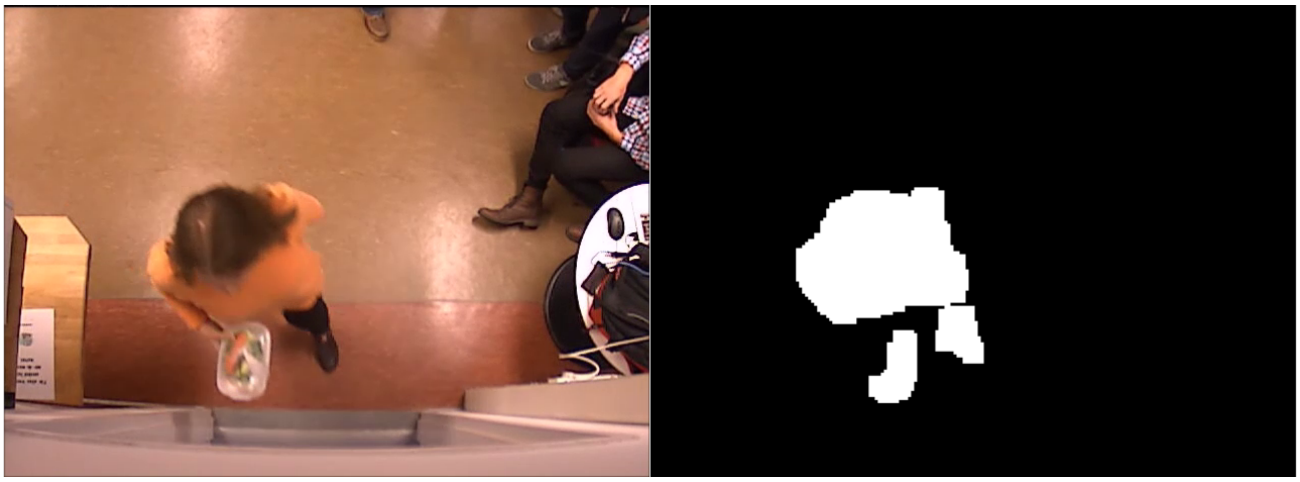

Figure A.1: *The left image shows a moving person who is detected by the background subtractor. The right image shows the binary image generated from the left image.*

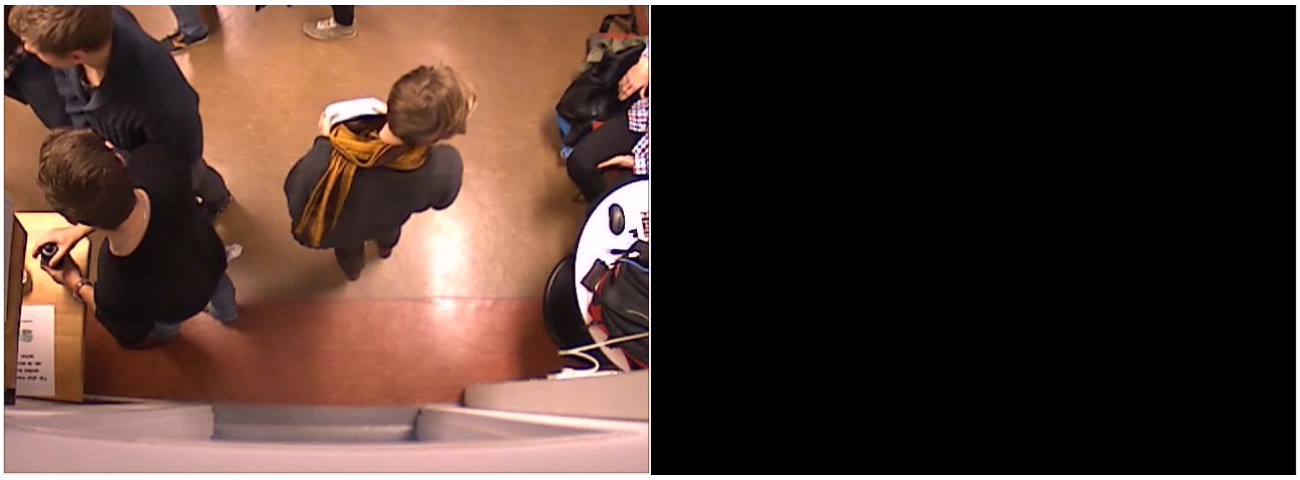

Figure A.2: *The left image shows a persons who is not detected by the background subtractor, because of insufficient movement. The right image shows the binary image generated from the left image.*

B 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 B.1 and B.2 below.

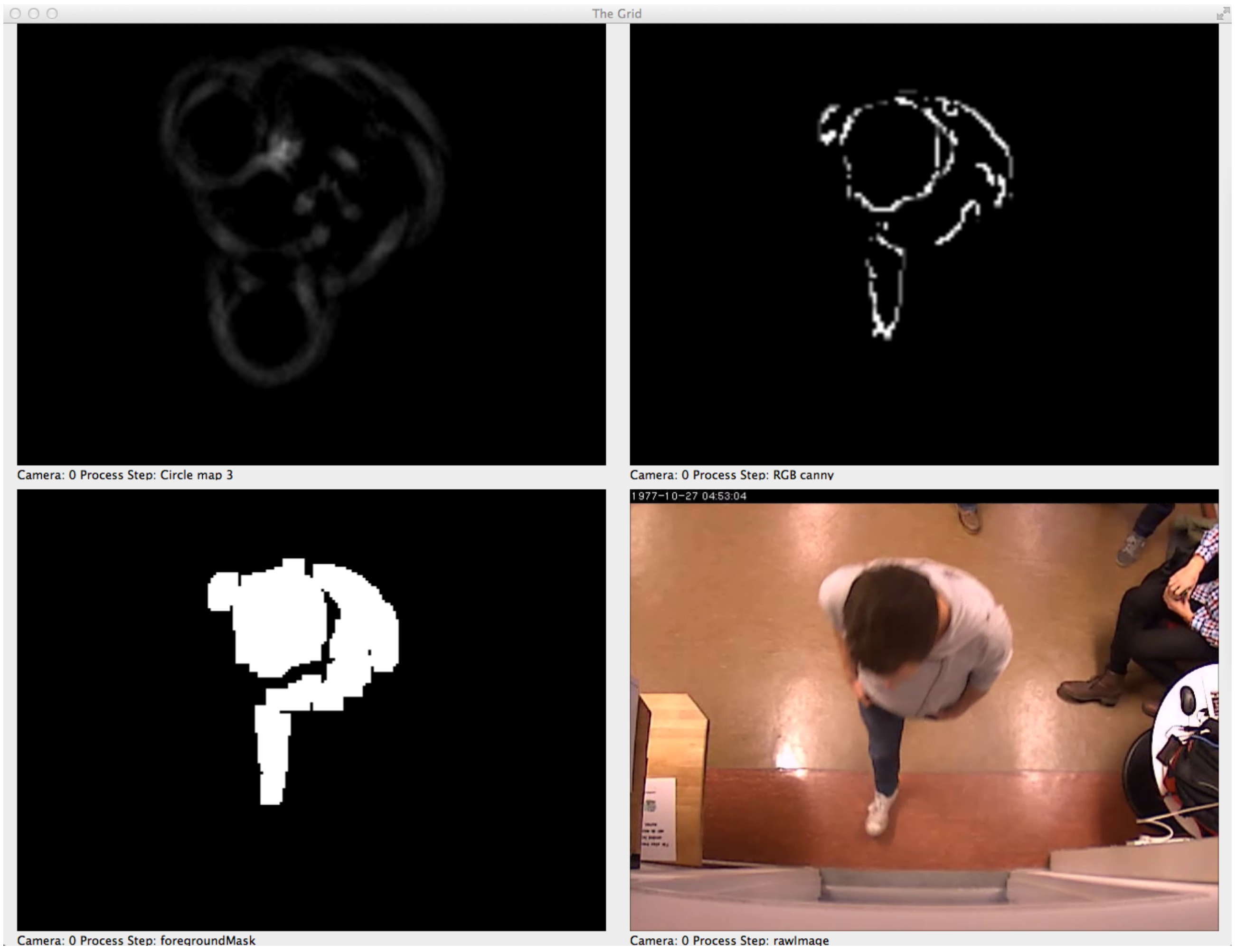
The Grid
Camera: 0 Process Step: Circle map 3
Camera: 0 Process Step: RGB canny
1977-10-27 04:53:04
Camera: 0 Process Step: foregroundMask
Camera: 0 Process Step: rawImage

Figure B.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.

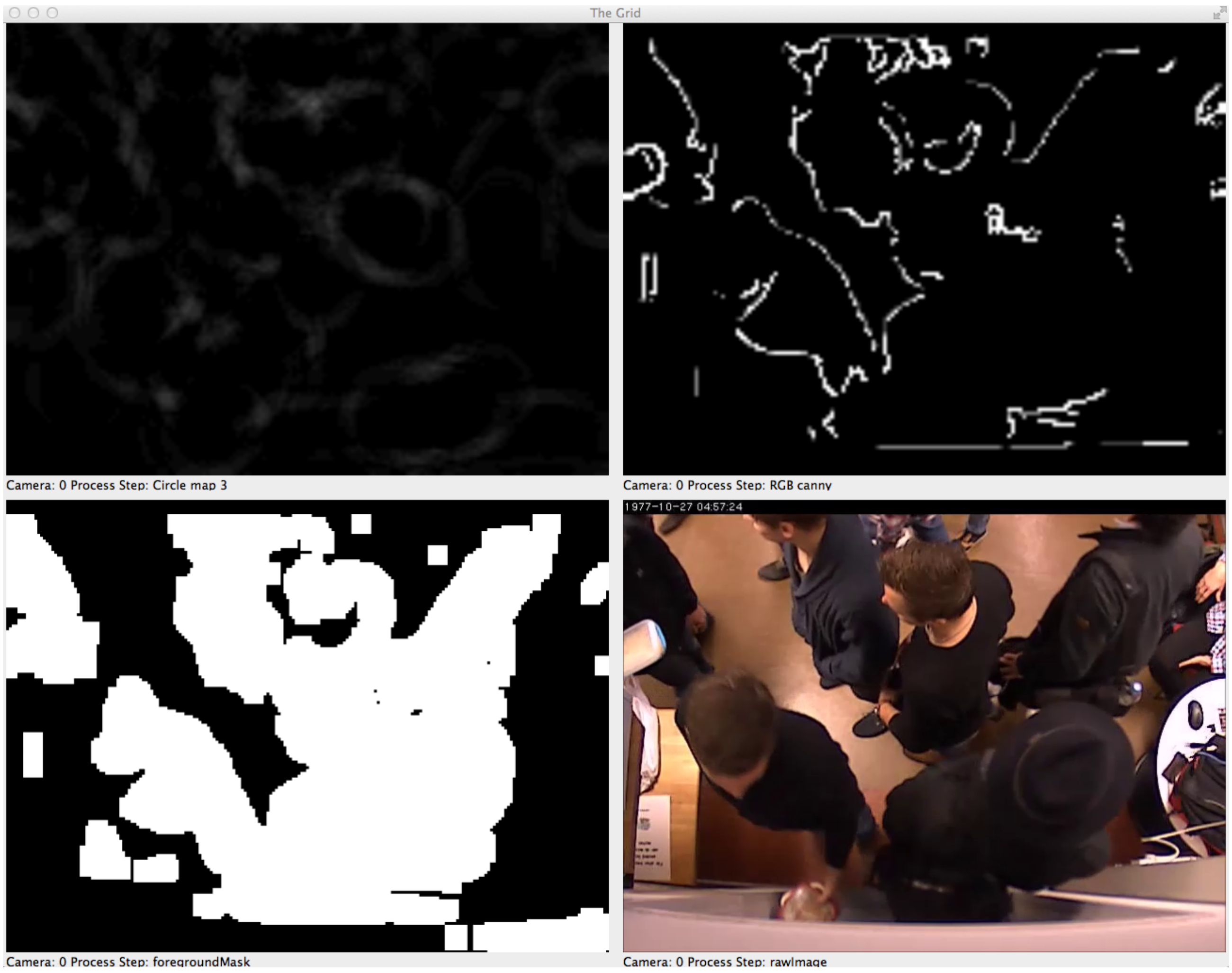
The Grid
Camera: 0 Process Step: Circle map 3
Camera: 0 Process Step: RGB canny
Camera: 0 Process Step: foregroundMask
1977-10-27 04:57:24
Camera: 0 Process Step: rawImage

Figure B.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.

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

EXAMPLE REFERENCES ONLY, REMOVE BEFORE HANDING IN

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

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

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

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