CHALMERS

VCAS: Vision-based Chassis Alignment System

Master of Science Thesis

KRISTOFER WEIDOW
LOUISE GEIJER

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2008
Report No. EX070/2008

Report No. EX070/2008

VCAS: Vision-based Chassis Alignment System

Louise Geijer and Kristofer Weidow

Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden, 2008

VCAS: Vision-based Chassis Alignment System
LOUISE GEIJER, KRISTOFER WEIDOW

Report No. EX070/2008
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31 772 1000

Department of Signals and Systems
Göteborg, Sweden 2008

VCAS: Vision-based Chassis Alignment System

Louise Geijer and Kristofer Weidow
Department of Signals and Systems
Chalmers University of Technology

Abstract

In this thesis work, we explore the possibilities of substituting a laser-based chassis alignment system for the loading/unloading of container ships with a vision-based solution. We test this approach using several different image analysis methods, including chamfer matching, connected component analysis and Kalman filtering.

The methods are applied to test data consisting of still images and video sequences, taken from two different ports. The results presented for the automatic positioning are compared to ground truth data, obtained by manual inspection of the video sequences. Furthermore, the test images are distorted with noise functions to obtain an indication of the system's robustness towards varying weather and illumination conditions.

The error found in the ground-truth data comparison is approximately 1-3 pixels and the system's noise robustness is overall good. The results indicate some of the possible benefits with a vision-based system as well as some of the key limitations. As only a first step in the development process, the thesis' results and conclusions can help inspire future work.

Keywords: container handling, automation, computer vision, image analysis, chamfer matching, connectivity, Kalman filter

Contents

Preface

We would like to thank the persons involved at Qualisys and ABB Crane Systems for providing the opportunity for this thesis work. This especially concerns Magnus Berlander and Fredrik Müller at Qualisys and Björn Henriksson and Christer Johansson at ABB Crane Systems.

We would also like to thank other persons who have contributed in different ways, including Tomas Gustavsson, Artur Chodorowski and Mikael Persson from Chalmers, Sten-Åke Oveborn from the port of Gothenburg and all employees at Qualisys.

In closing, we would also like to extend thanks to our friends and families.

Chapter 1

Introduction

This master thesis is part of a collaboration between Qualisys and ABB Crane Systems.

Qualisys is a company situated in Gothenburg, providing optical motion capture solutions for a wide range of problems. The systems consist of high-speed cameras and software for tracking and analysis of the data. Today, the products are mostly used to capture the motion of attached, reflective markers together with an infrared light strobe, but the cameras can also be used in high-speed video mode. Investigation of marker-free applications of the systems is of interest.

ABB Crane Systems, located in Västerås, is one of the leading companies in the crane automation industry. Among other things, they supply equipment for automation in controlling the motion of container cranes, reducing the loading/unloading times and increasing the efficiency in the harbor.

Today, Qualisys is an OEM (Original Equipment Manufacturer) for ABB Crane Systems, delivering parts of an anti-sway solution for the cranes. ABB Crane Systems are interested in investigating the possibility of using vision-based solutions for other applications in the automation of the container handling process. Since Qualisys already has experience from delivering camera-based solutions to the particular outdoor environment, the collaboration in such an investigation is natural.

Possible applications of machine vision in container handling include general identification of containers and various truck chassis onto which the containers are loaded. One such application is to replace the existing laser-based solution for positioning truck chassis while loading and unloading the ship.

Trying to reproduce the functionality of an existing system is a good starting point since the necessary functions and requirements on accuracy and speed are clearly stated. If the problem can be solved successfully, the extension to other applications is possible. The problem includes identification and positioning of both containers and truck chassis and their large quantities make the attachment of markers impossible. Therefore, a marker-free approach with cameras used in video mode is necessary.

As a first step in the process of developing a complete solution for the application, this thesis mainly examines suitable image processing algorithms for replacing the laser-based system.

1.1 Thesis objectives

The objective of this thesis is to investigate if a marker-free vision-based system can provide the same functionality as the laser-based solution for the chassis positioning.

The key features to examine are

- Suitable image processing algorithms
- The system specifications needed
- The accuracy, speed and robustness achievable under various illumination and weather conditions

It is also of interest to see if there are additional things that can be accomplished by the use of a vision-based solution for this particular application.

1.2 Report outline

In the following chapter, we briefly introduce some basic concepts of the container industry and describe the current solution for chassis positioning. Next, based on the functionality of the existing system, we set up the necessary system functions for a corresponding vision-based solution.

In chapter 4, the collection of images and video sequences used in testing the algorithms are described and exemplified. Chapter 5 explains the algorithms used and the implementation for their special purposes in the application.

Results of the implementation are presented in chapter 6 after which the work is discussed in chapter 7. Finally, the appendix contains a manual for the graphical user interface created in the implementation phase.

Chapter 2

Background and current solution

The shipping and handling of containers is a multi-billion dollar industry, with worldwide operations. The evolution toward today's industry started in the 19th century, when shipping companies first began using containers, clearly improving the efficiency. The development process continued when purpose-built ships were taken into use from the 1950's. At this time, every shipping company had their own standards, which were not compatible. The subsequent standardization of shipping containers is widely considered one of the most important improvements of the logistic business during the 20th century. The first step to global standardization began with an ISO standard document, originally published in 1968, that primarily regulated the various dimensions of the containers. This standard is still in use today and regulates the container width to eight feet. The three most common lengths are 20, 40 and 45 feet [1].

Another important ISO standard was published later in 1968, regulating the marking and identification. The identification code consists of four letters followed by seven digits that uniquely identify the container and the shipping company that owns it. These codes are registered at every loading/unloading station and are used for tracking and security purposes [2].

2.1 Container cranes

This section is included to make the reader familiar with some concepts regarding the crane and its functions. In [3], a slideshow introduces many of the basic container crane parts and functions. Here, some of them are briefly described.

A container crane is used for loading and unloading containers onto and off a ship. It is usually mounted on rails so the entire crane can be moved along the dock. One distinguishes between the area towards the ship and water as the seaside or waterside and the area towards land as the landside. In figure 2.1, a couple of the container cranes used at the port of Gothenburg are shown.

Figure 2.1: Container cranes at the port of Gothenburg.

A crane can move the cargo in three different directions. The lowering and raising of the lifting device is called hoisting. The movement perpendicular to the dock, i.e. forward and backward is done in the trolley direction. To move in the direction parallel to the dock, the entire crane has to move. This direction is called the gantry direction.

The moving device of the crane is called the trolley and consists of the operator's cabin, the lifting device and the machinery for the hoisting. The trolley moves in the trolley direction along a horizontal girder.

The lifting device is usually called spreader. When lifting a container, the spreader is first lowered to the top of the container, after which so called

twistlocks are locked into matching openings at the container top's corners, called corner fittings. The container can then be safely hoisted and moved. The spreader is expandable so it can handle 20ft containers as well as 40ft and 45ft containers.

2.2 Current solution: CAS

The container crane is usually positioned over one or more parallel traffic lanes where truck drivers enter with either a loaded or an empty chassis. The trucks drive in the gantry direction, perpendicular to the trolley movement. Since the trolley has a very limited displacement capability in the gantry direction, the truck drivers need to position the chassis very accurately under the crane [4].

ABB Crane Systems has developed a system that automatically guides the truck driver to the correct position by the means of traffic lights. This minimizes the time needed to align a chassis to the correct position. The system is called *Chassis Alignment System* (CAS) and consists of a 3D-laser scanner, software and communication interface, traffic lights placed on the crane legs and controls and indicators in the operator's cabin. The laser is placed outside the girder between the landside and the waterside legs. The mounting height is about 30-50 meter which allows the laser scanner to cover all lanes and the full length of the chassis, independent of driving direction [5].

The truck drivers are directed to drive into a specific lane for receiving or delivering a container. In the operational cycle, this is referred to as the active lane and is where the laser will measure. The chassis are sometimes of different types, varying in the construction and have different configurations of loading positions. The different chassis types will have different ideal stop positions under the crane and in order to position a chassis correctly the chassis type must therefore be known [4]. If the chassis types are easily distinguished from each other, the CAS can detect and identify the chassis, otherwise the crane operator has to select the chassis type using a switch in the operator's cabin. For the chassis to be recognized properly, the different types used for loading/unloading must be loaded into CAS target database [5].

In a job cycle, the crane operator decides the active lane number, lane direction, the load position on the chassis and possibly the chassis type. Depending on the chassis type, the load position can be front, center or rear.

The different load positions also imply different ideal stop positions for the chassis [4]. CAS then receives an order from the crane control system to start measuring the position of a chassis given the input from the crane operator. The input also include the position of the crane, weather the operation is a set-down or a pick-up and possibly the chassis type. The operation type information comes from the twistlocks status over the ship. If the twistlocks are locked, the cycle is a set-down and CAS will start measuring for an empty chassis and vice versa. The laser then scans the selected active lane, and when the rear part of the chassis is found, a signal is generated and the traffic lights indicate this to the driver [5], [4].

The traffic lights consist of a control panel with arrows to indicate forward and backward driving directions and a two-digit display for the remaining distance to the ideal stop position. As the chassis moves forward in the lane, the traffic lights are updated according to the relative distance between the desired stop point and detected rear point. When the chassis is within predefined distances from the ideal position, different arrows light up on the traffic lights. As the driver reaches the ideal stop position the traffic lights indicate to the driver to stop. When the chassis has remained still at the ideal position for a fixed time, the CAS also measures the position in the trolley direction. This is done by measuring at the middle of the chassis. A skew measurement can also be done and the information about the chassis position in gantry and trolley direction is sent to the crane control system [5].

2.3 Why use a vision-based system?

The scanned images from the laser depend only on the distance to the object. This makes the system robust and reliable in all weather conditions [4]. The robustness, the coverage of all lanes independent of driving direction and the 3D information about the chassis position constitutes some of the major advantages with CAS. One of the disadvantages of the system is that it is expensive. It also has a limitation in that the laser can only measure at one lane at a time.

Given that a vision-based system can perform sufficiently under the different weather conditions, such a system could possibly decrease the cost. Theoretically, the system could also able to analyze all lanes at once, and thus have a larger range of operational control. However, the practical limitations are dependent on having software and hardware capable of sufficient performance.

Another feature of a vision-based system is that additional interesting information such as the container identification codes possibly could be extracted during the positioning. One disadvantage of a vision-based system with a single camera is the loss of the height information.

Chapter 3

VCAS: System specification

The *Vision-based Chassis Alignment System* (VCAS), is the proposed machine vision system that attempts to replace the 3D-laser with a camera and tries to imitate the functionality of CAS. In this chapter, we outline the specifications that need to be met and describe the different parts and issues of the setup of the vision-based system.

3.1 Camera issues

Apart from power supply and communication interface with other crane control systems, some of the other issues concerning the camera are briefly described here, even if the scope of the thesis does not entirely cover these areas.

3.1.1 Calibration

In order to be able to relate the position of a chassis found in the image coordinates to the position of the fixed point on the crane, the system needs to be calibrated.

The aim of the calibration is to obtain the camera matrix relating the image coordinates and the world coordinates. This depends on both camera parameters such as the sensor size, the focal length of the lens, the position of the principal point, and on the position and orientation of the camera [6].

The calibration procedure is not addressed here and need to be dealt with at a later stage. In the implementation we assume that the system is calibrated and that the camera matrix is available.

3.1.2 Field of view

The field of view is defined as the portion of the scene that projects onto the camera sensor. It depends on the camera focal length and the area of the sensor CCD and is can be defined as the angle 2ϕ where

$$\phi = \arctan \frac{d}{2f} \tag{3.1}$$

where d is the sensor diameter and f is the camera focal length [6]. The relation can be seen in figure 3.1.

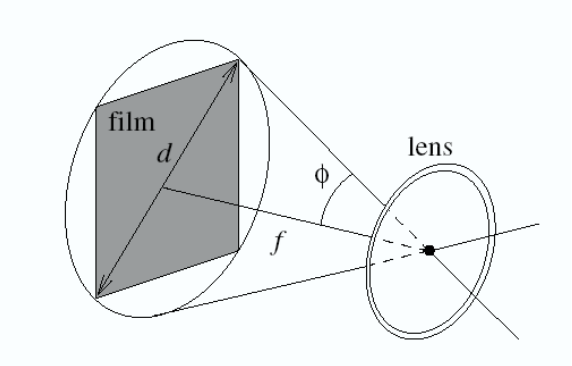

Figure 3.1: The field of view of a camera, defined as 2ϕ and depends on the focal length f and sensor diameter d [6].

Since different types of chassis implies different ideal stop positions, in order to be able to correctly control the traffic lights, the chassis need to be classified and its rear end needs to be located before the chassis is within a predefined distance from the ideal stop position. For CAS, this parameter is set to 3 meters [5]. Together with the requirement of the coverage of all lanes independent of driving direction, this set the needs on the field of view.

Being mounted outside the girder at a height of 30-50m, this enables the laser to have a measurement range of $\pm$ 26 degrees in both gantry and trolley direction [5], which can be used as a guideline for the necessary camera field of view.

3.1.3 Housing

The marine location can involve many different weather conditions which puts high demands on the housing of the camera. The existing system has the following environment specifications of the laser housing [5]:

- Protection is IP65 or NEMA 4X
- Withstands vibrations of 1G, 10-200Hz
- Operation temperature interval is -20 to +43°C
- Humidity interval is 0-100%

Since Qualisys already delivers systems with cameras housings that meet the requirements of this environment, these issues are at least partly solved already.

3.2 System performance

The vision-based system should be able to provide the accuracy and speed needed. These requirements are set by the performance specification of CAS [5]:

- The resolution of the laser is 20 mm
- The accuracy of the system is ±40 mm
- The traffic lights updating time is 0.5 s

These factors will require certain hardware specifications such as camera resolution and frame rate but also put demands on the image processing algorithms in order to meet the accuracy and have low enough computational load.

As mentioned earlier, only the software is considered at this stage. From the results of the implementation, we give an estimate of the needed number of pixels/cm and show the relative computational load of the algorithms.

3.3 System functions

Since the functionality of the systems should be equivalent, for the same inputs, the same outputs should as much as possible be provided.

In short, given the input of active lane number, lane direction, operational type and possibly the chassis type and load position, VCAS must be able to position the empty or loaded chassis to the correct position by controlling the traffic lights. Also, the position in the trolley direction should be measured.

In different parts of the operation cycle, VCAS should be able to

- Detect incoming motion in the active lane
- Identify the chassis type or container type
- Locate the center of the rear of a chassis
- Relate the rear position to the ideal stop position
- Control the traffic lights
- Measure the position in the trolley direction and measure the skew

The system should be operational under all weather and lightning conditions. Before being able to position the chassis, some information need to be obtained and loaded into the system in an offline phase. This involves a calibration of the camera to relate the image coordinates to the position of the fixed reference point on the crane. The systems also need to know where to start measuring given the input information regarding active lane number and direction.

Since different chassis types and different load positions will imply different stop positions under the crane, in accordance with CAS, the system needs to have a database of the different chassis type and load positions.

Extracting information from an intensity level image will require a certain contrast for the system to be able to discriminate the objects from the background. Under certain weather conditions, this will probably require the use of additional lamps. Due to the limitations of this thesis, such issues are not addressed here but the possible illumination distortions of the images are being accounted for.

The mounting onto a part of the crane will most definitely imply camera oscillations. This issue needs to be considered when relating the chassis position to the fixed point on the crane.

Chapter 4

The input data

4.1 Images from ABB

At the start of the project, a pack of images from a large port was provided. The images were taken with a handheld digital camera with a resolution of 2592×1944 pixels. Though most of them had been taken during daytime, there were a few with varying illumination and weather conditions. The image set consists of 4 images of a loaded ship, 15 chassis loaded with containers and 17 images of empty chassis taken from various angles.

Figure 4.1: A road chassis (left) and a terminal chassis (right), both images taken from the crane structure.

4.1.1 Synthetic video sequences

In order to test our algorithms on video sequences containing moving chassis, we had to create these synthetically, as we had no access to a site where these

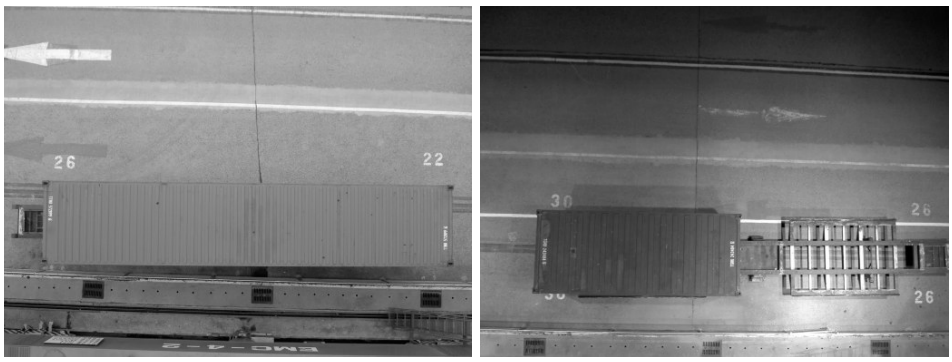

Figure 4.2: A 40ft container (left) and a 20ft container (right), both images taken from the crane structure.

types of chassis were being used. Using the backgrounds from the original test data, and by manually extracting chassis templates, we created video sequences where the incoming chassis had a certain pixels/frame velocity. The downside of this approach is that the sequence generator doesn't take factors such as illumination or perspective distortion into account.

4.1.2 Sequences from the port of Gothenburg

About halfway into the project, we made a visit to the port of Gothenburg, where we were allowed to collect test data from one of the cranes during operation. As the vantage point, we chose the platform that is normally used as an access ramp from the crane structure to the trolley. The location was chosen due to the broad overview of the lane area beneath (unobscured by the moving trolley), and the fact that while crane structure can be vastly varied between different cranes, most cranes probably have a point very similar to this one. Another reason to choose this point is that operations at the port of Gothenburg take place behind the crane instead of under it. As such, to film from the same location as where the CAS is usually located would not give satisfactory test data while this location provides the possibility to obtain data from both under and behind the crane. From this location, we took still image and video sequences of the operational area.

From the acquired video sequences, we selected three sequences containing an entire load cycle. The cycle begins with an empty lane, in which a straddle carrier enters. It drives up to the designated stop position, drops the container and then continues down the lane before it disappears. Two of the cycles contained single 40 feet long containers while the third contained a

twin load of two 20 feet long containers.

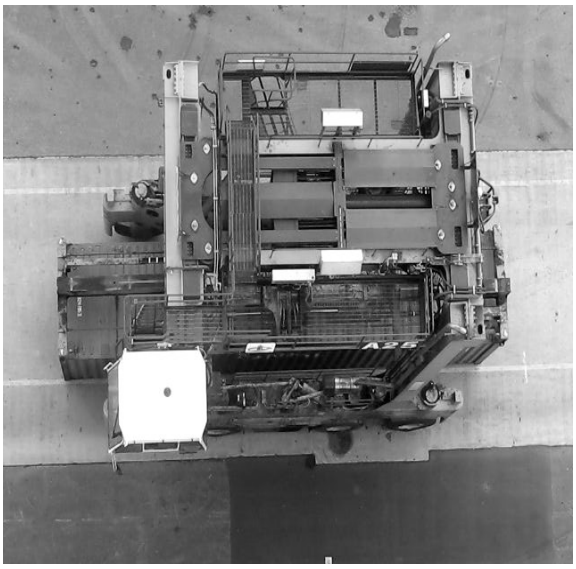

Figure 4.3: A straddle carrier from the port of Gothenburg, as seen from the access ramp between the crane structure and the trolley.

The test sequences taken on site are all acquired during a single visit, from one crane. As such, the background is very similar in all sequences. The weather is also somewhat similar, although it was sunny with a few clouds blowing past, giving examples of sun reflexes and rapidly changing illumination in one of the video sequences.

4.2 Noise distortions

Natural changes in illumination and weather are very hard to simulate realistically. However, as the system is required to be robust, testing the performance of the algorithms after the addition of noise to the test data should provide an indication of whether it will be capable of handling different conditions. As such, we have chosen four different noise types to distort the frames. For all noise types, we go through all the pixels of the frame, and change the input pixel intensity s to the output pixel intensity t. As the images are 8-bit, the range of the pixel intensity is [0...255]. Fraction changes

are rounded to the nearest integer.

Gaussian noise

The addition of Gaussian noise uses a normally distributed, random variable z. The output value for the pixel intensity is calculated by adding z to the input value of the pixel intensity:

$$t = s + z \quad , \quad z \sim N(\mu, \sigma) \tag{4.1}$$

where t is the output pixel value, s is the input pixel value, μ is the desired mean of the intensity change, and σ is the standard deviation of the distribution. A negative mean will cause the output values to take on generally lower intensities, while a positive gives a higher probability for the output pixels to become brighter. The variance determines the range of the intensity change, where a high variance will lead to a higher degree of noise [7].

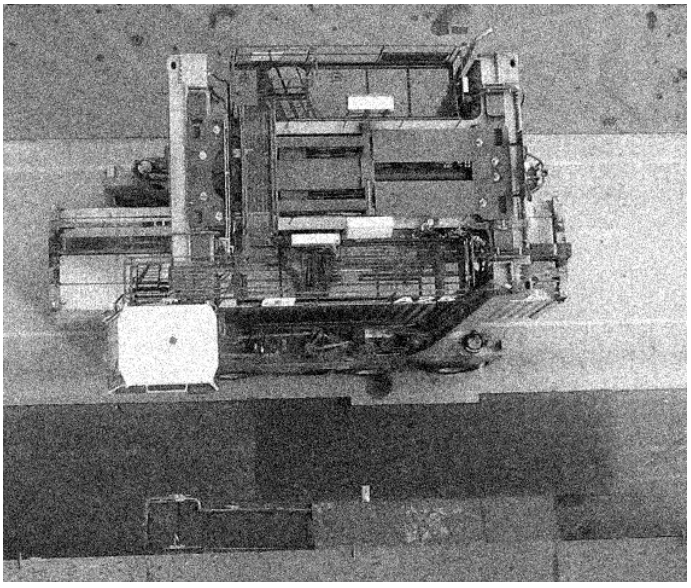

Figure 4.4: Test data corrupted with Gaussian noise.

Poisson noise

The addition of Poisson noise uses a random variable z with a Poisson distribution, $z \sim Po(\lambda)$. The λ parameter in the Poisson distribution is its

expectation value (mean), and in this implementation we set the parameter to s, that is the input intensity value. The output intensity t is simply updated with the value z that is drawn from the distribution:

$$t = z \quad , \quad z \sim Po(s) \tag{4.2}$$

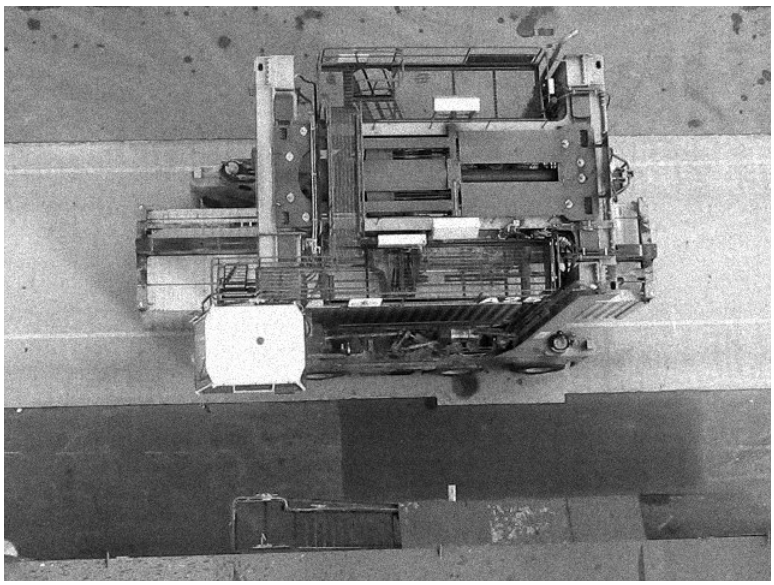

Figure 4.5: Test data corrupted with Poisson noise.

Speckle noise

Speckle noise is a multiplicative noise function which uses a random variable with a uniform distribution. The parameters for the distribution is calculated from the variance σ^2 and the noise function is:

$$t = s + sz \quad , \quad z \sim U(-\sqrt{3\sigma^2}, \sqrt{3\sigma^2}) \tag{4.3}$$

where t is the output pixel value and s is the input value.

Salt and pepper noise

The three noise types above all give similar outputs, where the disturbance is related to the original pixel value. To broaden the robustness test, we have decided to include salt and pepper noise among the other noise types. While

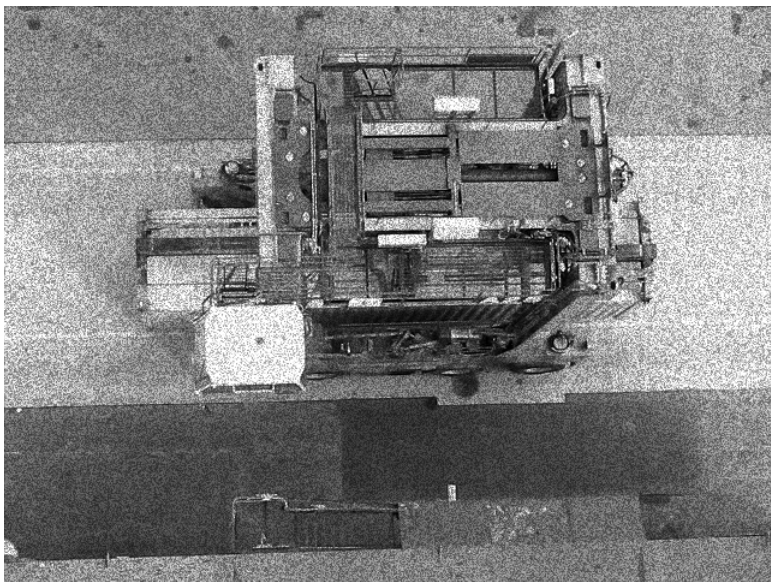

Figure 4.6: Test data corrupted with speckle noise.

salt and pepper noise is not a good description of any naturally occurring phenomenon, it is something that stands apart from the other noise types and will thus broaden the condition types the algorithms have been tested for.

The noise function takes a probability p as an input, and for every pixel there is an equal probability ($p/2$) that it will either become white (intensity level 255) or black (intensity level 0). The probability that the pixel remains unchanged is thus $1 - p$.

4.3 Pre-processing

The original image set provided was grayscale, and while the data obtained at the port of Gothenburg was in color, we choose not to use or rely on that information in any of the algorithms. This choice was made to limit the number of parameters in the algorithms, simplify the process and not have to investigate how illumination and weather affects colors in three dimensions instead of one.

The resolution was set to imitate the capacity of Qualisys' Oqus1 series, which have a sensor size of 640×480 pixels. The video sequences at the port of Gothenburg were obtained with a camera having this very resolution,

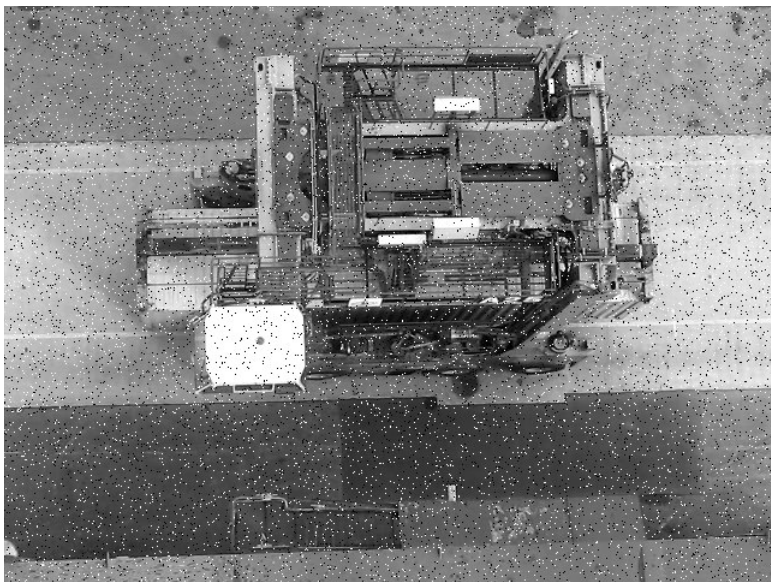

Figure 4.7: Test data corrupted with salt & pepper noise

although it automatically used a MPEG compressor which compressed the images to about 1/15 of the raw data size. In a real hardware implementation, a MPEG compressor must be avoided, but since the acquired sequences only contained straddle carriers, and the algorithm for tracking these carriers does not involve the edge map, it did not matter if the edges could be distorted by the compressor.

All still images acquired with a higher resolution than 640×480 were downscaled and also converted to grayscale. This was done for a few reasons, mainly to be able to perform the algorithms at higher speed, and also so that the still images could be treated as single frames of a video sequence. All conversions from RGB to grayscale are made by converting the RGB image values to NTSC coordinates, setting the hue and saturation components to zero and then returning to RGB space [8].

Chapter 5

Implementation

The focus in this chapter is the algorithms used and how they are implemented. The goal of the system is to guide the driver to the correct position by means of traffic lights. As outlined in chapter 3, the necessary intermediate system functions are

- Detect incoming motion
- Identify and locate the chassis, container or straddle carrier
- Relate the found position to a fix position

In the following sections, we first describe the proposed algorithms for the different parts of the system in a general sense and then the implementation of the algorithms for their special purposes in VCAS is described.

Unless otherwise stated, all operations described take place on 8-bit grayscale images containing $W \times H$ pixels, where $W \times H$ is the resolution. 8-bit images means that every pixel in the image can have intensity values in the range [0...255]. Basic image processing operations are explained in their relevant context in the text, but for a thorough and detailed explanation, we refer to a course book in image processing such as [7].

5.1 Motion detection

The operational cycle begins by detecting incoming motion in the active lane. This will trigger the search for the chassis, container or straddle carrier. Being able to detect moving objects is an important part of almost every vision system. A common technique for motion detection is to perform background subtraction. The pixels in the current frame are then compared to some kind

of background model. Pixels that deviate significantly from the background model are considered as foreground objects. Most background subtraction methods follow the same scheme including four major steps; preprocessing, background modeling, foreground detection and data validation [9].

Preprocessing of the raw data is usually simple image processing including temporal and spatial smoothing to reduce camera noise and environmental noise such as rain and snow. Frame-size reduction is also commonly used to lower the amount of data to be processed [9].

The next step is the subtraction of a background model from the current frame. The simplest model is the use of a static background, captured at a frame where no foreground objects appear. This can be used when the background does not change much over time.

Frame differencing uses the previous frame as the background. This works only for sequences having particular object speed and frame rate [10].
Median filtering is one of the most commonly-used background models. The background is then defined as the median at each pixel location a number of previous frames stored in a buffer. The background can also be modeled as the average of the frames in the buffer [9].

Running-average instead uses an updating of the background model B according to

$$B_{i+1} = \alpha F_i + (1 - \alpha) B_i \tag{5.1}$$

where α is the learning rate and F_i is the current frame [10].

All these background models have low computational load but the memory requirements are higher for the models using a buffer of previous frames [10].

The resulting difference image is then thresholded to find the foreground. The threshold is usually determined experimentally and three common thresholding methods are mentioned here [9].

The simplest and most commonly used thresholding method is

$$|F_i(x, y) - B_i(x, y)| > T \tag{5.2}$$

Some implementations use normalized statistics for the foreground detection

$$\frac{|F_i(x, y) - B_i(x, y) - \mu_d|}{\sigma_d} > T \tag{5.3}$$

where μ_d and σ_d are the mean and the standard deviation of $F_i(x,y) - B_i(x,y)$ for all pixel positions (x,y).

The threshold should ideally be lower for regions of low contrast. One modification that somewhat incorporates this is to threshold the relative difference

$$\frac{|F_i(x,y) - B_i(x,y)|}{B_i(x,y)} > T \tag{5.4}$$

The result of the thresholding is a binary mask for the foreground. As a final step, the foreground pixels sometimes need to be validated. This usually involves removing false positives and improving the foreground mask [9].

5.2 Chassis classification and localization

The problem After having detected a chassis in the active lane, we need a way of recognizing the chassis in order to be able to position them. Questions that need to be solved are: What information in the image can be used to solve this? How can the chassis be recognized from this information? How will we know when the chassis has been recognized and found?

Regarding what information, or which features, in the image to use, these have to be robust enough for changes in illumination and weather conditions and be simple to extract. Also, the selected feature must incorporate enough information so that the object can be recognized.

In order to recognize the chassis from the features chosen in the image, we also need to have the same features for the chassis to be recognized. A model of the chassis must, in accordance with CAS, be loaded into the system before any operation. This model should also incorporate the information needed to relate the found position to the fixed reference point on the crane. The model also needs to be flexible so that it can be used for all lanes and work even if some of the chassis is occluded in the image.

We then need an algorithm that tries to find the features of the model in the extracted features of the image and determine if the chassis is found or not.

Proposed solution The different weather conditions and illumination changes in the outdoor environment make it hard to explicitly use the grayscale information in the images since this information will change over time. The shape

of the chassis and containers will however not change under different lightning conditions, this is also somewhat how the laser recognizes the objects in the system used today. A representation of shape is to use the edges of an object. Edges or object boundaries are places in the image matrix where we have strong changes in image intensities. An edge detection is used to identify these changes [11].

Using edges for the recognition also has a another major advantage in this application since the relative position that controls the traffic lights is located at the edge of the rear beam of the chassis. This means that the chassis can be recognized and localized in one step which lowers the computational load.

The corresponding features of the chassis and containers models are the contours of the known shapes. In order to get the flexibility needed, the models should incorporate all the different perspectives for the different lanes and lane directions. Using an extensive amount of templates could be very memory consuming. Instead, letting the templates be parameterized could solve the issue. The template models are then built up by defining the vertices of the polygonal representing the outer shape. This information could perhaps be obtained by importing a CAD-model of the chassis types. Having only the vertices as the model, this gives a parameterized template model which can be transformed by means of scale, rotation or other transformation parameters. Ideally, the vertices should be transformed by a projective transform, using the camera matrix from the calibration and the spatial information of the active lane. After a suitable transformation, edge models can be created by filling in the pixels between two successive vertices. Additional information such as the load positions and the rear measuring point should also be added to the model.

It then remains to find a suitable algorithm for the matching of the two patterns. A matching procedure using these concepts for localization and classification while being simple and easy to implement is chamfer matching [12].

5.2.1 Chamfer matching

At the core of the implementation for identification and localization for the chassis and containers lies a pattern matching algorithm called chamfer matching. The algorithm is well-known and commonly used in object recognition in computer vision [13].

The algorithm is used to search for a specific pattern in an image, this is called template matching. The patterns consist of feature points extracted from the template to be found and the same features are extracted from the image. Chamfer matching most commonly uses points on edges as the features but other interest points can be used [13].

Matching two patterns need a measure of the similarity. In chamfer matching, this measure is a generalized distance between the edge points in both model and the image. If the set of feature points in the model is $\mathcal{U} = \{\mathbf{u}_i\}_{i=1}^n$ and the set of feature points in the image is $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^n$, the edge distance d measure of the similarity of the two edge patterns and is defined as [12]

$$d(\mathcal{U}, \mathcal{V}) = \frac{1}{n} \sum_{u_i \in \mathcal{U}} \min_{v_j \in \mathcal{V}} ||u_i - v_j|| \tag{5.5}$$

The search for a pattern in an image means calculating this edge distance at all pixel locations. This can be efficiently computed if the image is first distance transformed meaning that each non-edge pixel is given a value that is the distance to the nearest edge pixel. The edge pixels get the value zero. The distance to each edge pixel should ideally be the Euclidean distance but often an approximation is used. The calculation of the distance transformed image is done using a two-pass sequential filtering with local operators [14].

Having the distance transformed image, the edge distance at a specific location is easily found. This is done by superimposing the template edges on the distance image at the pixel location, summing the values hit by the template edge pixels and finally taking the average. Finding the pattern in the image then simply becomes a linear filtering, using the template edges as a summing filter. A perfect fit between the two edge patterns will result in edge distance zero, as each template edge pixel will then hit an edge pixel in the distance image, having the value zero. An imperfect fit result in a non-zero edge distance but the smaller the value is, the better the match [14].

The position corresponding the minimum edge distance will be the position where the searched feature pattern best fit the image feature pattern. The edge distance value in this position also gives a measure on how good the patterns match. In this way, the result of the chamfer matching, not only give the best location but also a value for the similarity by the edge distance measure meaning that we can also use it to classify objects. If one out of two objects is to be found in an image, both model edges are correlated with the distance transformed image edges and the minimum edge distance are found

for both models. The model corresponding to the lowest value is the better match. In figure 5.1, the chamfer matching algorithm is depicted.

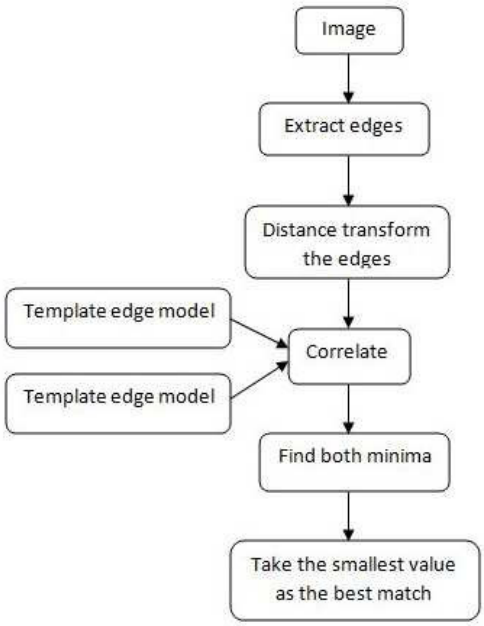

Figure 5.1: Chamfer matching algorithm, depicting how to identify which one of two templates that best matches the image edge pattern.

Chamfer matching is simple to implement and shown to be robust to clutter and occlusion but the matching is not invariant to scale and rotation. The algorithm is then suitable for situations where the templates can incorporate the different appearances of the objects in the image [12].

A couple of pointers can also be given here. Since the matching is a linear filtering, the best found position in the image corresponds to the upper left corner of the template edges. The linearity of the filter also means that the correlation can be done in the Fourier domain which can lower the computational load. Since the method uses both image and models edge maps this requires a robust edge detector and good template representation.

Edge extraction

For a good matching performance, the algorithm also needs a good edge extraction technique. Ideally, the edge detector should locate all the important edges, the resulting edge map should be binary with a single pixel wide edges to suit the chamfer matching algorithm and the detected edges also need to be optimally placed at the real edges in the image.

One of the most used edge detectors is the Canny algorithm for edge detection. The detector was first proposed with the objective to fulfill the above requirements and its simplicity and accuracy has made it widely used in computer vision applications [7].

The Canny edge detector has three parameters; the standard deviation of a Gaussian smoothing filter and two thresholds. The choice of parameters will affect the results of the edge detection. The smoothing filter is used for noise reduction and is the first stage of the edge detection. In a later stage, the two thresholds are used to make a hysteresis thresholding [7].

5.3 Tracking

When tracking a moving object it is desirable to be able to predict where the object will appear in the consequent frame. This has two major advantages. Firstly, the computational load can be lowered since the search window used can be set smaller. Secondly, it can also be useful if the position of the object is lost over a few frames, it can then be recovered in later frames based on the predicted position. A simple and widely used predictive filter is the Kalman filter. This has been implemented here to predict the position of a chassis in the next frame.

5.3.1 The Kalman filter

The Kalman filter is a recursive, linear filter that estimates the state of a discrete dynamic linear system based upon information about the system in form of noisy measurements taken at the previous time step [15].

The true state is assumed evolve in discrete time steps according to the difference equation

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{5.6}$$

where x_k is the state vector at time k, A is the state transition matrix, B is the control input model, u_k is the input vector at time k and w_k is a random variable.

The state vector x of a model consists of all parameters needed to determine the objects configuration. An object undergoing a one-dimensional uniform motion can for example be described by a state vector consisting of position and velocity. The state transition matrix describes the dynamics of the system in absence of inputs and noise. The control input model B relates any possible inputs u to the state [16].

The random variable w_k describes the process noise. For example, if the model assumes that the acceleration is constant between two successive time steps, any non-constant acceleration will then be considered as process noise.

In every iteration, the Kalman filter predicts the process state, then receives feedback from the system in form of the noisy measurements of the true state and finally corrects the prediction based on the feedback [15]. These measurements are described by the vector z

$$z_k = H_k x_k + v_k \tag{5.7}$$

where H is the measurement matrix relating the state vector to the measurement and v_k is another random variable describing the measurement noise.

The Kalman filter assumes w_k and v_k to be independent, normally distributed random variables with zero mean:

$$p(w) \sim N(0, Q) \quad ; \quad p(v) \sim N(0, R) \tag{5.8}$$

where Q and R are the noise covariance matrices [15].

The equations of the Kalman filter can be collected into two different groups. The time update equations projects the state in time, evolving the estimated state at the previous time step to estimate the state at the current time step. The estimated error covariance is also computed [15]. The time update equations are

$$\hat{x}_{k+1}^{-} = A\hat{x}_k + Bu_k \tag{5.9}$$

$$P_{k+1}^{-} = A_k P_k A_k^T + Q_k \tag{5.10}$$

The other group of equations, responsible for correcting this estimate due to the measurements, is called the measurements update equations. First, the

Kalman gain is computed. This decides how much of the difference between the actual observation and the estimated observation variables that should be used in the correction [16]. The estimated state from the time update equations are then corrected with the weight factor of the Kalman gain to obtain the filter output for the state estimate. The error covariance is then also updated due to the weighted measurement correction. The equations for the measurement update are [15]

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + R_k} \tag{5.11}$$

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H_k \hat{x}_k^-) \tag{5.12}$$

$$P_k = (I - K_k H_k) P_k^- \tag{5.13}$$

As seen in equation (5.11), if we have a large measurement error the Kalman gain will be small and less credibility will be given to the measurement when computing the next state.

When implementing the filter, process and measurement noise covariance matrices Q and R need to be set. This often involves measurements prior to operation of the filter and empirical testing. The filter must also be initialized with both a state vector and an error covariance matrix [15].

5.4 Binary image processing

5.4.1 Morphological operations

The system uses binary morphological operations, mainly erosion and dilation, for post-processing. These take a structure element S as input. The structure element is a small, binary sub image whose form is dependent on the kind of operation that is of interest.

Erosion

The erosion method takes the structure element S and the image I as input. The erosion of I by S is then defined as

$$I \ominus S = \{z | (S)_z \cap I^C = \oslash\} \tag{5.14}$$

With $\oslash$ being the empty set and I^C is the complement of I, the erosion can be considered the set of all points z for which S, translated by z, does not share any elements with the background. The translation of a set S by the point

$z = (z_1, z_2)$ is acquired by taking the (x, y) coordinates in S and replacing them by $(x + z_1, y + z_2)$. In practical terms, what the erosion accomplishes is to eliminate pixels in the image that doesn't fit into the structure element. This is useful for cleaning up small islands of noise pixels spread throughout the image. For this purpose, a square-formed structure element is generally preferred.

Dilation

The dilation is very similar to erosion, but achieves the opposite goal. Instead of eliminating pixels, dilation expands the structures. It is thus used to bridge gaps between components. Structure elements often take the form of vertical or horizontal stripes. The dilation of I by S is defined as

$$I \oplus S = \{z | \hat{S}_z \cap I \neq \oslash\} \tag{5.15}$$

We note that $\hat{S}$ is the reflection of S, that is the set of pixels in S whose (x, y) coordinates have been replaced by $(-x, -y)$. The dilation then becomes the set of displacements that causes $\hat{S}$ and I to overlap for at least one pixel.

Open and Close

The subsequent combinations of erosion and dilation operations are called Open and Close operations. Open is erosion followed by dilation, and it is used to smooth contours and eliminate sharp peaks and noise. The Close operation is dilation followed by erosion and is used to fuse gaps, without a lasting bloating of the entire structure [7].

5.4.2 Connected component analysis

Connected component analysis comes from graph theory, and is almost exclusively used on binary images. We define a path from pixel s (with coordinates (x_s, y_s)) to pixel t (with coordinates (x_t, y_t)) as a sequence of pixels with coordinates

$$(x_0, y_0), (x_1, y_1), ..., (x_n, y_n) \tag{5.16}$$

where the pixels (x_i, y_i) and (x_{i+1}, y_{i+1}) are adjacent for $0 \leq i \leq n - 1$ and $(x_0, y_0) = (x_s, y_s), (x_n, y_n) = (x_t, y_t)$. If we let S represent a subset of pixels in a processed frame, two pixels s and t are said to be connected if there exists such a path between them. For any pixel in S, the set of all components

connected to it is called the connected component. The size of a connected component is defined as the number of pixels it contains [7].

5.5 VCAS implementation

The implementation of the selected algorithms is made using Matlab (Matlab 7.0, R14), with heavy use of the provided image processing toolbox (IPT). The biggest advantage when using Matlab is the vast amount of available functions in said toolbox, which speeds up the implementation phase, and allows for more approaches to be tested in a shorter time span. The disadvantage is that Matlab itself is very slow, which means we can't get reliable data on how fast the proposed algorithms will process the information (and thus at what update frequency the positioning will work).

The implementation of VCAS is done with a GUI (graphical user interface) to make it easier to experiment with, and to provide a simulator feeling. A lot of work has been put into making the GUI intuitive and easy to work with. With the GUI, the user can load video sequences and run the analysis with our implemented algorithms. The source code is commented to enable further extensions and new algorithms to be incorporated without having to rewrite the entire program. This provides good opportunity for further extensions of VCAS, as well as verification possibilities for future test data. A manual for the GUI is provided in appendix A.

5.5.1 Motion detection

The detection of incoming motion is implemented using background subtraction with the simplest background model, a static background. Using predefined entering and exiting areas of the active lane, these areas are cropped and used as the static background models of the areas. No preprocessing of the data is done.

During the motion detection, the background is subtracted from the entering area. Foreground pixels are detected with a statistical threshold, using the Matlab IPT function *graythresh* which is a global image threshold using the Otsu's method [8]. No data validation is done and motion is considered detected when enough foreground pixels are detected in the entering area. The same procedure is used in some sequences for detecting the outgoing motion. A flow chart of the motion detection implementation can be seen in figure 5.2.

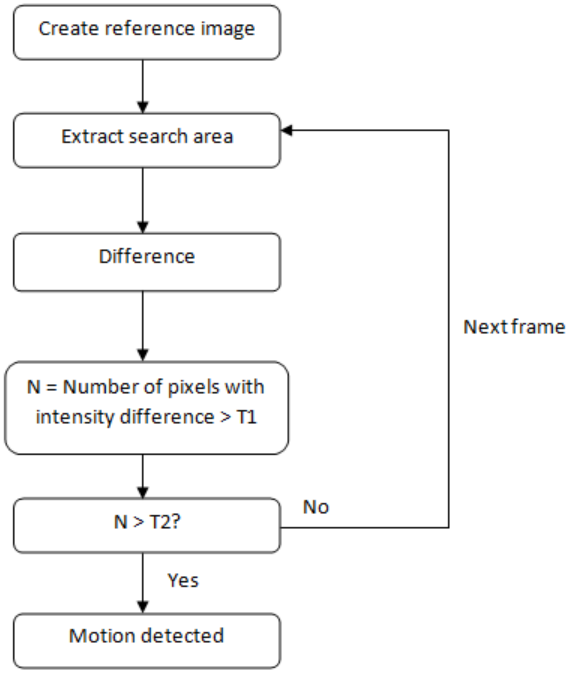

Figure 5.2: Flow chart depicting the motion detection algorithm. T1 and T2 are thresholds.

5.5.2 Chassis classification and localization

The template models

The template models used for the matching consist of vertices defining the polygonal shapes of the empty chassis' outer beam configuration and the roof of the containers.

Two chassis templates models have been created, defining the shapes of a terminal chassis and a road chassis. The container templates created are for 20ft and 40ft containers. An example of a terminal chassis template can be seen in figure 5.3.

Since different lanes will have different perspective distortions, the templates need to be spatially transformed with parameters set for a specific lane so that

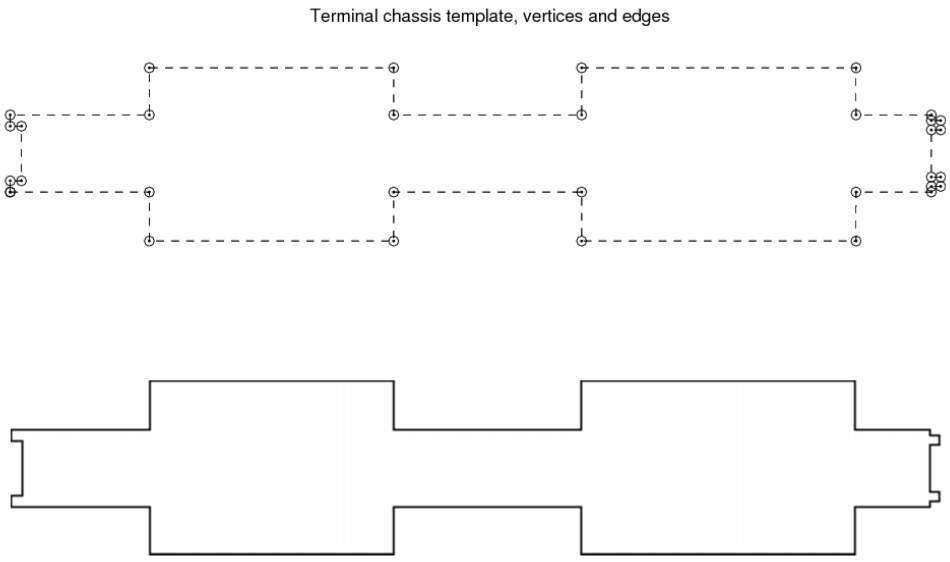

Figure 5.3: Terminal chassis templates and corresponding edge map.

the perspective can be accounted for. This can be done using the camera matrix obtained from the camera calibration. Different templates will be loaded into the system depending on lane and direction. Which parameters to choose need to be determined in some sort of training phase and of course a calibration of the camera is needed to obtain the camera matrix. However, since no camera matrix is available we use the simpler affine transformation in this implementation. This includes rotation, scaling and shearing in the x- and y-directions. The affine transformation has the general form [7]

$$[x', y', 1] = [x, y, 1]T \tag{5.17}$$

where $[x', y']$ are the transformed image coordinates, $[x, y]$ are the original image coordinates, and T is the transformation matrix built up by multiplying the different transformation matrices for rotation scaling and shearing.

$$T = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & s_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ s_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

where θ is the rotation angle, s_x, s_y, c_x and c_y are the shearing and scaling in the x- and y-directions respectively. The transformation is applied to the template vertices by using the Matlab IPT functions *maketform* and

tformfwd [8].

After the transformation, the template edges are created by filling in pixels between two successive vertices. A flow chart of the edge template construction is shown in figure 5.4.

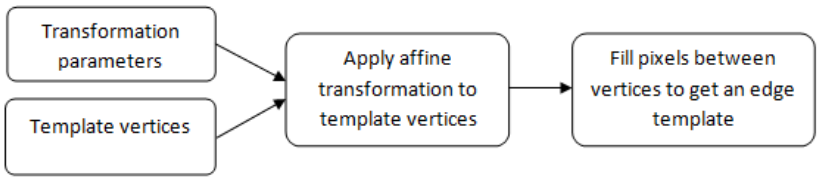

Figure 5.4: Flow chart depicting the construction of the template edge models.

Also included in the template structure is the rear measuring point and the code extraction areas for the container templates. In a later implementation, if existing, the relative position shifts needed if the container is to be loaded at the front, rear or center of the chassis, should also be included in the template structure.

Container code extraction

The container codes are used in verifying a position found by the chamfer matching. The extraction of the codes is an interesting additional feature of using a vision-based system.

The algorithm begins by extracting the search area of the code incorporated in the template structure, given the found position. The morphological operation dilation is then used with a square structure element. The median value of the search area is calculated and a threshold is set to a value higher than the median. The resulting image from the dilation is thresholded and then a connected components analysis is performed on the resulting image. The found bounding boxes of the connected regions are then checked to have the right proportions of a container code. If the correct shape has been identified, the code is assumed found and the region can easily be extracted from the image using the bounding box and displayed. In figure 5.5, these steps are illustrated.

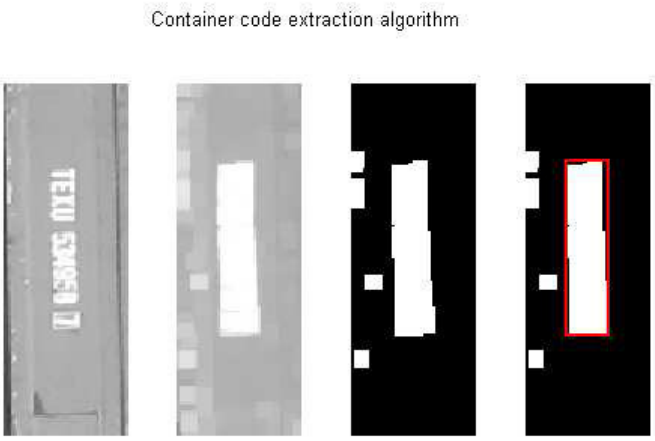

Figure 5.5: Images depicting the container code extraction algorithm. From left to right: the original extracted area from the matched position, the resulting image after the dilation and thresholding and finally the identified code area with the bounding box displayed.

Single image matching

A number of single images have been used in the test implementation of the chamfer matching.

The implemented procedure mostly follows the same scheme.

- Transform the template vertices with pre-set parameters
- Extract edges from the image
- Calculate the distance transform for the extracted edges
- Correlate the distance image with the template edges
- Find the minimum of the correlation

The template model can then be superimposed on the image located at the minimum position found to display the matching results.

The template transformation parameters are rotation, scale and shear. This need to be set before any matching can take place. Since the provided test images are taken at different camera angles and scales, the parameters had

to be set differently for each image. A real implementation will not need this in the same way but the different perspectives for the different lanes and lane directions must be determined in a training phase.

The edges are extracted from the image using the Canny edge detector. This is one of built-in edge detectors incorporated in the function *edge* in the Matlab IPT. The parameters used is the default value of one for the standard deviation of the Gaussian smoothing filter, and the thresholds are set automatically by the function using the found gradient magnitudes.

The distance transform of the resulting image is then obtained using the function *bwdist* [8]. The correlation of the template edges is done in the Fourier domain.

For empty chassis matching, the two template edges for terminal chassis and road chassis are correlated with the distance image and minimum values and positions for both correlations are found. The template model corresponding to the lowest value is taken as the match.

For the images depicting a single container or chassis, the classification is done somewhat differently. In the container matching, we the first try to locate a 20ft container and if this fails, we try to locate a 40ft container. This is done because of the possibility of a twinload and the container code verification used. This verification step is described more thoroughly in the next section.

Loaded chassis classification and positioning

The implementation for a pick-up operation is tested for two synthetically created sequences, one with a 20ft container and the other with a 40ft container. The scenario could also be a twin load of two 20ft containers or a 45ft container but the extension is quite straight forward and is not implemented here. The synthetic sequences are created to see the principles and methodology of using only parts of the template edges for matching, the use of windowing and the implementation of the Kalman filter.

In figure 5.6, a flow chart for the positioning of a known container type is shown. The classification is done by checking for the existence of container codes in the correct positions.

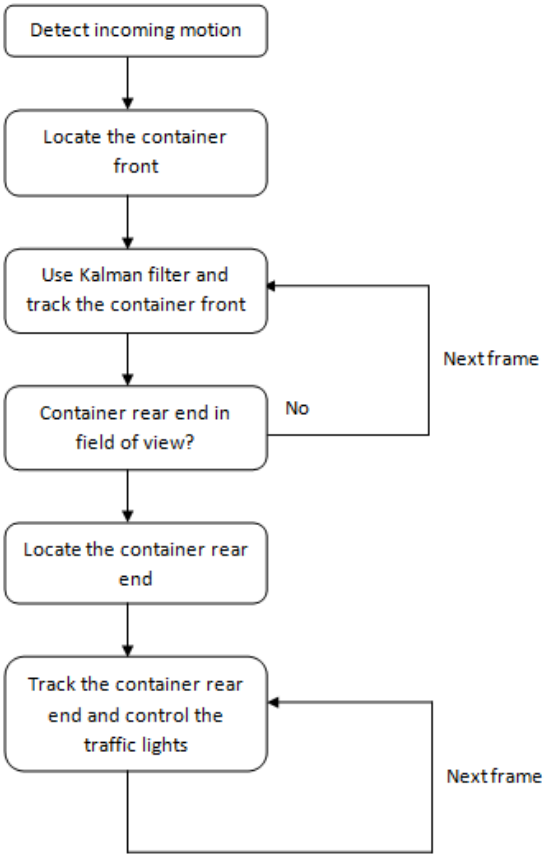

Figure 5.6: Flow chart depicting the positioning of a known container type.

The operation cycle starts off with detection of motion in the active lane. After an incoming object is detected the container front end is first located. The appearance of the container front is the same regardless of the container type. Edges are extracted from the pre-defined search window and the resulting edge map is distance transformed.

The front part of the container template edge map is then correlated with the distance transformed edge map and the minimum value is found. The corrugated container roof might produce strong edges and this can produce false minima in the correlation function. The position corresponding to the minimum is therefore verified by checking the existence of the container identification codes and twistlocks in the correct relative positions and the in the corners.

The search areas for the codes are incorporated in the template structure and the twistlock search areas can easily be set from the four template vertices. If the position can't be verified, the minimum and a possible neighborhood are removed from the correlation. A new minimum is then found and tested for verification. The number of tries can be set due to computational load. The extra step minimizes the false positives but sometimes makes the positioning in the current frame fail.

For the twistlock existence detection, we use the fact that they are almost always appear to be darker than the container surface. Having the position tested for verification, the twistlock area is extracted from using a neighborhood at the template model vertices. The morphological operation erosion is then applied to the extracted area using a square structure element. The resulting image is inverted and thresholded. The twistlock is assumed found if we can find connected pixels after the threshold. This verification could be improved since it does not always give pleasant results. If a position is not found in a certain frame, this mainly depends on the twistlock existence check.

When the position of the container front has been verified, the search window for the next frame can be set smaller. A Kalman filter is initialized and used to track the position.

In the next frame, the process is repeated, now using the smaller search window. The edges are extracted, distance transformed and correlated with the template edges to find the position. The found position is finally verified. In this way, the front end of the container is then tracked until the rear end of a 20ft container possibly is in the field of view. This can be determined by relating the rear measuring point to the image width. With the rear end possibly is in the field of view, this is verified or discarded by searching for the container code in the area for a number of frames.

If the code can be found, the container is classified as a 20ft container. The rear end of the container is then located in the same way as the front by the use of the rear part of the container template edges.

If the 20ft container hypothesis can be discarded, the tracking of the front end is continued until the rear end of a 40ft container possibly is in the field of view. The same verification procedure takes place and if the container code is found, the container is classified as a 40ft container. Following the

same scheme, the rear end is the located with the template edges.

When the container is classified and the rear end located, the rear measuring point is related to the given spreader position. The traffic lights are controlled in the same way described in [5] with different indications as the chassis is within predefined distances to the ideal stop position.

Empty chassis classification and positioning

For the set-down operational type, one sequence with a terminal chassis has been synthetically created to test the implementation. In figure 5.7, a flow chart of the empty chassis positioning is shown.

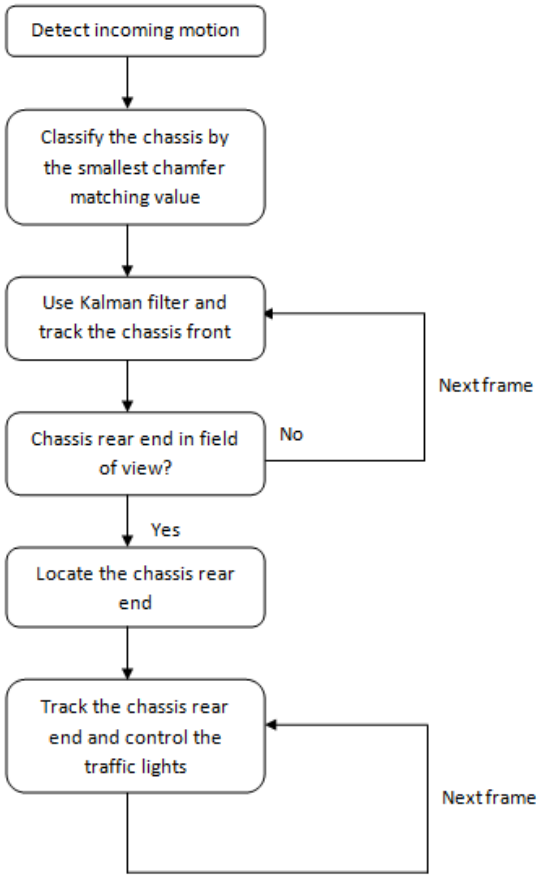

Figure 5.7: Flow chart depicting the positioning of an empty chassis.

The operation is quite similar to the loaded chassis positioning but for an empty chassis the classification is done differently. When the incoming motion in the active lane is detected we use the front part of both the chassis templates edges to locate and classify the chassis. The front edges are chosen so that enough shape information needed to discriminate between the chassis type is included.

Edges are extracted from the search window of the current frame and a distance transformation is done. The front edges of both templates are then correlated with the distance image. The classification is done by choosing the template corresponding to the lowest minimum value in the correlation functions.

The chassis construction can sometimes be very symmetrical, leading to strong minima in the correlation and false positives. A verification step should be added but since no characteristic features such as the container identification codes or twistlocks are available on the empty chassis, this is not straight forward. Incorporating more information about the chassis inner beam configuration can help in improving the matching but a verification step should be added as well. We have not been able to find a sufficiently robust verification and more work need to be done to address this issue.

As the front is located, the search window is set smaller and the Kalman filter is initialized. The chassis front is tracked following the same scheme until the rear end is in the field of view. The rear end is then located and the rear measuring point is related to the spreader position and the relative distance controls the traffic lights.

Tracking of the chassis' position

The motion of a chassis in a particular lane is almost completely constrained to move in a straight direction. The position of the chassis in the image coordinate system is given by its x and y coordinates. We assume that the acceleration between two successive frames is constant and use the following state model with process noise

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} + w_k \tag{5.19}$$

where $[x, y]$ is the position and $[\dot{x}, \dot{y}]$ is the velocity of the chassis. The process noise v_k will incorporate any actual acceleration.

The measurement vector is the position of the chassis which means that

$$z_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} + v_k = \begin{bmatrix} x \\ y \end{bmatrix} + v_k \tag{5.20}$$

The filter is initialized with a position specified as the middle point of the search area for motion detection of the lanes and with zero velocity.

The predicted position for the next frame is used to define the search window. If the system is not able to verify a position in the frame the position used is the Kalman estimated position. Then a slightly larger search window is used in the next frame. An additional feature of the filter is that the estimated velocity of the chassis also is obtained since it is included in the state.

5.5.3 Straddle carrier positioning

Besides terminal and road chassis, some ports utilize straddle carriers. These are large truck-like vehicles with wide legs, capable of straddling a container and lifting it. The test data collected on site at the port of Gothenburg contains exclusively straddle carriers working on the landside. Certain straddle carrier models are capable of lifting containers high enough into the air to be able to stack them onto each other.

The port of Gothenburg does not use a CAS to assist their straddle carrier operations. Instead, they rely on reflexive markers on the side of the carriers. As the carriers are large and complex structures it is harder to track them using the same methods and algorithms as for the road and terminal chassis. The edge model becomes very detailed and complex, and as the height is much larger than for ordinary chassis, the change in perspective causes problems. Instead, we have investigated if there is some characteristic feature of the straddle carriers that can be used for the tracking problem.

Tracking straddle carriers

The straddle carrier model used in the port of Gothenburg have a feature that is very useful in image processing - the roof of the driver's cabin is com-

pletely white, apart from a red warning light. This means that the roof will be a prime candidate in what to look for when localizing and positioning the carrier.

The motion detection algorithm is the same that is described in 5.5.1. With the incoming straddle carrier detected, each frame is histogram equalized using the Matlab IPT function *histeq* [8].

After thresholding into a binary map, we search for the largest connected component in the image, making sure it contains a certain minimum of pixels to ensure it's of adequate size to be a roof candidate. Once identified, the coordinates of the connected component's bounding box are used to extract a sub image from the original image. This sub image should contain mostly bright pixels, except for a small island in the middle - the warning light. As the extracted sub image sometimes contain the outer edges (and thus darker pixels) of the cabin, there is noise in the outskirts that must be dealt with. The sub image is now thresholded and inverted to obtain a binary map where the warning light consists of a small circle of white pixels, and the rest of the roof is made of black pixels, except for small parts in the outer edges. After this thresholding, we use the definition of connectivity in 5.4.2 and remove all components connected to the outer edge of the sub image. What is left is now the small circle of pixels belonging to the warning light. The center of mass of this island is taken as the position of the straddle carrier, and its value is fed into the traffic light system. In figure 5.8, we see a flowchart for the extraction of the warning light.

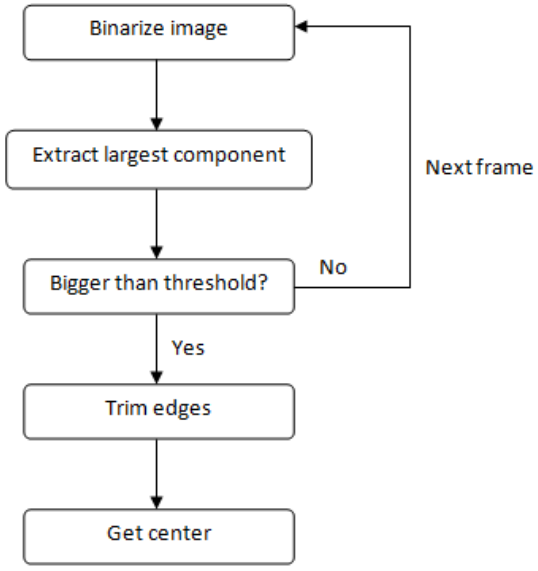

Figure 5.8: Flow chart depicting the warning light extraction algorithm.

Unloading carriers

As the straddle carrier reaches the stopping point, a counter is started. If the carrier stands still for a number of frames, it is considered to have reached its destination and is unloading the container. The tracking is halted, and the program moves to the next step.

Leaving carriers

The last step of the straddle carrier's cycle is when the carrier leaves the lane area. Once the carrier has dropped its cargo (or picked it up, depending on the type of operation), a background subtraction is initialized. The threshold used for this operation is preferably set higher than for the entry. As soon as the straddle carrier has been confirmed as leaving the camera's field of view, an attempt is made to classify the container and extract the identification code using chamfer matching.

In figure 5.9, a flow chart of the straddle carrier positioning is shown.

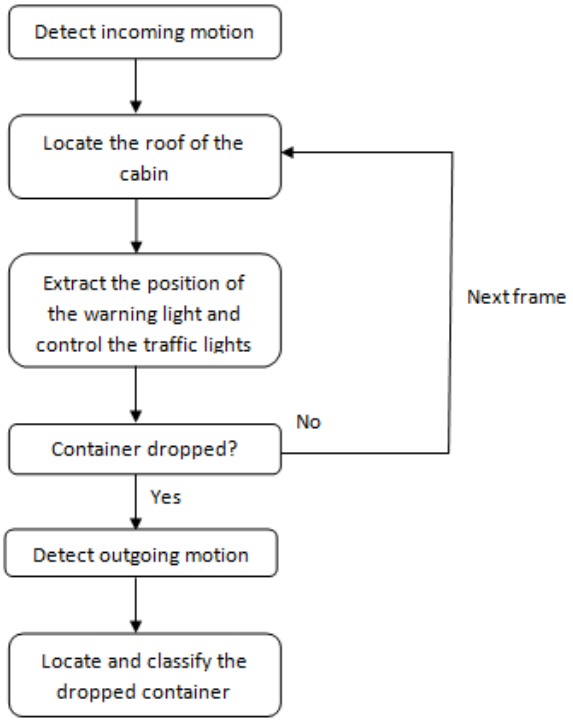

Figure 5.9: Flow chart depicting the straddle carrier positioning.

Chapter 6

Results

6.1 Motion detection

The motion detector has no problems detecting an incoming chassis even with the addition of Poisson noise. Other noise types provide a bigger challenge though. Also, the detections made are seldom consistent from sequence to sequence. For certain straddle carrier sequences, the detection is made as soon as the straddle carrier's shadow enter the search area, while other sequences requires the entire carrier to come into view before triggering the motion detection.

The motion detection used two different thresholds; one for entering chassis and one for leaving ones. The reason for this is that when the straddle carrier has completed its drop-off and begun to leave the area, it still blocks the camera's view of the container for a short period of time. As the post-drop-off container analysis is begun as soon as the system notes that the carrier is leaving, it will give unsatisfactory results. Having a higher threshold for exiting chassis will ensure that the carrier travels further from the container before it is detected as leaving.

6.2 Chassis results

Chassis classification, single images

The template matching for the empty chassis works well, the chassis is correctly classified and the right position is found for all test images used. The method seem to work well even for the noisy images. The matching results are very dependent on the edges that can be extracted from an image. Using

only parts of the templates for matching, the more of the template edges used in the matching, the better the matching results. This is because of the symmetrical chassis configurations.

Examples of the results for an empty chassis classification is shown in the figures 6.1 and 6.2, displaying the results for the same image, with and without noise. The noise added is Gaussian of variance 0.01. The figures show the original image of the lane, the resulting edge map after using the Canny edge detector and the found position of the chassis.

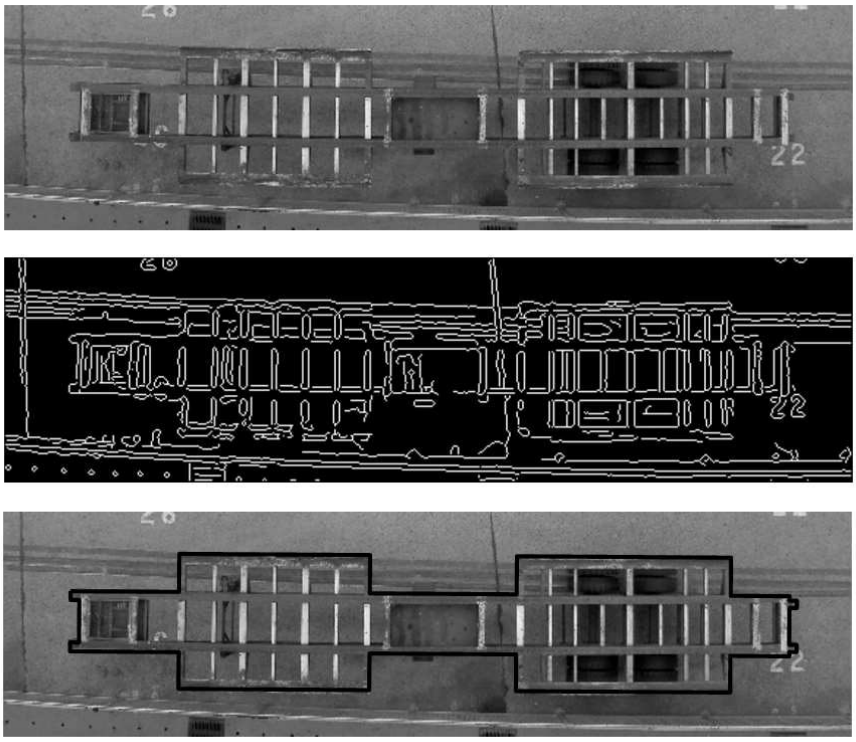

Figure 6.1: From top to bottom: the original image, the Canny edge detector results ($\sigma = 1$ and the thresholds automatically set to 0.05 and 0.13) and the chassis matching results.

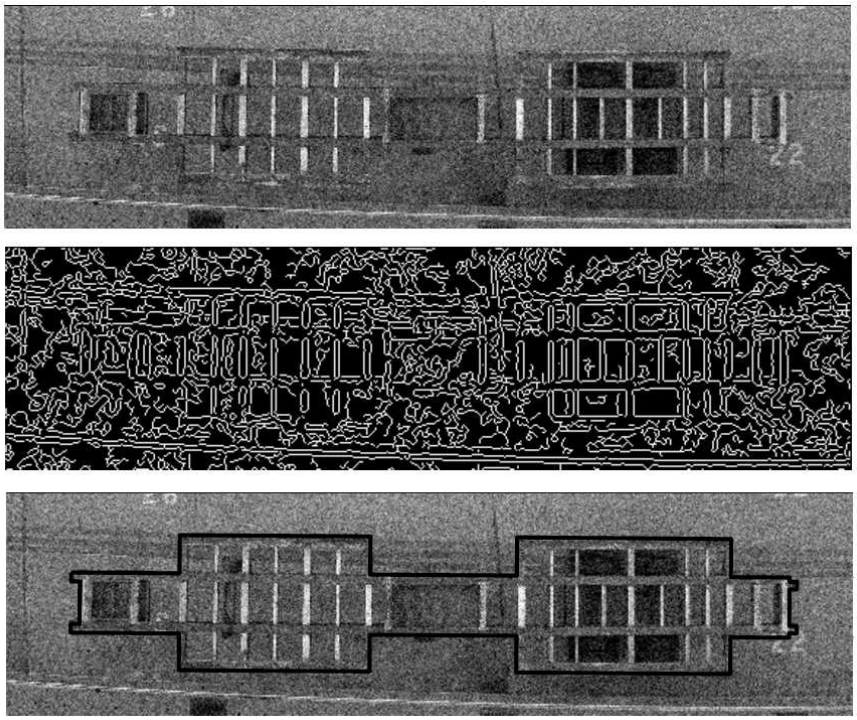

Figure 6.2: From top to bottom: the original image with added Gaussian noise (zero mean and variance 0.01), the Canny edge detector results ($\sigma = 1$ and the thresholds automatically set to 0.06 and 0.16) and the chassis matching results.

Container classification, single image

For the images taken at daytime, the classification and localization of the containers works well given that also a verification step is added where the container code existence detection is used. For the images taken at night and during bad weather conditions, the matching sometimes fails, mostly due to the fact that not enough edges can be found. The verification using container codes fails when strong illumination from lamps creates reflections that severely distorts the image.

Some result of the container classification is shown in figures 6.3 and 6.4. The figures show the results of a container classification and localization for the same image of a 20ft container, with and without noise. The noise added is salt and pepper noise with density 0.03.

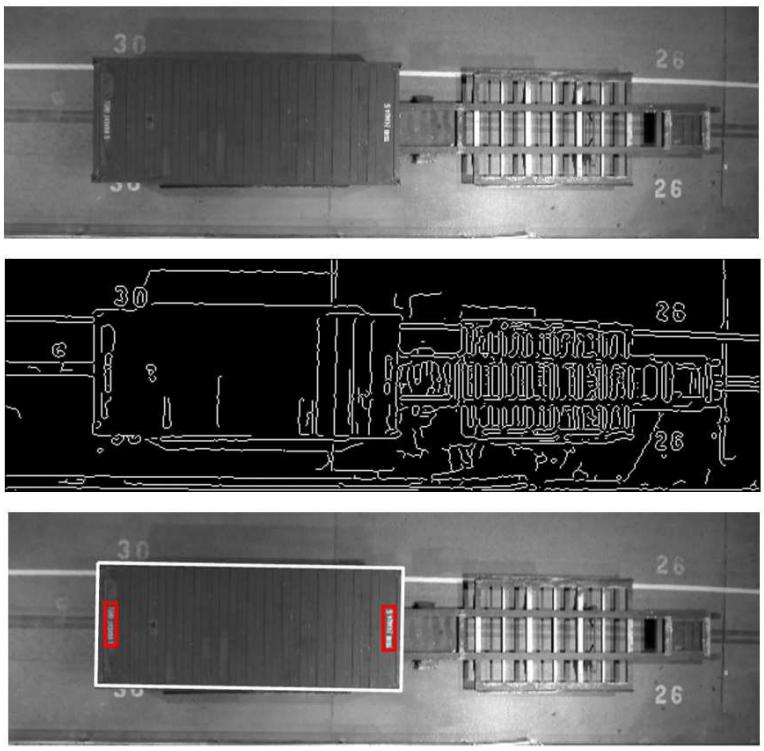

Figure 6.3: From top to bottom: the original image, the Canny edge detector results ($\sigma = 1$ and the thresholds automatically set to 0.03 and 0.06) and the container matching results.

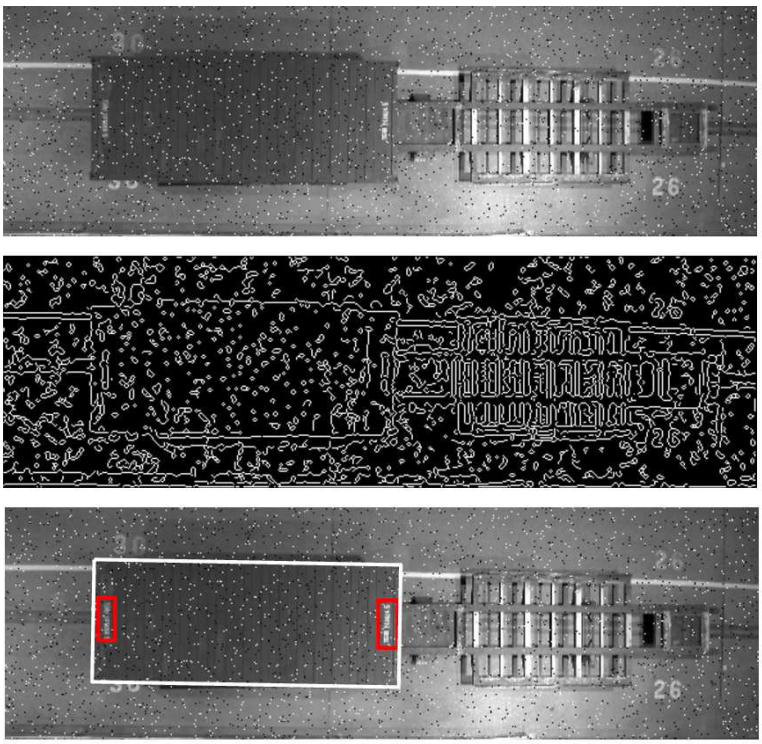

Figure 6.4: From top to bottom: the original image with added salt and pepper noise (density 0.03), the Canny edge detector results ($\sigma = 1$ and the thresholds automatically set to 0.04 and 0.11) and the container matching results.

6.2.1 Chassis positioning and tracking

Some results for a synthetic sequence (40ft container pick-up) are given below. The system is able to classify the container correctly and controls the traffic lights in correspondence with the existing positioning system. Comparing the position found by the matching to the ground truth data manually extracted from the sequence, the measured position is found to deviate from the ground truth in some frames with an error of 1-3 pixels. The comparison can be seen in figure 6.5, where the measured position is the x position for the center of the chassis' rear beam. Here we also see the loss of the position in some frames. This is mainly due to the verification step where the search for the corner-fittings fails in some frames.

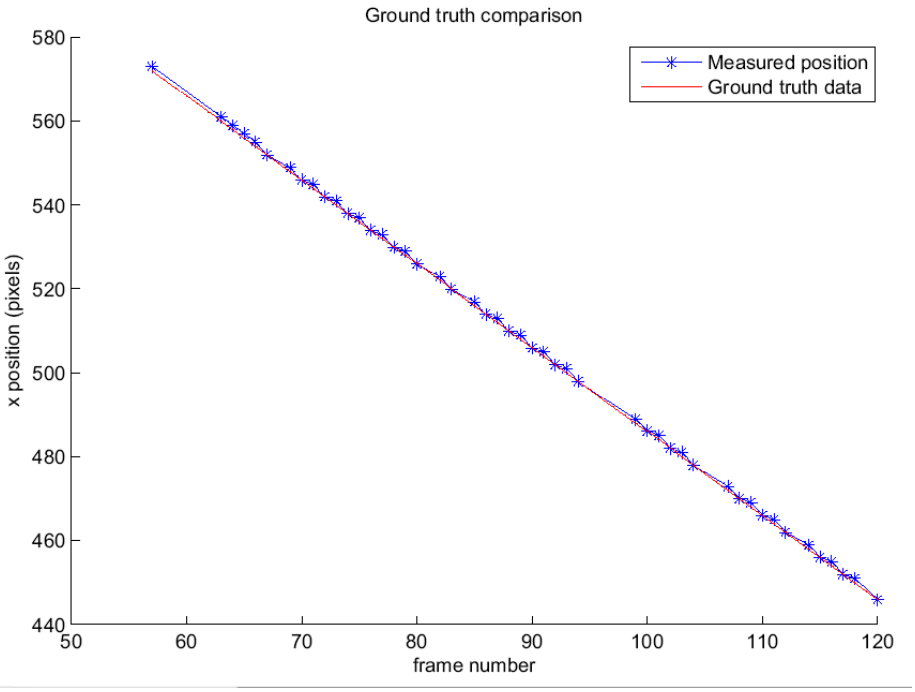

Figure 6.5: Ground truth comparison results for a synthetic sequence. The measured position is the x position of the center of the chassis' rear beam.

The Kalman filter prediction of the position in the next frame works well for the synthetic video sequences which is expected since the motion model is perfect. An example of the results of the tracking can be seen in figure 6.6. Here, $P_0 = 100$, $Q = 0.1$ and $R = 1$.

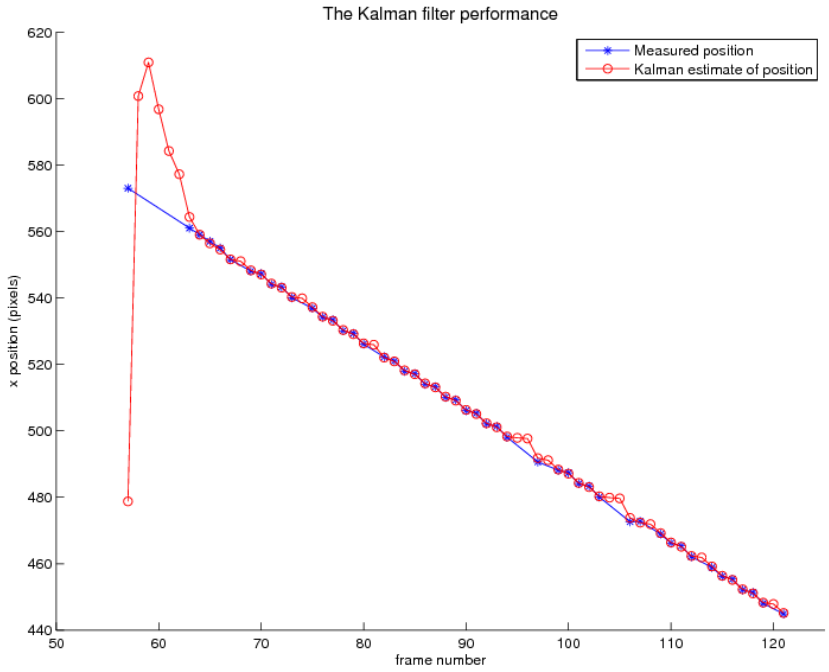

Figure 6.6: Kalman filter tracking results for a synthetic sequence. The measured position is the x position of the center of the chassis' rear beam.

As can be seen, the predicted position quickly adjusts to the motion model and also give good prediction results when the position of the chassis is lost over a few frames.

6.3 Straddle carrier sequences

For the straddle carriers, we tested the algorithms on video sequences containing an unloading cycle.

6.3.1 Cabin detection and tracking

The result of the analysis was compared with manually extracted ground truth data. We also ran a video sequence corrupted by Poisson noise and one with zero-mean white Gaussian noise, with a variance of 0.01. Tests with salt and pepper noise and speckle noise failed to detect the cabin in a satisfactory way.

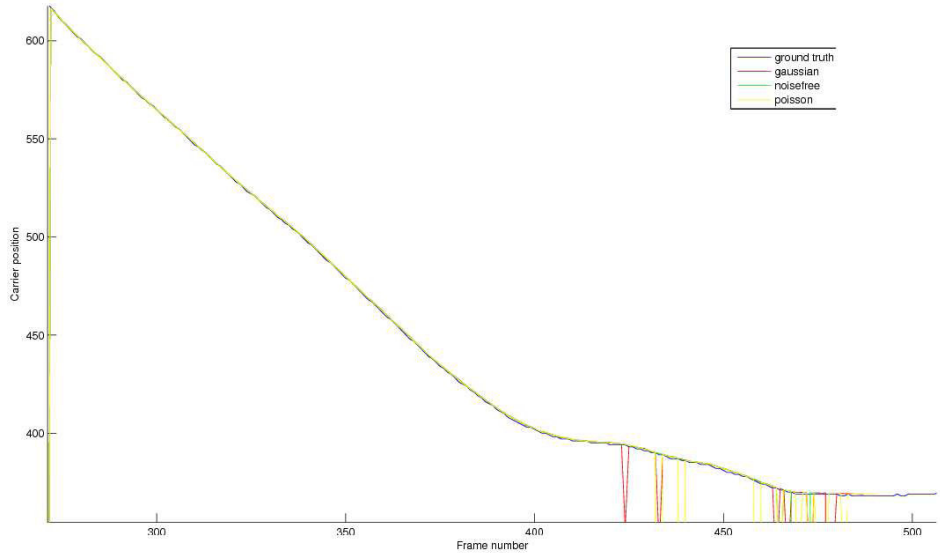

Figure 6.7: Tracking results compared with ground truth data for the straddle carrier. The measured position is the center of the warning light.

As the analysis shows, the algorithm comes very close to the ground truth, even with the addition of noise. What is interesting is to note that the noise does not give noticeable worse results than the noise free analysis. We also note that the algorithm fails to detect the cabin in several frames towards the end of the cycle, resulting in several spikes. This is most probably caused by the fact that as the straddle carrier slows down, a small wire that previously became blurred by the motion of the carrier now can be seen. The wire extends from the edge of the roof to the warning light. As this wire has lower intensity than the roof, it will sometimes cause problems with the removal of edge components. A possible solution would be to experiment with erosion operations as described in 5.4.1.

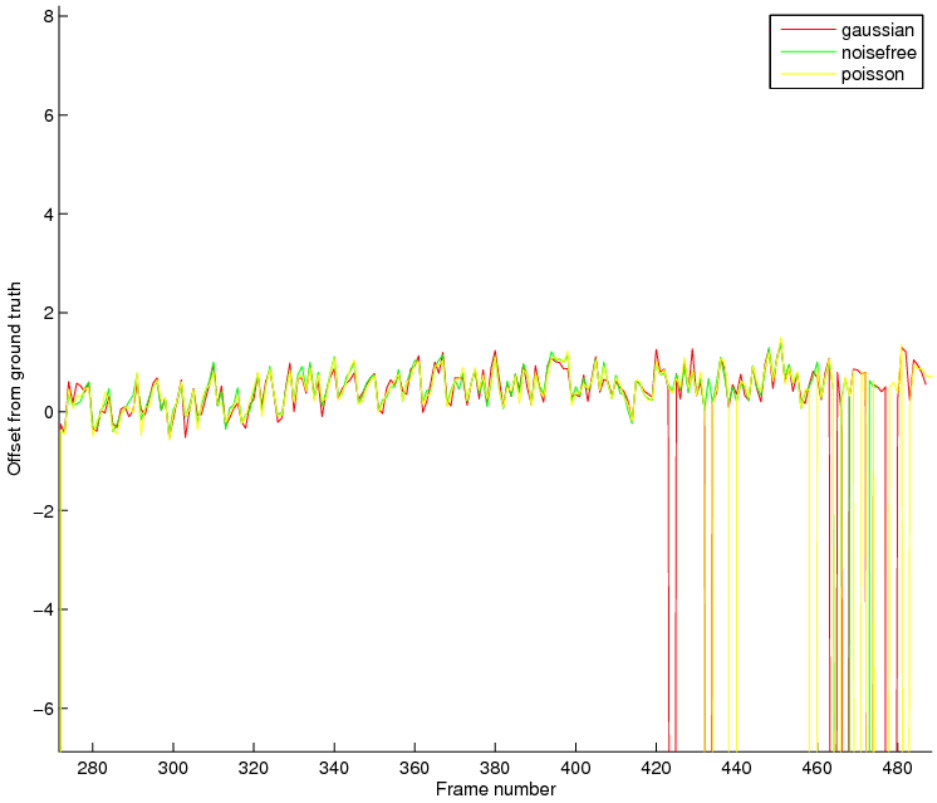

Figure 6.8: Offset from ground truth values for the tracking of the straddle carriers. The measured position is the center of the warning light.

Chapter 7

Discussion and conclusions

7.1 Chassis positioning

The chamfer matching shows promising results for both classifying and locating the chassis. The use of matching only front or rear parts of the template edges makes the method less computational heavy. The recognition and localization of the chassis and containers relies on a good edge extraction, good model representation and a consistent matching criterion. The noise robustness is very much dependent on the edge detector. The Canny edge detector has shown good results.

The verification step for the containers works well and makes the position more accurate although sometimes makes the positioning fail over a couple of frames, mostly depending on an insufficiently working twistlock existence detector. This should be improved. A verification step for the empty chassis is a necessary addition. When matching only parts of the chassis templates edges, the found position is sometimes located to the inner beams of the chassis configuration. This is partly due to the highly symmetric beam configurations. Incorporating more shape information about the chassis can help in improving the matching.

The container code extraction works well under certain weather conditions and could be a highly interesting addition to the system. However, if the container code should be readable to human eye or usable in an automatic OCR system, this probably increases the resolution needed. The extraction also needs to be extended to incorporate containers having dark container identification codes. The extraction also fails in some of the images where bright spots of illumination distort the image, this must be dealt with as well.

The implementation of a Kalman filter for tracking the chassis position has shown useful. Even though the use of the filter in the test implementation show promising results, an evaluation using more realistic test data is clearly necessary. This might show the need of another motion model that incorporation the acceleration as well. The parameters also need to be fine-tuned. Given the constrained vehicle motion in the lane, the Kalman filter should perform well in predicting the position given that a sufficiently good motion model is used. The two major contributions of the filter implementation are the smaller search window size and the ability to retain the chassis position even if it is lost over a number of frames.

The small amount of test data for chassis and containers have made the evaluation and implementation limited in many ways. No proper models of the chassis have been available so the template vertices have been set manually, adjusted to fit one of the test images. The matching has been implemented only for single images and synthetically created sequences, leaving the evaluation somewhat insufficient.

Due to the limitations and scope of this thesis, setting the required number of pixels/cm is not straight forward. This specification depend on numerous parts including the accuracy the calibration, implications of camera oscillations and the accuracy of the image processing algorithms. Here, we can only estimate the accuracy of the algorithms, which from the implementation has proven to give an error of approximately 1-3 pixels. Depending on the camera perspective for each lane, the needed position shift to compensate for the height information loss will differ and need to be known. The uncertainty in this shift must be added to the pixel error. Adding this and possible other factors not included for, a "worst" error estimate is probably around 5-7 pixels. Given that the accuracy of the existing system is ± 40 mm, this implies that the necessary resolution is approximately 1-2 pixels/cm. How this specification is obtained due to the camera's sensor size, mounting height and field of view need to be investigated.

Another objective was to estimate the system's performance under various illumination and weather conditions. We have shown that the technique works for the usable images provided and is robust to a certain degree of noise distortions but much work remains. One of the main issues is the edge extraction. If the correct features can't be extracted, the matching will fail. A more extensive evaluation of the edge extraction is therefore necessary. Additional lamps might be necessary to ensure that enough contrast is obtained.

The relative computational load of the algorithms should also be mentioned. In the chamfer matching, the edge extraction has the overall highest computation time, followed by the distance transform calculation. In comparison, the other parts have almost negligible computation times. The implementation of certain parts in the camera hardware has not been investigated in this thesis is a prime candidate for future work.

7.2 Straddle carrier positioning

Motion detection using a static background works well for all our sequences, overall having the same illumination. If implementing this part in a later stage, another background model that adapts to the changes in weather and illumination must be used. If working well, the background subtraction could also be used for partly segmenting the object from the background, helping in the feature extraction process.

The implementation used to find the straddle carrier's cabin hint of the advantages using markers. Due to the construction, it is possible to equip each straddle carrier with a marker, which together with a high-precision camera surely can provide sub-pixel accuracy in the positioning problem.

The tracking of straddle carriers works well with the sequences we have access to. In the beginning stages, it is fairly robust towards noise, handling both Gaussian and Poisson noise without any larger problems. In the few frames where the center point is completely dropped, a prediction filter and hard-coded requirements about how far the center point is allowed to travel from frame to frame could eliminate the problems that arise.

For both the chassis positioning and the straddle carrier positioning, the implementation is able to process 3-4 frames per second. The current system updates the traffic lights twice every second, which should mean that it's possible to achieve the same functionality in this aspect. It should also be noted that several parts of our systems could be eliminated in a real implementation. To illustrate, merely displaying the frames and analysis by drawing each frame in Matlab takes roughly one third of the total processing time.

7.3 Concluding words

In general, VCAS gives some promising results. However, having only grasped at the surface of the problem, a lot of issues remain to be solved and worked through.

The developed GUI and code is suited to be further tested and developed. Extensive code commentary is included, and the appendix of this thesis report contains a technical reference manual for using the software to evaluate further test data (along with guidelines for how to add functions).

To take this further, a real setup must be made and a large test database must be established. A key area is to introduce adaptive parameters to the system. With an extensive amount of test data available, correlations between certain frame features (i.e. contrast, average intensity, optical flow) and optimal threshold levels could be made, which is a requirement for any automatic system. With some further developing and fine-tuning of the algorithms, it would be interesting to run the program parallel to CAS on site, to see if the data provided would be sufficient for the requirements under real operation cycles.

We hope that our thesis can inspire future developments in these areas!

Appendix A

User's guide to the VCAS simulator

A.1 About

This is the technical reference manual for the MATLAB implementation of the proposed vision-based solution to the chassis alignment problem in the shipping industry.

The purpose of the program is to test the efficiency of vision-based algorithms on video sequences containing chassis and straddle carriers loading and unloading in a dock. The user imports a video sequence and can then analyze it to obtain performance information for the VCAS system.

A.2 GUI

The Graphic User Interface is split into four main parts. In the upper left corner, the playback window is located, showing the video sequence and the results of the analysis. To the upper right is the cabin panel. Here is where all the adjustments are made that are meant to be controlled from the crane cabin. The lower right part of the window contains the status window, where information from the system's analysis is transmitted to the user in text form. Finally, in the lower left are the technical controls for the system. These includes controls for playback (including reversal, fast forwarding play/pause, etc), adjustable parameters and the controls for importing video sequences. A screenshot of the GUI can be seen in figure A.1.

Figure A.1: A screenshot of the Graphical User Interface (GUI) running an analysis.

A.2.1 Layout

The graphic user interface is optimized for a PC running Windows XP at a resolution of 1280×1024.

A.2.2 Playback window

Displays the current frame, and plots the results of analysis on it. To the right of the playback window are the traffic light controls, displaying the distance between the spreader position and the position of the chassis determined by VCAS.

A.2.3 Cabin panel

This window represents all parameters that should be set by the crane operator, such as active lane and chassis direction.

A.2.4 Status window

The status window is, together with the image display itself, the source of information and results for the user. When an analyze sequence begins, the window is automatically flushed. As soon as the program has new information (for example if a chassis is detected), this is updated in the status window together with the frame number during which the event occurred. To manually flush the status window, press the button labeled *Flush status*.

A.2.5 Technical controls

Tutorial

To begin analysis of a video sequence, start the program and click the button *Import AVI*. A dialogue box will pop up and let you select a video clip to import. If Matlab is able to read it, the first frame will show up in the display window. Next, set the lane areas by clicking the button *Set lane areas*. The program now awaits you to select a small portion of the left side of the active lane. Repeat the process for the right part, and the program will plot the rectangles you selected. Should you not be satisfied, simply press *Set lane areas* again to repeat the process until you have your entrance/exit areas the way you want them.

Now the program is ready to analyze the video sequence. Note that by pressing *Show advanced controls*, you can see what values are used for the various parameters. An example is the cm/pixel field, which is automatically calculated from the width of your left and right lane entrance/exit. This is done using the measured width (in pixels) of the selected areas and the known length between the yellow lane lines (449cm as measured during our visit to the port)

To begin the analysis, once the parameters are set, simply press the button labeled *Analyze all*. The program will now go through the steps described in the implementation chapter, and present the result in the display and status windows.

Pre-processing panel

In the pre-processing panel, the user can choose between various operations to perform on every frame before the analyzing parts of the program is initiated. As of now, the pre-processing panel is used to add noise to the sequences.

Add noise - By checking this radio button, the selected item in the adjacent listbox will determine what type of noise is added to the image. All noise types use a call to Matlab's internal *imnoise* function. Depending on what noise function is selected, the corresponding parameter fields (mean, variance) are enabled. They are preset for Matlab's default values.

The architecture is open, and the panel is intended to be usable for extensions of the project - if someone wishes to integrate a new algorithm, for example video stabilization or shadow removal, a suitable block of the source code has been explicitly marked for where an insertion of additional code would be appropriate.

Export settings

Under the *File* menu tab, the user can use the option *Export settings* to export the current parameter settings as a .mat file.

Import settings

The *Import settings* function is reached from the *File* menu. The program imports settings and parameters such as pre-selected video sequence and set lane areas.

Export to AVI

A small checkbox to the left of the control panel is labeled *Export to AVI*. Once checked, a filename field becomes visible, where the user can specify what filename to export the result of the analysis too.

NOTE: Matlab has issues handling video files, and sometimes crash while opening and closing them. Save all other work before exporting avi files.

Supported formats

The limitations on supported codec formats are dependent on the local workstation and Matlab. The program is able to use video sequences compressed

by any codec that Matlab itself can read, which almost always translate to all codecs the workstation as a whole can manage.

Known issues

While Matlab 7.0.0 and 7.1 run the GUI without problems, Matlab 7.0.1 is not supported. There also have been some unexplained graphics behavior when running the program on a Windows Vista machine.

Bibliography

[1] Iso 668. Series 1 freight containers. Classification, dimensions and ratings.

[2] Iso 6346. Freight containers – Coding, identification and marking.

[3] G. Blum. *Pacific Maritime Association: Container Crane e-Learning.* http://garyblum.com/pma/splash.htm.

[4] C. Heidenback and C. Johansson. *Chassis alignment system*, 2006. Patent No: 7123132.

[5] ABB Crane Systems. *Chassis Alignment System 2.0 Technical Reference Manual*, 2007. 3AST004186 Rev. A.

[6] D. A. Forsyth and J. Ponce. *Computer vision : a modern approach.* Prentice Hall, 2003. ISBN: 0-13-191193-7.

[7] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition).* Prentice Hall, 2007.

[8] Mathworks. *Image Processing Toolbox^{TM} 6, User's Guide*, 2008.

[9] S. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. volume 5308, pages 881–892. SPIE, 2004.

[10] M. Piccardi. Background subtraction techniques: a review. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 4, 2004.

[11] O. Javed A. Yilmaz and Mubarak Shah. Object tracking: A survey. *AACM Computing Surveys (CSUR)*, 38(4):13, 2006.

[12] A. Thayananthan, B. Stenger, P.H.S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. volume vol.1, pages 127 – 33, 2003.

[13] E. Pissaloux J. You and H.A. Cohen. Hierarchical image matching: a chamfer matching algorithm using interesting points. pages 70 – 5, 1995.

[14] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[15] G. Welch. An introduction to the kalman filter. Technical report, 1995.

[16] S. Klein Goldenstein. A gentle introduction to predictive filters. *Revista de Informatica Teorica e Aplicada (RITA) XI*, 1:61–89, 2004.