Positioning and Docking of an AGV in a Clinical Environment

Master‘s Thesis in Biomedical Engineering

DAVID ANDERSSON & ERIC MÅNSSON

Department of Biomedical Engineering
Signals & Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012
Master's Thesis EX031/2012

REPORT NO. EX031/2012

Positioning and Docking of an AGV in a Clinical Environment

DAVID ANDERSSON
ERIC MÅNSSON

Department of Biomedical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012

Technical report no EX031/2012
Department of Biomedical Engineering
Signals & Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Phone +46 (0)31-772 1000

Göteborg, Sweden 2012

Abstract

The purpose of this project was to investigate options and find a solution to dock an Automatic Guided Vehicle (AGV) to a predefined path within an unique working cell. Precision, repeatability, fast docking time and minimization of load for the operator should be ensured to preserve quality of the tasks. The primary aim was to be able to maneuver the AGV between predefined positions with an accuracy of ± 5 mm.

By investigating prior works (articles) covering AGV control, we decided to proceed with dead-reckoning using odometry and a vision-based system together with an optical path. A path tracking technique called *Quadratic curve* was used together with the Extended Kalman filter algorithm for estimation of location.

Dead-reckoning was used to reach the optical path, after which the vision-based system was enabled to track it. With 30 repetitions of the docking phase, we reached a mean offset (Euclidean distance) of 5.1 mm with a standard deviation of 3.0 mm. The data were derived from the Kalman filter states where the measurements were performed by the vision-based system as an offset from the indicated stopping points.

We came to the conclusion that odometry has simple calculations and is easy to implement, but suffers from sensitivity to both stochastic (floor irregularities, collisions, skid) and systematic (cumulative) errors. The kinematic model was affected by the lack of measurement tolerance in the assessment of vehicle dimensions.

The vision-based system proved, as predicted, to be very accurate but also sensitive to proper light conditions. The low speed of the AGV (maximum 200 mm/s) was very advantageous in relation to the relatively low camera frame rate (approximately 15 frames per second). The accuracy for the vision system was limited by the frame rate. Note that computer vision is computational heavy and thus quite demanding on the system performance.

With an improved prototype set and access to calibration tools for camera mounting and assessing necessary dimensions together with favorable light conditions and an optimized object detection algorithm, we believe that we could have reached well within the requested accuracy.

Keywords: *Automatic Guided Vehicle*, *Dead-reckoning*, *Odometry*, *Computer Vision*, *Extended Kalman filter*, *Quadratic curve path tracking*

Acknowledgements

We would like to acknowledge Lars Hammarstrand, Post-Doc Research Fellow at the Department of Signals and Systems, Chalmers University of Technology for support in constructing the Extended Kalman filter algorithm and Jonas Linder, Ph.D student at the Division of Automatic Control, Linköping University, for giving us a crash-course in sensor fusion with Kalman filtering. Both contributed to needed repetition and support which saved us plenty of time. We would also like to give a lot of credit to the company whom offered us the thesis work. It has been very instructive and great fun working with them.

David Andersson and Eric Månsson, Göteborg 5/6/2012

Contents

1

Introduction

THE MAIN PURPOSE of this chapter is to introduce the reader to the full scope of this project, including background, purpose, objective, limitations and work method. The time schedule Gantt chart can be found in Appendix B. This chapter serves as an organized template for verification and validation in the wrap-up phase of the project.

1.1 Background

Imagine a manufacturer where several departments with multiple cells have an automated installed robot that performs a certain task. Due to variations in the inflow of materials or orders to each department/cell, the usage efficiency of each individual robot will never go above 80%. Also, a complete breakdown of this installation will have a significant economic impact upon the manufacturer.

Consider the potential benefits if this robot could be mobilized and easily transfered between respective departments and cells. The installation and maintenance costs of the robots will decrease significantly since fewer mobile robots can replace the installed ones. The ability to easily switch a malfunctioning robot will also decrease the severity of a total breakdown. The potential is very demonstrable in the medical equipment sector. Equipment could be transfered to ski resorts during seasons. Immobilized patients could receive medical attention at home. Hospitals and care givers could share heavy installations to increase efficiency. The list goes on. The bottom line is that modern technology allows for lean, light and portable solutions that could substitute installed technical equivalents.

An innovative company would like us to investigate options and find a solution to dock an Automatic Guided Vehicle (AGV) to a predefined path within an unique working cell. Secondarily, the unit shall be able to automatically align to a set of devices in the working cell. The purpose of the automated docking is to facilitate the operator. Precision, repeatability, fast docking time and minimization of load for the operator should be ensured to preserve quality of the tasks. Other desired features will be discussed later in this chapter. Due to confidentiality constraints, the reader has to settle with the sole fact that the AGV is supposed to be operating in a clinical environment.

1.2 Purpose

The purpose of our task is primarily to enable automated docking of this AGV to a predefined path which is parallel to a work station in an unique working cell. Once docked, the unit shall align to an operating tool on the station in a parallel manner when commanded by the user. Secondarily, the operating tool of the AGV shall follow the height (z-position) of the operating tool belonging to the working station. The working cell environment is illustrated in Figure 1.1 below.

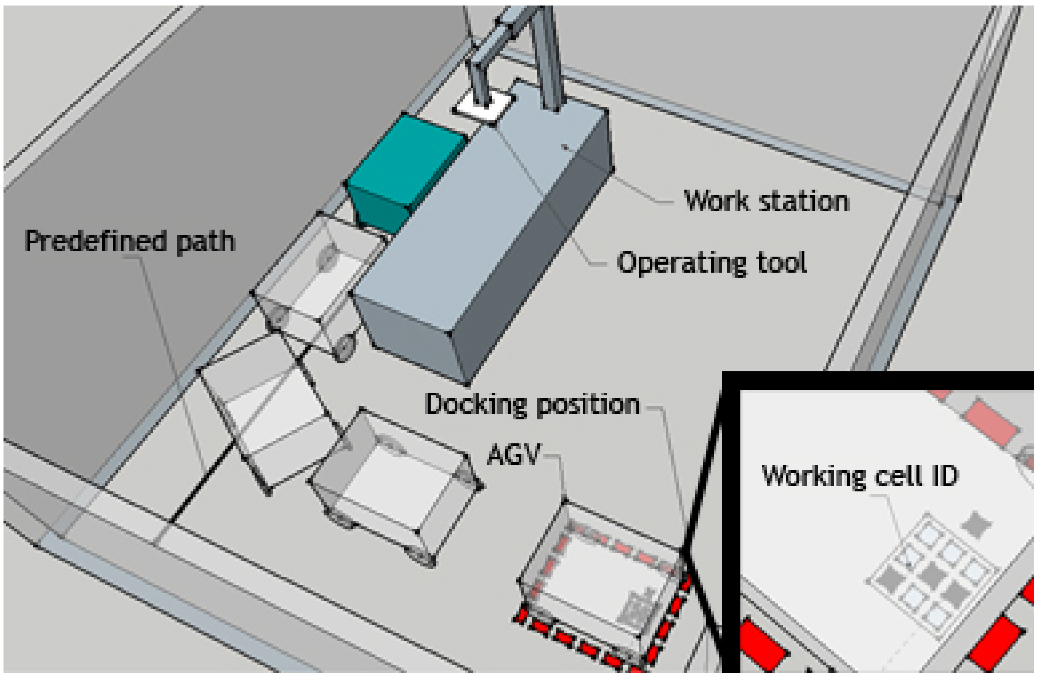


Figure 1.1: In the unique working cell (room) is the AGV placed manually within a marked area. An unique working cell ID is acquired which describes the necessary cell environment parameters. Upon command, the AGV docks automatically to the predefined path which is parallel to the work station.

1.3 Objective

The following list is the requirements for the whole AGV unit, specified by the company. Items are listed in descending order of priority. The automated control features of the AGV are described in 6(a)-(e).

1. The added feature of preference should be an option, i.e. the basic version of the unit shall not bear costs for the selected features.
2. It should be possible to add the feature during field operations or at the working cell.
3. The added feature should be applicable to different kinds of working cells with different setups of operating equipment.
4. It should be easy and fast to install the added feature at the working cell.
   (a) The installation at one working cell should be performed in less than four hours.
   (b) Installation without mechanical tampering on the working site floor is preferred.
5. The added feature should be robust against soiling, wear and tear.
6. The added features will exists in different versions with certain levels of performance.
   (a) The unit is only aligning to a predefined position (point) within the working cell.
   (b) The unit is following a predefined path, but the drive command is given manually by the operator.
   (c) The unit is aligning to the work station automatically, along the predefined path.
   (d) The operating tool of the unit is adjusting height with regard to input coming from the work station.
   (e) Combinations of the above features.
7. The accuracy of the absolute position must be within ± 5 mm with verifiable repeatability. This condition is not relevant while the unit is moving.
8. The unit should be able to recognize which working cell it is currently situated in. The ID of the working cell will determine which features, choices and information that are available to the operator.

The main research questions can be summarized as follows

AGV positioning. Is it possible to achieve a positioning system for the AGV within an accuracy of 5 mm?

AGV navigation. Is it possible to navigate the AGV to a predefined path and enable following that path, without any mechanical tampering on the floor?

Working cell identification. Is it possible for the AGV to identify which working cell it is currently situated in?

Problem identification

Many of the challenging problems for this project will probably be generated by the combination of the given objectives. Combining a mobile unit and high accuracy positioning is identified as the greatest challenge, especially when considering the preference of no mechanical tampering on the working site floor.

Furthermore, continuously verifying the accuracy of the estimated stop position, choosing sensor techniques that allow for robustness, development of software with all required functions (such as sensor fusion) and solving all required interfacing are identified as major challenges. The interfacing can be divided into AGV-sensors, sensors-PC and PC-user.

The parallel and height alignment mentioned above does not have to be in real time, but can be executed with a certain (limited) delay. In a more practical perspective, we have to find a solution for docking the unit and allowing it to align to the parallel movement of the working station. The software architecture must be able to manage common errors and hold robustness against user operations and/or extraordinary sensor inputs that might result in severe consequences. If it fits within our time frame, we would like to implement the z-alignment of the operating tool versus the working station.

1.4 Limitations

The following items will not be covered by the project work

- Handling of the unit until it has reached its position for automated docking.
- Handling of the unit after it has been used in the automated mode.
- Software related safety philosophy according to any specific ISO-standard. The company will not rely on our solution to be final in that sense, but we will have to consider and work with such principles in mind.
- Hardware related safety philosophy, including collision avoidance for the AGV.
- Real time following of the work station.
- Selection of hardware and/or software outside our working prototype.
- Selection of DC motors and frame body.

1.5 Method

This section covers the planned work flow and approach for solving the specified tasks. The work flow described below follows the Gantt-chart available in Appendix B. The Use-Case (Appendix A) will define the technical objectives of the project. The method and template for the Use-Case was provided by the company. Major check-ups with the company will be performed at the Toll Gates after each phase. Minor check-ups are scheduled every second week.

Definition Phase

The Use-Case describes interaction between user and system in a step-by-step manner. This will be the primary benchmark for our working progress. The Design Architecture can be initiated once a crude draft of the Use-Case is complete. It has to be noted that the Use-Case will be continuously updated during the project as more information and details are gathered.

The task of defining Design Architecture includes searching for different system solutions. These solutions are rough approaches to reach the objectives, and will not include detailed information on hardware and software components. Information for creating the different system solutions will be gathered through brainstorming, consulting with supervisors, web search and literature search. We will also attempt to acquire insight information via study visits at hospitals and environments where similar solutions are used (e.g. AGVs in warehouses).

Design Phase

The first priority of the Design Phase is to evaluate and choose an approach which fulfills the benchmarks of the Use-Case according to preferences, feasibility and the demands specified by the company. The Test Specification will determine how the Detailed Design will be tested and evaluated. Defining the Test Specification will be tailored to the Use-Case. The Detailed Design phase includes choosing hardware and producing software. The Test Specification is solely a communication tool between us and the company and will not be included in this report.

Verification Phase

The Verification Phase starts with merging the designed system with the complete prototype setup provided by the company. A large portion of this phase will be dedicated to modify the design system to the prototype (the complete setup with all wheels, bottom plate, motors, sensors and system logic). The resulting solution with modification will be explained in a separate Test Report which is communicated with the company at Toll Gate 3 (TG 3). The Test Report will not be included in the report.

Validation Phase

The validation process will be carried out as a workshop involving the company as a jury. A demonstration of the prototype with the added features will be a final measure of achieved results.

2

Design Architecture

IN THIS CHAPTER we will discuss a few possible approaches to solve the task. As stated in the introduction, the unit during the docking and alignment phase can be considered as an AGV. With this condition, the acquisition of possible solutions is fairly straightforward. The utility of AGVs is increasing, which is noticeable in several industries. Besides being able to perform multiple-shift operations and repetitive sequences, they allow a more rigorous tracking of material, improves safety and reduces labor costs and material damage [1].

All AGVs today are equipped with embedded systems (sensors, bumpers and/or cameras) [2]. The AGV concept that is used for this unit is *Dual rear drive and steer wheels* which is suitable for forward- or single-direction movement [2]. The steering concept is called *Steered-Wheel steer control* which is similar to that of a shopping cart.

Figure 2.1 below illustrates a common architecture in AGV design. The sensors acquire pathway and ambient information. The actuators represent activation of engine movement, lifting/loading or any action the AGV is designed to perform. The communication infrastructure manages information exchange between other AGVs and the main controller system. The AGV Controller is the information hub for all hardware. The control sensitivity is determined by a *logical map*. The map can be magnet tape, virtual path, laser guidance et cetera. The AGV Agent embeds the system logic.

The working prototype for this project will not have any main controller system since there is only one unit (AGV). In other words, there will be no need for any communication infrastructure initially. The focus will mainly be on the actuators controlling the DC motors for the rear wheel pair.

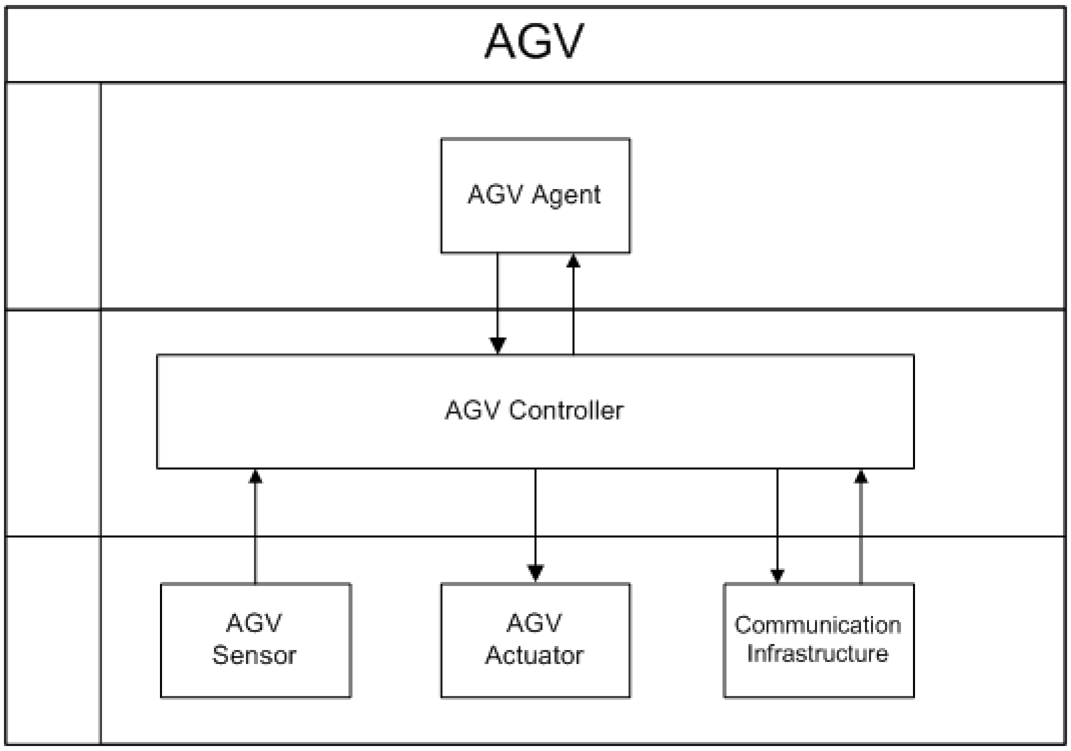


Figure 2.1: Common AGV Architecture [3].

2.1 AGV Perception

This section discusses common sensor techniques that ought to be the perceptive interface for the AGV.

2.1.1 Optical

There are several ways to navigate AGVs with optical systems. Izosaki et al. [4] suggested a solution with cameras in the ceiling which had a band-pass filter for IR light. An IR LED marker pattern were attached to the AGV to be able to detect the position of the AGV. Unfortunately the maximum error of 11.7 mm provided by this solution is not sufficient for our requirements [4].

Another very common optical approach is the use of laser that can either navigate by reflectors in the environment or find euclidean distances to certain landmarks [5]. The mean error reported here was 13 mm, but these values varies with the weight and stature of the AGV. Many of the optical systems are not constructed to handle relatively small deviations such as ours (±5 mm). The specifications that are emphasized by the AGV-system OEM is mostly maximum payload and AGV speed [6]. In a short conversation with *Atab*, a Swedish manufacturer of AGVs, the company states that most systems require a precision down to ±5 mm. The main issue with the laser navigated AGVs is that the system require a rotating device that emits beams to the reflectors. Such a device might not be suitable in a clinical environment due to moving personnel and other equipment in the surroundings which can block the beam path. The state-of-the-art systems is seemingly tailored for larger navigation patterns such as warehouse complexes.

2.1.2 Vision-based

Passive systems such as vision-based sensors are a growing application in the automotive industry [7]. One obvious downside of this approach is the relatively computational-heavy image processing. Note that the computational load in a clinical environment with low speed and simple patterns (such as a color tape against the floor) would be significantly mitigated. Also, the vision-based system in a clinical environment would be considerably more robust due to the lack of changing environment conditions such as rain, fog and sun. The low requirements on the performance parameters on the image acquisition device such as resolution, depth, stereo vision and colors will also decrease the costs [7].

2.1.3 Magnetic

Today there are a few different methods/ideas of using magnetic fields as guidance systems. One idea that is being tested in the automotive industry is based on bipolar cylinder magnets that are hidden in the road with an equal distance and creating binary codes by alternating which pole that is upwards [8]. In this way it can provide both lane guidance and information about road conditions.

A more common application is guidance of AGVs in the manufacturing industry where the magnetic systems often consists of roughly a magnetic tape and a magnetic field sensor [3]. There are numerous of companies that provide complete solutions towards the industry but there are also providers of the separate components.

These kind of systems are often developed towards a specific application. Hence literature based research is often unavailable or nonexistent, which means that information about the systems is in the form of data sheets. A common factor for these systems is that there often is not a high demand for positioning accuracy.

Designing a magnetic tape guidance system that has an accuracy better than ±5 mm will set high demands on both hardware and software.

2.1.4 Encoders

Shaft encoders converts the angular motion of a shaft to bits or an analogue signal. Shaft encoders allow for *dead-reckoning*, which is the simple positioning technique that determines the AGV position based on the distance traveled by each wheel. This measuring technique is generally referred to as *odometry* [9]. The accuracy of the positioning is limited by the amount of pulses per wheel revolution produced by the encoder. The odometry has an inherent systematic error since the dimensions of the mechanical system cannot be measured exactly. Thus, it will accumulate error over time that must be accounted for.

An odometric system is also relatively sensitive to stochastic errors in the model. If the AGV ought to slip or skid due to some irregularities in the floor, or slightly bump into something, the system will suffer from an offset bias in the measurements. Unless these stochastic processes are accounted for in the model, the AGV will not be able to recover the true position without another complementary sensor input.

2.1.5 Other techniques

Sonar systems were mostly used to find operating boundaries for the AGV [10]. These systems are somewhat obsolete due to the increased computational power during the last two decades. The vision-based systems has succeeded this role.

Advancements in microcomputers allows AGVs to be adaptive and intelligent, storing information about routes and previous actions. A set of AGVs typically navigates with a ”global” map system of the local landscape. The software related to navigating this system can be rather expensive, depending on the complexity of the environment and the amount of AGVs operating in it [11]. This type of advanced path planning is out of the scope of this project, but might be useful when confirming the actual position of the AGV versus another navigation system.

There are studies suggesting WiFi (802.11) for navigating AGVs in large environments [12]. A stochastic model represents the delay dynamics in the wireless system is weighted in a Kalman filter with a measured delay. The downside with this system is that the control algorithm requires a very accurate model of the delay dynamics in the WiFi [12]. The same results could be achieved with Bluetooth/ZigBee communication, but at shorter range [12].

Another suitable setup is the use of accelerometers combined with a gyro sensor [13]. This is particularly useful with an AGV that has omni-directional wheels or is easily affected by skid by any other environmental condition. Accelerometers has an inherent weakness due to the second integral [13].

2.1.6 Sensor fusion

Indoor AGV control is inclined to a number of systematical and non-systematical errors. Typically, odometric sensors are combined with optical sensors to handle different levels of accuracy [14]. Minor mismatches in AGV wheel geometry is something that commonly has to be accounted for and measured [14]. As discussed earlier, Kalman filters can also be used to weight information from different sources [15].

2.1.7 Evaluation of sensor techniques

As seen in the previous section, systems for AGV positioning can be constructed in many ways using several different techniques. Most of these systems/techniques can be discarded due to the fact that our demand on high accuracy can not be achieved. Having an on-board solution, at least when it comes to the logic of the system, is preferred since it is another parameter to consider in the step of choosing an approach. The same goes for complexity, cost, robustness and so forth. From our basic research we found three techniques that seemed to have potential for satisfying the goals of this project, namely; a magnetic solution consisting of a magnetic tape and a few magnetic field sensors; a vision-based system using a contrast line on the floor; a solution using an array of infrared diodes (or any other optical switch) to detect a line on the floor.

Each of the above listed systems comes with both positive and negative aspects which is gathered below.

Optical

Prō

+ Suitable to navigate a fleet of AGVs.

Contrā

- Not suitable in an environment with moving equipment and personnel that might block the optical path.
- High accuracy, optic systems are quite expensive and installation-heavy.

Vision-based guidance

Prō

+ Allows for high resolution depending on hardware.
+ May be easier to realize absolute positioning.

Contrā

- Can demand high computational power.
- Sensitive in some environments.

Magnetic tape guidance

Prō

+ Allows for high accuracy while moving along a magnetic tape.
+ Robust against rough environment conditions.

Contrā

- Magnetic fields are complex.
- May be sensitive for interference.
- It may be difficult to realizing absolute positions due to the complex geometries of the magnetic fields.

Encoders

Prō

+ Simple implementation.
+ Fast calculations.

Contrā

- Accumulates error over time due to uncertainties in geometries.
- Relatively sensitive to stochastic errors (such as wheel slip).
- Can not recover from errors alone.

Infrared diode array guidance

Prō

+ Simple hardware setup.

Contrā

- Low accuracy, depending on the distance between each diode.

2.1.8 Selection of sensor techniques

After discussion and evaluation of the different sensor techniques, we concluded that odometry fused with a vision-based system should satisfy the demands of this project. At the first glimpse of the utility of magnetic guidance in warehouses, it seemed like a suitable approach. We realized quite soon with a simple examination of Hall effect sensors that it will be very difficult, if not impossible, to achieve absolute positioning in the docking phase. Also, high-accuracy magnetic sensor arrays are in the range of 10 000 - 20 000 SEK, which is quite expensive for a seemingly plain task. Ordering such a device for the sake of experiment is indeed objectionable.

We decided to proceed with a vision-based approach instead. The accuracy is limited by the resolution of the camera, the image processing algorithm and the illumination/contrast conditions of the environment. This approach also allowed for a fairly simple test, since we were already in possession of a web camera. To find a corresponding starting point along the predefined path is a mere geometric interpretation of an image collected from the camera. Fused with encoder measurements (odometry), the system should have satisfying redundancy.

2.2 AGV Navigation

This section covers the concept of AGV *path planning* and *localization.* As the terms suggests, the aggregation of these two will determine the manner in which the AGV performs its navigation. The methods and algorithms that constitutes the navigation is the very foundation for success in this project.

2.2.1 Path planning

Path planning is an activity on the cognitive level of the AGV. Typically it involves aggregating partial ambient information so that the AGV can navigate in a node-to-node manner. On a higher cognitive level of the AGV, there is a navigation architecture which assembles *path planning*, *exploration*, *obstacle avoidance* and *localization.* This project will only cover *path planning* combined with *localization.* Navigation architecture mainly differs in the way that the strategy is decomposed [16]. It is critical that the strategy is well-reasoned and carefully selected to ensure proper utilization. To have a successful path planning algorithm, the following aspects should be considered

- Ambient and/or path information must be updated continuously.
- Unanticipated events must be managed in a constructive manner.

The path planning algorithm is considered *complete* if and only if the AGV states, trajectory with discrete nodes, sub targets and main target exists such that the AGV may complete its task [16]. There are mainly three rather significant simplifications that will constitute the foundation of the path planning algorithm. One may argue regarding the information overlap in the following statements. First, since the AGV is operating at relatively low speed (maximum operating speed below 50 mm/s when in parallel alignment mode), dynamics such as inertia, slip and lateral acceleration rarely requires any regard. This advantage also facilitates control issues on the cognitive level of the AGV. The second simplification is that a differential-drive AGV, such as in this project, is *holonomic*[1]. The states describing the position of the AGV (x, y, θ) has three degrees of freedom (DoF), and has the holonomic constraints $(\dot{x}, \dot{y}, \dot{\theta})$ [16]. Since a differential-drive AGV can rotate around its center point, the movement can be regarded as holonomic. Third, the AGV is modeled as a point in a 2-dimensional Euclidean space.

Although it is usually a very large portion of the problem solving, *obstacle avoidance* is not within the scope of this project. The present environment will not include any obstacles of consideration. In a general case, different approaches visualizes obstacles in different manners (polygons, Voronoi diagrams, cell decomposition, potential fields etc.) [16].

[1] *Holonomicity* describes the relation between total degrees of freedom in the AGV, or more generally, in any system, versus the amount of controllable degrees of freedom. If the the amount of controllable degrees of freedom is *less* than the total amount, the system (AGV) is considered non-holonomic. If the amount is equal, the system is considered holonomic.

We decided to evaluate three different techniques for the implementation of path planning, namely *Follow-the-carrot*, *Pure pursuit* and *Quadratic curve*. These techniques are presented below.

Follow-the-carrot

Follow-the-carrot is one of the most intuitive and easy path planning algorithms that exists. A line is drawn between the local AGV origo and the current "carrot"-point (sub target). The *orientation error* is defined as the angle between the local AGV coordinate system and the carrot point. Some control regulator (P/PI/PID) minimizes this error and the AGV drives towards the point. The obvious downsides of this implementation are that it cuts corners and allows for large oscillations if the carrot-points are adjacent and the AGV is operating at higher speed [17].

Pure pursuit

The pure pursuit algorithm calculates a circular segment between the local AGV coordinate system and the goal point. The local origo and the goal points share the same circle center. A steering angle is calculated related to the circular segment, and is translated into corresponding outputs to the differential-drive wheels. The goal point is continuously updated in the pursuit. The algorithm can be viewed as follows [17]

1. Fetch the local AGV coordinate system position.
2. Obtain the goal point (in the global reference system).
   (a) Compute an (arbitrary) point close to the AGV with a common center point to the origin of the AGV.
   (b) Compute the Euclidean distance between these points.
3. Transform the goal point coordinates into the local AGV coordinates.
4. Compute the circular segment (path) between the two point.
5. Move the AGV towards the goal point along the circular segment.
6. Fetch a new position of the local AGV coordinate system.

This algorithm is normally more stable than the *Follow-the-carrot* approach [17]. The smoothening of the path into a curvature allows for less oscillations.

Quadratic curve

Yoshizawa et al. [18] proposed a quadratic curve path tracking method, which they claim is simple to implement, requires low computational power and follows the desired path accurately. This section will summarize that proposal.

The quadratic curve algorithm uses a reference point along the desired path which is a given length away from the AGV position and then a quadratic link is calculated between the two points. As a second step the control inputs to the AGV are calculated as rotational and translational velocities in a way that the AGV will move along the quadratic link. A graphical explanation is shown in Figure 2.2.

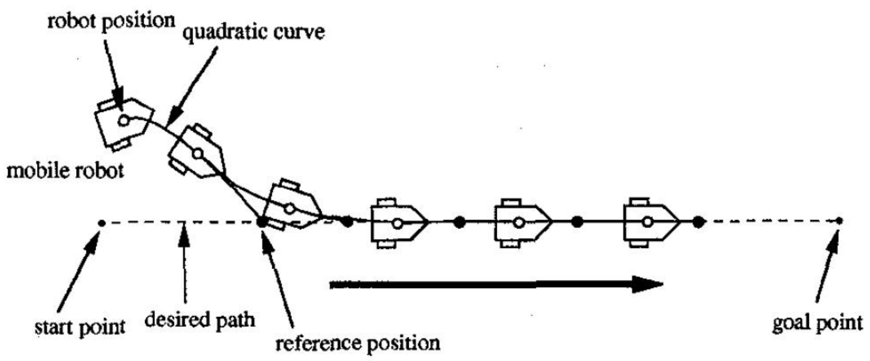


Figure 2.2: Concept of the quadric curve method [18].

Implementation of the method requires a few mathematical expressions, defining the reference state vector as $X_r = (x_r, y_r, \theta_r)^T$ and the AGV state vector as $X_c = (x_c, y_c, \theta_c)^T$. The quadratic curve in local AGV coordinates is calculated by

$$y = Ax^2, \quad A = \frac{\text{sign}(e_x)e_y}{e_x^2} \tag{2.1}$$

where the error vector, e, is calculated by transforming the global error, $(X_r - X_c)$, according to the following equation

$$\begin{pmatrix} e_x \\ e_y \\ e_\theta \end{pmatrix} = \begin{pmatrix} \cos\theta_c & \sin\theta_c & 0 \\ -\sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_r - X_c \end{pmatrix} \tag{2.2}$$

To make the AGV move along the quadratic curve, the AGV control signals are derived as

$$v = \text{sign}(e_x)\sqrt{\dot{x}^2\left(1 + 4A^2x^2\right)} \tag{2.3}$$

$$\omega = \frac{2A\dot{x}^3}{v^2} \tag{2.4}$$

where ω is the rotational velocity and v is the translation velocity. Thus the AGV will move forward when $e_x > 0$ and backward in the other case. If x at time $n\Delta t \leq t < (n+1)\,\Delta t$ is given by

$$x = K_n(t - n\Delta t), \quad K_n = \text{sign}(e_x)\frac{\alpha}{1 + |A_n|} \tag{2.5}$$

then equations (2.3) and (2.4) can be approximated to

$$v_n \simeq K_n \tag{2.6}$$

$$\omega_n \simeq 2A_nK_n \tag{2.7}$$

if the sampling period Δt is small enough. These two variables will be the desired input to the AGV and to transform them into speed references for the motors the following relationship can be used,

$$\begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} = \begin{pmatrix} r/2 & r/2 \\ r/W & -r/W \end{pmatrix}^{-1} \begin{pmatrix} v_n \\ \omega_n \end{pmatrix} \tag{2.8}$$

where ω_r is the angular velocity of the right wheel, ω_l is the angular velocity of the left wheel, r is the wheel radius and W is the distance between the two rear wheels.

2.2.2 Selection of path planning technique

We decided to choose the Quadratic curve algorithm for path planning due to the lack of precision of the other techniques. The *follow-the-carrot* technique cuts corners and the *pure pursuit* limits the virtual path to sections with circular segments.

2.2.3 Localization

Localization is a process where the AGV recognizes its position in a defined reference system by estimation techniques. Typically, the reference system is a three-dimensional Cartesian system constituting the room occupied (partly) by the AGV. Below we will discuss a couple of approaches that we came across in the pre-study phase.

Kalman filtering

To determine the state of the system, i.e. the position and velocity of the unit, a typical approach is to use a Kalman filter algorithm. The Kalman filter utilizes data fusion between sampled measurement data from the unit and a linear dynamic model of the system. The recursive algorithm estimates the internal state of the system from a set of measurements that are noisy. In this context, the linear dynamic system would be represented by a kinematic model of the AGV, with measurements from the wheels to determine position and velocity. Weighting between these two information sources will produce estimated state variables. For a regular Kalman filter, there are some requirements to provide an optimal algorithm

- The dynamic system is linear. There are extensions that can handle nonlinearities but there are draw-backs. One is that a linearization can produce unstable filters if some assumptions are incorrect [19].
- The parameters of the dynamic system must be accurately represented.
- The noise sources have a Gaussian distribution and remain Gaussian after passing through the dynamics of the linear system [20].

Other estimation techniques

Another useful tracking filter is the *Alpha-Beta* which can be considered as a light version of a Kalman filter. The Alpha-Beta filter requires less memory and computational power since it inherently has less internal states and is suitable for simple systems that does not require a rigorous state-space representation [21]. The advantage here is that it allows for even faster real-time tracking. Although this real-time performance is not required for the AGV in this project, the Alpha-Beta filter is a plausible solution.

Variants of the Kalman filter such as *Ensemble Kalman filter*, *Extended Kalman filter* and *Fast Kalman filter* are suitable for filtering problems that are demanding; large number of state-space variables, nonlinear systems or real-time requirements.

2.2.4 Selection of localization technique

The Kalman filter estimation technique, or rather the Extended Kalman filter was selected for position estimation (localization). The reason was that we had earlier experience with the technique, and we found a AGV model representation that suited the purpose. The Extended Kalman filter deals with non-linear models, such as the one described in the next section.

2.3 Model representation

A system model is described by filtering equations, where the variables are multivariate Gaussian distributions. The model yields a probabilistic representation of the relationship between the measurements and the selected states. The kinematic model used for this AGV is from Wang [22], given below (Figure 2.3).

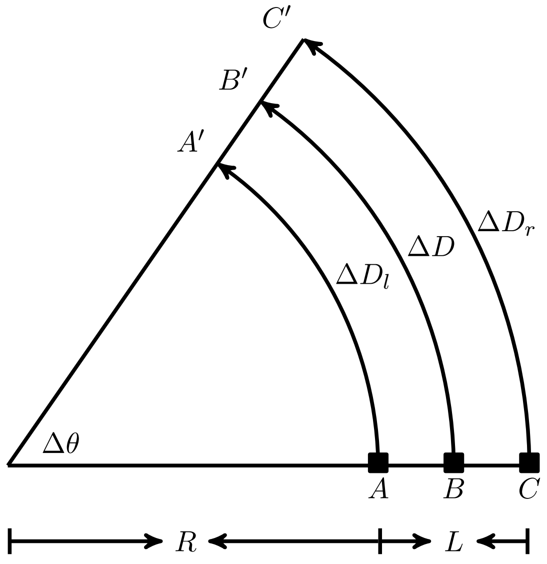


Figure 2.3: Representation of AGV rear wheel pair travel path.

The rear wheel pair of the AGV has a common axis along the circular segment B to B'. This represents the "true" position of the AGV. The left wheel travels along the path A to A' and the right wheel travels along the path from C to C'. L indicates the distance between the rear wheels. The distance traveled ΔD by the common axis and the changed angle $\Delta\theta$ yields

$$\Delta D_r = (L + R)\Delta\theta,\ \Delta D_l = R\Delta\theta$$

Giving,

$$\Delta D = (\Delta D_r + \Delta D_l)/2 \tag{2.9}$$

$$\Delta\theta = (\Delta D_r - \Delta D_l)/L \tag{2.10}$$

ΔD is thus the average of of two wheel paths ΔD_r and ΔD_l and $\Delta\theta$ is proportional to the difference. Further, the position P of the AGV in 2-D space at a given sample n is represented by

$$P_k = (X_k, Y_k, \theta_k)$$

If the previous position is given by X_{k-1} and Y_{k-1} and the previous orientation is given by θ_{k-1}, there is a common method to approximate the position along an arbitrary path given by

$$X_k = X_{k-1} + \Delta D_k \cos\left(\theta_{k-1} + \frac{\Delta\theta_k}{2}\right) \tag{2.11}$$

$$Y_k = Y_{k-1} + \Delta D_k \sin\left(\theta_{k-1} + \frac{\Delta\theta_k}{2}\right) \tag{2.12}$$

$$\theta_k = \theta_{k-1} + \Delta\theta_k \tag{2.13}$$

2.3.1 Extended Kalman filter

To estimate the position of the AGV, an Extended Kalman filter (EKF) needs to be implemented. The EKF is adapted to handle non-linearities in the dynamics of the physical model. The positioning system is composed of a dead-reckoning subsystem which utilizes rotary encoders on the separate rear wheel pair and a vision-based system (camera) that is directed downward to the floor.

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1} \\ \mathbf{z}_k &= h(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{2.14}$$

where $\mathbf{w}_k$ is the system noise, which is all disturbances and uncertainties in the subsystems such as slip, wheel diameter, real wheel distance, uneven substrate and constraints on all wheels. In addition, the vision-based system have pixel errors and image artifacts that contributes as well as systematic errors such as camera placement bias. $\mathbf{v}_k$ represents all measurement/observation noises. The system and measurement noises with covariance matrices Q and R respectively are assumed to be zero mean random variables with Gaussian distributions that depend on the state variable $\mathbf{x}_k$

$$\begin{bmatrix} \mathbf{w}_k \\ \mathbf{v}_k \end{bmatrix} = N \sim \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \right)$$

The state transition matrix $\mathbf{F}$ is defined by the following Jacobian

$$\mathbf{F}_{k-1} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_{k|k-1}}$$

with the following states

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \theta_k & \dot{D}_{r,k} & \dot{D}_{l,k} \end{bmatrix}^T$$

where x, y and θ are given by equations (2.11), (2.12) and (2.13) respectively. The EKF algorithm is then calculated as follows below. The superscript $(^-)$ denotes predicted value and the hat accent $(\hat{\ })$ denotes estimated values. Step 2 to 6 are iterated for each measured sample.

Step 1. Set initial values

$$\hat{\mathbf{x}}_0 \text{ and } \hat{\mathbf{P}}_0$$

Step 2. Predict state and error covariance

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \mathbf{F}\hat{\mathbf{x}}_{k-1}\\ \mathbf{P}_k^- &= \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}\end{aligned}$$

Step 3. Compute Kalman gain

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1}$$

Step 4. Compute estimate

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-)$$

Step 5. Compute error covariance

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{H}\mathbf{P}_k^-$$

Step 6. Update Jacobian

$$\mathbf{F}_{k-1} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_{k|k-1}}$$

Note that the measurement matrix H will be different depending on the measurement subsystem. The odometry will measure $\dot{D}_{l,k}$ and $\dot{D}_{r,k}$ (or rather pulses translated into speed) and the vision-based system will measure y_k and θ_k at each sampling instant respectively. z_k is a measurement at time k.

Selection and tuning of parameters

Setting a "proper" initial error covariance $\hat{\mathbf{P}}_\mathbf{0}$ is critical in the Extended Kalman Filter algorithm. $\hat{\mathbf{P}}_\mathbf{k}$ can be regarded as the accuracy of each estimation. Since $\hat{\mathbf{P}}_\mathbf{0}$ represent the error covariance of the first ($k = 0$) estimation, the values can be very close or equal to zero if the initial states $\hat{\mathbf{x}}_\mathbf{0}$ are known, i.e. defined.

Selecting the measurement error covariance matrix $\mathbf{R}$ might be difficult for some applications. The values on the diagonal represent the error covariance of the respective measurement. Unless there exists prior experiments with the exact same conditions, setting these to a suitable value might be non-trivial. In our experience, the pursuit of these values is a trial-and-error process with an initially qualified guess.

There are different methods for specifying the system noise covariance matrix $\mathbf{Q}$. One of these methods is described here. Basically, the $\mathbf{Q}$ matrix reflects how the system can change between every sample. Depict a system that describes a train traveling on a railroad with two parameters, position along the railroad and its speed. Then a normal assumption is that the speed of the train will not change between two samples if the sample time is small enough. So what happens when the train accelerates? This is where the **Q** matrix comes in. If the acceleration can be in the size of a and the system states are x and v ,then the **Q** matrix can be calculated by[2]:

$$\mathbf{Q} = \mathbf{B}\mathbf{Q_1}\mathbf{B}^T = \begin{bmatrix} 0 \\ v \end{bmatrix} \begin{bmatrix} a \times \Delta T \end{bmatrix} \begin{bmatrix} 0 & v \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & v^2 \times a \times \Delta T \end{bmatrix}$$

where **B** is selected as the columns from the system model that represents speed (system change per time unit) and the diagonal of $\mathbf{Q_1}$ should be assigned in the size of the systems acceleration. Determining the **Q** matrix in this manner would probably not be necessary if the system is linear and has a constant sampling time since **Q** would be constant, a trial and error method could be sufficient. But in the case of the opposite, a linearized system with a varying sampling time, the above described method enables improved compliance.

[2]Lars Hammarstrand, Post-Doc Research Fellow at the Department of Signals and Systems, Chalmers University of Technology

2.3.2 Overview

In Figure 2.4 below is an illustration of the estimation sequence used in this project. The vision system collects observations y_o, θ_o at a sample rate limited by the frames per second that the PC and/or camera can process. A rotary encoder collects observations (pulses) as often as the device allows. The pulses are translated into respective wheel speed determined by time between samples, wheel radius and amount of pulses per wheel revolution. Using previous reference states x_R, y_R, θ_R from the system model (stored in software variables), the Kalman filter fuses the data in different steps (described earlier in section 2.2.3) into new estimated states.

In the AGV Control "box" is the quadratic curve algorithm that tracks the estimated states in conjunction with the respective virtual path that is stored in a database.

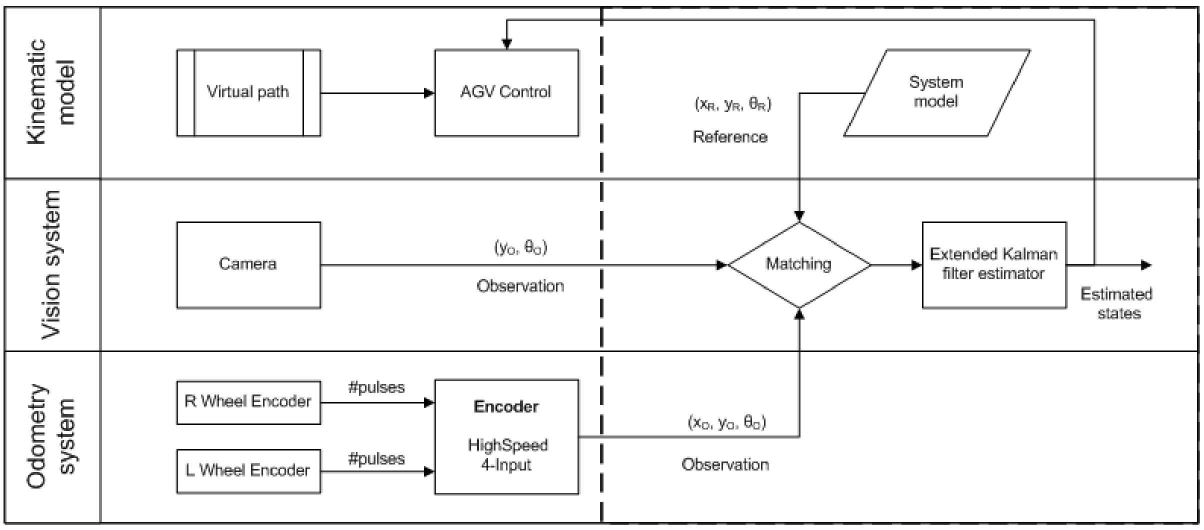


Figure 2.4: The Extended Kalman filter fuses observed data from the camera y_o, θ_o and the odometry x_o, y_o, θ_o and generates new estimated states in conjunction with the non-linear system model.

2.4 Software development

The task of developing software always requires numerous choices to be made before any actual programming can be initiated, and these choices highly depend on the target hardware and purpose of the application. The first and major choice is which programming language to use. In some cases this choice is fixed (e.g. an Android application) but for this project there was no such limits. Therefore we choose the programming language that we felt most comfortable with, C++. After this point the search for development environment and libraries started.

2.5 AGV prototype environment and conditions

The operating environment of the AGV determines conditions and limitation on the methods of implementation. The provided AGV prototype is the foundation of the physical constraints, such as turning radius and control maneuverability. This is described below.

2.5.1 AGV prototype description

The AGV prototype is based on a power-driven wheelchair with combined indoor/outdoor use. The model is a *Storm*^{3}, provided by *Invacare*®, stripped of seat, leg support and arm support, see Figure 2.5. The electrical unit in the back was moved to the front and a scaffold bottom plate was welded to the top to bear the two 12 V batteries. Cabling to the respective lamps were also removed. The rear wheel pair are pneumatic, puncture-proof with dimensions $14'' \times 3''$ and separated by a distance of approximately 0.57 m.

The experimental environment where the prototype was tested did not resemble the ideal conditions of the actual operating environment. The base frame body of the prototype is exposed to the ceiling light which casts shadows under the body where the camera is collecting visual information. Casings and apron were not available at the time to shield from the direct illumination of the ceiling light sources.

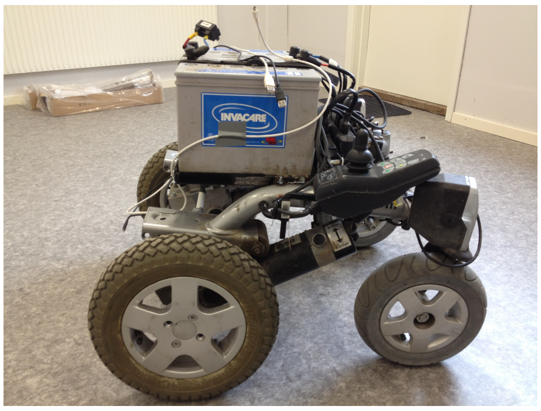


Figure 2.5: The stripped and modified power-driven wheelchair.

The environment also postulates an obstacle-free surrounding, or at least not objects in the near vicinity of the AGV that requires consideration. Thus, no obstacle detection or handling is needed. Such constraints can easily be added to the virtual map representation.

The first setup of the optical path consisted of a simple black electrical insulation tape with a relatively reflective surface. The tape was attached to the floor in a straight line (without any tools for correction). The background (the floor) was a canescent, mottled pattern, which is far from the ideal contrast conditions. The optical path is ideally straight, i.e. one-dimensional in a two-dimensional Cartesian system.

Other mechanical conditions worth of note were the characteristics of the front wheel pair. The standard front wheel pair followed the *Storm*^{3} base had a wheel diameter of 25 cm and a width of 7 cm (measured). The wheel width combined with the rough rubber surface provided a friction that required an unnecessarily large torque from the differential drive. The body frame was slightly twisted on some shafts, resulting in a uneven distribution of forces from the front wheel pair to the ground.

2.5.2 PC system information

The AGV control is running on a Windows 7 (32-bit)/PC environment with system specifications of Intel® Core™ Duo CPU T6600 @ 2.20 GHz, GeForce GT 130M and 3 Gb usable RAM.

2.6 Error awareness

To be able to predict and correct for stochastic and systematic errors, these must be addressed before implementation. Although there most certainly will manifest unforeseen errors during the project progress and after, awareness will save a lot of time.

Stochastic errors

- Skid.
- Sudden change of illuminating conditions.
- Mechanical deformation.
- Floor irregularities.
- Collision.
- Sudden image artifacts.

Systematic errors

- Wheel diameter.
- Rear wheel distance.
- Camera fixation (angle, placement).
- Image acquisition noise.

2.7 Summary

The AGV prototype is based on a power-driven wheel chair with combined indoor/outdoor use. The AGV concept is dual rear drive and steer wheel with a steering mode called steered-wheel steer control. A vision-based system fused with dead-reckoning from encoder measurements are used for the AGV perception. An algorithm called quadratic curve performs the path planning sequence in the navigation. The AGV estimates its position with an Extended Kalman filter algorithm.

3

Implementation

THIS CHAPTER covers the implementation of the selected approach, including hardware interfacing, software and driver installation, modifications and parameter tuning.

3.1 Components used for realizing the solution

A number of components were added to the wheelchair described in Section 2.5. These can be divided in two categories, measurements and control. Together with the component list below, Figure 3.1 presents a schematic of how the parts where connected.

1. PC laptop
2. PhidgetEncoder HighSpeed 4-Input
3. PhidgetAnalog 4-Output
4. Low-price web camera
5. Voltage divider
6. Optical pulse encoders
7. Wheelchair - Storm^3 with an embedded motor controller.

For odometry measurements two pulse encoders from *Henglster*[1] were used, one for each wheel. They produces 720 pulses per revolution or 2880 when also counting rising and falling edges of each pulse. The encoder shaft was connected to drive shaft via a couple of cogs and a timing belt. Due to the method of mounting the encoders the relationship

[1] `http://www.hengstler.com/`

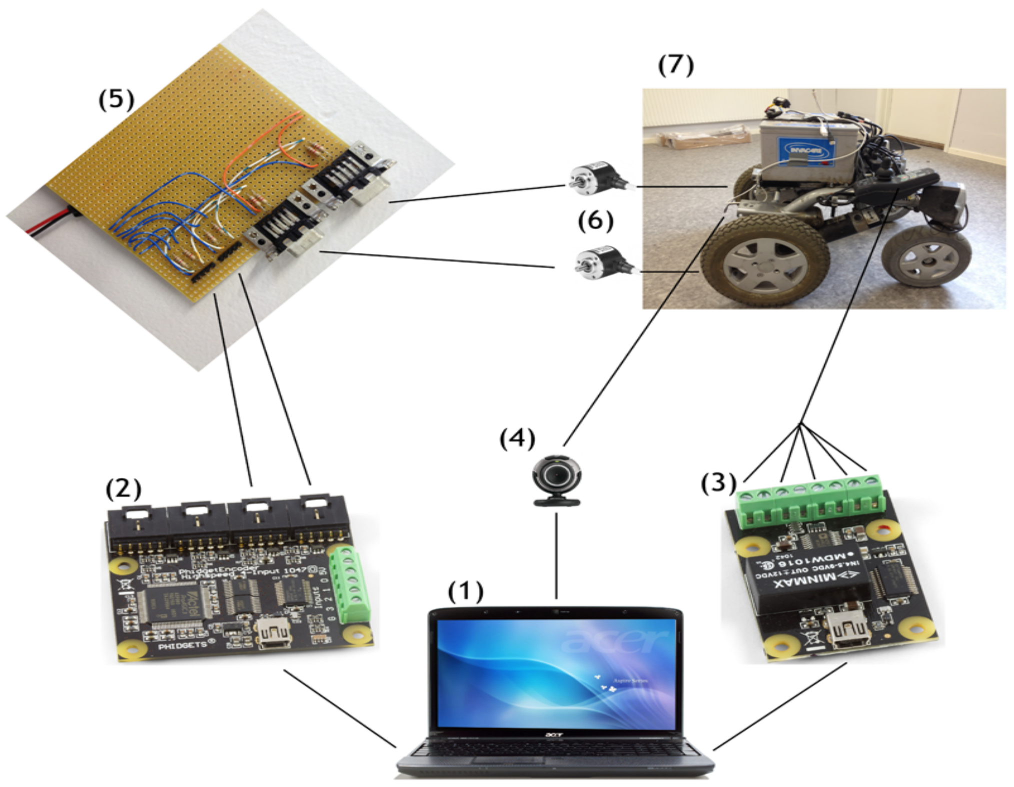


Figure 3.1: Prototype setup for this project.

between encoder rotation and wheel rotation became 7:20, i.e. one wheel revolution corresponds to approximately three encoder revolutions. Each pulse is represented by a voltage that corresponds with the input voltage which could be in the range 10 and 30 V. The pulses were measured using the high-speed-counter USB-gadget from *Phidgets Inc.*[2]. Since the counter required a pulse voltage between 1.8 and 5.0 V a voltage divider was constructed and placed between the encoders and the counter.

Optical measurements were realized with the low-price USB web camera. This camera produces a maximum of 30 images per second, depending on light conditions, with a resolution of 640×480 pixels. The camera was mounted between the two rear wheels with a distance of approximately 15 cm from the ground level. The camera center point should be following B to B' as described in Section 2.3. This will allow for easier measurements since they will correspond directly to the AGV coordinates.

To be able to control the electrical motors on the wheelchair the joystick was replaced by the analog output USB-gadget from *Phidgets Inc.* and in this way still using the embedded motor controller. From measurements with a multimeter it was determined that the joystick produces different voltages on four channels. Two channels were for setting forward/backward speed and the other two controlled the turning speed. Both pairs of channels functioned in the same manner. A center voltage (joystick in resting

[2]`http://www.phidgets.com/`

stage) of 2.5 volt and joystick maneuvers that resulted in equally sized voltage changes in opposite directions on the two channels. The embedded motor controller had only current controlled feedback.

3.2 Graphical User Interface

For this project a graphical user interface (GUI) would not be necessary but it enables easy simulation, verification and interacting. The time spent on creating a GUI paid off good. Main features are the ability to view the AGV‘s global position, what the camera sees and detects (such as lines and patterns), important parameters and the occurrence of any unwanted event.

3.2.1 Simulation

Before the developed software was executed on the real system, a number of simulations and testing were performed with the assistance of the GUI. The most important steps are listed below:

- Simulating path generation.
- Testing Kalman filter with simulated values.
- Testing motor control algorithm, i.e. path tracking.
- Simulating software sequences.
- Testing image processing algorithms.

These tests would be very hard to perform without the GUI, and in some cases impossible.

3.2.2 Verification

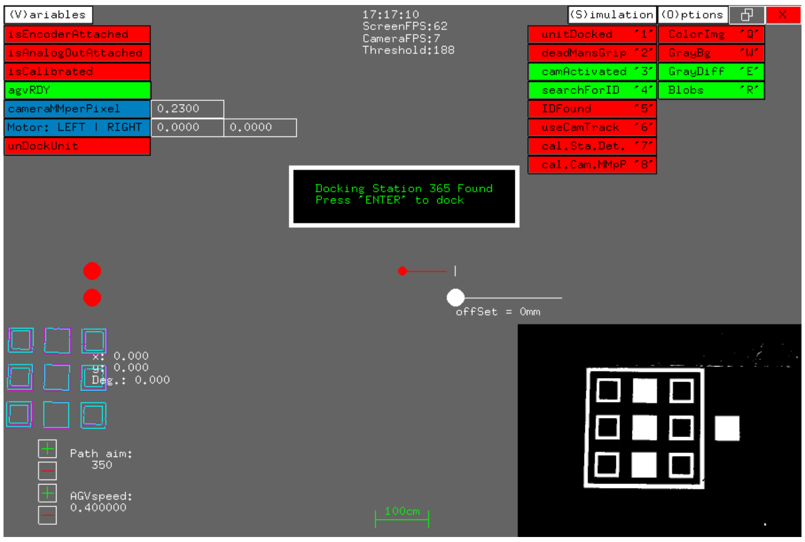
When the software first was executed on the target system, the GUI was mainly used to verify measurements from the encoders and the camera. Later on it also served as a tool for verifying Kalman filter parameters, global position and calibrating rear-axis length and tire radius on the AGV.

3.2.3 Interacting

In the final software version user input is only needed in a few occasions, but there are possibilities to get a lot of information about the system‘s current state. An example of when user input is needed is demonstrated in Figure 3.2.

3.3 Software Implementation

We faced basically two major programming challenges, namely implementing Extended Kalman filtering and designing image processing algorithms for camera measurements.



```
Docking Station 365 Found
Press 'ENTER' to dock
```

Figure 3.2: Final graphical user interface.

For both tasks, testing started out in MATLAB since it allows for easy access to mathematical operations, which is not the case with C++ where the standard library only supports simple arithmetic operations. When it was time for porting the Kalman filter algorithm from MATLAB to C++ we directly discovered that C++ has no built-in support for matrix operations. This was solved by using an already existing C++ library called *Newmat*[3], created by Robert Davies.

For the image processing part, no MATLAB code was ported to C++, mainly because it was created in a very early testing stage and that there is a widely used C++ library [23] for real-time image processing, *OpenCV* (Open Source Computer Vision Library). *OpenCV* was not used directly but through an open source C++ toolkit, *openFrameworks*[4]. *openFrameworks* is designed to make it easier for programmers to make use of commonly used libraries. Note that the image analysis/object detection algorithm of OpenCV will not be disclosed in this report. This information is available in the OpenCV documentation.

All C++ coding was performed in *Code::Blocks*[5] which is an open source cross platform Integrated Development Environment (IDE). The main reason for choosing *Code::Blocks* as IDE was its compatibility with openFrameworks.

[3]`http://www.robertnz.net/index.html`
[4]`http://www.openframeworks.cc/about/`
[5]`http://www.codeblocks.org/`

3.4 Camera measurements

The vision-based system (camera) is used to extract information from three different situations that will be described in this section. The first situation where camera measurements are used is when the AGV is standing still at a docking station and needs to acquire information about the environment (working cell), which is stored in a database.

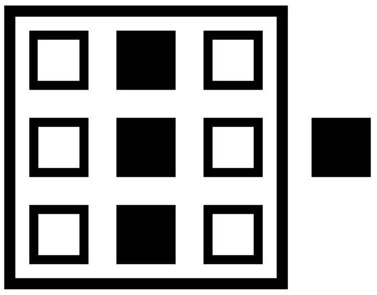


Figure 3.3: Example of a station ID pattern.

The environment is defined as a *Station*, each with a unique ID, with corresponding coordinates for important locations. To get access to the right *Station* information, the *Station* ID is acquired by reading an optic pattern on the floor similar to the one shown in Figure 3.3. The pattern is a binary sequence that corresponds to a unique Station ID.

The optic pattern is built-up by eleven squares, where one big square encloses nine smaller squares which can be either filled or just a frame. The last square in the pattern is located adjacent to the cluster of squares and is a reference for measuring the angle of the AGV at its current position. The center point of the cluster is defined as a coordinate in the environment and the displacement to that point is measured. The displacement together with the angle defines the AGV's current position in a global coordinate system.

The second situation is when the AGV is traveling along a straight optic path and needs information about displacement and angle to increase the accuracy of its position. An example is presented in Figure 3.4. The measurements will only be registered if the line does not "touch" the right or left side of the camera image and if the detected path has the correct width.

In the third and last situation the AGV is standing still along an optic path and needs to calibrate its global position. In this situation a white circle, with a predefined radius and predefined coordinates for its center point, will be present on the optical path as shown in Figure 3.4. By measuring the AGV displacement in comparison to the circle mid point the global position variables X and Y can be updated with high accuracy. The angle of the AGV in the global coordinate system will be updated in the same way as in the second situation described above.

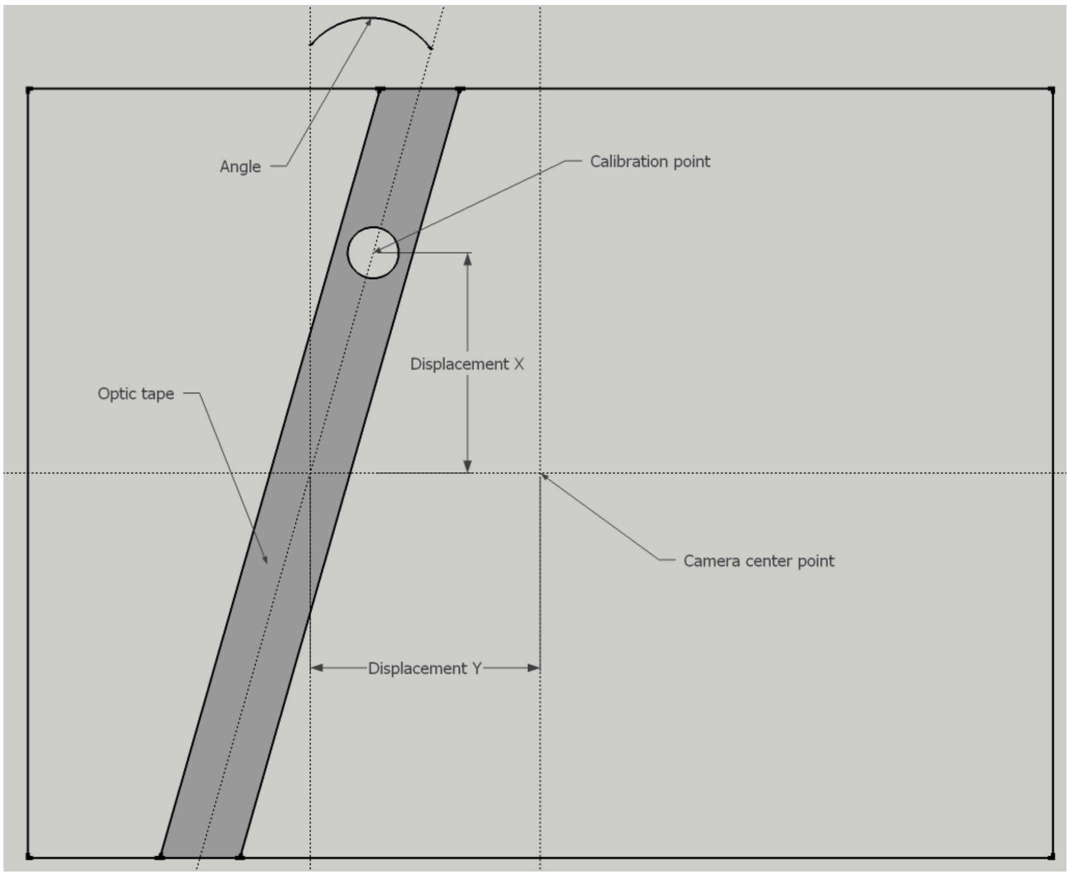


Figure 3.4: Description of parameters extracted from camera images when a line and calibration point is detected.

The accuracy of these measurements is determined by the camera resolution, the distance between the camera focal point and the floor and the view angle of the camera. With these three parameters a relation between camera pixels and real world distance can be determined. This setup can be viewed in Figure 3.5. In our setup one pixel represented approximately 0.2 mm. The image noise affected our measurements with a change of one or two pixels, which corresponds to a tolerance radius less than 0.5 mm on the centre point of a measurement.

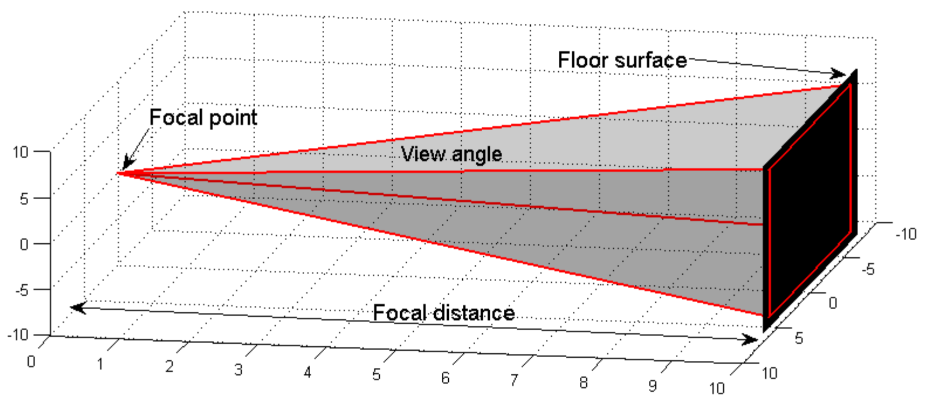


Figure 3.5: Illustrating important variables to consider when mounting the camera.

Regarding the software part of these measurements there is mainly one parameter, *threshold*, that needs to be tuned depending on light conditions and the contrast between background image and the captured frame. The background image can be either manually set to e.g. white or a captured image of the floor (without any obstacles present). The latter was not an option in our case since the camera was moving and the floor had a canescent mottled pattern. Then every captured frame is compared with the background image, generating a new image. As a last step before the image is processed with object detection algorithms, the *threshold* is applied. This action discards any pixels that does not satisfy the *threshold* value, i.e. does not have enough contrast compared to the background image.

3.5 Extended Kalman filter

As mentioned in section 2.3.1, the measurement matrix H will be different depending on if we use the odometry (H_{odo}) or camera together with odometry (H_{cam}) measurements.

$$\mathbf{H}_{odo} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_{cam} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In the same manner, the measurement noise covariance matrix $\mathbf{R}$ depends on which measurement that is being performed

$$\mathbf{R}_{odo} = \begin{bmatrix} r_{k1} & 0 \\ 0 & r_{k1} \end{bmatrix}$$

$$\mathbf{R}_{cam} = \begin{bmatrix} r_{k2} & 0 & 0 & 0 \\ 0 & r_{k2} & 0 & 0 \\ 0 & 0 & r_{k1} & 0 \\ 0 & 0 & 0 & r_{k1} \end{bmatrix}$$

where $r_{k1} = 10^{-3}$, representing the error covariance from the odometry and $r_{k2} = 10^{-6}$ representing the error covariance from the camera measurements. These were empirically set by simulations and test runs. The size of the measurement $\mathbf{z_k}$ is dependent on the amount of measures performed. The system noise covariance matrix $\mathbf{Q}$ was as follows

$$\mathbf{Q} = \mathbf{B} \times \mathbf{Q_1} \times \mathbf{B^T}$$

and

$$\mathbf{Q_1} = \begin{bmatrix} q_1 & 0 \\ 0 & q_1 \end{bmatrix}$$

where $q_1 = 0.01 \times \Delta t$ and Δt is the sampling time interval. The initial error covariance matrix P_0 was empirically set to

$$\mathbf{P}_0 = \begin{bmatrix} p_0 & 0 & 0 & 0 & 0 \\ 0 & p_0 & 0 & 0 & 0 \\ 0 & 0 & p_0 & 0 & 0 \\ 0 & 0 & 0 & p_0 & 0 \\ 0 & 0 & 0 & 0 & p_0 \end{bmatrix}$$

where $p_0 = 10^{-10}$. The scalar p_0 represents a small uncertainty covariance since we initially define the AGV position.

3.6 Odometry/dead-reckoning test

To be able to test the performance of the odometry on the AGV, a nascar-like track (Figure 3.6) was modeled as a simulated trajectory (virtual path) for the AGV to drive. The Kalman filter states were recorded in the software between each sample and stored on the PC to be evaluated later. This was also used to calibrate the rear axis length of the AGV model in a trial-and-error manner. Depending on the offset resulting from each lap in the track, the parameter could be changed to correct for the error. This was used as a second step in our two-step calibration method.

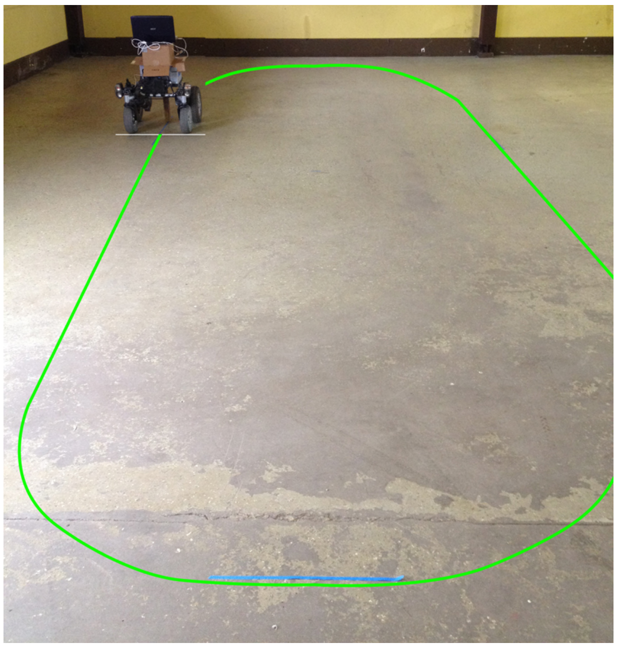


Figure 3.6: Nascar track for calibration.

As a first step the tire radius of the AGV model was calibrated by using a 15 m long straight virtual path (and also marked with tape on the floor) instead of the nascar track.

This was done until the AGV drove straight and the traveled distance error between the real world measurement and the softwares estimation was less than 10 mm. Although not very scientific, there was really no way around it using odometry alone with the crude prototype model that was provided,

In the second step, the AGV should drive three laps along the virtual path of the nascar track with a path aim of 450 mm (corresponding to the reference point in the quadratic-curve algorithm). The AGV should be able to drive both clockwise and counterclockwise with roughly equal performance. The x and y offset from the real path should be measured with an inch ruler to provide sufficient information to the calibration.

The path, as mentioned earlier, is nascar-like and has a total path length L_{tot} of

$$L_{tot} = \#laps \times (2 \times L_{horizontal} + 2 \times L_{turning}) = 3(2 \times 6 + 2 \times pi \times 1.5) \approx 64.274\,[m] \quad (3.1)$$

3.7 Summary

The components used in the project was a PC laptop, PhidgetEncoder HighSpeed 4-Input (usb interface), PhidgetAnalog 4-Output (usb interface), a web camera, a voltage divider, two optical pulse encoders and the power-driven wheel chair. A Graphical User Interface (GUI) was made to facilitate simulation, verification and interacting with the AGV unit. Simulations started in a MATLAB environment and was ported to C++ later. The vision-based system extracts unique environment information from a predefined binary binary pattern. It also measures the AGV displacement from an optical path.

4

Results

This chapter presents results from odometry calibration procedures and the final result of the task.

4.1 Kalman filter simulation on odometry

In MATLAB we simulated the performance of the filtering algorithm on the odometry system with an added pseudorandom Gaussian noise. The simulated path is arbitrary generated to represent a turning sequence. In Figure 4.1 below is a simulation where the filter values and the true values are almost aligned.

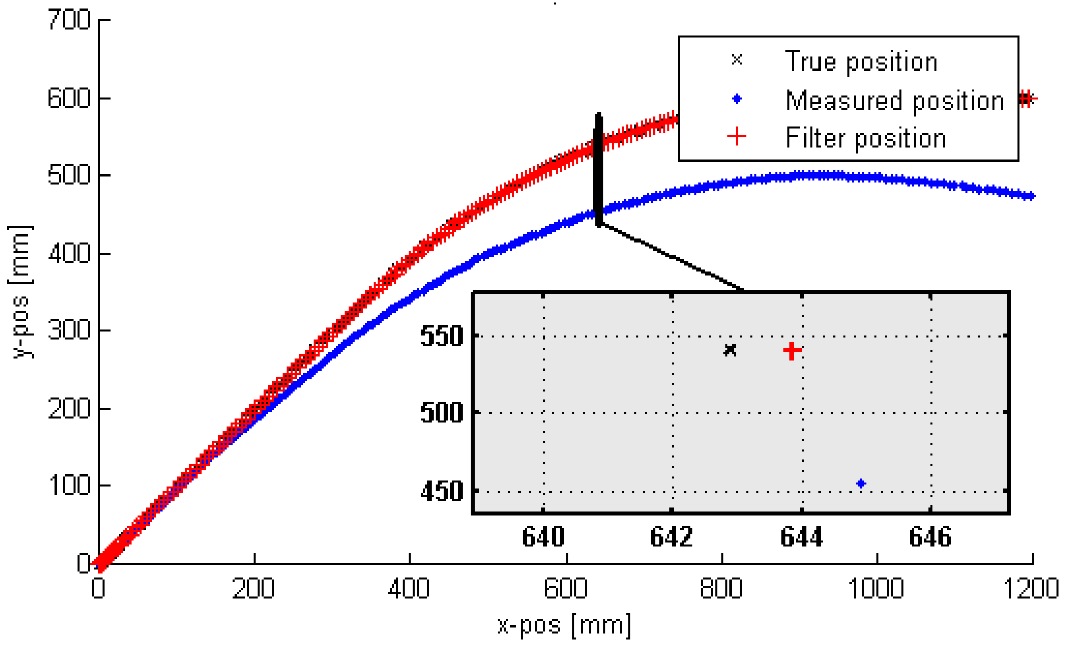
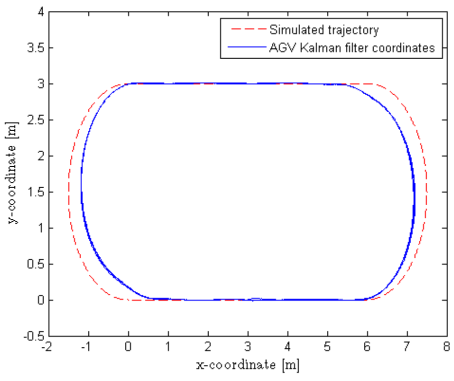


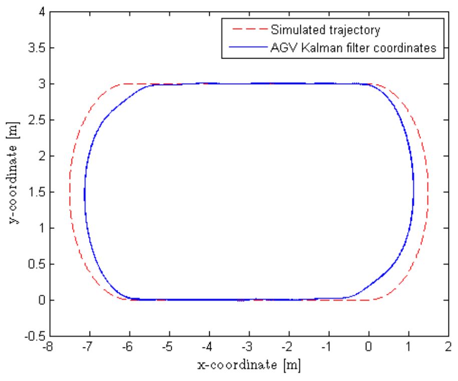
Figure 4.1: Kalman filter perfomance on the dead-reckoning.

4.2 Odometry/dead-reckoning calibration

The process of calibrating the odometry/dead-reckoning was quite time-consuming. We had no way to measure the prototype dimensions (rear axis distance and rear wheel radius) better than a 1 mm tolerance at best. Note that we could not trust the provided wheel dimensions (14") since the wheels were pneumatic. This made the adjustment to same diameter difficult as well. The weight of the AGV (50-60 kg) also contributes to a tire profile different from what is measurable when the tire is unloaded. Another issue was that the frame body had been damaged, resulting in a skew frame structure.



(a) AGV driving counter clockwise.



(b) AGV driving clockwise.

Figure 4.2: Dead-reckoning calibrations.

Although the benchmark above is crude, it reveals that the current calibration is within acceptable levels. In Figure 4.2a and 4.2b is the result of three laps in the tracks described in Section 3.6. Due to the software setting to aim on the path 450 mm ahead - as a result to the quadratic curve algorithm - the AGV will "cut corners" as seen in both figures. The path aim relates to the smoothness behavior of the AGV control. We can also derive from Figure 4.2a and 4.2b that the filter states does not deviate much from lap to lap.

With a regular inch ruler, the following data in Table 4.1 were acquired. The values are momentary states in the Kalman filter.

Table 4.1: Odometry simulation data.

	Kalman Filter States			Measured parameters		
	x[m]	y[m]	θ[deg]	x[m]	y[m]	θ[deg]
Simulation 1	3.138	-0.021	1078.214	3.04	0.03	N/A
Simulation 2	-2.832	0.016	-898.151	-2.96	-0.07	N/A

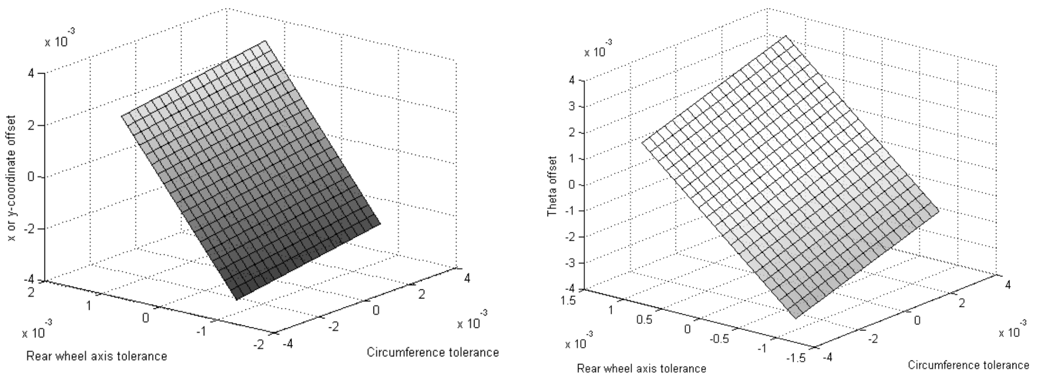
Due to the difference in wheel diameter, the AGV will behave differently depending on whether its a left turn or a right turn. Determining a fitting rear axis distance L that ought to match both laps was difficult. It is almost impossible to distinguish between very small changes in parameters, such as rear axis distance, and variations from stochastic noise (bumps et cetera) in the environment.

The key knowledge from this calibration procedure is the sole performance of the odometry system alone. We have already identified that this system will not satisfy the demands on its own.

Note that the angle θ is growing for each lap. The physical angle was not measured.

4.3 Dead-reckoning error accumulation

In Figure 4.3 below is the simulated error offset per wheel rotation using 580 mm as "true" rear axis wheel distance and and a tolerance error of $\pm$ 1 mm. 350 mm is used for the wheel diameter with $\pm$ 1 mm tolerance error (offset). The measurement sample is representing a turn with a radius of 1.5 m. The errors are represented as accumulated error per wheel rotation in this turn. To examplify, if the measured rear axis distance differs with 1 mm from the actual distance, the AGV will be displaced roughly 2 mm per wheel rotation in the x (or y since they are affected equally) plane. This figure is not meant to display exact number but rather to visualize the sensitivity of the system.



(a) x or y-coordinate offset per wheel rotation. **(b)** θ offset per wheel rotation.

Figure 4.3: Error offset per wheel rotation.

In this particular turning scenario, the dead-reckoning is quite sensitive to tolerance errors in both parameters. Note that a straight forward drive will not be affected by errors in the rear axis wheel distance.

4.4 Docking phase

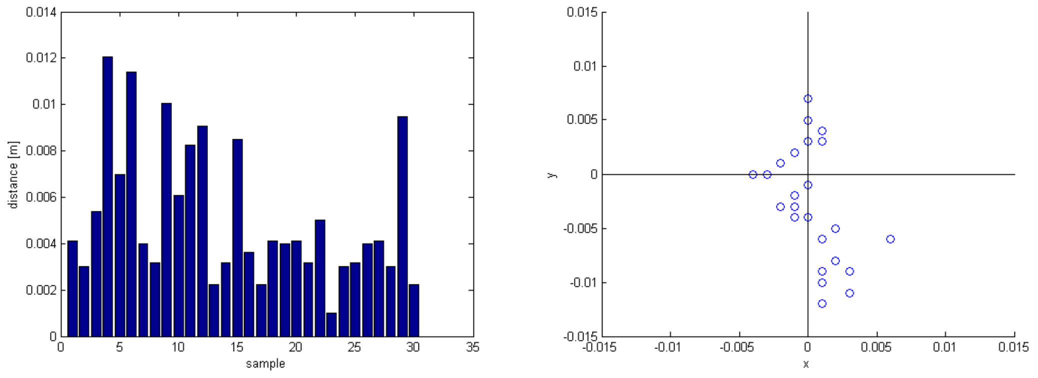
The docking sequence (see Appendix A) was repeated 30 times and the final filter estimates from the camera measurements of the origo was noted. The first column represent the mean along a one dimensional axis. This value indicates whether the AGV tends to stop before or after the desired stopping point. The absolute values (second data column) represent a mean Euclidean distance to the stopping point. The mean values and standard deviation of the Kalman filter states $\hat{x}$ [m], $\hat{y}$ [m] and $\hat{\theta}$ [deg] at the finished sequences were

Table 4.2: Docking phase data.

$\hat{x}$	0.0013	σ_x	0.0020	$\lvert\hat{x}\rvert$	0.0015	$\sigma_{\lvert x\rvert}$	0.0014
$\hat{y}$	-0.0023	σ_y	0.0051	$\lvert\hat{y}\rvert$	0.0045	$\sigma_{\lvert y\rvert}$	0.0032
$\hat{\theta}$	-0.0036	σ_θ	0.0967	$\lvert\hat{\theta}\rvert$	0.0784	$\sigma_{\lvert\theta\rvert}$	0.0549

The complete data set of measurements can be found in Appendix C. The average Euclidean distance to the calibration point was calculated to $\overline{dist} = 0.0051$ m with a standard deviation $\sigma_{\overline{dist}} = 0.0030$ m. Examining the bar plot in Figure 4.4a yields a good perspective of the AGV performance.

The corresponding scatterplot of all measured $\hat{x}$, $\hat{y}$ can be seen in Figure 4.4b. Together



(a) Measurements bar plot.

(b) (x, y) position of 30 measurements.

Figure 4.4: Docking phase results.

with Table 4.2 we can conclude that the y-offset is worse than the x-offset, and that the AGV tends to end up in the fourth quadrant (positive x, negative y). The bar plot also reveals that a few distance values above 0.01 m contributes to a large mean value over the relatively small sample size.

Another interesting figure in this sequence might be the GUI representation at that instance. A snapshot of the docking phase can be seen in Figure 4.5. The AGV (red) is illustrated on the virtual path (green). As stated in Chapter 3, the GUI has been a most supportive tool. The center of the circle in Figure 4.5 represents the global origo (0,0) in the two-dimensional Euclidean space.

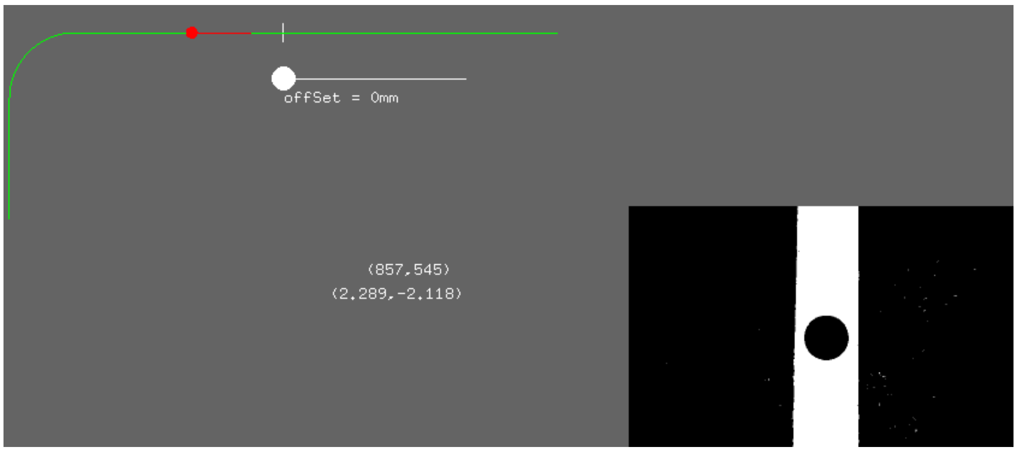


Figure 4.5: The lower right box is the camera perspective, displaying the AGV stopped at the calibration point (circle). The white line and circle with the subtext ”offSet” is a software function that represents the position of the operating tool.

4.5 Aligning phase

Instead of acquiring position data from the operating tool, we marked the optical path with a couple of white circles separated by 1 m each representing standard locations for the tool. Between those distances, the AGV drove back and forth without complications. The accuracy of this phase was not recorded since we were examining a scenario with much more favorable conditions. Thus we assumed that the accuracy is at least that of the docking phase. The x-coordinate is provided as a distance from the circle, negating the effects of the dead-reckoning. The procedures of the aligning phase can be found in Appendix A.

4.6 Software execution times

When the software development was completed we performed execution time measurements of important program sequences. The results of these measurements are displayed as mean values in Figure 4.6 below. Data was collected during a full docking sequence to make sure that all program sequences were executed an adequate number of times.

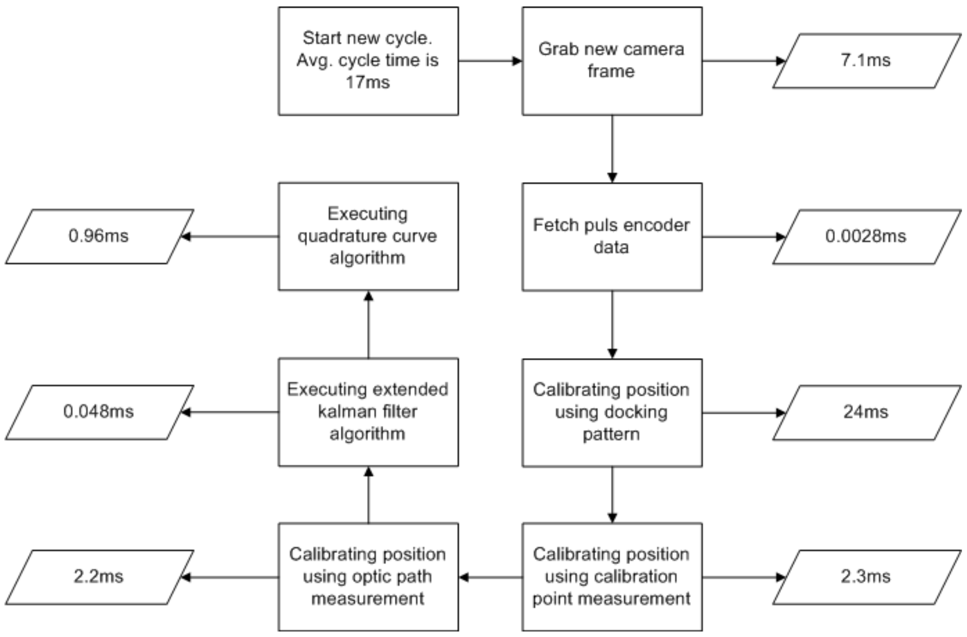

Start new cycle. Avg. cycle time is 17ms
Grab new camera frame
7.1ms
Fetch puls encoder data
0.0028ms
Calibrating position using docking pattern
24ms
Calibrating position using calibration point measurement
2.3ms
Calibrating position using optic path measurement
2.2ms
Executing extended kalman filter algorithm
0.048ms
Executing quadrature curve algorithm
0.96ms

Figure 4.6: Average execution times for important program sequences.

The average cycle time for the whole software was measured to 17 ms, though minimum and maximum values are quite far away from that value. The main reason for these disseminated measurements is the fact that execution of most program sequences are conditional. For instance when the algorithm that identifies station IDs is executed the cycle time more than doubles due to heavy computations.

4.7 Exclusions

Reviewing the objectives in Section 1.3, we particularly addressed item 5 to 8. Item 6(d) was excluded due to lack of time and item 6(c) is partially fulfilled by the fact that the AGV can align to predefined coordinates along the optical path, representing standard locations for the working tool.

Whether we successfully completed item 5 (robust against soiling, wear and tear) is arguable. If the wheels are dirt they will soil the textile tape after some cycles. The same goes for footsteps. The current optical marker thus needs regular cleaning.

5

Discussion and conclusion

To successfully predict the outcome of each phase in a project of this magnitude is rather complicated, especially when you have no experience of such an extensive task prior to the planning. We would like to address some issues encountered during the working progress and identify some ”aha-experiences”.

We would also like to prepare peers encountering similar problems to be solved. The results of this project are interesting and appealing for development and future work. Below we discuss each element of the project worth of note.

5.1 Pre-study work and preparations

In this section we discuss preparations and strategies that could have been realized differently. We also address issues of implementing the different techniques.

5.1.1 Odometry calibration

Implementing dead-reckoning and odometry was new for us and very worthwhile. We anticipated that there would be an inherent systematic error in the sensor setup due to the lack of precision in geometries and dimensions (wheel radius, axis length). We figured that the accumulated error over the short distance that the AGV was supposed to travel (initially approximately 3.7 m from the docking station to the calibration point) would not be a major issue. What we failed to prepare was a proper calibration strategy for the odometry.

As soon as the AGV has to do any form of turning, the odometry system will suffer from tolerance errors in the measured wheel diameters, the relations between the (rear) wheel diameters and the measured rear axis distance. This is another issue worth emphasizing. How do you, within an acceptable tolerance, measure the circumference or the effective wheel diameter on the loaded AGV? The dimensions provided with the donated prototype were obsolete due to the wear and deformations and there were no sophisticated tools available for us to determine them. As seen in Section 4.3, the model is quite sensitive to inaccuracies in the mechanical dimensions.

The rear wheels had pneumatic tyres, which had tiresome consequences during the calibration process. The desiccated tyre tubes of the condemned power-driven wheelchair deflated after hours of calibration work which set us back both in time and morale. We had to procure new tyres from a local store.

The same applies to the rear axis/wheel distance. Determining the actual contact point of the wheels and then actually measuring the distance within a tolerance of ± 1 mm is impracticable. The skew frame body might also contribute to an axis length displacement while turning.

The optimum conditions would have been solid core wheels with heavy rubber enfoldment. The wheels should have been industrially mounted with exact dimensions and tolerances provided.

5.1.2 Camera mounting and calibration

Implementing the vision-based system was very exciting. Besides unfolding the potential of the system, we discovered limitations with the selected approach. Mounting the camera in zenith (lens and floor parallel) to the optical path is impossible without any sort of calibration range and proper attachment. The web camera we used had a ball-and-socket joint with three independent rotational degrees of freedom (DoF) together with three translational DoF for the mounting. Subsequently, mounting the camera with a center point on the mid rear axis - representing the AGV‘s true position - is another source of error.

The floor-lens distance was successfully determined with a 10×10 mm square printed on a paper, giving the pixel proportions to the software. A similar strategy could have been adopted to determine the mounting angle. By printing a known geometric grid, the angle displacement could have been acquired.

5.1.3 AGV speed and motor output voltage

Working with the embedded motor control resulted in some issues. The motor control was only current controlled so we had no proper speed control embedded. The motor control logic also differentiated between backward and forward movement, resulting in software issues in the motor output algorithm.

The major problem with the embedded logic was dealing with small output voltages and trying to achieve low speed. Speeds below approximately 50 mm/s gave a chopping, uneven motion which indeed is an issue with the high accuracy demand. The most straight-forward solution here is to replace the old embedded logic with one that allows for higher output voltages, thus a higher resolution control.

5.2 Sensory technique and hardware

In this section we discuss the selection of sensory technique and hardware and its impact upon the results. The part of selecting control building blocks was almost completely foreign to us. Selecting these caused a natural delay in the project progress and we were quite uncertain of the outcome of implementing these Phidgets.

5.2.1 Vision-based system

Selecting a proper tape for the optical path was not as easy as we predicted. At an early experimental stage, we found that the surface of the tape has to be dull and black (ideal blackbody). The black textile tape satisfied the optical requirements, but was also quite susceptible to soiling. On the contrary, the plastic electrical insulation tape (reflective) we used in the design phase would not have absorbed as much soil.

The contrast conditions of the vision system had a large impact on the detection algorithms. The floor of the workshop where we experimented had a canescent, mottled pattern and the ceiling armature had a set of unshielded fluorescent lamps. With contributions from the room windows, the changing light conditions enacted a constant problem. The threshold value (see Section 3.4) of the object-acquiring algorithm needed to be adapted to the changing conditions. As the AGV moved and turned, new shadow geometries were casted within the camera view. We tried to shield off the worst angles with a cardboard apron.

Another issue worth of note is the camera update frequency in frames per second (FPS). The sampling rate here obviously determines the accuracy of the control algorithm. If the AGV unit travels at 100 mm per second and the camera FPS is 10, the camera will travel 10 mm between each sample. As a consequence, it is difficult to stop within ± 5 mm of any given stop point with the vision system alone. With better light conditions and a more process efficient operating environment, for instance a lean Linux kernel and a system with better performance, it would be possible to increase the FPS.

One important prerequisite is that the light conditions should not change. Changing conditions forces the camera settings to enhance certain features to be able to acquire objects. In other words, on-the-fly adjustments are not wanted. As long as the noise level and artifact interference is low, camera measurements are very precise and mostly limited to the resolution.

Before the algorithm for acquiring objects from a camera frame is executed there is one important parameter that needs consideration, namely the threshold as discussed in Section 3.4. This parameter is highly correlated with the contrast between the background image and the grabbed frame. When the light conditions change, the threshold needs to be changed which is not a straight forward task to do on-the-fly.

A very interesting advantage of using a vision-based system is that it can acquire any known geometry or pattern translated into meaningful information. The amount of information that can be stored in a *StationID*-tag is immense. Within or adjacent to the scope of this project, a *StationID* could contain

- Coordinate set generating the required virtual path.
- Object boundaries for obstacle avoidance.
- Auxiliary coordinates for e.g. a charging platform for the AGV batteries.
- Tool-specific parameters for operation.

5.2.2 Odometry

Using pulse encoders in combination with dead-reckoning as the main positioning technique allowed for easy simulation and getting started in an early stage. One should be aware that this technique requires a well defined mechanical system to be able to deliver high accuracy positioning. Miss-match between the model (measurements of tyre radius and rear-axis length) and the mechanical system will rapidly result in extensive positioning errors as the miss-match grows.

The choice of pulse encoders will, of course, also play a role. Number of pulses per revolution is the main variable of interest, though we were delayed by the fact that the voltage output from the encoders were not suited for our hardware. If one pulse represents 1 mm travel for a wheel, the positioning accuracy will be limited to 1 mm. We mounted our pulse encoders with a gear ratio which resulted in a resolution of about ten pulses per millimeter traveled, which from our point-of-view was more than enough.

5.2.3 USB interfacing

Working with the Phidgets and implementing these USB plug-and-play components were surprisingly straight-forward. With code samples for several programming languages provided, the testing and understanding were made easy. The application programming interfaces (API) documentation and libraries were all sound. Also, they were quite cheap and easy to procure.

The HighSpeed rotary encoder provided high resolution measurements from the axis shaft via a couple of cog wheels and a small timing belt.

5.3 Programming environment

Porting the matrix operation-heavy Extended Kalman filter algorithm to C++ were made easy with the *Newmat*-library we found.

OpenCV by it self is a quite complex library which requires good C++ knowledge to use. Here *Code::Blocks* was a great tool since it comes with an *OpenCV* add-on. The add-on together with with a code sample provided by *Code::Blocks* made it easy to use the OpenCV tools needed for this project. For a project more strictly based on computer vision we would advice to take a closer look at *OpenCV* and the potential it carries. Compared with other integrated development environments (IDE), e.g. *jGRASP*, that we have worked with before, *Code::Blocks* has a lot of benefiting features and we could recommend it as an option when looking for a freeware IDE.

5.4 Alternative approaches

In this section we discuss reasonable alternatives to the selected solutions and attempt to compare performance of substitutes. In some cases we can address problems, and whether that problem is worth solving or has any priority what so ever is also voiced in the review.

5.4.1 Optical path

As an option to optical path on the floor, the path also could have been attached to the ceiling. There were several reasons why we decided to go with the floor approach, although there are some drawbacks. The ceiling approach would require an even ceiling, free from armature or cabling that could interfere with the optical path or create image artifacts. Also, attaching anything to a ceiling is obviously more problematic than a floor attachment. The distance between the camera and the optical path would have been increased by possibly several tenfolds.

Theoretically, this should not create an issue as long as the camera could perform zoom or the width of the optical path is increased proportionally to maintain the robustness of the algorithm. Besides the additional performance demands on the camera, the setup will be more sensitive to lateral distortions (vibrations et cetera). To increase the robustness of the docking phase, one alternative could have been to have an optical path all the way from the docking station. One downside here is that there will be more impact on the working cell.

We also discussed a solution with an infrared emitting optical tape, passive or active. Such an approach would have been robust to soiling and resistant to light pollution in the visible spectra. Unfortunately this idea was discovered in the wrap-up phase of the project.

Another issue is that it might be more difficult to move the optical path from the docking station to the parallel path elsewhere if new geometric constraints occur in the working cell. Reprogramming the StationID with a new virtual path is easily done since we had a database (text file) with certain parameters required to draw it.

5.4.2 Path tracking

The quadratic curve algorithm performed surprisingly well. Most impressive was how it handled the issues that occurs when switching direction. The front castor wheels forces a twist in the front section of the AGV since the castor wheels have a vertical axis of rotation offset from the contact point. The rolling constraints are identical to that of a standard wheel as long as the motion is aligned to the wheel plane.

The balancing term is the path aim (reference position - see Section 2.2). The motor output will become less aggressive with higher path aim, at the cost of offset from current position to desired path. The path aim is related analogically to a proportional gain in a PID-controlled motor output. The gain is thus also dependent on the current velocity of the AGV. The report we examined for the quadratic curve algorithm demonstrated good simulation results with a similar kinematic model for the AGV.

5.4.3 Localization technique

Tuning the parameters (r_{k1}, r_{k2} and p_0) of the Kalman filter was quite time consuming, but the MATLAB model we had was an excellent tool. It is the nature of the Extended Kalman filter that it is subject to divergence due to the recursive linearization process. As presented in Section 2.3.1, if the initial estimates are wrong or the non-linear model is incorrectly formulated, the filter will diverge rapidly.

Fusing two measurement sources from two different sensory techniques was also intriguing. Initially we anticipated that multi-rate sampling and switching sensory input from different phases would be a difficult task. But with proper software timing control and support of boolean operations, the implementation proceeded quite well. At first the Kalman filter suffered from systematic errors from an (at the time) unknown source. We later found out that the standard C++ software processing time clock had too few decimal placements to accurately represent the actual time intervals between samples. A relatively large bias factor.

The Kalman filter worked quite satisfactory eventually. The natural question here is; was it necessary to implement the Extended Kalman filter, or any other estimation technique for estimation? The answer is; it depends on whether you can successfully implement tolerance parameters (bias factor) in the measurement model of the Kalman filter, as follows

$$z_k = \begin{bmatrix} y_k + \delta_{y,k} \\ \theta_k + \delta_{\theta,k} \\ \dot{D}_{r,k} + \delta_{\dot{D}_r,k} \\ \dot{D}_{l,k} + \delta_{\dot{D}_l,k} \end{bmatrix}$$

The localization was performed without the Extended Kalman filter as well. The performance was almost as good within the short distance that the AGV was planned to travel. To really asses the difference, we should have had an equal set (30) of measurements without the filter active. Note that the Extended Kalman filter is not an optimum filter for several reasons. One of the optimality criterion is the Minimum Mean Square Error (MMSE), which estimator is linear in data only if the densities are Gaussian and the models linear. Since EKF is used when the model are non-linear, it can not be MMSE. From an academic point of view, it was a very worthwhile experience to have implemented it.

5.4.4 Additional redundancy

Another interesting notion that arises from the fusion of the measurements system is whether the current setup yields sufficient redundancy to solve the task of localization. Indeed, every measurement of the same variable adds new information to the filter.

With a camera setup facing downwards, as in this project, a stereo system would not add much redundancy but rather add unnecessary computational load. With a different stereo system setup, for instance two cameras facing forward, the system could have provided another distance dimension (x) to the filtering algorithm.

One fundamental issue remains. The attachment point of both camera has to be known, and another camera might also be another source of error. Calibrating attachment points for two cameras requires an acceptable angle and distance bias. Although, as mentioned earlier, such uncertainties can be embedded in the measurement model.

What about another system to measure the absolute position? Sensory alternatives were discussed in Section 2.1. It is difficult to imagine a simpler system than the odometry with the rotary encoders. Also, considering cost and implementation, odometry has an advantage compared to other positioning systems.

A gyro would have been a suitable complement to compensate inadequacies of odometry. In addition to further redundancy to the dead-reckoning, a gyro can manage stochastic errors such as collisions, skid and floor irregularities.

It is debatable whether further redundancy would add *significant* value to the performance of the AGV positioning. Since the distance between docking point and calibration point is short and several other points along the optical path provides absolute positions, the current setup is probably enough.

5.5 Conclusions

Coupling to the main research questions stated in the introduction, we can conclude as follows:

AGV positioning. With the use of odometry and a vision-based system, we reached a mean offset of 0.0051 m with a standard deviation of 0.0030 m. With improved mechanics and a suitable motor controller, we believe that we can achieve the requested accuracy of ± 5 mm. We encountered problems with calibrating the software (embedding the kinematic model) for the crude prototype dimensions we had.

We used the Extended Kalman filter algorithm for location estimation which performed satisfactory but allows for improvement. The path tracking algorithm called *Quadratic curve* controlled the motor output in a smooth manner.

AGV navigation. The AGV could successfully navigate without any mechanical tampering on the floor. Odometry is simple and easy to implement but sensitive to stochastic errors and has the nature of accumulating errors (systematic). Vision-based systems are precise and very potent in acquiring information, but limited by the relatively computational-heavy algorithms that makes acquisition difficult at higher speeds.

As an optical marker, we used a black textile tape which had good optical properties (non-reflective) but was quite susceptible to soiling. The camera we had was cheap and had an attachment unfit the purpose of adjusting angle and placement.

Working cell identification. The vision-based system identified the current working cell by interpreting a binary pattern on the floor, corresponding to a unique working cell ID. The vision-based system also acquired parameters from the binary pattern regarding the working cell geometry to determine the virtual path.

As a final measure of success we used the feedback received from the company that provided us with the task. They acknowledged our solution as successful and recognized it as a potential add-on on future products.

5.6 Future work

The resulting prototype is an alpha version (or even pre-alpha) in terms of development stage. After completion of the project, we will use our alpha to aggregate the acquired knowledge and channel it into a beta version that is a first official demo prototype for intended investors and customers. Components related to our project that will improve the performance is

Improved front wheel pair. Castor wheels with smaller offset distance between the center axis of the shaft and the contact point gives a better compliance when changing direction. The new wheels also has less friction and a solid core.

Improved rear wheel pair. The rear wheel pair will have a solid aluminum core supported by a steel ring with a tough rubber enfoldment. These wheels are robust to aging and will (hopefully) not change dimensions over time.

Improved body frame. The body frame will be an extruded aluminum structure. The alpha version had a skew geometry.

Improved camera attachment. Calibration of the camera position with real calibration tools and a proper attachment will improve the measurement model.

Improved light conditions. A surrounding apron and proper diffuse light sources in the undercarriage will enhance the vision system significantly. This also includes an improved optical marker and a camera optimized for the task.

Improved motor controller. The motor controller hardware in the alpha had issues dealing with small output voltages, resulting in a chopping motion at low speeds. The beta motor controller will have feedback directly from the encoders rather than being current controlled.

Additional aspects worth considering in future work and improvements are *Improvement of object-detection algorithm*, *System model with bias factors*, and *Porting the setup to a lean and efficient environment*. Improving the object-detection algorithm (not available in the report) is one step on the road to enhancing the system performance. As of now, there is much redundancy in the algorithm to ensure that ”correct” object is being tracked. Since we are searching for relatively basic geometries and few objects at one sampling instance, we did not put much effort in streamlining the code or evaluate program cycles.

Another improvement to the localization estimation could have been to add the bias factors to the measurement model. This issue has been covered. Porting the setup to a lean and efficient environment could most certainly increase the FPS of camera, thus improving the accuracy when reaching calibration points along the optical path. Consider that the program is running on a Windows 7 (32-bit)/PC environment with system specifications of Intel® Core™ Duo CPU T6600 @ 2.20 GHz, GeForce GT 130M and 3 Gb usable RAM. Probably it would be better with a Linux kernel free from heavy background processes.

Bibliography

[1] T. Meyer, “AGVs: The Future Is Now,” *Material Handling Management*, p. 14, October 2009.

[2] D. E. Mulcahy, *Materials Handling Handbook*, 1st ed., D. E. Mulcahy, Ed. McGraw-Hill, 1999.

[3] A. Helleboogh, T. Holvoet, and Y. Berbers, *Testing AGVs in Dynamic Warehouse Environments Cover Image*, D. Weyns, H. V. D. Parunak, and F. Michel, Eds. Springer Berlin / Heidelbergl, 2006, vol. 3830.

[4] N. Izosaki, D. Chugo, S. Yokota, and K. Takase, “Camera-based AGV Navigation System for Indoor Environment with Occlusion Condition,” Beijing, China, August 2011.

[5] D. Ronzoni, R. Olmi, C. Secchi, and C. Fantuzzi, “AGV Global Localization Using Indistinguishable Artificial Landmarks,” Shanghai, China, May 2011.

[6] Anonymous, “Laser point the way for Indumat’s AGV vehicle,” *Motor Transport*, pp. 16–17, 2004.

[7] M. Bertozzi, A. Broggi, M. Cellario, A. Fascioli, P. Lombardi, and M. Porta, “Artificial vision in road vehicles,” *Proceedings of the IEEE*, no. 7, pp. 1258–1271, July 2002.

[8] P. D. Team, “Magnetic Guidance System (MGS),” http://www.path.berkeley.edu/PATH/research/magnets/, April 2008.

[9] M. Wahde, “Introduction to Autonomous Robots,” Department of Applied Mechanics, Chalmers University of Technology, Tech. Rep., 2012.

[10] J. P. Huissoon, “Curved ultrasonic array transducer for AGV applications,” *Ultrasonics*, no. 4, pp. 221–225, 1989.

[11] H. Martinez-Barbera and D. Herrero-Perez, "Development of a flexible AGV for flexible manufacturing systems," *Industrial Robot: An International Journal*, July 2010.

[12] C. Lozoya, P. Martí, M. Velasco, J. Fuertes, and E. Martín, "Simulation study of a remote wireless path tracking control with delay estimation for an autonomous guided vehicle manufacturing systems," *The International Journal of Advanced Manufacturing Technology*, vol. 52, pp. 751–761, June 2010.

[13] K. Jungmin, W. Seungbeom, K. Jaeyong, J. Do, K. Sungshin, and B. Sunil, "Inertial navigation system for an automatic guided vehicle with Mecanum wheels," *International Journal of Precision Engineering and Manufacturing*, vol. 13, no. 3, pp. 379–386, March 2008.

[14] A. Azenha and A. Carvalho, "Dynamic analysis of AGV control under dead-reckoning algorithm," *Robotica*, vol. 26, pp. 635–641, March 2008.

[15] C. Soria, E. Freire, and R. Carelli, "Stable AGV corridor navigation based on data and control signal fusion," *Latin American applied research*, no. 2, June 2006.

[16] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots.* Massachusetts Institute of Technologyl, 2004.

[17] M. Lundgren, "Path Tracking and Obstacle Avoidance for a Miniature Robot," Master's thesis, Umeå University, 2003.

[18] K. Yoshizawa, H. Hashimoto, M. Wada, and S. Mori, "Path Tracking Control of Mobile Robots Using a Quadratic Curve," Tokyo, Japan, September 1996.

[19] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," Department of Engineering Science, The University of Oxford, Tech. Rep., 1997.

[20] R. E. Kalman, "A new approach to linear filtering and and prediction problems." *Trans. ASME Ser. D. J. Basic. Eng.*, no. 82, pp. 24–45, 1960.

[21] J.-C. Yoo and Y.-S. Kim, "Alpha–beta-tracking index (α–β–$\bigwedge$) tracking filter," *Signal Processing*, vol. 83, no. 1, pp. 169–180, July 2003.

[22] C. M. Wang, *Location estimation and uncertainty analysis for mobile robots.* Springer-Verlag New York Inc., USA.

[23] J. Coombs and R. Prabhu, "OpenCV on TI's DSP+ARM platforms: Mitigating the challenges of porting OpenCV to embedded platforms," http://www.ti.com/, July 2011.

A Use-Case ver. 1.2

As mentioned in the introduction, a Use-Case describes interaction between user and system in a step-by-step manner. The purpose of a Use-Case in this context is that it brings a sequential template for software development. Although it is quite simple and lean, it provides a general concept of how the unit is supposed to act and if it fits the acting that the company had in mind. The Use-Case is updated continuously. More detailed information will be added during the project as more data is acquired on how to solve the task. This version was final at the time of completion of the project.

Docking

This Use-Case describes the docking sequence of the unit.

Pre-conditions

No error indications from the unit (light or sound signaling).
A site is predefined by tape or any equivalent marking to demonstrate where the unit is to be place to be able to initiate docking. The site also has an optic marker where the unit shall be able to perform parallel tracking.
The station ID at the docking site is linked to a global coordinate system which contains the calibration point coordinates, virtual path of the optical marker for parallel tracking and the docking point coordinates itself.

Normal flow

1. The user maneuvers the unit to the predefined marked site (docking site), facing a predetermined direction.
2. A vision system acquires station ID, angle and distance offset from a predefined optical binary pattern.
3. The unit indicates (display/light/sound signaling) that it is ready to initiate docking.
4. The user commands the unit by pushing button(s) to initiate docking.
5. The unit calculates a virtual path based on the acquired angle and distance offset between the current position and a predefined optical path.
6. The unit awaits user to push dead-mans grip button(s) on a hand held remote device to enable drive along the path towards the predefined optical path.
7. The unit drives along the virtual path by dead-reckoning using odometry measurements on each wheel.
8. The unit reaches the predefined optical path and initializes vision tracking. The vision measurements are fused with the odometry to enable elimination of the inherent systematic (cumulative) errors from the odometry alone.

9. The unit drives along the predefined optical path until it recognizes the calibration site (pattern recognition) that indicates an absolute position along the path.
10. The unit stops at the calibration site and acquires an offset angle and distance from the calibration marker and indicates docking mode complete (display/light/sound signaling).

Alternative flow

Start at 7. The user releases the button(s) before the unit has reached the calibration point.
- 8. The unit stops immediately and puts the docking sequence on hold.
- 9. The unit awaits new command from user to continue with the docking sequence.
Continue at 7.

Alternative flow

Start at 7. The unit has driven a maximum allowed distance along the optical without recognizing the calibration point.
- The unit changes direction to make another attempt on finding the calibration point.
Continue at 1.

Post-conditions

The unit is ready to begin alignment from the predefined calibration site along the path.

Special requirements

The button(s) to command the unit to perform the docking sequence must be held continuously to secure that the sequence is monitored and ensure that the sequence can be cancelled easily.

Parallel alignment against the work station

This Use-Case describes alignment with the work station.

Pre-conditions

No error indications from the unit (display/light/sound signaling).
The docking sequence is completed.
Position data from work station is available and offset between calibration site and work station is verified.

Normal flow

1. The user moves the work station operating tool to desired position.
2. The user commands the unit by pushing button(s) to initiate parallel alignment of the unit against the work station.

3. The unit drives towards the acquired position of the work station, corresponding to a parallel position on the path.
4. When the unit approaches the desired position, it decelerates and stops within a ±5 mm accuracy of this position.
5. The unit indicates (display/light/sound signaling) that it is within a ±5 mm accuracy of this position and is ready to perform work task.

Alternative flow

Start at 3. The user releases the button(s) before the alignment sequence is complete.
- 4. The unit stops immediately and puts the alignment sequence on hold.
- 5. The unit awaits new command from user to continue with the docking sequence.
Continue at 3.

Alternative flow

Start at 3. The unit loses tracking of its position or path.
- 4. The unit stops immediately and puts the alignment sequence on hold.
- 5. The unit indicates (display/light/sound signaling) that it has lost position or path.
- 6. The unit is set to manual driving mode and is required to be moved manually to the docking position.
Continue at 1 in Docking.

Alternative flow

Start at 2 or 3. The unit fails to acquire the position of the work station operating tool or acquires a non-valid position of the work station operating tool.
- 4. The unit stops immediately and puts the alignment sequence on hold.
- 5. The unit indicates (display/light/sound signaling) that it fails to acquire the position of the work station operating tool or acquires a non-valid position of the work station operating tool.
- 6. The unit alerts the user to contact a technician and/or enables manual drive mode.

Post-conditions

The unit is ready to *Perform work task* or *Release unit from docked mode.*

Special requirements

The button(s) to command the unit to perform the docking sequence must be held continuously to secure that the sequence is monitored and ensure that the sequence can be cancelled easily.

Perform work task

This Use-Case describes the sequence of the unit's work task execution.

Pre-conditions

No error indications from the unit (display/light/sound signaling).
Operating tool parallel alignment completed.

Normal flow

1. The user commands the unit to perform work task by pressing button(s).
2. The unit indicates (display/light/sound signaling) ”work task in progress” and starts to execute work task.
3. The unit indicates (display/light/sound signaling) ”work task completed”.

Alternative flow

Start at 2. The user releases the button(s) before the work task execution is complete.
- 3. The unit stops immediately and puts the work task execution on hold.
- 4. The unit awaits new command from user to continue with the work task execution.
Continue at 2.

Post-conditions

The unit is ready to *Perform work task* or *Release unit from docked mode* or initiate *Parallel alignment against the work station operating tool.*

Special requirements

The button(s) to command the unit to perform the docking sequence must be held continuously to secure that the sequence is monitored and ensure that the sequence can be cancelled easily.

Release unit from docked mode

This Use-Case describes the sequence of release the unit from docked mode.

Pre-conditions

No error indications from the unit (display/light/sound signaling).
In Docked mode.

Normal flow

1. The user commands the unit to *Release from docked mode* by pressing button(s).
2. The unit drives to and stops at to the docking position.
3. The unit indicates (display/light/sound signaling) that it is ready for manual driving mode.
4. The user commands manual driving mode.
5. The unit is no longer in docked mode and can be driven manually.

Alternative flow

Start at 2. The user releases the button(s) before the unit has reached docking position.
- 3. The unit stops immediately and puts the release sequence on hold.
- 4. The unit awaits new command from user to continue with the release sequence.
Continue at 2.

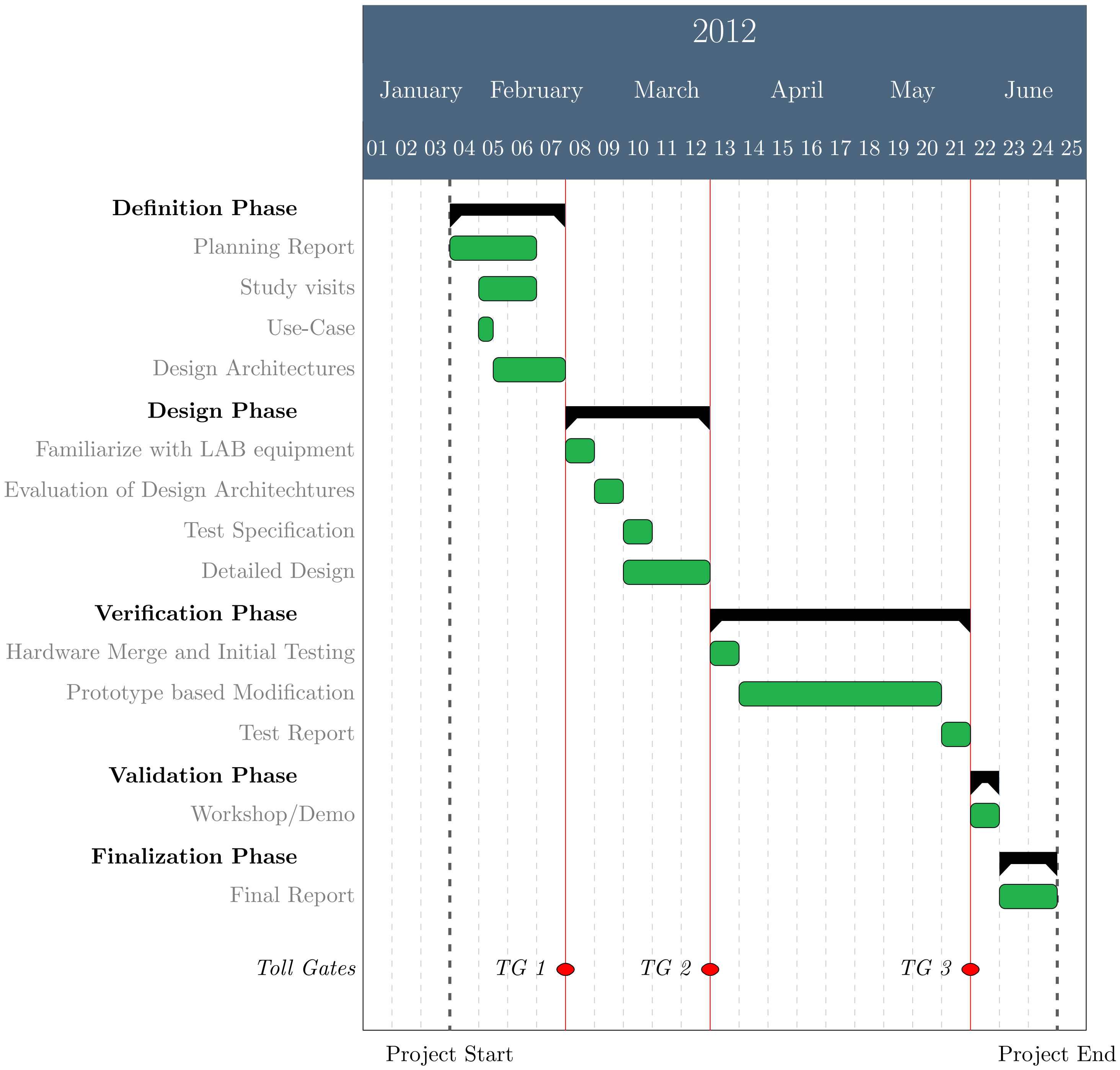
Post-conditions

The unit is ready for manual driving.

Special requirements

The button(s) to command the unit to perform the docking sequence must be held continuously to secure that the sequence is monitored and ensure that the sequence can be cancelled easily.

B Time schedule



Time schedule evaluation

The **Definition Phase** planning were matched almost to the day with the expected progress. The Use-Case though was continuously updated during the project. The **Design Phase** was completed earlier due to a fast decision to drop the initial thought to proceed with a magnetic solution for AGV guidance. The two first phases are also the most uncertain ones and at the planning stage, we felt we kept a good pace.

What delayed the progress was the order time of the hardware and installation with mounting of encoders that allowed us to start testing our prepared software. Once all the hardware were set, much correction in the software was needed to make the AGV behave as we wanted. Tuning the mechanical dimensions to the kinematic model and selecting proper values in the covariances matrices of the Extended Kalman filter were very time consuming.

Once satisfied with the solution, we prepared a demo to an Open House session as a part of the **Validation Phase**. The demo consisted of a nascar-track, much like the ones in the calibration stage of the odometry but with the optical path along one long side to demonstrate the vision-based system.

C Docking phase data

$\hat{x}$	$\hat{y}$	Dist.	$\hat{\theta}$
0.001	0.004	0.0041	0.029
-0.003	0.000	0.0030	-0.146
0.002	-0.005	0.0054	0.100
0.001	-0.012	0.0120	0.079
0.000	0.007	0.0070	0.023
0.003	-0.011	0.0114	0.191
0.000	-0.004	0.0040	0.017
-0.001	-0.003	0.0032	-0.042
0.001	-0.010	0.0100	0.077
0.001	-0.006	0.0061	0.048
0.002	-0.008	0.0082	0.109
0.001	-0.009	0.0091	0.065
-0.002	0.001	0.0022	-0.116
-0.001	-0.003	0.0032	-0.037
0.000	-0.006	0.0085	0.020
-0.002	-0.003	0.0036	-0.115
-0.001	-0.002	0.0022	-0.035
-0.001	-0.004	0.0041	-0.067
0.000	-0.004	0.0040	-0.014
0.001	0.004	0.0041	0.093
0.001	0.003	0.0032	-0.040
0.000	0.005	0.0050	0.035
0.000	-0.001	0.0010	-0.009
0.000	0.003	0.0030	-0.143
-0.001	-0.003	0.0032	-0.063
-0.004	0.000	0.0040	0.200
0.001	0.004	0.0041	0.065
-0.003	0.000	0.0030	-0.154
0.003	-0.009	0.0095	0.171
0.001	0.002	0.0022	-0.049