Real-time people counting system using video camera

Roy-Erlend Berg

Masteroppgave
Master i Teknologi - Medieteknikk
30 ECTS
Avdeling for informatikk og medieteknikk
Høgskolen i Gjøvik, 2008

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Real-time people counting system using video camera

Roy-Erlend Berg

2007/05/30

Abstract

In this MTech thesis experiments will be tried out on a people counting system in an effort to enhance the accuracy when separating counting groups of people, and non-human objects. This system features automatic color equalization, adaptive background subtraction, shadow detection algorithm and Kalman tracking. The aim is to develop a reliable and accurate computer vision alternative to sensor or contact based mechanisms. The problem for many computer vision based systems are making good separation between the background and foreground, and teaching the computers what parts make up a scene. We also want to find features to classify the foreground moving objects, an easy task for a human, but a complex task for a computer. Video has been captured with a birds eye view close to one of the entrances at the school about ten meters above the floor. From this video troublesome parts have been selected to test the changes done to the algorithms and program code.

Keywords: Computer vision, background subtraction, shadow removal, feature extraction.

Preface

After two years studying Media Technology at master level at Gjoevik University College (GUC), the grand finale is the making of this thesis. Video processing has been a major part of the master education. And luckily for me, this topic area is of good interest for me, as I choose to immerse myself in elective course for advanced video processing. This started the corporation with, whom has become my supervisor for this thesis work, Prof. Faouzi Alaya Cheikh. I would like to thank Prof. Faouzi Alaya Cheikh for the support and help, and for pushing me to get things done when the motivation and morale has suffered. Because I think it is inevitable that morale drops some times when work is only up-hill and dark sky's. Therefore I would also like to thank my fellow students sharing experience, ideas trough good times and bad times at the master lab at GUC.

Roy-Erlend Berg, 2007/05/30

Contents

List of Figures

1 Introduction

This project is a continuation of the work done by four students at Gjoevik University College in the spring of 2007, "Real-time people counting system using video camera" [1]. The goal of this people counting system was to count the number of people entering and exiting an closed area like a building, museum or shopping mall. This should be accurate and not costing too much. A prototype for this system has been build and tested by LeFloch [1], where a camera was mounted almost vertically from the roof covering an entrance to a building. This system could count people entering and exiting, and count multiple people if they where not walking to close each other.

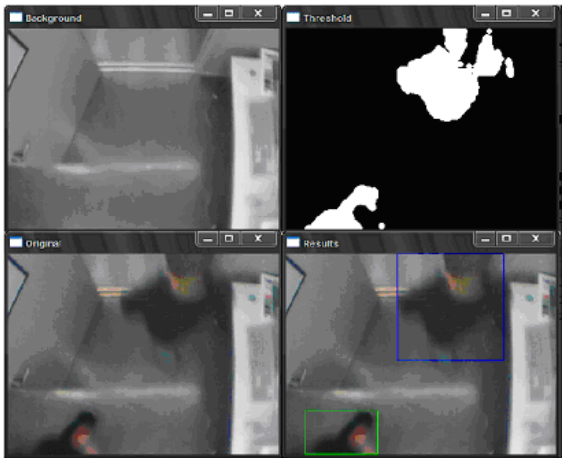

Figure 1: An image form LeFloch's work, with the background, object blobs, input image and bounding boxes

In the figure 1 some of the process is shown. The system identifies a background (upper left in the figure), moving object blobs (upper right), input image (bottom left) and the moving objects bounding box. When two or more people walk too close, the system will have problems counting how many people that is. These problems are described in [1] in a chapter for possible improvements. One of the most difficult problems is people occlusions, where the camera has a limited view of each individual in a group. This will be the main focus of this thesis, to separate each individual from a group. Some work in the area of people flow estimation uses multiple cameras to create 3D models of the scene and it's contents, this is however too slow to be used in real-time due to complex algorithms and is heavily dependent on good calibration of the cameras. Though multiple cameras are the most accurate, a single camera will be used in this project.
Later in this thesis there will be a discussion about some of the related work, and which algorithms has been implemented and tested. the results will be analyzed and discussed before conclusions are made based on the project work.

1.1 Motivation

The motivation for such a system is to gather data about how many persons there is inside an building at any given time. This will help the owners to set up fire extinguishing equipment or the size and placement of fire exits. Building owners are required by law to have enough of this equipment based on how many people that can gather inside. A computer vision counting system have the advantage of not disrupting the flow of traffic like contact based systems might do, and more robust then simple photoelectric cells.
The original project was started as a collaboration between Norwegian Color Research Laboratory and P.i.D Solutions. P.i.D Solution is a supplier of knowledge and equipment for Health, Environment and Safety.
Knowing when and how many costumers are inside a shopping mall could also help to optimize labor scheduling and monitor the effectiveness of promotional events. Optimization of security measures is also a possible benefit from this, knowing how many security guards should be assigned, and hot-spots inside the mall for them to patrol.

1.2 Problem description

The goal of this continuation is to enhance the accuracy of the People counting system introduced in 1. To achieve this, a better separation of groups of people is useful. Also classification of moving non-human objects like a trolley to prevent these from being counted.
Methods for classification of people and objects have been proposed by other groups and teams before, as described in 2. Selected methods will be put to the test to find one that is accurate in classification as well as being able to work in an real-time environment. This limits the complexity of methods for detection, tracking and classification that can be implemented.

1.3 Choice of methods

The experimental work of this thesis had an implementation of the People counting system of LeFloch as a basis. From this basis changes will be made to better suit the new scenarios introduced by using the AXIS ip cameras 207W (small cam) and 223D (large cam), both for real-time and later processing. These cameras was mounted from the ceiling at three different scenarios, later specified in 4. 1st was with the smaller 207W cam at close range, 2nd was the large 223D cam in a place at GUC with approximately 10m distance between the ground and ceiling. The 3rd time a 207W was mounted where the 223D had been, since a permanent fixing of the cam was not desirable by the school. The implementation of the People counting system was available for MATLAB early in the project, and a discussion will follow in 4.1 why MATLAB is chosen for a test bench. From the video, groups of moving objects are be found by separating the foreground from the background. The detected foreground has to be cleaned by removing as much shadow as possible and small objects caused by noise and such. These groups will be merged or split based on some rules about distance, coherent motion and speed to form each individual moving object. The results of changes made to the algorithms was mainly evaluated by the author alone there and then, and some results discussed with the supervisor. This is partially because good statistics about different segmentation methods was hard to find/calculate, and effort saved from involving more people in the evaluation process. The goal was to get as clean segmentation as possible without loosing moving objects in

any frames. The moving objects would later be handled by methods for object classification and tagged as person(s) or non-person, by extracting useful features that could give some semantic meaning.

1.4 Surveillance ethics

Norwegian law do not allow for ground owners to just put up cameras for surveillance where and how they like. In the 1st scenario in the classroom at GUC, students captured in the video file agreed to help with the creation of data for this project. Later in the scenarios 2 and 3 the rules for surveillance had to be checked with the school, and Datatilsynet who can direct and guide those who want to set up cameras, and are set to enforce and monitor those who has surveillance applications. In the overhead position, with the far distance between the camera and the people walking under it this project was ok since people cant be identified, and permission don't need to be gathered from everyone. This was confirmed by a telephone to Datatilsynet and in a booklet on their homepage [2].

1.5 Research questions

For this thesis there will be a three research questions exploring different aspects of this project.
Can we achieve better separation of groups into individuals? Exploring algorithms and methods so that each individual can be detected, tracked and counted.
Can we find features to discriminate people from objects? Good features need to be found and combined to make good decisions about foreground objects.
Can we still maintain real-time operation? Limit the processing cost of this system so that it can run at real-time speed, without the need of some super computer.

2 State of the art

In this section we will try to give an overview of the research done in recent years in the field of counting and classifying people and objects. There has been significant research interest in object classification for automated surveillance applications. The evolution of computing power allow computer-vision algorithms to be more sophisticated in complexity, as there are several steps in the counting and classification tasks that require allot of processing.
Classification systems typically perform either object detection without prior segmentation or object classification after moving objects detection. Systems of the first type is often used when a full surveillance system are not implemented and a specific object type needs to be detected, like a human face [3]. Second type systems are usually an integrated part of a larger system for tracking and detection, like the case of the people counting system [1].

2.1 Foreground/Background segmentation

For computer vision systems, which people counting system is part of, the first challenge is to separate moving objects from the background (foreground and background separation). There are several proposed methods to do this, which can be grouped into two groups: foreground subtraction, background subtraction or directly by modeling either the objects of interest or the background pixel values. In the rest of this section we will discus each of these types of techniques and focus on background subtraction since it is the former technique in this work. In the papers [3] and [4], they compare the current frame with a stored background reference image, which slowly updates (adaptive background).
To get good data for foreground objects to analyze, the paper by Tian et al.[5] has a large focus on good background subtraction (BGS); this is done by modeling each pixel as a mixture of Gaussians and using an on-line approximation to update the model. Other papers also use this method, like Javed et al [6]. The Gaussian distributions of the adaptive mixture model are then evaluated to determine which are most likely to result from a background process. Each pixel is classified based on whether the Gaussian distribution which represents it most effectively is considered part of the background model. Foreground objects can then be traced, and are evaluated when the object is left static. For the focus of this paper is to track objects people leave, like luggage at an airport, that in recent years has become a security concern to many surveillance applications as a response to increased threat from terror groups. The Gaussian mixture method should be more robust to shadows and changes in lighting conditions. Implementation of this method however requires allot of changes to the project work done by LeFloch last year.
In the paper by Schofield et al. [7] a different method of separating foreground and background is used. A RAM based neural networks is used to discriminate between parts of each image that belonged to the background scene and foreground objects. In addition it uses a dynamically adjusted spacing algorithm in an approach to solve occlusion

problems. This papers focus is in segmentation and counting, while tracking and classification issues are not handled. Shio et al. [8] uses a method to improve the background separation by simulating a phenomenon of the human visual system called perceptual grouping. This refers to our visual system ability to extract features of important regions in an image and group them to make a meaningful structure. This algorithm first estimates the motion from consecutive frames and uses this information to help background subtraction. The idea is that even if it is impossible to distinguish the boundaries between two objects at a particular frame, they most likely can be cleanly distinguished in another frame. Motion information from a sequence of images can also provide useful data for detecting boundaries, merging regions belonging to the same object and solving occlusion problems. It was found that using an object model is a good improvement for the segmentation and a possible way of dealing with occlusions. However the method fails in other situations, like when a group of people move in the same direction at equal speeds.

2.2 Tracking

When the foreground is separated from the background, the objects needs to be tracked. The general problem of tracking people is complicated because of factors like the changing shape of a people as body parts move, occlusion and unpredictable paths. The tracking can be done by following the moving objects regions, blobs. This usually relies on properties of this blob like size, color, shape, velocity and/or centroid. Properties like this can suffer from occlusions, resulting in merged blobs and errors in tracking. One object can also be split into several regions, in which regions need to be merged if they are close enough to each other, and moving in the same speed and direction. Likewise when a blob seems to move in two different directions they need to be separated.
Kalman filters can be used to track and predict movement of objects, and are used by Strauffer et al. [9] and Rosales et al. [10]. Methods have been proposed to solve occlusion problems, like using multiple cameras, usually two. Tereda et al creates a system that can determine the direction of people's movements [11], thus enabling counting as they cross a virtual line. The strength of such systems is their robustness to occlusion problems, but they need good calibration between the cameras. Conrad and Johnsonbaugh [12] simplified some aspects of people counting by placing the camera overhead, since no one should be totally occluded by objects in front of it. To save computation their algorithm also limit the detection and tracking to a sub window in the frames perpendicular to the flow of traffic.

2.3 Counting

There are three main categories of counting typically used, these are the contact-type systems, sensor-type systems and vision-based systems. Contact-type systems like turnstiles obstruct the passage way and can cause congestion and frustration for the crowd as only one at a time can pass. This is though a fairly accurate system. Sensor-type systems often rely on a beam of infrared light, or heat sensors, that do not block the passageway. Beams are however unreliable as people can be passing by in the "blind zone" of the sensor.
For vision-based systems processing of multiple frames is needed. The counting process consist of determine the direction of blobs by tracking as they pass a virtual line to increment either a in or out counter [1]. There is two general ways to implement this; one is

whit two virtual lines, one for in and one for out, blobs that completely passes over one of these lines increment the corresponding counter. The option is using one virtual line for both in and out, which separates the in and out area combined with the tracking data to determine the blobs direction.

2.4 Classification

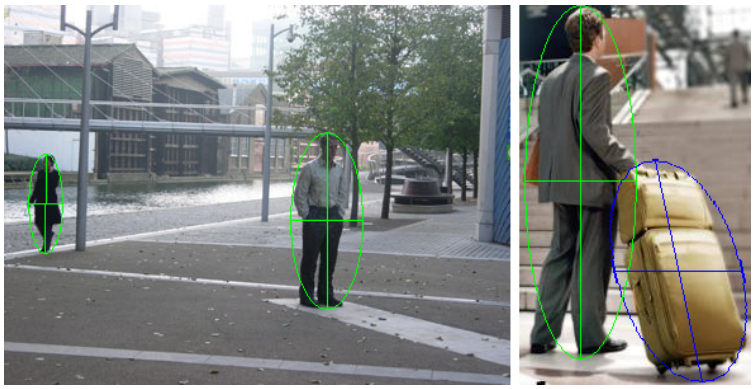

Figure 2: Ellipsoid fitting to different type of objects.

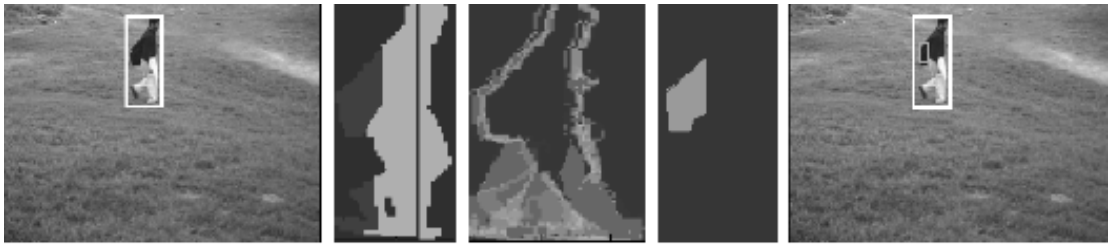

Figure 3: a) Shows the bounding box, b) is showing the symmetry line, c) is the recurrent motion image, d) shows non-symmetric pixels.

Features of the moving object can be collected and analyzed after the foreground/background segmentation. In [3] a two phased method is used, combining evidence from color, texture and motion. Features like ellipsoid fitting (minor and major axis), magnitude of velocity, direction of motion and object blob size are extracted. The result was a good method for separating people from vehicles. Ellipsoid fitting is illustrated in figure 2. In the second phase normalized features are computed to improve classification. The final system made by Brown et al. [3] used normalized features based on size only, the ellipse axes and blob area. This method managed good separation of people and cars, but groups of people could be identified as vehicles. The ratio between the major and minor axis in ellipsoid fitting is also used in [13], together with data about corners inside the bounding box of an object. This approach also gave good results in separating people from cars.
For the project described in [6] classification is done as a view based technique. The bounding box is analyzed for repetitive changes in shape of the object. For global move-

ment (moving toward the camera makes bigger projection from 3D space to 2D space) of the objects to not interfere with this analysis of local changes, the object is scaled to it's original size when it was first detected. The authors of this paper have developed their own feature vector called a "Recurrent Motion Image" (RMI) to calculate recurrent motion of objects. The idea is that different type of objects yield very different RMI's and can be classified based on this. An example of this is shown in figure 3. They also have a chapter on "Carried item detection", as a carried item (here an backpack) turns up as "non-symmetric" pixels as an symmetry axis is made vertically through the silhouette of the walking man in the example.
Bose et al. [14] consider classification in another view. For them the classification is a process that takes a set of observations (represented using suitable features) as input, and produce probabilities of belonging to different object classes. By using these probability scores instead of hard decisions this knowledge can be used for further processing, like when to alarm the operator of suspicious activity. This paper also addresses the conflicting requirements of high classification performance, and still be able to transfer classifiers across scenes. And aim to make this method work without prior knowledge about camera position, orientation or scale of the objects of interest. By using scene-independent features this project wants to use multiple cameras covering a single entrance to teach each other when the flow of traffic is less for some of them, and higher for others, to increase overall accuracy of classification of the system.
Over time, a good number of these features have been proposed, and an effort to evaluate the contribution or importance of these in [4]. Metrics are selected which are computationally inexpensive, reasonable effective for small numbers of pixels on object (far-field), and invariant to lighting conditions. This paper then presents a metric, "contribution ratio", which is obtained from selected features and weights in the AdaBoost training, and validated by demonstrating positive correlation with the performance of an artificial neural network based method.

3 Algorithms and implementation

This section will describe the algorithms used in this research project and how they have been implemented. Some of the MATLAB code was already available early in the project from the thesis work done by LeFloch in 2007. The possibility to get the code early is important so that it wont be necessary to "reinvent the wheel", and saves allot of development time. However it also means some development choices will be hard or impossible to change. A flow chart for the system is illustrated in figure 4, parts of this system will be explained in more depth explained in later sections in this Algorithms and implementation chapter.

3.1 Shadow detection and removal

The figure 5 shows the steps involved in the shadow removal. For calculating the H, S and V values the equations used are (3.1), (3.2) and (3.3) respectively. After these values are calculated the shadow mask is created with the condition in (3.4). The shadow removing function was coded and implemented by LeFloch for his project in 2007 [1].
The input frames are in RGB, these frames are sensitive to shadow since the darker shadows changes the RGB values of the background much, making the segmentation algorithm trigger on them as moving objects, or extension of a true moving object. Shadows can be considered semi-transparent regions in which the scene reflectance undergoes a local attenuation. It is possible to identify those shadows by analysis their photometric properties. The RGB images will thus be converted to the Hue-Saturation-Value (HSV) color space to separate luminosity and chromaticity [15] [16]. The aim is to estimate how the H, S and V values change in the presence of shadows, as shadows should mainly affect the V value (shadow pixel V should be smaller then non-shadow pixel V). The shadow detection algorithm is only used for detected objects to reduce computational cost.

$$H_{(x,y)} = \begin{cases} \text{undefined} & \text{if} \quad MAX = MIN \\ 60 \times \frac{G-B}{MAX-MIN} & \text{if} \quad MAX = R \wedge G \geq B \\ 60 \times \frac{G-B}{MAX-MIN} + 360 & \text{if} \quad MAX = R \wedge G < B \\ 60 \times \frac{B-R}{MAX-MIN} + 120 & \text{if} \quad MAX = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240 & \text{if} \quad MAX = B \end{cases} \tag{3.1}$$

$$S_{(x,y)} = \begin{cases} 0 & \text{if} \quad MAX = 0 \\ 255 \times \left(1 - \frac{MIN}{MAX}\right) & \text{otherwise} \end{cases} \tag{3.2}$$

$$V_{(x,y)} = MAX \tag{3.3}$$

$$SM_{t(x,y)} = \begin{cases} 1 & \text{if } \alpha \leq \frac{I^v_{t(x,y)}}{B^v_{t(x,y)}} \leq \beta \wedge \left| I^v_{t(x,y)} - B^v_{t(x,y)} \right| \leq \tau_s \wedge D^h_{t(x,y)} \leq \tau_h \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

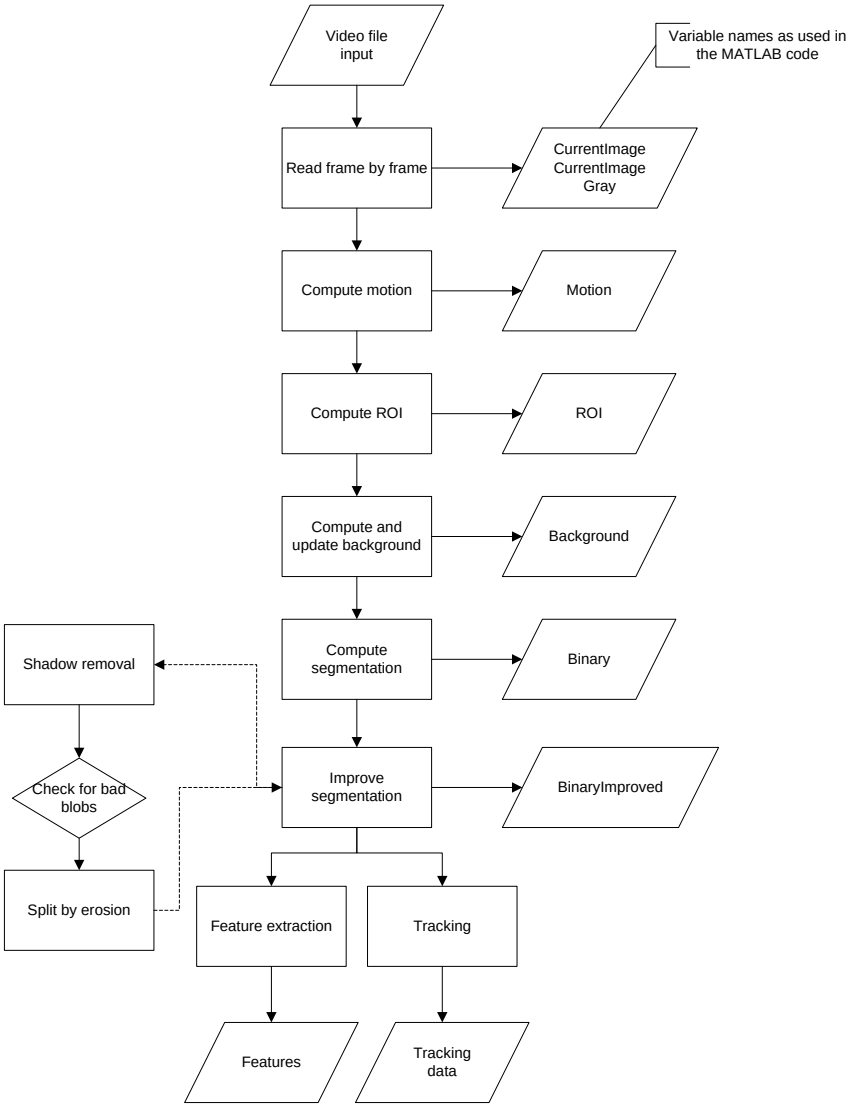

Figure 4: Complete system overview flowchart.

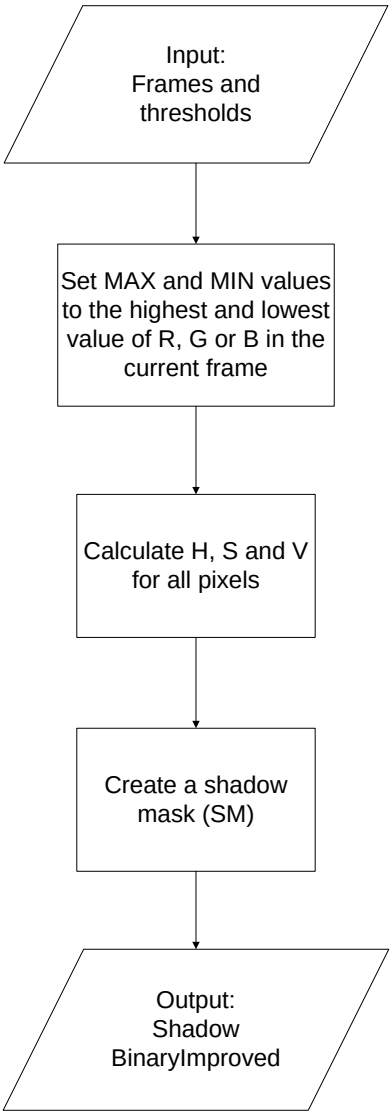

Figure 5: Flowchart illustrating the shadow detection algorithm.

Where D_t^h represents the angular difference between the hue channel of the current image I_t^h and the background B_t^h as defined in equation (3.5)

$$D^h_{t(x,y)} = \min\left(\left|I^h_{t(x,y)} - B^h_{t(x,y)}\right|, 360 - \left|I^h_{t(x,y)} - B^h_{t(x,y)}\right|\right) \tag{3.5}$$

and the two thresholds ($\alpha, \beta \in [0,1]$) where α defines the maximum value of the darkening effect that is proportional to the light source intensity (the higher intensity on the light source, the lower α should be); β is used for preventing over classification of shadows where the darkening is to small compared to shadow effects. The two other conditions τ_s and τ_h correspond to the chrominance and saturation channels and are not crucial to the detection of shadows. Referring to [17] all these thresholds are an empirical dependence on scene luminance parameters such as the average image luminance and gradient.
This MATLAB function removes detected objects below the threshold set by *Blob_Size_Min* which should filter out some false object detections.

3.2 ACE - Automatic Color Equalization

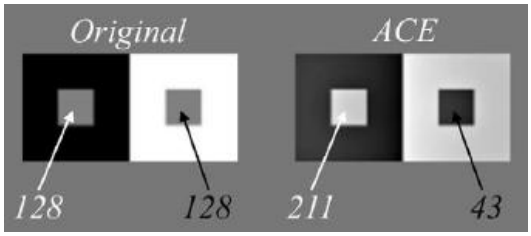

Figure 6: The ACE algorithm changes the two middle patches similar to what the human visual system do.

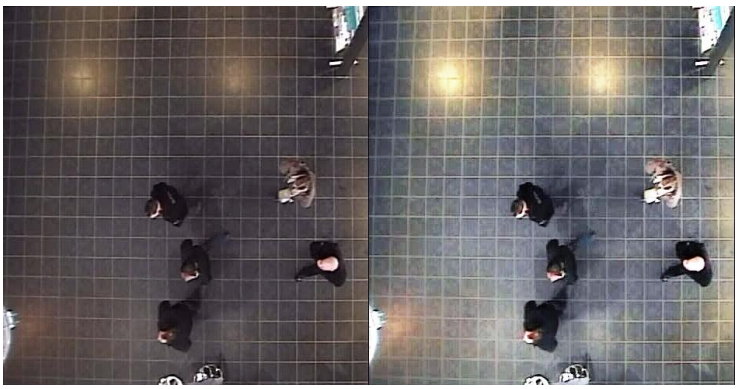

Figure 7: Left: Original frame, Right: ACE enhanced frame.

Automatic Color Equalization (ACE) is an algorithm developed by Gatta et al. [18]. Figure 9 shows a basic block schema for the ACE algorithm. The aim of this work has been to develop a digital image automatic enhancement algorithm that mimics some of the human visual system behaviors. Such as the ability for the human visual system (HVS) to steadily perceive a scene regardless of the changes in lighting conditions, a phenomenon called *color constancy* [19]. *Color constancy* enable us to perceive the true color

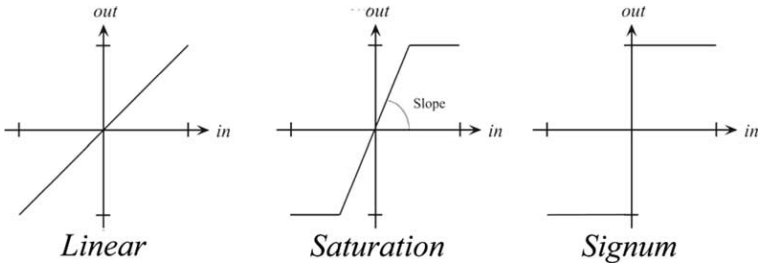

Figure 8: Different implementations of $r(\cdot)$.

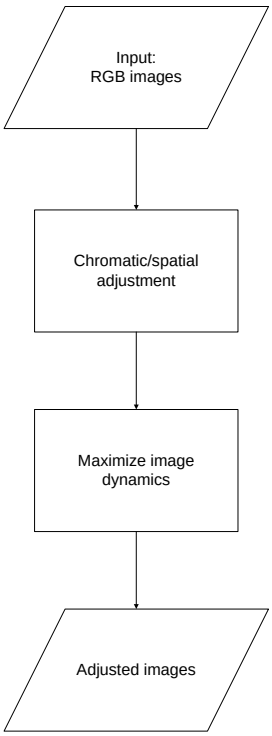

Figure 9: Flow chart for the ACE preprocessing.

of an object even if the light source changes, like when parts of an object are covered in shadow. In addition the HVS makes the perception of objects reflectance dependent on the chromatic and spatial composition of the scene [20]. The goal of these mechanisms is to maximize the image dynamics and the information content in the perceived scene. Other adaptation properties used by the HVS, and modeled a simpler version of in ACE is called *White patch* and *Gray world* [21]. The HVS in some cases normalizes the color channel values maximizing toward a hypothetical white reference area, achieving color constancy. This is the mechanisms in *White patch* (WP), and is what the HVS already does when we perceive the two patches of gray level 128 to be different in figure 6. When the mean luminance of the scene changes, our visual system performs various mechanisms to adapt to the new conditions. This allow us to have comparable luminance perception despite the change. The same mechanisms are used by photographers when they adjust the lens aperture and shutter time to acquire image gray levels fitted in the available dynamic range. This is the mechanisms in *Gray world* (GW).
In general the ACE algorithm measures how a pixel is influenced by all pixels around it using a weighted function. Then it maps the pixel into a new image, with an improved luminance and removes color cast.
Each pixel p in the output image R_c is computed separately for each channel c

$$R_c(p) = \sum_{j \in Image, j \neq p} \frac{r\left[I_c(p) - I_c(j)\right]}{d(p,j)} \tag{3.6}$$

$I_c(p) - I_c(j)$ accounts for the lateral inhibition mechanism [21], while $d(\cdot)$ is a distance function that weights the amount of local/global contribution between pixels. $r(\cdot)$ is the function that accounts for the relative lightness appearance of the pixel and act as an contrast tuner, and different functions have been tested by Rizzi et al. such as linear implementation, saturation and signum, shown in figure 8. From tests done in [22], a function for the distance $d(\cdot)$ that provides good results is the *Euclidean distance*, or it's approximation like *(*Manhattan distance). This means different implementations of $r(\cdot)$ and $d(\cdot)$ is important for the resulting image.
The basic ACE algorithm is reported to have an computational cost as $O(N^2)$ with the N being the number of pixels in the image, making it fairly slow. The ACE application used for the preprocessing step has an option for sub sampling of x that only calculates the values for every xth image pixel and interpolate the missing values between the fully computed pixels. This option saves a large amount of processing.

3.3 Blob separation by erosion

Figure 10 illustrates the steps involved in the splitting algorithm. Even when most shadows are removed, some blobs still merge due to a little shadow in between them. And since the camera is over head the walking person's feet stick out from the blob in front and in back as illustrated in figure 11, which also can cause merging errors. A method of separating these errors was made by eroding off the edge of the bad blob, until a split occurs (also depends on the size of the parts, as the leg are split from the body in figure 11 c)). When a split is detected the bounding boxes of the new objects are enlarged to include the most of the original blob without merging them again. The new bounding boxes are used on the blob in figure 11 b), resulting in the two parts that make up figure 11 b) in figure 11 e) and figure 11 f). To split the two parts, the common of figure 11 e)

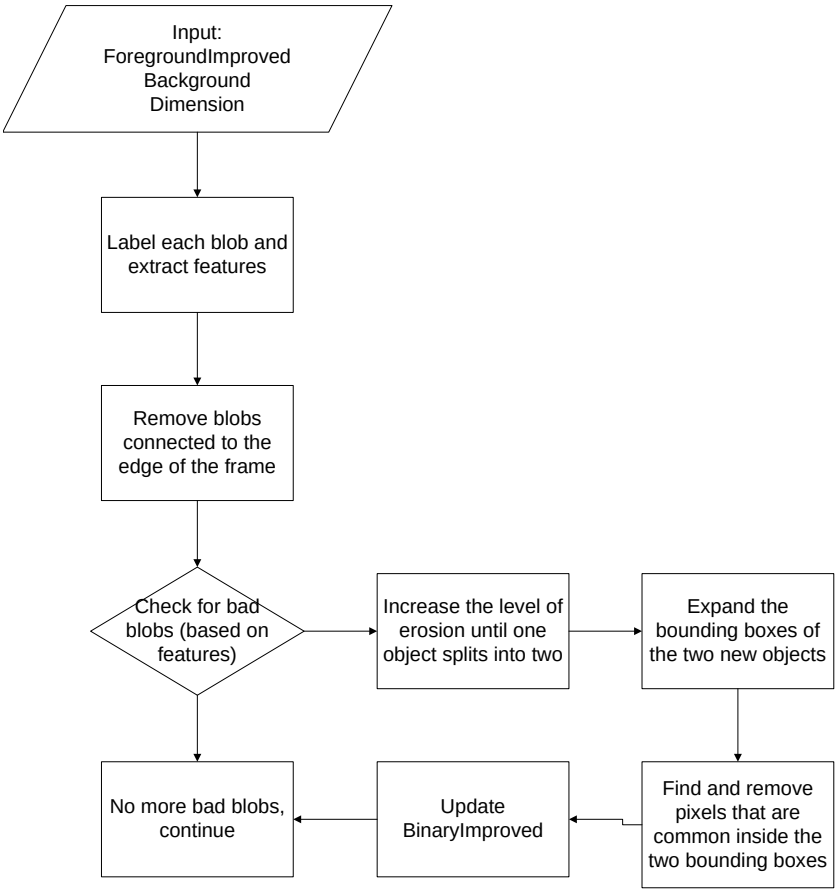

Figure 10: Flow chart showing the steps in the splitting algorithm.

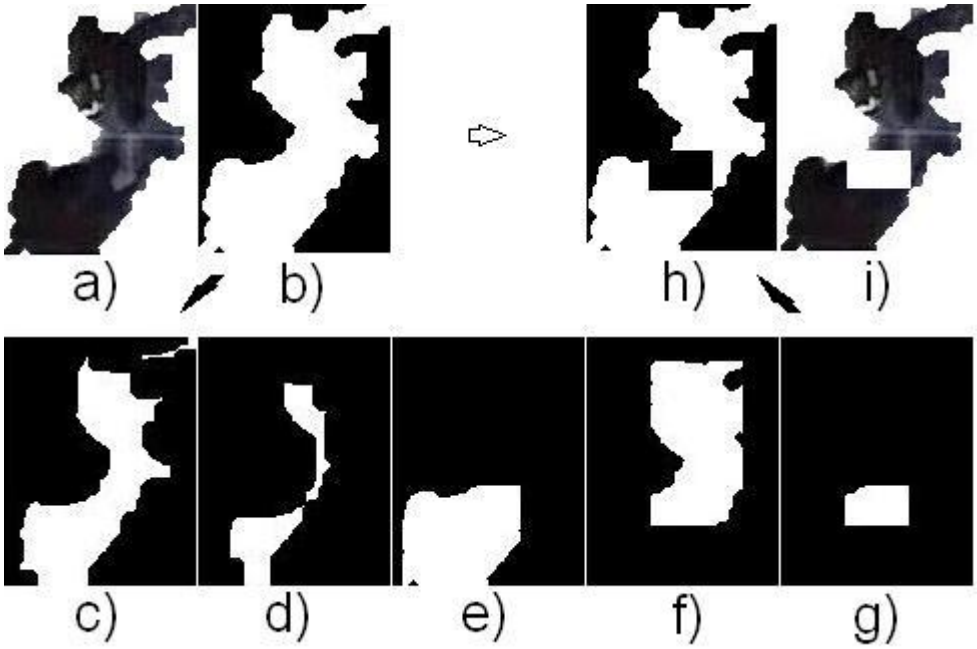

Figure 11: This shows different steps in separating a bad blob. In a) is the color starting blob, b) is the corresponding blob, c) is an erosion of 10, d) is an erosion of 22 and a separation occurs. e) is part one of the original blob, f) is the second part, g) shows what these has common. In h) the common part is removed, and the final blobs is shown in i).

and figure 11 f) as shown in figure 11 g) is removed from the original blob. The result is presented in figure 11 h) and finally the two separated blobs in colors are shown in figure 11 i). After a split has been done the image stored in the variable *BinaryImproved* (containing the cleaned up image of detected blobs) has to be updated with the separated blobs. In addition blobs that was connected to the edges of the image is deleted, meaning people entering or exiting the field of view will not be traced or extracted features from. The script code made to split these blobs can be found in the appendix B.

4 Experimental setup and results

In this section there will be a discussion about the results of experimentation on different thresholds, values and algorithms done in MATLAB. The observations made is mainly done by using a test video and observe the output.

4.1 The test bench

MATLAB has been chosen as a platform to build a test bench for a few simple reasons, Pro's:

- Built-in and external functions
- Image processing is mathematics
- Data can easily be illustrated
- Flexibility

Con's:

- Speed

Starting with the Pro's access to both basic and advanced image processing functions and algorithms, saves development and implementation time. There is a wealth of image processing functions readily available in MATLAB with good documentation that saves time and effort from "reinventing the wheel". The software provider also gives access to a user community on line, *Matlab Central*, for developers to exchange files, ideas and experience with each other. For the early stages in this research project, work started in developing code to stream video from the AXIS network cameras to be processed, but the strategy was later changed in favor of the latest implementation of LeFloch's work on the People Counting system[1]. This has been the platform which new algorithms and strategies has been tested. Working with MATLAB variables is easily stored, retrieved and illustrated proving very useful when debugging and to help find the reason when the results where not as expected. The last, but not least important is the flexibility. Parts of the code or subroutines can easily be modified or replaced, making this an ideal test bench for this research work.
The downside is the running speed of the developed code, that has not been close to real-time during the development.
The program provided by LeFloch takes a video file as input with the significant advantage that results are now comparable as the output is controlled by the changes made to test bench. This also make it possible to focus on specially problematic events in the video, as situations can be hard to reconstruct in front of the camera later.

4.2 Classroom ceiling (scenario 1)

In the beginning a camera was placed in the ceiling by the entrance of a classroom, about 2 meters of the floor. A sample frame is illustrated in figure 12. On the positive side there is almost no shadows from the people moving under the camera. However the algorithms

Figure 12: A sample frame from the camera fixed to the roof in a classroom.

in the test bench had big problems making good segmentation caused by the objects being relatively large in the scene. This in turn made it hard to calculate what pixels where in motion, large surfaces where static for many frames causing big problems for the ROI mask (Region of Interest, made so the adaptive background would not be updated with data from the foreground, called ghosting) and cascading trough all future processing steps. Foreground pixels that are not inside the ROI are pushed to the background and create huge problems for the segmentation algorithm in future frames as they conceal any changes. So pieces of the foreground is pushed to the background the segmentation algorithm fails to make a good decisions. In an effort to correct some of the problems and make better output several methods were tested, the *frameskip* parameter was increased making the program skip frames, instead of processing every frame now only every 2nd, 3rd or so frame. This increased the difference between the frames compared, effectively increasing the magnitude of motion between the processed frames. Also the ROI mask was morphologically dilated, meaning the regions are enlarged to stop some of the foreground being pushed to the background. However this made the blob separations harder. Blobs also often contained holes in them where the foreground had not separated correctly, in an attempt to clean up the detected foregrounds the MATLAB function *imfill* was used. In addition to the desired effect it made the separation harder in the same way as *ROI* dilation, and holes between two separate blobs was filled creating one large blob hard to separate into each individual. By increasing the *alpha* parameter (the learning rate of the adaptive background) ghosting from the foreground was reduced, for the cost of problems if lighting conditions change or movement of the camera. Reducing the image resolution proved more effective by reducing the number of detected objects in the scene, reduction in ghosting and improving processing speeds. A method of extracting the image blobs and eroding them was tried. For this the MATLAB function *imerode* was used with a erosion function that used a percentage of the minor axis length of the specific blob. This method could in some cases make the separation needed, as long as the point of merging was not to big as it is in figure 13. A point of concern is also the effect this has on the statistics of the blobs, as this ideally should enable the classification of the foreground object (on the bright side it affect all the blobs relatively the same way).
Still this could not solve the problems in bad separation and wrong merging of blobs, example frames in figures 13 and 14. To be fair, the People Counting system was not developed for this type of "close up images", but rather smaller objects moving across the scene. Discussing with the supervisor, another professor at GUC and checking with Datatilsynet what we could and could not do 1.4, the decision was made to capture a

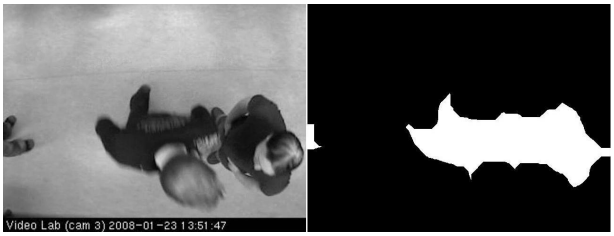

Figure 13: Wrong merging.

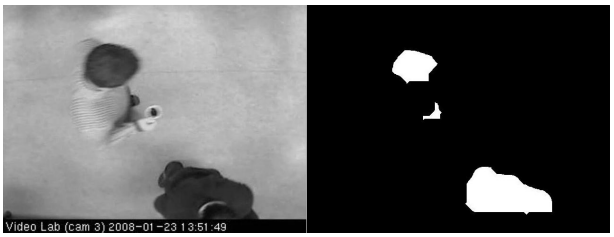

Figure 14: Bad separation.

new video, this time about 10 meters off the floor in another part of the school.

4.3 High up mounted camera (scenario 2)

Figure 15: Sample frame from the second video.

Many of the earlier problems with detection and tracking of objects improved with the new input (see figure 15). The objects are much smaller now relative to the scene, this also mean a larger entrance can be tracked, and more people can be tracked. Objects move faster, preventing the large static surfaces that caused problems before. In many cases the separation is also improved thanks to the methods used by Lefloch working better on this scenario.
A comparison has been made in figure 16. The new problem to good separation is now the shadow, as there is allot of in this scene. The shadows cause objects to merge or appear as false objects. A suggestion for a preprocessing step was proposed to equalize the image histograms, increasing the contrast between the foreground and background, that could help the implemented algorithms to make cleaner segmentation. To test this the MATLAB function *imhist* and *imadjust* was tested, both in gray scale frames and on all channels in RGB frames. However this did not seem to have any positive effect for the separation. The problem is when a person moves into the scene, the dynamic range

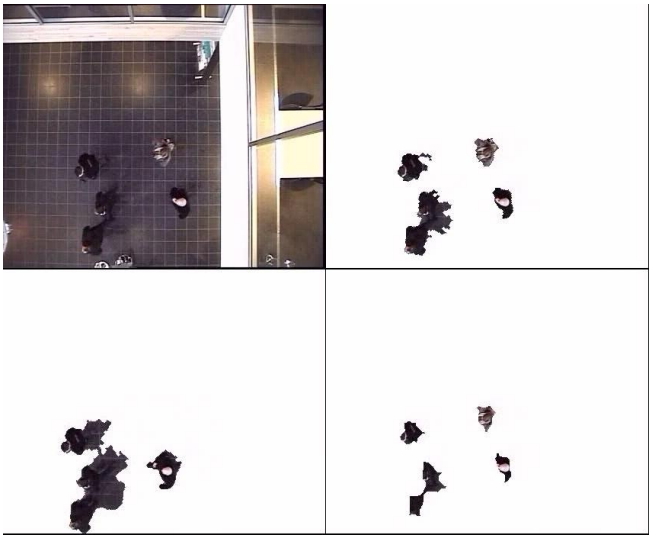

Figure 16: Upper left: Input image, upper right: Lefloch's original code, lower left: code modified for 1st scenario, lower right: original code and blob erosion.

of the colors changes considerable and the whole background changes color/intensity. As in the example in 16 the lower right image show the blobs eroded 15% of the blob's minor axis lenght could almost separate the two people merged. The problem increasing this factor is in some cases people are eroded away instead of separated and counted.
The code provided by LeFloch includes a shadow detection and removing algorithm, described in 3.1. By checking the *Shadow* (a binary image showing where shadows has been detected) it was clear that this algorithm performed poorly in this context. Changing the α and β thresholds could not improve the detection much.
Three of MATLAB demo's are *viptrackpeople*, *viptraffic* and *viptrafficof* were put to the test with the video captured from the ceiling in scenario 2. The 1st is made to track people, by using adaptive background estimation, then separating out the moving objects. This had huge problems detecting and tracking the people in the scene. It is possible it needs very specific conditions to work properly, like large objects similar to scenario 1. This far the viptrackpeople was the weakest against shadows, making the tracking useless.
The demo viptraffic is made to count cars moving on a road, and are in design close to that of LeFloch's, with similar adaptive background estimation and separation. Although this demo had not an implemented shadow detection algorithm, detection and tracking worked ok. It is hard or impossible to make good statistics for how good these different solutions work, the traffic demo's are made to display how many cars are in the scene in that frame. When used on the test video some people are tracked all the time, some are not tracked in every frame, some are almost never detected and there is the false detections to consider. The movie contain many frames, therefore a completely accurate statistic is missing. Back to the viptrafficof does things a little bit differently as it's detection and tracking is modeled by the help of "optical flow", a method of approximating the motion of objects in an image by finding correlations between near frames in the video, generating a vector field showing where each pixel or region has moved. This method had problems detecting a person moving, resulting in multiple counting of a single person. From the output video of this demo this might be the method so far with the

least problems dealing with shadows, but the detection is too poor. Optical flow might be useful to implement in a future project with the time to explore it's possibilities.

4.4 High up mounted smaller camera (scenario 3)

After working with the video from scenario 2 for a while it was evident that some other data was needed containing hard type objects. To save time a smaller AXIS 207W camera was used, mounted at about the same place as the large camera used in scenario 2. The aim of this session was to capture one or more rigid body objects. A trolley borrowed from the cafeteria was used for this purpose. Learning from a mistake done in scenario 2 the video quality was set lower for the computer to capture all the frames delivered from the camera. The resolution and frame size was reduced to do this. Working with this video a problem occurred in the way thresholds are used in our project, they are hard coded, meaning many changes had to be made to the system to fit the new video. All these changes also mean features collected from the two last scenarios are not directly comparable.

4.5 ACE preprocessing

Figure 17: Left: Original frame, Right: ACE enhanced image.

When testing this algorithm as a preprocessing stage for the People Counting system the hope was that shadows in the images would not be so problematic. A frame from the scenario 2 video is shown in figure 17. The shadow problem was not solved by this preprocessing, however the ACE made the separation more stable (parts or whole objects tend to flicker before) and the boundary between background and foreground was improved. This is probably because the increase in contrast between foreground objects and the background. The *Shadow* image was discovered to detect an huge increase in shadows compared to the testing mentioned in 4.3.
This preprocessing was not implemented in MATLAB, instead the input frames was processed beforehand. Interesting sections of the captured videos was converted into a image file for each frame and added to the ACE application created by Rizzi and Gatta [18]. To speed up the process a sub sampling of 8 was selected. The processed frames was then reassembled as a video file for MATLAB to use. In the case of a real-time people counting system a different approach is needed like the STRESS (spatio temporal retinex-like envelope with stochastic sampling) by Kolaas made for the GEGL library (http://gegl.org/operations.html#op_stress), made to be a faster implementation of ACE

for use with video.

4.6 Shadow detection and removal

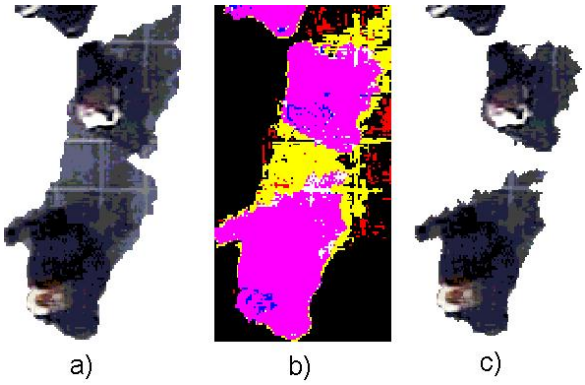

Figure 18: Finding the right settings for shadow detection algorithm. a) shows blob detection before, with shadow merging two people to one blob. b) shows an image from the code cycling trough settings of the detection as yellow shows detected shadows, and the pink is the resulting new blobs, and c) showing how the new detected blobs looks like.

With the already implemented shadow detection algorithm now working much better, the correct settings of the input variables (described in section 3.1) for this specific scene had to be found. As illustrated in figure 18 shadows could cause people to be merged by a connecting shadow.
A script in MATLAB was made with the purpose of testing and finding good settings for *alpha*, *beta*, *Threshold_S* and *Threshold_H*. As an input a frame with allot of shadows was needed, and start values for the thresholds was set to:

- *Alpha_V* = 0, +0.01 pr iteration
- *Beta_V* = 1, -0.01 pr iteration
- *Threshold_S* = 0, +1 pr iteration
- *Threshold_H* = 0, +1 pr iteration

Changing all attributes at once made it hard to find good values, so only one value would change, with the others set to LeFloch's default values. The result images for *Binary*, *Shadow* mask and *BinaryImproved* were put into a color image as the respective channels Red, Green and Blue as in figure 18 b). The goal was to get the pink blobs (red+blue) match the people in this frame, and to detect the shadow between them in yellow (red+green), enabling splitting of the people in this example. Good values was frozen, and the next is unfrozen, listing the final settings:

- *Alpha_V* = 0.45 (changed from 0.8)
- *Beta_V* = 1 (changed from 0.99)
- *Threshold_S* = 80 (changed from 30)
- *Threshold_H* = 20 (changed from 30)

A side note however, is in this scene the light source conditions are not uniform through-

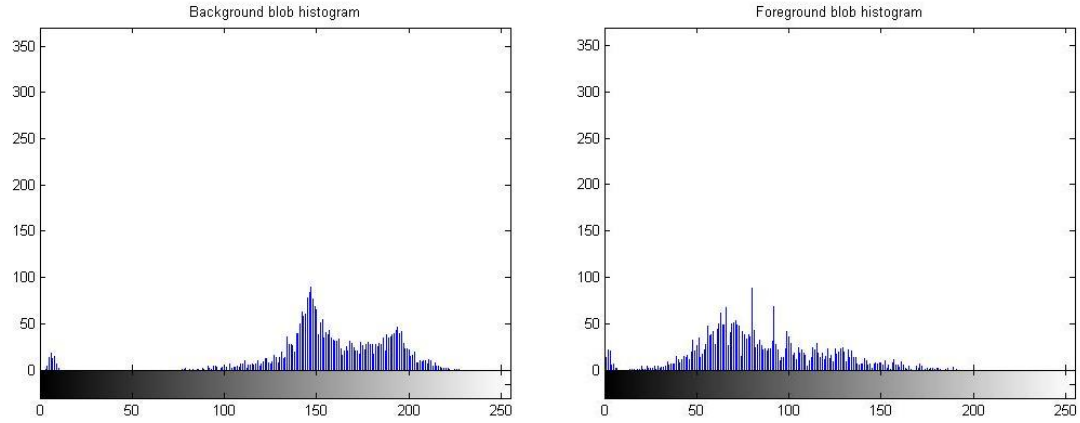

Figure 19: Histograms of a detected blob only containg shadow, and its background.

out the scene. As can be seen by the two bright yellow areas where lamps are closer to the floor in figure 17 upper part. This can change the accuracy of the shadow removal for different parts of the frames.
Early testing in removing detected blobs only containing shadows included a histogram comparison, where the idea was that the topology of the histogram would not change much except shifting the pixels to darker values. Figure 19 shows two histograms of the same area where a blob with only shadows is detected. Topology preservation is subject in the paper [23]. This was shortly explored, but was dropped as shadow detection and removal really is a complex unsolved problem at the basis of all computer vision systems. And there was the need to progress this thesis project after using allot of time on the issues caused by shadows in the scene.
The script made to test the configuration of the shadow detection algorithm can be found in the appendix C.

4.7 Feature extraction

To classify objects some features of them has to be extracted. Some features like size and bounding box dimensions are regarded less complex and are easier to extract. The following list shows the features currently extracted:

- Blob size - number of pixels.
- Bounding box dimensions - number of pixels in height and length.
- Centroid motion - X and Y position of the centroid in the frame.
- Solidity - how much of the blob fills the convex hull. [0,1]
- Roundness - A formula for calculating the roundness of an object. [0,1]

The four first features are extracted using the MATALB functions *bwlabel* (for the roundness calculation it had to *bwboundaries*) and *regionprops*, which has the possibility to gather many other features also. The ones selected in this project is much the same as by [3] and [13]. If the low level features are enough to make good decisions in classification this would be preferable as the computational requirements and the ease of getting these.

4.7.1 Blob size

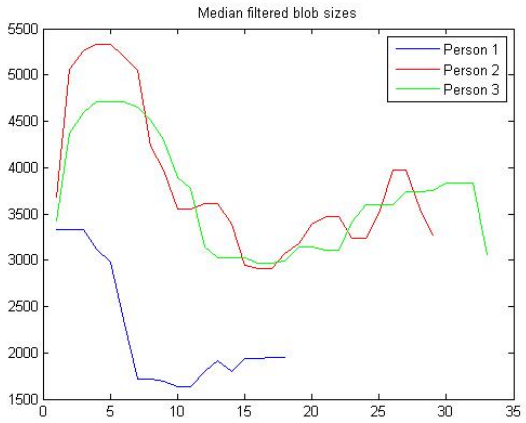

Figure 20: Compares the blob sizes from three persons.

In figure 20 the size of three different people is compared, extracted from three sample video's from scenario 2. In all three videos the moving object was usually well segmented from the background without to much shadow. In [24] the blob size is used as an estimate of how many people make up a large group. This is currently not implemented for our project, but it should be in the future, also discussed in 6. The estimation used in [24] is shown in (4.1) to (4.8) where PSI is people size interval, LB is lower bound and UB is upper bound, M is mean, D is distance, N is the number of pixels in the blob and P is the number of people.

$$PSI = [LB \cup UB] \tag{4.1}$$

$$M = \frac{LB + UB}{2} \tag{4.2}$$

$$D = M - LB = UB - M \tag{4.3}$$

$$P = 1 \quad \text{if} \quad M - D \leq N \leq M + D \tag{4.4}$$

$$P = 2 \quad \text{if} \quad M + D < N \leq 2M + 0.5D \tag{4.5}$$

$$P = 3 \quad \text{if} \quad 2M + 0.5D < N \leq 3M + 0.8D \tag{4.6}$$

$$P = 4 \quad \text{if} \quad 3M + 0.8D < N \leq 4M + D \tag{4.7}$$

$$P = 5 \quad \text{if} \quad 4M + D < N \leq 5M + 1.2D \tag{4.8}$$

Back to figure 20 all three blobs tend to be largest when they enter the scene. This can partially be explained by the geometry where directly under the camera the 3D to 2D translation makes the objects smallest. To the edge of the scene the height of the object starts to influence the blob size. This suggest the features should not be captured from the moment they enter the scene, but rather in a window in the center field of view for the camera.

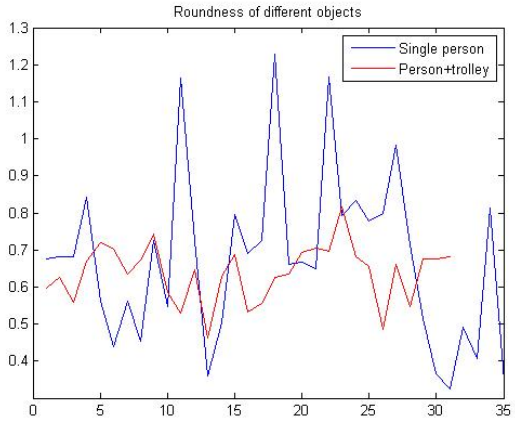

Figure 21: Roundness calculated for two object types.

4.7.2 Roundness

The roundness feature is calculated from a equation found on MATLAB Central, the earlier mentioned community provided for MATLAB users, equation is shown in (4.9) where *area* is the number of pixels in the blob, and *perimeter* is an estimation made from the (x,y) coordinates of an objects boundary pixels.

$$\text{Roundness} = \frac{4 \times \pi \times \text{area}}{\text{perimeter}^2} \tag{4.9}$$

The roundness should only be equal 1 for circles. However during the calculation of roundness for objects in the video from scenario 3 some values was higher then 1. This could be an error caused by the denominator being close to zero. This should be fixed by adding a small value to prevent division by zero problems. Discussing with the supervisor we expected the roundness to change more for walking people as the feet and arms change the shape of the blob, compared to the rigid shape of hard body objects. The figure 21 compare the calculated roundness for a single person, and for a person moving a trolley. This first test looks promising, but time constrains do not allow for extensive exploration and gathering of better data for analysis.

5 Conclusions

The focus of this thesis was in the beginning at the classification process as an extension to LeFloch's people counting system. During the project period the issues taking up most of the time are getting good separations between foreground and background, and the shadow detection and removal. Feature extraction is heavily dependent on good data to analyze, justifying the time spent on these issues. The code from LeFloch's prototype has been used as a basis for this thesis experiments.
Therefor we focused on a three research question in this thesis, and conclusion of the work on these questions will be presented here in respective sections. Due to time limit statistics to show improvements are missing. As mentioned before good ways to make statistics for this is hard. The detection of all persons in a frame are perfect for some frames, but other times persons are still merged.

5.1 Can we achieve better separation of groups into individuals?

The ACE algorithm described in 3.2 improved the already implemented shadow detection algorithm, improving the background and foreground segmentation. This alone solved some merging problems, but are still not enough to achieve perfect separation of all groups. In some cases two people could merge due to feet sticking out of the body in front and in the back resulting in a blob with two larger parts and a narrow "bridge" between them. A separation algorithm was proposed to erode the larger blob until the "bridge" was broken 3.3. This solved bad merging for some groups further increasing the accuracy.

5.2 Can we find features to classify people and indoor objects?

The roundness feature described in 4.7.2 looks as a potentially good feature to separate hard body objects from people. Further analysis might show this feature has to be combined with features like blob size. The roundness changes much from frame to frame when people are walking, and are more stable for hard body objects. The size of the combined person and trolley blob is also naturally larger then a person alone. The number of persons in a large blob can also be estimated, referring back to chapter 4.7.1 and equations (4.1) to (4.8).

5.3 Can we still maintain real-time operation?

At this time the simple answer is no. This part has not been that important during the project work either. The preprocessing of color equalization is currently done on the video file before it is imported to MATLAB, and a stored file is used as a video source, rather than live feed from a camera. This will be further discussed in 6. MATLAB is not the ideal platform for a complete application, but it has been used since it is an excellent development tool for the tasks at hand in this project. This was also discussed in 4.1.

6 Future work

In this chapter ideas for future work and possible solutions to some problems we encountered will be discussed and presented. The captured videos had some issues, with one in scenario 2 being captured at a frame rate the computer could not process, resulting in frames duplicated a changing number of times to maintain 30 FPS when about one third of the frames was actually unique. Later recording a new video for scenario 3 measures was taken to ensure correct frame rate sacrificing image quality and resolution. A better solution here would be to have the larger AXIS 223D camera permanently mounted so the data can be acquired as often and fast as needed and for a live test. The smaller 207W cameras image quality was too low to be used as a permanent setup, so 223D is the preferred choice.

For the code the thresholds were too rigidly coded. Many parameters can be changed, but is time consuming when input data change major characteristics like resolution, frame rate ect. This was the case changing from scenario 2 till scenario 3. This process should be more automated in the future. One problem in working "off line" on a stored video is that feature statistics are not saved. These statistics could be used to teach the system how to calibrate thresholds on it's own. Already discussed in 4.7.1 automated thresholds for the people size boundaries are proposed. What we believe to be most difficult to automate is the shadow detection settings, since just making a good system for detecting shadows is complex already.

Easier access to live streaming video from the cameras, and the possibility to gather statistics over time also opens for other features then what is currently extracted. Due to time constraints the features extracted was not throughly investigated either, and I wish I had more time to investigate this part.

To explore the speed/performance of this system an application, based on e.g. C++, would have to be developed to see what performance is achievable. The current use of ACE should also be revised or changed to STRESS as mentioned in 4.5 or other illumination-invariant color spaces as discussed in [25]. The color equalization by itself still is a good idea though, but it currently is to complex for what we need.

With statistics over time a learning algorithm should be useful for making good decisions from the data at hand. Learning algorithms are though complex and potential a problem in a real time environment. Implementation of this has not been tested in this project, but could be a natural extension later.

Bibliography

[1] LeFloch, D. Real-time people counting system using video camera. Master's thesis, Gjoevik University College, Universite de Bourgogne, 2007.

[2] When are you allowed to have camera surveillance?, naar har du lov til aa overvaake med kamera?

[3] Brown, L. M. View independent vehicle/person classification. Technical report, Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks, pages 114-123, 2004.

[4] Masamitsu Tsuchiya, H. F. Evaluating feature importance of object classification in visual surveillance. Technical report, The 18th International Conference on Pattern Recognition (ICPR'06), Vol 2, pages 978-981, 2006.

[5] Ying-Li Tian, M. L. & Hampapur, A. Robust and efficient foreground analysis for real-time video surveillance. Technical report, Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol 1, pages 1182-1187, 2004.

[6] Omar Javed, M. S. Tracking and object classification for automated surveillance. Technical report, Proceedings of the 7th European Conference on Computer Vision-Part IV, pages 343-357, 2002.

[7] Schofield A.J, S. T. & P.A, M. A ram based neural network approach to people counting. Technical report, Image Processing and its Applications, 1995., Fifth International Conference, pages 652-656, July 1996.

[8] Shio A., S. J. Segmentation of people in motion. Technical report, Visual Motion, 1991., Proceedings of the IEEE Workshop, pages 325-332, October 1991.

[9] Chris Stauffer, W. G. Adaptive background mixture models for real-time tracking. Technical report, Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference, Vol 2, pages -252, June 1999.

[10] R. Rosales, S. S. 1998. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *IEEE Conf. on Computer Vision and Pattern Recognition. Workshop on the Interpretation of Visual Motion, pages 228-233*.

[11] Yamaguchi, K. T. D. Y. S. O. J. October 1999. A method of counting the passing people by using the stereo images. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference, Vol 2, pages 338-342*.

[12] Gary Conrad, R. J. 1994. A real-time people counter. In *Proceedings of the 1994 ACM symposium on Applied computing, pages 20-24*.

[13] Qi Zang, R. K. 2004. *Object Classification and Tracking in Video Surveillance*. Springer Berlin / Heidelberg.

[14] Bose, B. Classifying tracked objects in far-field video surveillance. Master's thesis, Indian Institute of Technology, Delhi, 2002.

[15] Cucchiara R., Piccardi M., P. A. 2003. Detecting moving objects, ghosts and shadows in video stream. *IEEE Trans. PAMI. v25 i10. pages 1337-1342*.

[16] Cucchiara R., Prati A., M. I. & M.M., T. July 2003. Detecting moving shadows: Algorithm and evaluation. *IEEE Trans. on pattern analysis and machine intelligence, vol.25, no.7*.

[17] Piccardi, C. R. P. A. G. C. & S., S. August 2001. Improving shadow suppression in moving object detection with hsv color information. *IEEE Intelligent Transportation Systems Conference Proceedings - Oakland (CA) USA*.

[18] Alessandro Rizzi, Carlo Gatta, D. M. A new algorithm for unsupervised global and local color correction. Technical report, Pattern Recognition Letters 24, pages 1663-1677, 2003.

[19] Wyszecky G., S. W. 1982. *Color Science: Concepts and Methods, Quantitative Data and Formulas*. J. Wiley & Sons, New York.

[20] De Valois R.L., D. V. K. 1988. *Spatial Vision*. Oxford University Press.

[21] Rizzi A., G. C. & M., M. From retinex to automatic color equalization: issues in developing a new algorithm for unsupervised color equalization. Technical report, Journal of Electronic Imaging 13(1), pages 75-84, January 2004.

[22] Rizzi A., G. C. & M., M. Color correction between gray world and white patch. Technical report, Proc. SPIE, Vol.4662, pages 367-375, 2002.

[23] H., Y. W. W. Q. L. Q. L. & S., M. Topology-preserved diffusion distance for histogram comparison. Technical report, National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences, 2002.

[24] Velipasalar S., T. Y. & A., H. Automatic couting of interacting people by using a single uncalibrated camera. Technical report, Princeton University, Electrical Engineering and IBM T.J. Watson Research Center, 2006.

[25] Bascle B., B. O. & V., L. Illumination-invariant color image correction. Technical report, France Telecom R&D Lannion, France, 2006.

A Source code, PeopleCountingVideo.m

This code has mainly been written by Damien LeFloch [1], and changed to better fit the needs in this thesis.

```
clear all;
close all;
clc;

%filename1 = strcat('PETS2006_34','.avi'); %create current filename
%filename1 = strcat('F:\Master\Testvideo\features_test1','.avi');
filename1 = strcat('F:\Master\Testvideo\features_test5','.avi');

cheat_fix = false;
output_mov = true;
output_mov_file_name = ('features_test5output.avi');

%Stored features for statistics:
blob_size = [];
bounding_box_size = [];
solidity = [];
orientation = [];
centroid = [];
roundness = [];

%Moving average filter settings
span = 3; % Size of the averaging window
window = ones(span,1)/span;

if( exist(filename1,'file')~=0)%if file not exist skipped the process

    %     NbImX=2;NbImY=2;
    NbImX=1;NbImY=2;

    info = aviinfo(filename1);
    NumberFrames = info.NumFrames;
    clear info;

    BufferFrames = 10;
    StartFrame = 1;
    i = StartFrame;%For the loop (all the frames of the video)
    bufferIndex = i;
```

```
AviBuffer = [];

%TimePaused = 0;
TimePaused = 1; %(Time waited for taking the Next Frame in second)
frameskip =1;
FPS = 0;

Buffer_Centroid_Bound =[];
LengthBuff = 10;

PreviousSecondImageGray = [];
PreviousImageGray = [];
BinaryImprovedEroded = [];
ForegroundEroded = [];

Neta = 3;
Str_el_Ray = 2;

%Blob_Size_Min = 650;
Blob_Size_Min = 200;
BlobsAreas = [];

%Threshold Shadows
%Alpha_V = 0.8;
%Beta_V = 0.99;
%Threshold_S = 30;
%Threshold_H = 30;
Alpha_V = 0.45;
Beta_V = 1;
Threshold_S = 80;
Threshold_H = 20;

TailleEspace = 4;%deviation between each image in order to display

Color = true;%Work in Color or grey intensity
DefaultColor = uint8([255;255;255]);%default color of the foreground image

% %
if( output_mov )
    mov = avifile(output_mov_file_name);
end
% %

dt=1;
g=6;
% Kalman filter initialization
```

```
Kalman.R=  eye(4).*abs(randn(4));
Kalman.C=   [[1,0,0,0,0,0,0,0];...
    [0,1,0,0,0,0,0,0];...
    [0,0,1,0,0,0,0,0];...
    [0,0,0,1,0,0,0,0]];
Kalman.CT = Kalman.C';
Kalman.Q=0.01*eye(8);
Kalman.A=   [[1,0,0,0,dt,0,0,0];...
    [0,1,0,0,0,dt,0,0];...
    [0,0,1,0,0,0,dt,0];...
    [0,0,0,1,0,0,0,dt];...
    [0,0,0,0,1,0,0, 0];...
    [0,0,0,0,0,1,0, 0];...
    [0,0,0,0,0,0,1, 0];...
    [0,0,0,0,0,0,0, 1];];
Kalman.AT= Kalman.A';
Kalman.Bu = [0,0,0,g]';

while(i<=NumberFrames)
    %display(strcat('Frame = ',num2str(i)));
    if(bufferIndex + 1 > length(AviBuffer))
        bufferIndex = 1;
        if(i == StartFrame)
            endIndex = StartFrame + BufferFrames*(frameskip+1) -1;
            AviBuffer = aviread(filename1,StartFrame:(frameskip+1):endIndex);
        else
            endIndex = i + BufferFrames*(frameskip+1);
            if(endIndex > NumberFrames)
                endIndex = NumberFrames;
            end
            AviBuffer = aviread(filename1,i+1:(frameskip+1):endIndex);
        end
    end
    CurrentImage = uint8(AviBuffer(bufferIndex).cdata);
    %
    %CurrentImage = imresize(CurrentImage, 0.7);
    %CurrentImage = CurrentImage(62:404,1:330,:);
    %CurrentImage(:,:,1) = imadjust(CurrentImage(:,:,1));
    %CurrentImage(:,:,2) = imadjust(CurrentImage(:,:,2));
    %CurrentImage(:,:,3) = imadjust(CurrentImage(:,:,3));
    %
    Dimension = size(CurrentImage);
    if(length(Dimension) == 3)
        CurrentImageGray = RGB2LUM(CurrentImage);
        %
        CurrentImageGray = imadjust(CurrentImageGray);
```

```
            %
            if(Color == false)
                CurrentImage = RGB2LUM(CurrentImage);
                Dimension = size(CurrentImage);
            end
        else
            CurrentImageGray = CurrentImage;
            color = true;
        end
        if(i == StartFrame)
            Background = CurrentImage;
            TailleX = NbImX*Dimension(1) + TailleEspace*(NbImX-1);
            TailleY = NbImY*Dimension(2) + TailleEspace*(NbImY-1);
            ImDisplay = zeros(TailleX,TailleY,3,'uint8') + 200;
            clear TailleX TailleY;
        end

        useframe = 1;
            if(useframe)
tic
        %%%%%Compute Motion%%%%%%%%%%
        %%%%%%Elapsed Time%%%%%%%%%%%
        %%%%%%5 millisecond%%%%%%%%%%
        [PreviousSecondImageGray PreviousImageGray Motion NonZero] = Compute_Motion(PreviousSecond
        Motion = Im_Dilate(Motion,2);Motion = Im_Dilate(Motion,2);

        %%%%%Compute ROI%%%%%%%%%%
        %%%%%%Elapsed Time%%%%%%%%%%%
        %%%%%%1.8 millisecond%%%%%%%%
        [ROI Alpha] = Compute_ROI(Motion,NonZero);
        ROI=imdilate(ROI,strel('rectangle',[1 5]));
        %ROI = Im_Dilate(ROI,4);
        %display(strcat('Alpha = ',num2str(Alpha)));
        %Alpha = Alpha*2;
        %Alpha = 0.8;

        %%%%%%%%%%%%%%%%%%Compute And Update Background%%%%%%%%%%%%%%%%%%
        %%%%%%Elapsed Time%%%%%%%%%%%       %%%%%%Elapsed Time%%%%%%%%%%%
        %%%%%%%%%%Color%%%%%%%%%%%%%%       %%%%%%%%%%GrayScale%%%%%%%%%%
        %%%%%%8 millisecond%%%%%%%%%%       %%%%%%3 millisecond%%%%%%%%%%
        Background = Compute_Background(ROI,Alpha,Background,CurrentImage);

        %%%%%%%%%%%%%%%%%%%%%%%%Compute Segmentation%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%Elapsed Time%%%%%%%%%%%       %%%%%%Elapsed Time%%%%%%%%%%%
        %%%%%%%%%%Color%%%%%%%%%%%%%%       %%%%%%%%%%GrayScale%%%%%%%%%%
        %%%%%%7 millisecond%%%%%%%%%%       %%%%%%2.1 millisecond%%%%%%%%
```

```
        if(Color == true)
            [Binary FrameDiff] = Compute_Segmentation(CurrentImage,Background,Neta,true);
        else
            [Binary FrameDiff] = Compute_Segmentation(CurrentImageGray,Background,Neta,fal
        end
%         Binary = Binary_Process(Binary,2);
        %Binary = imclose(Binary,(strel('disk',2)));
        %Binary = imopen(Binary,(strel('disk',2)));

        %%%%%%%%%%%%%%%%%%%%%%Improve Segmentation%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%Elapsed Time%%%%%%%%%%%       %%%%%%Elapsed Time%%%%%%%%%%%
        %%%%%%%%%%Color%%%%%%%%%%%%%%       %%%%%%%%%%GrayScale%%%%%%%%%%
        %%%%%%10 millisecond%%%%%%%%%       %%%%%%2.7 millisecond%%%%%%%%
        if(Color)

            [BinaryImproved Shadow] = Improve_Segmentation_ShadowRemoval(Binary,Backgrounc
            %S1 = imerode(Shadow,strel('square',2));
            BinaryImproved = immultiply(BinaryImproved,~( imdilate(Shadow,strel('square',3
            BinaryImproved = Binary_Process(BinaryImproved,2);
            BinaryImproved = imfill(BinaryImproved,8,'holes');

            se = strel('disk',3);
            BinaryImproved = imerode(BinaryImproved,se);
            %se = strel('disk',3);
            BinaryImproved = imdilate(BinaryImproved,se);

            [BinaryImproved BI2] = splitting(BinaryImproved, CurrentImage, Background, Dir
            L = bwlabel(BinaryImproved,4);
            [B,L] = bwboundaries(BinaryImproved,'noholes');
            s = regionprops(L,'all');
            if( numel(s) == 0 )
                BinaryImproved = [];
            else
                %Storing features:
                num=1;
                if(s(1).Area<100)
                    %num=2;
                end
                boundary = B{num};
                delta_sq = diff(boundary).^2;
                perimeter = sum(sqrt(sum(delta_sq,2)));
                blob_size(numel(blob_size)+1) = s(num).Area;
                bounding_box_size(numel(blob_size),1) = s(num).BoundingBox(3);
                bounding_box_size(numel(blob_size),2) = s(num).BoundingBox(4);
                bounding_box_size(numel(blob_size),3) = s(num).BoundingBox(3)*s(1).Boundir
                solidity(numel(blob_size)) = s(num).Solidity;
```

```
                orientation(numel(blob_size)) = s(num).Orientation;
                centroid(numel(blob_size),:) = round(s(num).Centroid);
                roundness(numel(blob_size)) = 4*pi*s(num).Area/perimeter^2;
                4*pi*s(num).Area
                perimeter^2
                4*pi*s(num).Area/perimeter^2+0.001

            end


            [BinaryImproved ForegroundImproved BlobsAreas BlobsCentroids BlobsBoundingBoxes] = Com

        else
            [BinaryImproved ForegroundImproved BlobsAreas BlobsCentroids BlobsBoundingBoxes] = Imp
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%Compute Tracking%%%%%%%%%%%%%%%%%%%%%%%%%
        [Buffer_Centroid_Bound,Blobs] = Compute_Tracking(Buffer_Centroid_Bound,LengthBuff,Kalman,B

        clear L;
        clear B;
        clear s;
time=toc;
FPS = 1/time;
%display(strcat('Frames Per Second = ',num2str(FPS)));
%             if(i>2)
%             BinaryImprovedEroded =
%             Compute_Binary_Eroded(BinaryImproved,Blobs,0.2);
%             end
%             if(size(BinaryImprovedEroded) == size(CurrentImageGray));
%             ForegroundEroded = immultiply(BinaryImprovedEroded,CurrentImage(:,:,2));
%             ForegroundEroded =
%             immultiply(BinaryImprovedEroded,CurrentImage(:,:,2));
%             ForegroundEroded = immultiply(BinaryImprovedEroded,CurrentImage(:,:,3));
%
%             %make the area outside the blobs white in stead of black:
%             for h=1:size(ForegroundEroded,1)
%                 for v=1:size(ForegroundEroded,2)
%                     if( ForegroundEroded(h,v,:) == 0 )
%                         ForegroundEroded(h,v,:) = 255;
%                     end
%                 end
%             end
%
%             end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%Displaying%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        ImDisplay = Display(NbImX,NbImY,Dimension,TailleEspace,ImDisplay,ForegroundImprove
        %ImDisplay = Display(NbImX,NbImY,Dimension,TailleEspace,ImDisplay,ForegroundImprov
        imagesc(ImDisplay); axis image off; title('Counting People');
        Draw_Blobs_Properties(1,1,Dimension,TailleEspace,Blobs);

        if(TimePaused == 0.0)
            drawnow;
        else
            pause(TimePaused);
            drawnow;
        end

            end%End of cheat_fix
        i = i+1+frameskip;
        bufferIndex = bufferIndex+1;

        if( output_mov && useframe )
            mov = addframe(mov,ImDisplay);
        end
    end

    %Movie out%
    if( output_mov )
        mov = close(mov);
        disp('Movie done');
    end
    %Movie out%

else
    string = strcat('File "',filename1,'" does not exist');
    display(string);
end
```

B Source code, splitting.m

```
function [ BinaryImproved, BI2 ] = splitting( BinaryImproved, ForegroundImproved, Backgrou
%SPLITTING Takes a Binary image of detected moving object blobs.
    %Checks some statistics about each blob, then decide if it might be
    %more then one person/object in one blob, and tries to seperate them by
    %eroding until the blob splits into two medium-large objects.
    %Author: Roy-Erlend Berg

    BI2 = BinaryImproved;
    % Label each object in the current frame
    L = bwlabel(BinaryImproved,4);
    % Extract features from each labeled object
    s = regionprops(L,'Area', 'BoundingBox', 'Image', 'Solidity', 'MajorAxisLength', 'Minc
    if(numel(s)>0)
        for i=1:numel(s)
            dim = size(s(i).Image);
            numPixels = dim(1)*dim(2);
            filledPixels = s(i).Area;

            %Region info for this specific blob (coordinates from the
            %frame)
            x_from = ceil(s(i).BoundingBox(1));
            y_from = ceil(s(i).BoundingBox(2));
            x_to = floor(x_from+s(i).BoundingBox(3)-1);
            y_to = floor(y_from+s(i).BoundingBox(4)-1);



            %Blobs that consist of only shadows should have a distance
            %metric much lower then true moving objects
          %  BinaryImproved = clean_up_output(BinaryImproved, ForegroundImproved, Backgrou

            %Separate smaller blobs where bounding box is connected to the
            %edge of the frame
            if( x_from == 1 || y_from == 1 || x_to >= Dimension(2) || y_to >= Dimension(1)
                if( s(i).Area < 200 )
                    BinaryImproved(s(i).PixelList(:,2),s(i).PixelList(:,1)) = 0;
                    disp('For liten, bort');
                end
            end

            %Input blob size is so big it is probably not a single person
```

```
            %Check fill ratio
            %if( s(i).Solidity < 0.7 && numPixels > 4000 )
            if( s(i).Solidity < 0.7 && numPixels > 400 )
                BinaryImproved(y_from:y_to,x_from:x_to) = split(BinaryImproved(y_from:y_to,x_from:
                disp('dig blob');
            end
        end
    end
end

function imout = split(imin, numPixels, filledPixels, dim)
    expand = 0;
    for erodelvl=1:20
        imout = imerode(imin,strel('square',erodelvl));
        L2 = bwlabel(imout,4);
        s2 = regionprops(L2,'Area', 'Centroid', 'BoundingBox', 'Image', 'Orientation', 'MajorAxisL

        %Remove small objects caused by eroding
        nosmallblobs = false;
        for blobs=1:numel(s2)
            if(numel(s2)>=2 && ~nosmallblobs)
                %Initial threshold
                minsize=numPixels;
                blobnum=0;
                for i=1:numel(s2)
                    if( s2(i).Area < minsize )
                        minsize= s2(i).Area;
                        blobnum = i;
                    end
                end
                %New blobs with size less then 5% of originally filled pixels
                %are removed
                if(minsize < filledPixels*0.05)
                    imout(s2(blobnum).PixelList(:,2),s2(blobnum).PixelList(:,1)) = 0;
                    L2 = bwlabel(imout,4);
                    s2 = regionprops(L2,'Area', 'Centroid', 'BoundingBox', 'Image', 'Orientation',
                else
                    nosmallblobs=true;
                end
            else
                nosmallblobs=true;
            end
        end

        if( numel(s2)>=2 && nosmallblobs)
            expand = ceil(erodelvl/2);
```

```
            break; %stops the execution of 'for erodelvl=1:20'
        end
    end

    %Post processing to recover most of the original blobs shape and size
    if(numel(s2)>=2)
        newbounds = calc_newbounds(s2, dim, expand);
        imout = extract_blobs(newbounds, dim, s2, imin);
    else
        imout = imin;
    end
end

function newbounds = calc_newbounds(s2, dim, expand)

    newbounds(4,numel(s2)) = 0;
    for i=1:numel(s2)
        newbounds(:,i) = s2(i).BoundingBox;

        bound4 = newbounds(2,i)+newbounds(4,i)+expand;
        if(bound4 > dim(1))
            newbounds(4,i) = dim(1);
        else
            newbounds(4,i) = round(bound4);
        end

        bound3 = newbounds(1,i)+newbounds(3,i)+expand;
        if(bound3 > dim(2))
            newbounds(3,i) = dim(2);
        else
            newbounds(3,i) = round(bound3);
        end

        bound2 = newbounds(2,i) - expand;
        if(bound2 < 1)
            newbounds(2,i) = 1;
        else
            newbounds(2,i) = round(bound2);
        end

        bound1 = newbounds(1,i) - expand;
        if(bound1 < 1)
            newbounds(1,i) = 1;
        else
            newbounds(1,i) = round(bound1);
        end
```

```
    end
end

function imout = extract_blobs(newbounds, dim, s2, imin)
    extractedblobs(:,:,numel(s2)) = zeros(dim);
    for j=1:numel(s2)
        extractedblobs(newbounds(2,j):newbounds(4,j),newbounds(1,j):newbounds(3,j),j) = imin(newbo
        common = (extractedblobs(:,:,1) & extractedblobs(:,:,2));
        imout = imin-common;
    end
end

function BinaryImproved = clean_up_output(BinaryImproved, ForegroundImproved, Background, x_from,

    mask(:,:,1) = BinaryImproved(y_from:y_to , x_from:x_to);
    mask(:,:,2) = mask(:,:,1); mask(:,:,3) = mask(:,:,1);

    imback = Background( y_from:y_to , x_from:x_to, :);
    imback = immultiply(imback,mask) ;

    imfore = ForegroundImproved( y_from:y_to , x_from:x_to, :);
    imfore = immultiply(imfore,mask);

    imdiff = imabsdiff(imfore,imback);
    if( mean(mean(mean(imdiff))) < 25 )
        BinaryImproved(y_from:y_to , x_from:x_to) = 0;
    end
end
```

C Source code, test_shadow

The script used to test how the shadow algorithm should be configured

```
Back = Background;
Fore = ForegroundImproved;
Bina = BinaryImproved;
Shad = Shadow;
Curr = CurrentImage;
% Variables to change:
alpha = 0.45;
beta = 1;
%taus = Threshold_S;
%tauh = Threshold_H;
taus=80;
tauh=20;

figure
clear im;
while(1)
[Bina Shad] = Improve_Segmentation_ShadowRemoval(Binary,Back,Curr,Blob_Size_Min,alpha,beta
    Bina = Binary_Process(Bina,2);
    [Bina Fore BlobsAreas BlobsCentroids BlobsBoundingBoxes] = Compute_RegionProperties(Bi
    Bina = imfill(Bina,'holes');
    im(:,:,1)=uint8(Binary*255);
    im(:,:,2)=uint8(Shad*255);
    im(:,:,3)=uint8(Bina*255);
    imshow(im);
    drawnow;
    clear im;
    alpha = alpha+0.01;
    %beta = beta-0.1;
    %taus = taus-1;
    %tauh = tauh-1;
    if(alpha >= 1)
        alpha = 0;
    end
    if(beta <= 0)
        beta = 1;
    end
    if(taus <= 0)
        taus=100;
    end
```

```
    if(tauh <= 0)
        tauh=100;
    end

    pause(0.10);
end
```