Queue Assessment via Images

Simon Bang Jacobsen

Kongens Lyngby 2012
IMM-M.Sc.-2012-106

Technical University of Denmark
Informatics and Mathematical Modeling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-M.Sc.-2012-106

Summary (English)

The goal of the thesis is to evaluate queue using computer vision and mathematical modeling. As a part of this project a queue assessment system has been developed to estimate the waiting time of a queue. In this development process studies of other queue assessment systems have been done, and we have furthermore been working on different approaches to do queue time estimation. Among these approaches people tracking techniques has been considered as a basis for queue assessment. The final solution is a system based on foreground segmentation and optical flow. The system has been tested in the main canteen at DTU and shows in tests to give a good estimate of the waiting times.

Summary (Danish)

Målet for denne afhandling er at vurdere kø via computer vision og matematisk modellering. Der er som en del af projektet blevet udviklet et kø vurderings system til at kunne estimere ventetiden på en kø. Som en del af denne udviklingsprocess er der blevet lavet et studie af andre kø vurderings systemer, derudover er der arbejdet på flere forskellige metoder til at vurdere kø. Der er blandt andet benyttet forskellige simple menneske sporings teknikker som grundlag til kø-vurderingen. Den endeligt udviklede løsning er et system der bygger på forgrunds segmentering og optical flow. Den endelige løsning er blevet testet på en kø i DTU's kantine og har vist i test at give nogle acceptable estimater af ventetiden.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modeling at the Technical University of Denmark in fulfillment of the requirements for acquiring an M.Sc. in Computer Science and Engineering.

The project was completed in the period between the end of January 2012 to the end of August 2012 under the supervision of Henrik Aanæs and Anders Lindbjerg Dahl.

The reader of this thesis is expected to have a basic understanding of image analysis and linear algebra.

Lyngby, 30-August-2012

Simon Bang Jacobsen

Acknowledgements

I would like to thank my supervisors:
Henrik Aanæs and Anders Lindbjerg Dahl

The Participants of our small vision group containing:
Mikkel Damgaard Olsen
Poul Kjeldager Sørensen
Ricard Moya Mata
Jannik Boll Nielsen
Kasper Skårhøj
Jeppe Walther
Søren Vinther Poulsen

The people in building 321, room 133 with a special thanks to
Tobias Lindø
Mircea Rohat
Camilla Falk Jensen

And thanks for the support from:
Mads Ingerslew Ingwar
Søren Rosdahl
Jennifer Jørgensen
Rasmus Viggers
Stefan Lyck
Christian Bang Jacobsen

Contents

Chapter 1

Introduction

Monitoring a waiting line or queue can be widely used to aid event organizers and many others. It can be used to get information on queue peakhours and the waiting time customers and guests experience.
In this project we will look at how computer vision and mathematical modelling can be used to provide information on waiting time of a queue.
Estimating the waiting time of a queue is not a trivial task, and several studies have been done to develop queue assessment systems. In this project we will describe some different approaches to do waiting time estimation. Furthermore we will go through our process of developing a novel approach to waiting time estimation. In the process of developing a system to do waiting time estimation, an initial tracking approach was tried. As the tracking approach turned out to be too ambitious with the time available, a simpler method was developed based on the experienced gained from working with the tracking approach. A critical evaluation of the final derived method concludes this report.

Chapter 2

Motivation

When considering an amusement park or a big event as a festival, it can be very difficult to assess the load on the different facilities. When a facility is overloaded it creates queue and waiting time for the guests. While people wait for a facility it can create frustration and overall lower the satisfaction and the experience.
For some facilities additional crew can minimize the queue in peak hours. For other facilities it might be needed to extend the capacity of the facility to be able to handle the amount of people. Essential for being able to solve the problem, information of the queue needs to be collected such that the correct solution can be found to minimize the queues.
In this project the focus will be on developing a system to automatically estimate the waiting time of a queue over time. Automatic queue assessment can provide a good overview of queues and ease the task of comparing different facilities over time.

Chapter 3

Goal

To assess a queue is not an easy task when challenges such as a changing environment, changing light condition, camera perspective and difference in peoples appearance has to be considered.
The goal of this project is to be able to develop a queue assessment system that is able to estimate the waiting time at different points in time. Several other projects have been working with how to estimate the length and the waiting time of a queue. In this project some of these methods will be considered to get an overview of the possibilities when developing a queue assessment system. The result of queue estimation is often classified into groups such as *large*, *medium* and *small*. One goal of this project is instead to give an estimate on the waiting time of the queue as it can be more precise.
When estimating a queue an obstacle can be that all people in the waiting area of the queue are not always a part of the queue. This is a problem that can be difficult to solve and the problem is often avoided when estimating a queue. In this project we will look into how to filter people who are not part of the queue.

Chapter 4

Other Queue Assessment Systems

Many different approaches to get a good estimate of queue size and the waiting time have been made.
In this chapter we will go through a part of the work, as a part of this we will look at some of the different approaches used, under what conditions they were operating and how well they worked.

4.1 Roskilde Festival Queue Projects

Among other projects done on queue size estimation two minor projects has been done at DTU in cooperation with Roskilde Festival 2010 and 2011. The first project named *Crowd Monitoring - Estimation of queue length* was developed to classify the queue size of an ATM as can be seen in figure 4.1

The second project named *Audience Tracking* was developed to estimate the queue size to the volunteers' wristbands, which can be seen in figure 4.2

The projects are very similar and the systems deal with problems that are very alike. Both projects make use of still images respectively updated each 30 sec-

Figure 4.1: Picture from the first Roskilde Queue System

ond and each 15 second.
Both systems have the challenges of a camera not ideal positioned and very changing light conditions. The changing light conditions are caused by the systems running day and night. This has an impact as the first system is running outside and the second system is running in a room with large windows.
Both systems uses Mixture of Gaussian(MoG) for background subtraction. This is a very used general pixel-based foreground segmentation method that works very well in changing light conditions and some of the other problems to consider when doing background subtraction(futher described in the theory chapter).
After foreground segmentation is done both method uses different pixel-based validation and classification methods to do classification of the queue size. The first project consider the foreground pixels in a line from the ATM to estimate and classify the status of the queue based on that. In the second project the classification is solely based on the overall number of total foreground pixels in a the waiting area. Even though each of the systems are fairly simple and limited with respect to the size of the projects some ideas can still be used.
The classification done in the two projects is hard to evaluate upon. The developed systems works in many cases, but it seems that the approach used is too simple, which might be caused by the quality of the input data in both systems.

4.2 Observation of Airport Queue Sizes using Surveillance Cameras

Another queue assessment system using still images have been used to estimate the size of queues. The project was titled *Observation of Airport Queue*

Figure 4.2: Picture from the Second Roskilde Queue System

Sizes using Surveillance Cameras done by Martin Jensen in his master thesis 2009 at DTU. It was a project done for Scandinavian Airlines using the input from surveillance cameras. The system works by using a Mixture of Gaussian background model and background subtraction to isolate people in the images similar to the two previous described projects.

Each camera input is setup with a queue area defined and a lot of work have been done to compensate for the cameras perspectives.
To compensate for the camera perspective and the setup calibrated, a calibration object is used. The calibration object is an object of known dimensions, that is placed on the ground in front of the cameras.
After the queue is isolated and the camera perspective have been taken into account, the size of the isolated foreground is taken as input to be able to estimate the size of the queue.
The queue is then categorized into 4 different categories which are *none*, *small*, *medium* and *large*.
To do the classification a linear regression model is used. By adjusting the model to some training data it was possible to achieve an accuracy of 86%. The problem mainly lies when classifying medium and large groups of people. Medium groups could be classified as a large group and a large could be classified as medium size group.
In general the system has a very high level of accuracy and is well suitable in many situation.

Figure 4.3: Crowd Motion Tracking[GM09]

4.3 Crowd Motion Analysis from Video

An interesting article that look at tracking people in a crowded environment is the "crowd motion analysis from video"[GM09] made by Gayathri Mahalingam, Chandra Kambhamettu and Benigno Aguirre.
It uses a simple method to track people done by using Minimum Mean Square Error (MMSE). Two subsequent frames are compared and it is then the goal to find best matching regions of the two images. The tracking is manually initialized and the tracking is done by finding the best matchings between the two frames. The best matches are found by using the Mean Square Error which is given by:

$$MSE = \frac{1}{MN} \sum_{M}^{y=1} \sum_{N}^{x=1} [I(x,y) - I'(x,y))]^2 \tag{4.1}$$

Where M and N are the dimensions of the frames and I and I' are respectively the current and the following frame.
The values gotten by I(x,y)(and I'(x,y)) is the RGB-value which is used for comparison. When looking for the best match only a small region of interest(ROI) is searched in I' which improves the speed of the tracking and takes peoples movement speed into account.
A picture of the tracking can be seen in figure 4.3.

The algorithm fails when occlusions occur and a person needs to be manually found again. By using MMSE the algorithm can automatically notice when the

error is getting to high and invoke the manually tracking whenever needed. The tracking done in this solution works quite well and it manages to look at:

- the number of people passing by
- average number of people
- number of people per minute
- and average walking speed.

The solution however is not complete in the way that initialisation of people have to be done manually. Some work also needs to be done to solve the problem when occlusions occurs, which is not a trivial task. A benefit of tracking people is that the waiting time of each individual can be gathered which gives the potential of making a very accurate waiting time estimation.

4.4 Design and Validation of a System for People Queue Statistics Estimation

A very good queue assessment system[VP11] has been developed by Vasu Parameswaran, Viney Shet, Visvanathan Ramesh. This system is developed and tested on queues at an airport. The system is divided into two main components:

- Queue Module
- Counter Module

The task for the queue module is to be able to estimate the number of people waiting in a defined waiting area. The task for the counter module is to be able to estimate the average service time per person at the counters and also to be able to estimate the number of open service counters. A picture showing the division can be seen in picture 4.4.

In the development experiments with two different camera angles are done. A high ceiling top-down camera and low ceiling oblique camera. This is illustrated in figure 4.5. The system is built to be robust towards changes in light and people's variation in appearance which include that people can carry bags.

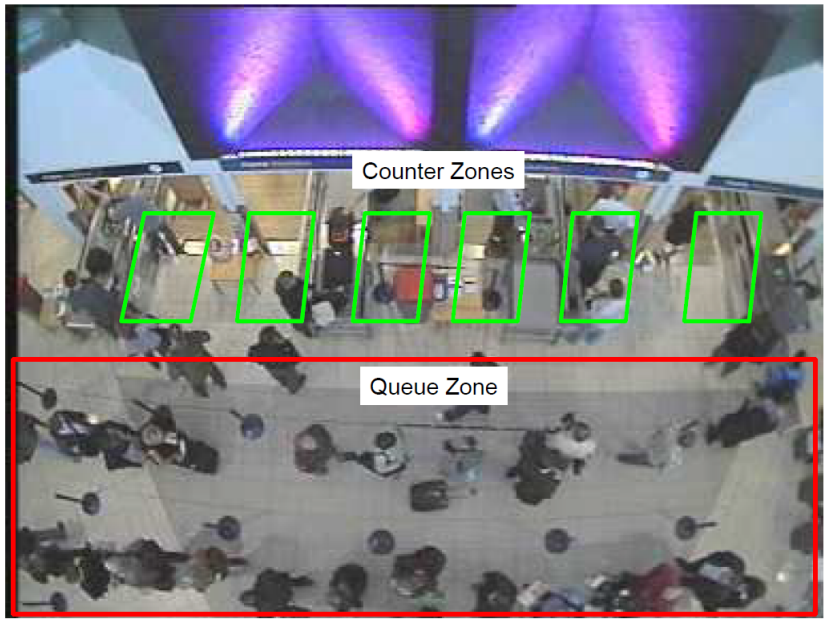

Figure 4.4: The counter and queue area.[VP11]

(a) (b)

Figure 4.5: High and Low ceiling mounted cameras positions.[VP11]

The basic idea behind the system is combining the two modules to be able to calculate the current waiting time. This is done by multiplying the average service time with the number of people waiting.
The Queue Module works by using a weighted area function, weighted to the size of people in the images according to their position. The queue size module is in general based on the following formula where C is the number of people(crowd count), H is the height and W is the width of the region.

$$C = \sum_{i=1}^{H} \sum_{j=1}^{W} \Theta(i) B(i,j) \tag{4.2}$$

They express C as a weighted area of B where B(x,y) is the isolated foreground and having Θ being the weight function. The accuracy of the crowd count estimation is found to be very depended on the position of the cameras. It has better results on a high ceiling mounted camera where less occlusion exists as seen in figure 4.5. To compensate for the problem with occlusion some probability measures is used to improve the estimation. To do the estimation the system is quasi-calibrated to work properly. As an example they define a minimum size of a person, which is defined by the minimum found head size.
The Counter Module's task is to find the service time for each person at the counters. It is based on detecting and localizing people in the defined counter areas. The input used for the detection and localization of people is similarly to the queue module quasi-calibration. The Counter Module also uses the isolated foreground, which is gained by using foreground segmentation.
The foreground image is then used to estimate the location of people based on the shape and size of the silhouettes. By using shape-matching some fairly good results is obtained when using high-ceiling cameras, but they experience that when changing to low ceiling cameras, a probability model is needed to give a good result. An image illustrating input and output of the localization and detection can be seen in figure 4.6.

To train the probability model a 3D model of the queue area and a simulation of people arriving and waiting is used.

The system achieves a high level of accuracy, which is caused by a good estimate on the number of people in the waiting area, and a fairly accurate service time estimation. The reason for the largest error occur when the state of a counter(if it is open or closed) is estimated wrong. The accuracy of the system when considering estimated waiting time has an average absolute error of 0.8736 minutes in the tests performed.

Figure 4.6: Input and Output of the people detection algorithm.[VP11]

4.5 Resume

After having looked at some of the interesting queue assessment projects, it has been quite clear that a lot of different approaches can be used to solve the problem of doing queue assessment. For queue assessment in general some of the projects that are very close connected and can be used, is systems that detect and localize people in images also used by the project of Parameswaran et al.. This involves shape matching, probability based methods and learning based methods. It has been chosen not to go into this area of research in this project. The systems considered in this chapter can overall be placed into two categories. The ones based on images and the ones based on video as input.
For the systems based on images we have the two Roskilde systems and the system done for SAS. In these systems they have several similarities. Among the similarities are that they both use background subtraction and that the result of the systems is a classification of the queue.
For the two systems using video as input more precise measures is attempted, such as the estimated waiting time and the number of people waiting. It has been noticed in all the projects that have been found during research, that one of the biggest problems when developing a queue assessment system is occlusion occurring when people are standing close.
When using images even with a relative large interval it is possible to give a quite good classification of the queue size which have been shown by the two Roskilde project and especially the project done for SAS. A classification is in many cases a sufficient output of a queue assessment system.
Considering the systems based on video input it has been seen that tracking people is an advanced task, but when tracking is made a very precise queue assessment can be given.
For all the systems developed for solving this problem they required either to

have a queue area defined or to be quasi-calibrated before being able to estimate the queue. For all the system developed specially the system developed by Parameswaran et al. it was noticed that the camera perspective have a very large impact on the necessary complexity of the system. It is in some queue area not possible to position the camera in a good angle and therefore most of the system compensate by calibrating their system based on the camera perspective.

Chapter 5

Theory

In this chapter the theory behind the developed solution will be described. As a basis for developing a computer vision algorithm to be used to do queue assessment, we looks into the a bit of theory on how people waits in queue. Then we will look into the more technical aspects which is how blob-detection works, the problems with foreground segmentation and then the Mixture of Gaussian foreground segmentation method is briefly described. Finally optical flow and the basics farneback's optical flow are described.

5.1 The concept of waiting in queue

When people are waiting there are some sociological informal patterns that people follow, these patterns are informally defined by factors such as culture and types of queue. One of the psychological theories interesting is the theory of personal space, which was first described by Edward T. Hall.[Hal66] The theory of personal space describe the fact that people have a bit of distance between each other. It is hard to specify the exact distance because it depends on the informal definition and it is dependent on the specific location of the queue and the people waiting. Most of us have experienced being in a crowded environment like a train, where we are standing a lot closer than we normally would.

This can also be the case when waiting in line. The location of a queue does also influence on the way people are standing[Man69]. Another pattern is that people wait one behind the other in a first come first serve manner, this is for most of us obvious, but that is not the pattern for all cultures and for all queues. This structure is only the case for a queue where everyone are priorities equally. In this project we will only look at a queue structure where people are prioritiest equally and jumpers[Man69] is not considered.

5.2 Blob Detection

Blob detection is the procedure of finding regions in an image that are connected, which in our case is color wise. Blobs is also known as connected components and detection of those can be done in many ways. A fairly simple approach is to do connected-component labeling. This is done by initially assigning each pixel with an unique label and the labels are then propagated and re-labeling is done, such that each pixel in each component end up being the same and that each region will finally have a unique label.
Blob detection is normally used on binary images, but can easily be used on other type of images such as grayscale or colored images. It only needs a clear definition of which pixels belonging to the blobs, which can be done by defining a threshold. An early approach to do component-labeling was developed by [RP66], the algorithm runs in two passes. The first pass goes sequential from top to bottom and at each row going left to right. The first run passes each one of the pixels and each time it encounters a component pixel, it labels it either with a new label or an existing label dependent on its neighbors.
In most component-labeling algorithm either 4-connectivity or 8-connectivity is used which is illustrated in figure 5.1.

Figure 5.1: Different connectivity used in blob detection algorithms

In the first pass of [RP66] the current pixel label is decided from the labels of

the neighbors 1,2,3 and 8(depicted for the 8-connectivity in figure 5.1). Whenever two different labels are registered to be labeled to the same component it is stored in an array. The second pass of the algorithm then goes through all labels and re-labels such that each pixel in belonging to a component has the same label.

Another connected component labeling method that improves the space complexity by removing the arrays containing the connection between the different labels used by [RP66] was developed by [RL83]. It works by initially giving each component pixel an unique label and then runs a pass through each row from top to bottom at each pixel where it looks at its 4-connected neighbors and where it finds the lowest label. In the end of each row it runs back and gives all component connected pixel the same label. At second pass it goes from bottom to top using the same approach(left-to-right and back) as the first pass, which secures that all the pixels have the same label for the same component[FCL03]. This approach do not work on complex regions such as regions connected diagonally.
In this project we use a hybrid between the two such that every pixels is initialized with an unique label and the lowest label is then propagated to all connecting pixels using 8-connectivity.

5.3 Foreground segmentation

Implementing foreground segmentation when having static cameras, is often done using background subtraction. The basic idea behind background subtraction is to have a model(which can be an image) of the background and by looking at the difference between the current image and the background model we find the foreground. If we have a good model of the background we will get a good foreground segmentation.
Although background subtraction often can be used and give good results it contains several problems and these are not trivial to solve. Most of the problems appear because the background cannot be assumed to be static in a real world scenario. The problem with pixel-based background subtraction has been described by [KT99] and categorized as following:

- **Moved objects**, objects considered to be part of the background is moved.
- **Time of day**, light conditions can change gradually during the day.

- **Light switch**, light conditions can change suddenly.
- **Waving trees**, a moving object such as a waving tree can be part of the background.
- **Camouflage**, a foreground pixel can blend into the background model and therefore not become detected.
- **Bootstrapping**, an image where only background is present might not be existing in some environments.
- **Foreground aperture**, when an object of the same color moves and some of the object is considered background it might not be detected.
- **Sleeping person**, A foreground object that becomes motionless, cannot be distinguished from a background object moving and becoming motionless.
- **Waking person**, A foreground object can become background after being motionless when it then starts moving, both the object and the background can become considered foreground.
- **Shadows**, foreground object often cast a shadow on the background, which is not considered a part of the foreground eventhough it gives changes to the background.

Several background subtraction methods have been developed to deal with some of these issues. Furthermore fine-tuning different background subtraction methods depending on the specific background can improve the foreground segmentation. It is in general hard to get a perfect background subtraction method that works perfect for all situation without finetuning, because it is very depended on the background and the speed of the moving foreground objects. It is most of the time a balance between not making background elements a part of the foreground and not making the foreground a part of the background. This can be a difficult problem to solve and an important point to be aware of is that some problems such as *camouflage* cannot be solved when using a pixel-based method[NF97].
For this project it is important that the update rate of the background model is not too fast to avoid people becoming a part of the background when waiting in queue. One of the very recognised methods of background subtraction is the Gaussian Mixture Model(GMM) also referred to as Mixture of Gaussian(MoG)[NF97] which is used in this project and the basics of the method is described in the next section.

5.4 Gaussian Mixture Model

A very used method to do background subtraction is the Gaussian Mixture Model(GMM)[NF97]. It is an unsupervised technique that uses a classification model on the pixel-level to build a background model. It works by building a model were only recent pixels are considered, which compared with the average of all images is a much better approach. When building the background model the more recent a pixel appears the more relevant it is. GMM classifies each pixels before updating the background model and based on likelihood it updates the matching model. GMM works generally with three classifications: foreground, shadow and background. When updating the background model each pixel is categorized and placed in a group of which the new pixels matches. It is assumed when doing the categorisation of the pixels that the background have the most evidence and the least variance. The normal number of groups for each pixel is between 3-6.
When avoiding the direct image-averaging approach, this can solve some problems but the method still suffers from slowly moving objects (if the update rate is too high) and camouflage, which cannot be solved by using a pixel-based approach.

5.5 Optical Flow

An optical flow method is used to find the motion within a sequence of frames. Optical flow is here the 2D representation of the 3D motion and from a 2D video sequence this is the only one available. It is difficult to approximate the right representation of a 3D motion from a 2D model video. Currently no perfect solution for finding the optical flow has been found although several methods have been developed. In the development of optical flow, algorithms as with most computer vision algorithms there is a trade-off between accuracy and speed. Therefore the increase in computer power increase the possibility of accuracy in real-time algorithms. To avoid going through all the different optical flow methods we make use of already made optical flow performance review made by Barron et al.[JB92] and have furthermore looked into a new rather approach made by Gunnar farneback[Far03] which we ended up using and which is described in the following section.

5.6 Farneback

Gunnar Farnebacks optical flow method was originally developed for the WITAS project, whose aim was to develop an autonomous flying car. Optical flow algorithm's is in general a tradeoff between accuracy and speed and Farnebacks new approach was able both to have a fairly good speed and a high accuracy. In Farneback's phd. thesis where the optical flow is described, two very similar methods are described. The first approach look at the temperallyspatial domain where several successive frames is considered. The second approach described and used in this project look at two successive frames. This is done to avoid existence of noise caused by either the camera moving or if the movement captured is not continuous. In this section we will give a description of the basics of farneback's optical flow method based on the article[Far03] describing the method and which the implementation of the method used is also based upon. The overall approach uses polynomial expansion to build models describing each of the two subsequent frames, thereafter displacement estimation are performed between the models. We hereby result in an optical flow field, which is a matrix describing the displacements.

Polynomial expansion

Polynomial expansion is used to estimate a small area to be used for comparision between two frames. Polynomial expansion provide a local signal model of the neighborhood of each pixel by the formula:

$$f_1(x) \sim x^T A_1 x + b_1^T x + c_1 \tag{5.1}$$

Where A is a symmetric matrix, b is a vector and c is a scalar and where the coefficients are estimated from a weighted least squares fit of the neighborhood.

Displacement estimation

When estimating the displacement it is based on the polynomial expansion formula so the displacement symbolized as d, can describe the displacement in a perfect translation by the formula:

$$f_2(x) = f_1(x-d) \sim (x-d)^T A_1 (x-d) + b_1^T (x-d) + c_1 \tag{5.2}$$

$$= x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1 \tag{5.3}$$

Then we can replace the following:

$$A_2 = A_1 \tag{5.4}$$

$$b_2 = b_1 - 2A_1 d \tag{5.5}$$

$$c2 = d^T A_1 d - b_1^T d + c_1 \tag{5.6}$$

This leads to:

$$f_2(x) = x^T A_2 x + b2^T x + c_2 \tag{5.7}$$

From 5.5 we can isolate d assumed that A_1 is non-singular, we get:

$$2A_1 d = -(b_2 - b_1) \tag{5.8}$$

$$d = -\frac{1}{2} A_1^{-1}(b_2 - b_1) \tag{5.9}$$

When estimating by using a single polynomial for each of the frames a single translation is quite unrealistic. After finding the coefficients A1(x), b1(x), c1(x) from frame one and A2(x), b2(x), c2(x) from frame two. $A_1 \neq A_2$ and A(x) is therefore estimated by:

$$A(x) = \frac{A_1(x) + A_2(x)}{2} \tag{5.10}$$

Another thing introduced to estimate the coefficients is:

$$\Delta b(x) = -\frac{1}{2}(b_2(x) - b_1(x)) \tag{5.11}$$

Instead of having a global polynomial over the whole frame, we have local polynomial approximation to be:

$$A(x)d(x) = \Delta b(x) \tag{5.12}$$

d(x) is introduced because of the local displacement estimation. From the equation 5.12 we are able to do a pointwise estimation but as it was discovered to be too noisy, this is instead done over a neighborhood. A movement is in general also most often several pixels moving in the same direction and not separate pixels moving, such that theory reflects practice.

Chapter 6

Queue and test environment

To collect data to be used for developing and testing the system two cameras was placed in the main canteen at DTU. In the main canteen people are waiting in up till four queues at peak hours and a lot of people are having lunch here each day. The queue here is a one-way queue where most people enter from the right and exists at the left. When one or more queues exists they are positioned in fairly straight lines. In the same area as the queues bread and other types of food is available and a specific queue area cannot be strictly defined. To capture the queue two fisheye cameras (Levelone FCS-3091) was used. The gathered data was collected from Monday to Friday between 11am and 2pm. The reason for the short period of time per day was because that it is the only time interval where queue exists where the peaks mainly appears between 12pm to 14pm, so it was decided to capture around these hours. By only having a few hours and by emptying the cameras each day it was possible to capture the queue in a high resolution 1600x1200 and with a frame rate of 15 frames per second which was the highest quality possible using the type of camera. By using the fish-eye camera we were able to have a good overview of the area, but some constraints to the position of the cameras existed as they could only be mounted by using the vents. In figure 6.1 a sketch of the queue area and the position of the cameras can be seen.

Furthermore the view of the two cameras can be seen in the picture 6.2.

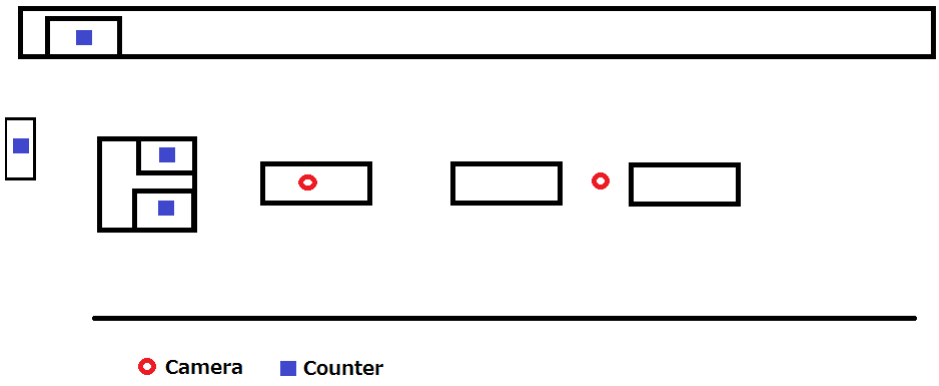

Figure 6.1: Sketch of the canteen area

Figure 6.2: The two cameras viewpoints

When positioning the camera the idea was to be able to follow people and use the input of both the cameras, but as it turned out that the distance between the two cameras where to large it was chosen only to use the data collected by the camera closes to the service counters.

Chapter 7

Initial Tracking Approach

It was decided to look into the idea of tracking people as it seemed to be a good approach to solve the problem of estimating the waiting time of a queue. This would also make it possible to look into a problem that have not been considered in any of the previous systems found during research, which is that many queues do not only have people waiting in the queue, but also people passing by within the same area.
It was noticed during research that people detection is a complex task and is therefore not considered a realistic goal of this project and will therefore be avoided. Instead we will take the approach of tracking different generated blobs. To be able to solve the problem of avoiding people who are not part of the queue, we consider tracking to be necessary. Therefore our initial tracking approach was to track the individual people or group of people and hereby be able to classify if they are waiting in the queue. Thereafter we can look at their movement pattern to be able to classify which people are part of the queue. We divided the task of tracking people into two tasks: Initialization of people and further tracking of people. Within these two tasks we experimented with two different approaches to each of these two tasks. When using tracking to estimate the queue size it was decided to use it on historical data. In the following section we go through the two approaches of the two different tasks and look at some of the problems that caused the two approaches not to work properly.

7.1 Initialization and Tracking using MoG and blob-detection

This approach was the first approach developed. it is a naive approach but an easy way to be able to see the problems occurring. It works by using MoG to built a background model over time and then uses background subtraction. By using a simple blob-detection algorithm, we find all the foreground blobs which is expected to be people. This algorithm works in four steps:

- Background Subtraction
- Blob-detection
- Blob-tracking
- Waiting time estimation

In picture 7.1, the queue after subtraction with the background model is shown.

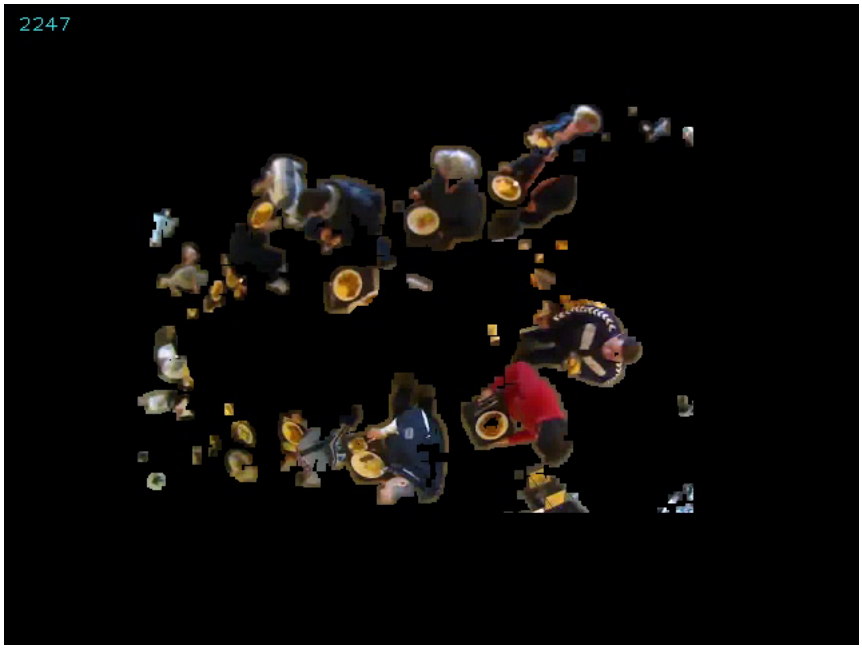

Figure 7.1: MoG used to isolate the foreground

When using MoG the challenge is to set the parameters correctly both according to the video frame rate and the background changes, such that changes in the background get adapted into the background model, but also such that the foreground does not become a part of the background model. To get the parameters suitable for the environment several experiments was made. Furthermore erosion and dilation was applied after the background subtraction was done, this was done to remove some of the noise appearing.

On top of the foreground segmentation the blob-detection is applied. In figure 7.2 the blob-detection is drawn as an square and the blob-size is written in the center of the blobs.

Figure 7.2: Tracking using the MoG approach

When doing the tracking all blobs where found on each frame. Every time a blob not matching a blob on the previous frame, it was initialized as a new person or group of people. We then stack all the found blobs by looking and similar blobs within the same area, it can hereby be either a person or a group. After implementing the algorithm it was quite surprising to experience that it worked fairly good, which can be seen in picture 7.2. But in tracking where several people and where occlusion occurred frequently the algorithm showed to be unsuitable for the tracking task even though the camera position where fairly good. When using MoG for foreground segmentation a number of errors occur

which is caused by many different things such as, the general reasons mentioned in the theory chapter, but also by dark background and by reflection in the steel table. To compensate for the errors in MoG an specially errors coming from shadows and reflections the small blobs was removed. When using this fairly simple approach a crowded scene could result in the following result seen in picture 7.3.

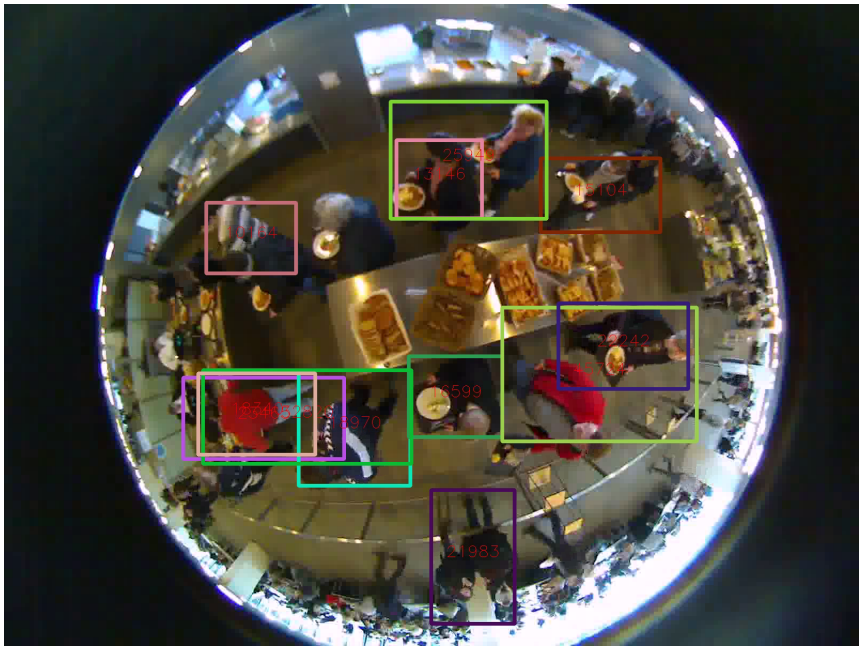

Figure 7.3: Crowded MoG blobs tracking

In the implemented algorithm two general problems where experienced which was when blobs merged and when they split up. This was an expected problem of the approach and it was therefore tried to generate the blobs by using feature tracking which is explained in the following section.

7.2 Initialization and tracking using feature tracking

The idea of using feature tracking for initialization came from the idea that everytime we have a moving person there will be a number of features moving in the same area as the person. By grouping these features we have an estimate

of a persons position. A number of different types features can here be used but it was chosen to use SIFT features as it was easy available in the opencv library. By looking at the moving SIFT-features in the video of the queue it seemed as a good idea as a large number of features was found in the video frames. Which can be seen in picture 7.4 where the found SIFT features is marked with red circles.

Figure 7.4: Sift features found on subsequent frames

General feature tracking works by finding features recognizable by the used feature tracker, descriptors of each feature are then stored, and on the subsequent frame features matching the descriptors is then located. In the tracking approach we filter features not moving and features that have moved a too far distance. By isolating the moving sift features between two subsequent frames can be illustrated as in picture 7.5. Where the lines illustrate the movement.

When grouping these features it was first considered to look at features moving in the same direction, but as this is not always the case when ex. a person is turning it was decided to group all nearby features. The grouping was done very simple by drawing a filled circle around the center of each moving feature. which can be seen in picture 7.6

Figure 7.5: Isolated moving features

The blob-detection is then used on top of these generated blobs, and a minimum size filter is applied to avoid noise. The tracking of the blobs were done very similar to the blob tracking of isolated foreground, the difference being that when people do not move their blob do not appear. It was therefore considered to track specific moving features such that people standing would still be found. The approach eventhough it seemed very promising turned out not to work properly because people change a lot in appearance between being directed towards the camera and directed away from the camera. The change in appearance is a problem when using a ceiling mounted camera.

7.3 Evaluation of the two approaches

When looking at the two approaches we look at how well initialisation works, and how well the tracking of people works. When we look at the algorithm developed it can be evaluated on many different criteria. In this project we have decided to take the tracking process and divide the evaluation into the following parts:

Figure 7.6: Red blobs generated from moving SIFT features

- Initialisation
- Tracking

When evaluating how well the algorithm is at finding people, several things can be considered such as: how *fast* people are found and how *accurate* the algorithm is when finding people. The main concern is here how *accurate* the initialization is. To do this we take a video sequence of a queue which contains many different scenarios and while the algorithm runs through the video we have it save a picture of the area where initializations occur. A perfect initialization looks like seen in picture 7.7

By going manually through the initialized trackings it can easily be seen if an initializations did work well. While performing manual evaluation it was experienced that cases exist where more people where initialized as one as seen in picture 7.8.

Figure 7.7: Perfect initilization

Figure 7.8: More people initialized together.

Figure 7.9: Failed initialization

Other initialization failed such as the one seen in figure 7.9 where and area of the wall where found as a person.

When evaluating the accuracy of initialization we have decided to divide it into three categories: *perfect initialization*, *more as one* and *wrong initialization*. Apart from having these three cases that we can find by looking the initialized pictures, some cases occurs where people never gets initialized. This does however not occur very often. The accuracy of the initialization can be seen in figure 7.10.

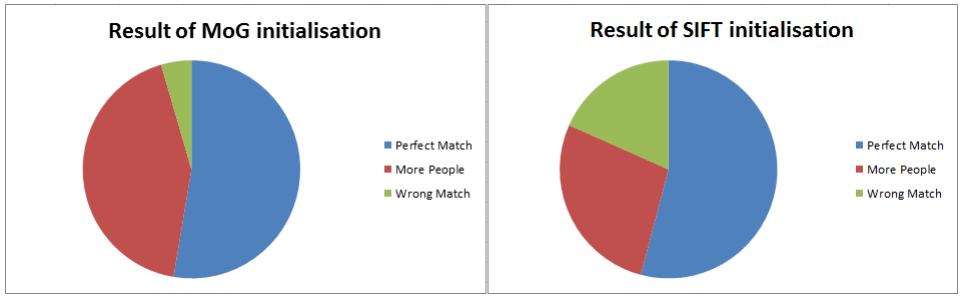

Figure 7.10: Evaluation of the initialization

In the evaluation of the MoG approach 70 perfect matches where found, 57 containing more people and 6 initialization were to be wrong. The wrong initialization where all caused due to reflections coming from metal surfaces. In total 133 initialization was used in the test. When considering the initialization of SIFT, 207 initialization were used in total, 112 Perfect matches were found, 57 with more people and 38 wrong matches. The wrong matches was mainly due to light changes. Both the tests were performed on the same test sequence. The reason that SIFT did more initialisations is that the initialisation is done again when the tracking is lost. A general problem of the two approaches is that the camera perspective is not taken into account.

The tracking of both methods does not work very well. The MoG based approach does not loose tracking as much as SIFT, but both has the problem of not being able to isolate people well enough for a crowded environment. MoG works well for an environment where no occlusion occur, it is thereby too simple for the problem we would like to solve. When developing the two approaches a very large number of variables can be changed, which makes it very hard to find the right setup to be used. For the MoG approach these parameters are:

- Threshold for MoG
- The distance between moving features
- erosion
- how much to do dilation

For the feature approach another set of parameters existed:

- Maximum distance for moving features
- Distance between grouped features
- How often a tracking should be confirmed to exist

For both the approaches the following parameters was available to chance and consider:

- When to initialise track
- When to remove a lost tracking
- Minimum size of valid blobs
- Maximum size of valid blobs

These parameters do have a very large impact on how well the tracking works. The large number of variables does it difficult to optimize the methods, because the running time of the approaches is relatively high(between 1-3 hours for a 8 min. video). The high running time was caused by unoptimized code and by the high resolution used and the number of frames per second.

7.4 Further improvements on the tracking

An approach which can improve the MoG tracking methods of which phd. student at DTU, Mads Ingwar is currently working on is by using the farneback optical flow method on top of MoG to be able to track individual people. He is so far able to separate people from each other based on their movement. If we look at figure 7.11 and 7.12, it also gives a picture that this can be used. Here two people pass by each other and occlusion occur, which is an example where the simple MoG/blob approach fails, but when looking at the flow field of the same situation, we are able to distinct the two people, based on the movement.

Figure 7.11: An example of two people occluding

7.5 Conclusion on the tracking approaches

In the development of these two tracking approaches it was experienced that both of approaches was not suitable as a solution for doing queue assessment. The different approaches had different issues. The tracking of individual people in a non-crowded environment works well when using the MoG/blob approach, but when the scene gets crowded the approach is too simple.

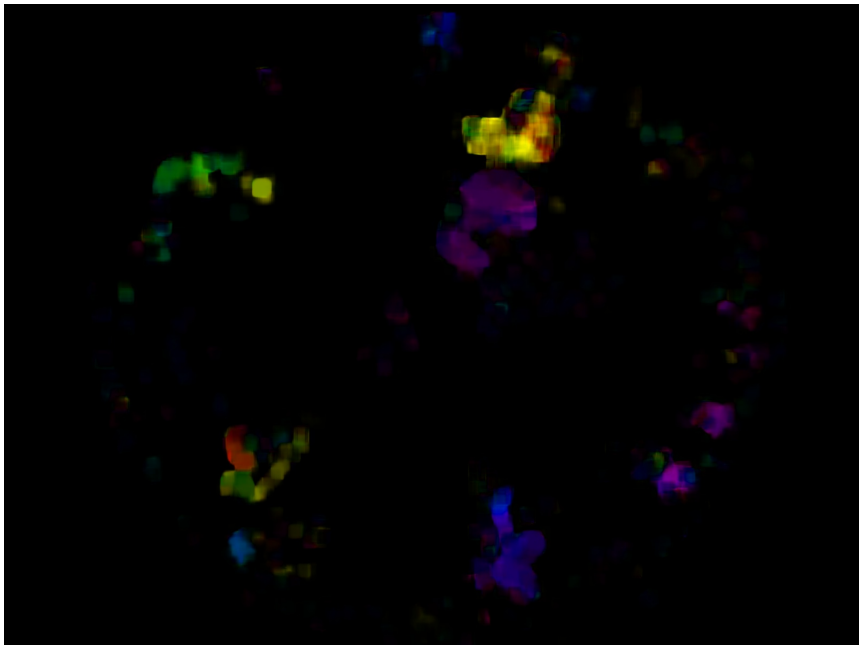

Figure 7.12: Optical flow field of two people occluding

A problem which needs to be solved in the MoG/blob approach. Is the problem occurring when people gets too close and some separation needs to be done to be able to do individual tracking. When looking at the problems of the feature based approach the main problem is due to the camera angle used. By having a ceiling mounted camera directly above the queue it causes the people too change a lot in appearance, which makes feature tracking such as SIFT unsuitable for tracking. We did not experiment with other types of feature tracking, but some improvement could be done if a feature tracker where able to find features that are always visible(such as the top of people's heads), but many feature tracking methods are expected to suffer from the same problem as seen when using SIFT. If we had succeeded in making a good tracking it would have been fairly easy to estimate the waiting time on historical data, but because of the poor tracking results for both the approaches we were not able to get some results that could be used for that. It would have required at least being able to track every tenth person because of the length of the video sequence. Because people tracking in a crowded environment showed to be a too complex task for the two developed approaches, we therefore came up with a different approach which will be described in the following chapter.

Chapter 8

Final Queue Assessment Method

By working with the initial tracking methods, it was found very hard to track people in a crowded environment. Therefore it was chosen to look at a solution where it was not required to isolate and track people, but rather to look at peoples movement. To be able to improve the tracking algorithm the optical flow was considered and the optical flow method is in this final approach used. The overall idea of this method is to follow the speed of peoples movement from when people enter the queue to people exits the queue. We thereby are able to estimate how fast the queue is moving and to estimate the waiting time of the queue.
An assumption when using this approach is that people are waiting when they enter the defined queue area. This is however not strictly correct when only a short queue or no queue is present because it will take a little bit of time to get to the end of the actual queue and start waiting. This difference does not have a big impact on the estimate and we therefore justify that the approach is valid although it can be argued otherwise. The steps of the algorithm can be seen in figure 8.1

The first step of the algorithm is similar to most of the systems considered during research where a specific queue area is isolated.
The isolated queue area is used as the input of both MoG and Farneback's optical flow method. To set the parameters of these methods to be optimal is not a

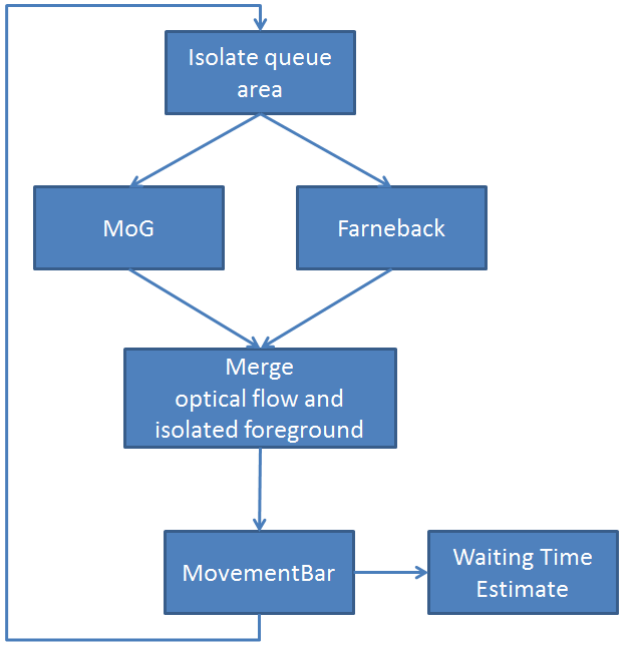

Figure 8.1: Overview of the Algorithm

trivial task. In the SAS system earlier described a large part of the time was used to quantify the accuracy of the foreground segmentation. The parameters chosen for MoG is based on what provided the best results when doing the initial MoG-blob tracking. As a part of the foreground segmentation erosion and dilation is applied to the result of the MoG background subtraction.
The parameters of the Farneback optical flow method were advised by Mads Ingwar and we have not been able to find any parameters that gave better results. After the foreground is isolated and an optical flow field is found these are merged. Such that all flow not part of the foreground is removed. This is done because some noise exists in the output of the optical flow method. This merge is inspired by the work on people tracking by Mads Ingwar. When using only the farneback method some noise is caused by light changes. This is dramatically minimized by merging the isolated foreground and the optical flow. The last step of the algorithm is the main part of the system which is the movementbar algorithm. It works by following the isolated flow in the queue and measuring the time it takes going from the beginning to the end of the queue, which is the output of the algorithm. In the next section the isolation of the queue area will be outlined. Then the details of the movementbar algorithm is described. Thereafter we will go through how the camera perspective is taken into account and finally we evaluate the algorithm.

8.1 Isolating the queue area

The queue area defined can be seen in picture 8.2 (marked with blue). The queue area is isolated and turned such that we get an image as the one seen in figure 8.3. When isolating the queue area it is assumed that the queue area can be defined as a square in the frame. This is done to simplify the isolation of the queue area.

8.2 The Movementbar Algorithm

This approach is based on square areas following the flow. The idea is to use the optical flow field to move the squares in the same speed as the queue underneath the squares. By tracking the time from the square is in the beginning of the queue until the end, we have an estimate of the waiting time of the queue. When using this approach we are not able to estimate expected waiting time. We can only estimate how long people currently at the counters have been waiting in line. In picture 8.4 the approach is shown.

Figure 8.2: Queue area used for testing

Figure 8.3: Isolated queue area used for testing

The first step of the algorithm is to initialize the rectangular areas(from now on refered to as movementbars) in the beginning of the queue area. We then look at the general movement within the frame and whenever movement appears within the movementbar in the expected direction of the queue, we look at the amount of movement and the speed of the movement, based on these factors the movementbar is moved. After a movementbar reaches the counter, the estimated flow of the queue is stored. An important part of this approach is to consider when to initialize the movementbars, the problem is that the estimation of the waiting time starts when the movementbar is introduced. Therefore it was decided to introduce the movementbars whenever movement of a certain amount is registered in the beginning of the queue. As a part of the initialization a limitation was implemented to limit the frequency of initializations.
Another important part of this approach is to be able to move the movementbars with the speed matching the speed of the queue underneath the move-

Figure 8.4: An illustration of the movementbar approach

mentbar. To match the speed the first step is to convert the x and y value of the optical flow field to an angle and magnitude, which makes it easier to filter the movement. This conversion is done using the following equations:

$$angle = \frac{180}{\pi} * \begin{cases} arctan\left(\frac{y}{x}\right) & x > 0 \\ arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ 0 & y = 0, x = 0 \end{cases}$$

After this conversion we can easily filter the flow, such that we only consider flow going directly towards the counter with a variation of 45° in both directions. Which means in the queue considered that only the flow that has an angle between 225° and 315° is used. A good feature of isolating the flow direction is that we are able to filter people moving in the opposite direction and which is not part of the queue. This is also good feature if monitoring a queue where people exits the same place as they enters.

When considering the input to estimate the speed of the queue the size of movementbar is important through testing a good width of the movementbars was found to be 20 pixels. This was further improved when considering the camera perspective which is described in the next section. After the relevant flow is isolated several tests where done to find a good speed estimate, initially it was tried to look at the histogram of isolated flow, and it was here assumed that the most frequent magnitude would be the best estimate. This turned out not to be the case as the queue often moved faster than the most frequent movement, which mainly occurred near the center of the queue and we expect that this is caused by the same problem as the one experienced when trying to use feature tracking. That the appearance of people change a lot near the center of the frame when using a ceiling mounted camera.

Instead of the most frequent magnitude, it was then tried to move the movementbar with the highest found magnitude which gives better results. It introduce some errors if people move their arm, but it has not been experience as a big source of error.

A problem by the movementbar algorithm is that people can appear and exit other places then defined by the initiation and exit of the movementbars. This however can be seen of minor importance as long as we are able to get the wait-

ing time of the majority of people waiting. As an example we have a queue with two entry points and we only monitor one of the queues. The waiting time of people waiting in the single monitored queue would still provide a good estimate of the general waiting time, as the queues merge. Similar if it is assumed that people make a uniform distribution between the open counters, then estimation done on one queue would match the overall waiting time. This is however not an assumption that can always be used. In the following section we look into how the camera perspective have been taken into account.

8.3 Camera Perspective

In the initial tracking approaches we did not consider the different sizes of people seen from the camera perspective. In this approach we will take that into account when considering an area of moving pixel. Several factors can influence the size of people, such as if a person is carrying something or if occlusion occurs and the camera perspective.

To get a good estimate of peoples size change seen from the camera perspective would be to do image warping. But this is not very practical when using a fish-eye camera as the borders get very stretched. Therefore it was decided to look at statistics of the size of people in a set of data. To get pixel-sizes correct a manually drawing on top of people where done which can be seen in picture 8.6, and the blob-detection was used to determine the size and position of people. In the training data we have a sample of 150 people with the location shown in figure 8.5.

In the sample people have been chosen with a good spread over the area and the center of peoples blobs is used to determine their position. In figure 8.5 it can be seen that the bread table unfortunately cause that no people are positioned in the center.

Then the correlation between the value of y-coordinate and x-coordinate position and the blob-size is considered, which can be seen respectively in figure 8.7 and figure 8.8.

The correlation between the y-coordinate and the blob size is here hard to see, which can be caused by several things such as the people used in the test, but also the limited spread of people along the y-axis.

When looking at the correlation between the x-coordinate and the blob-size in figure 8.8, it turns out that the intuitive guess that the closer people are to the

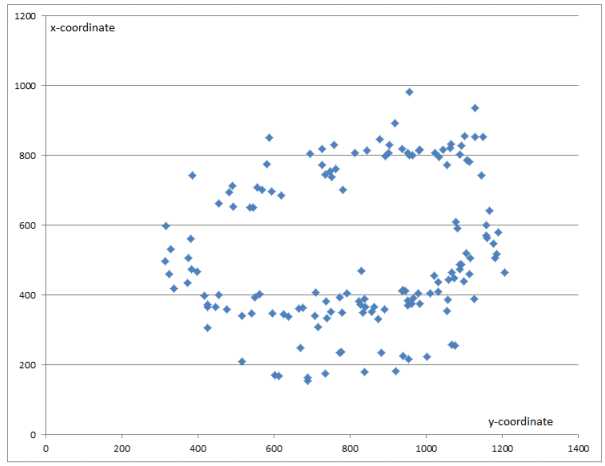

Figure 8.5: Position of people used in the test.

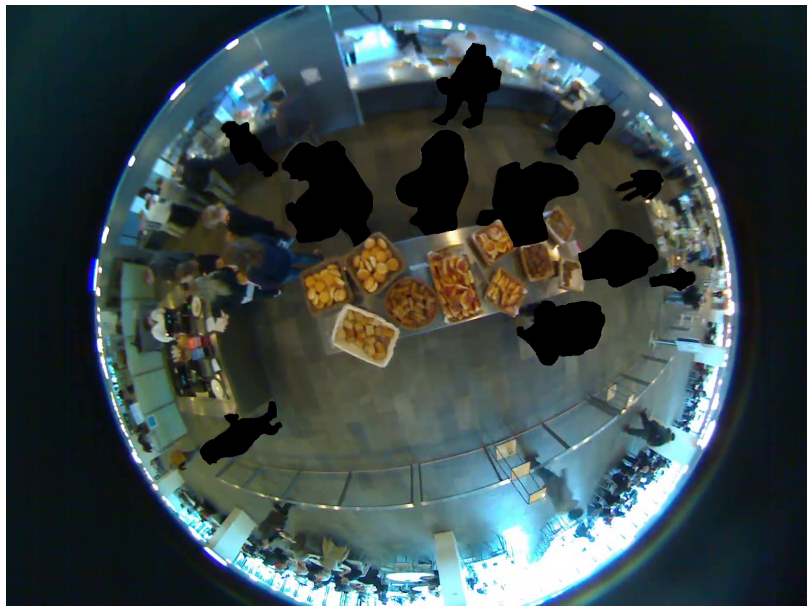

Figure 8.6: Manual drawing to find people sizes

camera the bigger they blob-size become. Is not the case when having a ceiling mounted fish-eye camera, because a larger area of people can be seen when not seen directly above. People then become smaller when getting further away. Caused by the longer distance along the x-axis used by the algorithm and the

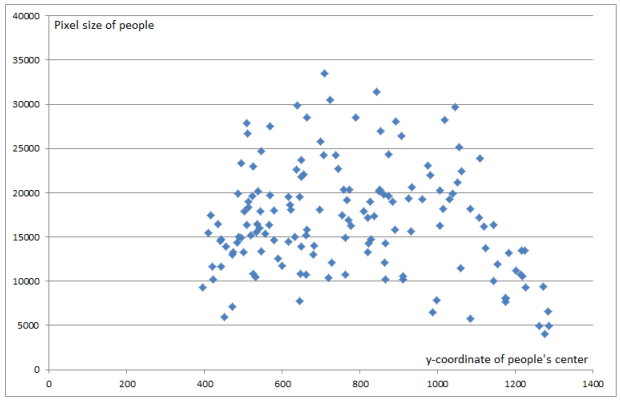

Figure 8.7: y-coordinate correlation with the blob size

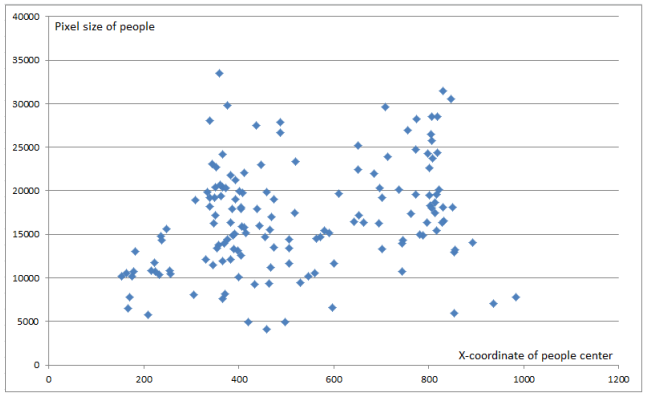

Figure 8.8: x-coordinate correlation to blob-size

rather unclear pattern seen when comparing the y-coordinate and the blob-size. It was decided only to use the correlation with the x-coordinate to generate the function for equalization. The first step when trying to estimate the correlation of people sizes over the x-axis, we divide the people sizes with the average people size. We then use a polynomial regression model to fit the data and generate a polynomial expression to describe the correlation. The approximation of the equation is done by taking vector A containing the x-coordinate values, and vector B containing the relative measure of the blob-sizes and estimating the fitting polynomial such that:
$C = [x^4 x^3 x^2 x 1]$
We thereby have a matrix where each column represent an order in the approximated polynomial. The approximation is then done by calculating Theta:

$Theta = B\backslash C$
This approximated polynomial can then be seen in figure 8.9.

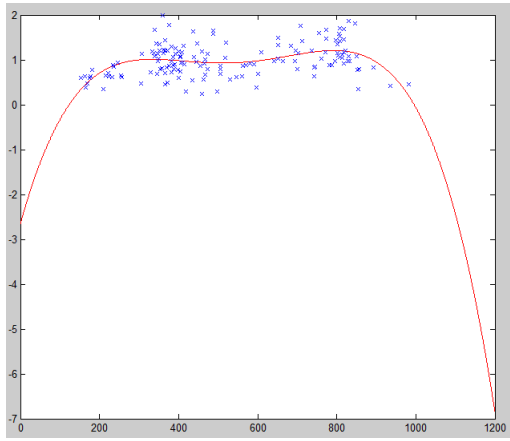

Figure 8.9: Relation between people size and x-coordinate

After the values of the polynomial describing the correlation have been found, this is then used to change the size of the movementbars as they are moved. When implementing this measure it was decided to define a lower bound of the movementbar sizes as it does not work if the movementbars gets too small, so the minimum size was therefore set to 50 percent of the normal size. When resizing the movementbars only the width of the bar is changed.

8.4 Evaluating the Movementbar method

To evaluate the movementbar method we have compared the result of the algorithm with ground truth. Ground truth have been obtained by manually going through the video used for testing. The test sample contains different queue sizes and is in total 28 minutes long.

The result of the test can be seen in figure 8.10.

It can be seen in figure 8.10 that the algorithm developed gives a fairly good estimate of the waiting time of the queue, but it can also be seen that the algorithm in some cases give some high estimates not matching the actual queue. The reason for high peaks is caused by that people enters the queue area and a movementbar is initiated, they then walk away again, and a new person arrives which the movementbar then starts to follow and the peaks hereby occur. To

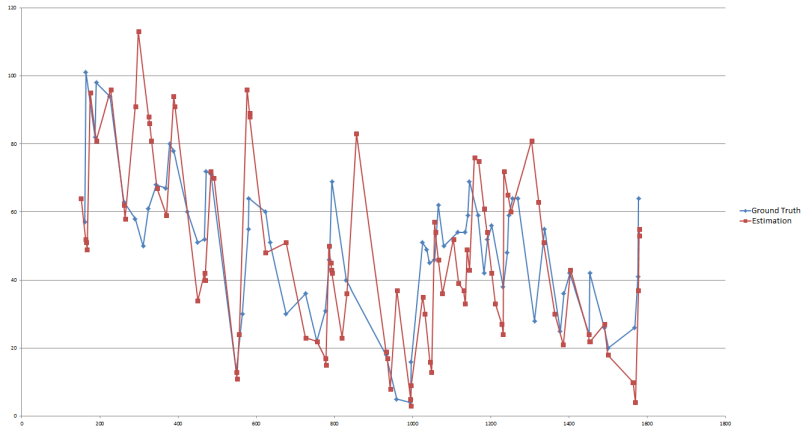

Figure 8.10: Movementbar Algorithm compared with ground truth

solve this problem it has been tried to limit the amount of time a movementbar can exist without moving. This has removed some of the errors, but a problem by this way of solving the problem is that the lower limit should not be lower than the amount of time people are standing still in the queue. Another problem with the algorithm appears when people moves while standing. To solve this it has been considered to look at the amount of movement when moving the movementbars, which might solve the problem. When looking at the difference between the average of the actual waiting time and the estimated waiting time the difference is 2,23 seconds. This does not mean that high accuracy in general has been achieved. A problem when considering the accuracy of the waiting time is that the queue used for testing changes rapidly. Because of the rapid changes small errors in the movement of the movementbars have a large impact on the estimate. For further work it could be interesting to test how well the algorithm works on larger and slower changing queues.
A smart feature of the developed approach is that the distance that people move and the actual movement speed is not considered and that only the movement in the 2D frame is considered. This makes it possible for the system to be moved to another location with few adjustments. When considering the queue used as an test for the movementbar algorithm it is not a very good queue as the waiting time does vary remarkably between each person waiting. Another problem of the queue used is the very limited number of people that can be captured by the camera. When considering the overall result of the algorithm it is able to provide an accuracy that provide a good overview of the waiting time to a facility. For the system to be used further testing needs to be done. It would furthermore be necessary to improve the system to use several cameras as input, if monitoring larger queues.

Chapter 9

Conclusion

When developing a queue assessment system a lot of considerations have to be done. It has during this process been found that modeling the world what seeming ideally is not always the easiest and necessary way to go within the field of computer vision. Some experience has however been found in the process which have ended up in the solution.
In this project it has been achieved to develop an algorithm that can be used to estimate the waiting time of a queue. The goal of separating people not being a part of the queue has not been fully achieved, the solution of looking at the direction of movement was able to filter some people not part of the queue, but other approaches are necessary to be able filter more people.
If we compare the developed approach with the systems found during research, the developed system is able to provide a waiting time estimate compared with a less precise classification. A drawback by the developed approach is that it cannot estimate the current status of the queue, only of how long people have been waiting when having gone through the queue. But to get an overview of the load of a facility the developed system is considered to give some acceptable results.

Bibliography

[Far03] Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. *Scandinavian Conference on Image Analysis*, page 8, 2003.

[FCL03] Chun-Jen Chen Fu Chang and Chi-Jen Lu. A liniar-time component-labeling algorithm using contour tracing technique. *YCVIU 1061*, 1/10/30:15, 2003.

[GM09] Dr. Benigno Aguirre Gayathri Mahalingam, Dr. Chandra Kambhamettu. Crowd motion analysis from video. *Proceedings of 2009 NSF Engineering Research and Innovation Conference, Honolulu, Hawaii*, page 5, 2009.

[Hal66] Edward T. Hall. *The Hidden Dimension.* Anchor Books, 1966.

[JB92] S.S. Beauchemin T.A. Burkitt J.L. Barron, D.J. Fleet. Performance of optical flow techniques. *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference*, 1:7, 1992.

[KT99] Barry Brumitt Brian Meyers Kentaro Toyama, John Krumm. Wallflower: Principles and practice of background maintenance. *International Conference on Computer Vision*, page 7, 1999.

[Man69] Leon Mann. Queue culture: The waiting line as a social system. *American Journal of Sociology*, 75 no 3:16, 1969.

[NF97] Stuart Russell Nir Friedman. Image segmentation in video sequences: A probabilistic approach. *In Proc. of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, Aug. 1-3:13, 1997.

[RL83] O. Zuniga R. Lumia, L. Shapiro. A new connected components algorithm for virtual memory computers. Computer Vision Graphics Image Process, 1983.

[RP66] Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13 Issue 4:24, 1966.

[VP11] Visvanathan Ramesh Vasu Parameswaran, Vinay Shet. Design and validation of a system for people queue statistics estimation. page 19, 2011.