An Application-oriented Open Software Platform for Multi-purpose Field Robotics.

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Kjeld Jensen

Faculty of Engineering, University of Southern Denmark
Odense, Denmark 2014

An Application-oriented Open Software Platform for Multi-purpose Field Robotics.

A thesis submitted to the Faculty of Engineering at the University of Southern Denmark in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Odense, Denmark, May 2014

Topic
Applied Robotics

Supervisor
Senior Researcher Rasmus Nyholm Jørgensen

University of Southern Denmark
Faculty of Engineering
Campusvej 55
DK-5230 Odense M
Denmark

Abstract

Field robotics has become an increasingly active research area in the past 50 years, and there is great potential in using autonomous field robots to perform precision tasks in biological production and related applications. But the products available on the market today are limited to small robots that solve simple tasks such as mowing, and automatic tractor steering that navigates through a planned route under the supervision of an operator.

The outdoor environment in which the robot operates is often very complex. This places great demands on the robot's ability to perceive the environment and based on this behave in a way that is appropriate and productive with respect to the given task while being safe for nearby people, animals and objects. Researchers are challenged by the considerable resources required to develop robot software capable of supporting experiments in such a complex perception and behavior. The lack of collaboration between research groups contributes to the problem, the scientific publications describe methods and results from the work, but little software for field robots are released and documented for use by others.

The hypothesis of this work is that an application oriented open software platform for multi-purpose field robotics will reduce the resources required for experimental research considerably due to reuse of existing work across projects and robotic platforms. This thesis describes the FroboMind field robot software platform developed in this work and presents FroboMind use cases in precision agriculture and humanitarian demining.

Use of FroboMind in various projects have shown that it does save resources using a common software platform across projects and robotic platforms and that it thus facilitates the task of carrying out experiments in the field. FroboMind is used today by other universities, and some companies use FroboMind as a basis to develop new field robotic products.

Resumé

Markrobotter er blevet et stadig mere aktivt forskningsområde inden for de sidste 50 år, og der er et stort potentiale i at bruge autonome markrobotter til at udføre præcisions-opgaver inden for biologisk produktion og relaterede applikationer. Men de tilgængelige produkter på markedet i dag er endnu begrænset til små robotter, der løser simple opgaver som eks. græsklipning samt autostyringer til traktorer, der kan navigere en planlagt rute under overvågning af en operatør.

Det udendørs miljø, som robotten opererer i, er ofte meget komplekst. Det stiller store krav til robottens evne til at opbygge viden om omgivelserne, og herudfra udvise en adfærd, som er hensigtsmæssig og produktiv i forhold til den givne opgave samt ufarlig for mennesker, dyr og genstande i nærheden. Forskere er udfordret af de mange ressourcer, det kræver at udvikle markrobot software, der kan understøtte eksperimenter inden for kompleks perception og adfærd. Og manglen på samarbejde mellem forskergrupperne bidrager til problemet. De videnskabelige publikationer beskriver metoder og resultater fra arbejdet, men kun lidt software til markrobotter er frigivet og dokumenteret til brug for andre.

Hypotesen for dette projekt er, at en applikations-orienteret åben software platform for markrobotter til forskellige formål vil nedsætte ressourceforbruget betydeligt pga. genbrug af eksisterende arbejde på tværs af projekter og robot platforme. I denne afhandling beskrives FroboMind markrobot software platformen udviklet i projektet, og der gives eksempler på FroboMind applikationer inden for præcisionslandbrug samt inden for humanitær minerydning.

Brugen af FroboMind i forskellige projekter har vist, at det sparer ressourcer at benytte en fælles software platform på tværs af projekter og robot platforme, og at det dermed letter arbejdet med at udføre forsøg i marken. FroboMind bruges i dag af andre universiteter, og flere virksomheder benytter FroboMind som udgangspunkt for etablering af nye markrobot produkter.

Preface

This thesis describes the majority of the work I have carried out as part of my PhD project in *Applied Robotics*.

The work deals with the application of field robotics to solve precision tasks in outdoor environments. Using both my academic and industrial educational background I have aimed towards creating innovative solutions through integration of novel and known technology and to optimize the process of moving from idea to experimental results.

In addition to following the PhD curriculum it has been a challenging and rewarding process to participate in the development of research in Biosystems Engineering at the University of Southern Denmark. I have enjoyed more freedom and have had more responsibility than most, and even though this has taken a considerable amount of time in addition to the regular curricular activities I would not want to be without.

The work has included a large number of field tests and experiments. I have spent many long days walking along a robot in the field trying to look at the laptop screen bathed in sunshine or protecting it from a sudden harsh rain, working late evenings in a machinery house or in the field repairing the robot or fixing a piece of software that was not properly tested in the lab. Nevertheless being in the field is the best part of this work and I would like to thank those of my colleagues and students who have struggled together with me to achieve our common results. It has been fun!

During the project period I have been so lucky to become the father of Asbjørn, Bertil and Sigrid, three wonderful kids. While playing with them I have often thought of how easily they learn to solve simple tasks which it would take many years to make a robot do. I believe that the work presented in this thesis is a small step towards achieving better and faster results in robotics research. It has been hard work, and yet the kids learn everything just by playing...

Being the father of three small children has also proven to be a very time consuming task, and I owe my family and friends a huge thank you for your support and helping out with the kids and thus giving me time to focus on the project. Thank you Ida for your love and support and great patience.

Kjeld

Acknowledgements

Field robotics is a highly interdisciplinary field and many areas of expertise are required to succeed in performing experimental work. The work presented here is a product of a close cooperation with a number of colleagues and external partners. I highly appreciate the support, contributions and sparring that I have received. To put it short, this work would not have been possible without. I particularly would like to thank:

- My supervisor Rasmus Jørgensen, Aarhus University for supervising my work. I highly value the support and constructive feedback I have received.
- My colleagues in the Biosystems Engineering group: Henrik Skov Midtiby, Morten Stigaard Laursen, Thomas Giselsson, Jørgen Maagaard Petersen and Keld Kjærhus Bertelsen for mutual collaboration and sparring.
- The students, student programmers and research assistants: Søren Hundevadt Nielsen, Morten Larsen, Leon Bonde Larsen, Kent Stark Olsen, Rudi Hansen and Martin Skriver for their significant contributions and sparring regarding the FroboMind software platform and its application to and implementation in several research projects.
- My colleagues at the robotics laboratory RoboLab and the Faculty of Engineering management and administration for supporting my work.
- Ole Green and Kongskilde Industries for collaboration regarding *Robotti*, the first commercial robot based on the FroboMind software platform.
- Tom Simonsen and Conpleks Innovation for collaboration on using FroboMind in industrial applications and on developing a low-cost single frequency RTK GPS.
- Larsens Landmåler Service for collaboration regarding the surveying robot.
- Aarhus University and Thorsen Teknik for collaboration regarding the *ASuBot*, the first garden tractor robot running a prototype of FroboMind.
- Rune Bech Persson and the organization DanChurchAid for providing advice on humanitarian demining and lending a large loop metal detector for this work.
- Jørgen Maagaard Petersen for mechanical designs for the robot platforms. Mikkel Kildemand Larsen for print design and production of the FroboMind Controller, Carsten Albertsen for print design and production of the WADS sensor interface and low-cost GPS module. The people at the faculty mechanical workshop for producing numerous different constructions often in very limited time.

- My colleagues at the Department of Engineering and Danish Centre For Food And Agriculture, Aarhus University, the Automation and Control group, Technical University of Denmark and the Instrumentation and Test Engineering, University of Hohenheim.
- Stig Hansen and the agricultural department, KOLD College for collaboration on preparing the experimental test fields.
- The companies and organizations Lynex, DataGrid, More Electronics, Norad A/S, Ebinger GmbH, it-vest and The Danish Coastal Authority for contributing with information, knowledge, support and equipment for this work.

Readers guide

This thesis is mainly based on three scientific papers produced within this work. The first paper proposes an open software platform for multi-purpose field robotics, the second deals with low-cost global pose estimation, one of the significant challenges with regards to commercialization of field robots. Paper three deal with using the software platform in a precision agriculture application. The thesis contains the following chapters:

- Chapter 1 gives an introduction to the project background and purpose. The project aim and hypothesis are formulated, relations to related work are explained. A summary of the papers is given in section 1.1 and the main contributions of this work is listed in section 1.2.
- Chapter 2 is the core part of the thesis. It deals with the FroboMind software platform developed in this work and is based on the paper "Towards an open software platform for field robots in precision agriculture".
- Chapter 3 deals with the FroboMind perception layer with regards to global pose estimation. A substantial part of the chapter is devoted to low-cost positioning which is very important to the commercial potential of field robotics. The chapter is partly based on the paper "Evaluating the performance of a low-cost GPS in precision agriculture applications".
- Chapter 4 deals with the FroboMind action layer with regards to global navigation based on waypoints.
- Chapter 5 presents a use case in the domain of precision agriculture tasks. It is based on the paper "Autonomous precision spraying trials using a novel cell spray implement mounted on an Armadillo robot".
- Chapter 6 presents another use case within humanitarian demining applications.
- Chapter 7 presents some of the field robot prototypes and other projects that are related to this work.
- Chapter 8 concludes on the work and describe future perspectives.

Abbreviations

This section lists abbreviations used in the dissertation alphabetically. Where applicable units are indicated with [unit] after the abbreviation.

2D	Two dimensional
3D	Three dimensional
4wd	Four wheel drive
AGM	VRLA battery type: Absorbed Glass Mat
Ah	Electric charge: Ampere-Hour $(3600s) * (A) = 3600C$
AMS	Robot owned by UH: Autonomous Mechanisation System
ATV	All Terrain Vehicle
AU	Aarhus University
Back-EMF	Counter-electromotive force
CAN	Controller Area Network
CLARAty	Coupled Layer Architecture for Robotic Autonomy
CMU	Carnegie Mellon University
dB	decibel, a logarithmic unit expressing the ratio between two values of a physical quantity
DC	Direct Current
DCA	Humanitarian organization: DanChurchAid
DoF	Degrees of Freedom (independent parameters)
EKF	Extended Kalman Filter
E	East
EMI	Electromagnetic Interference
ENU	East, North, Up Cartesian coordinate system typically used for land vehicles
ERW	Explosive remnants of war
F	The focal length of an optical system
GC	Geometric Center
GLONASS	GLObalnaya NAvigatsionnaya Sputnikovaya Sistema
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPX	GPs eXchange format
GUI	Graphical User Interface
ha	Hectare = 10,000 m^2
HDOP	Horizontal Dilution Of Precision
HMI	Human Machine Interface
ICR	Instantaneous Center of Rotation
IED	Improvised Explosive Devices
IMAS	International Mine Action Standards
IMU	Inertial Measurement Unit
IPC	Inter Process Communication

L1	GPS carrier frequency: 1575.42 MHz
L2	GPS carrier frequency: 1227.60 MHz
LCRS	Low-cost RTK system
Lidar	Light Detection And Ranging / Laser Imaging, Detection and Ranging
MDI	Mine Detection Implement
mm	Length: Millimeter: $1 * 10^{-3}$ m
MRDS	Microsoft Robotics Developer Studio
NE	North-East
NED	North, East, Down Cartesian coordinate system
NID	Normal and Independent Distributed
NMEA (0183)	A specification for device communication authored by the National Marine Electronics Association
NW	North-West
μg/l	Concentration: 10^{-6} grams per liter
μm	Length: Micrometer: 10^{-6} m
OROCOS	Open Robot Control Software
P	Proportional (controller)
PCB	Printed Circuit Board
PD	Proportional-Derivative (controller)
percentile	The value below which a given percentage of observations in a group of observations fall
PI	Proportional-Integral (controller)
PID	Proportional-Integral-Derivative (controller)
pose	State information consisting of of position and orientation of an object
PPP	Precise Point Positioning
ROS	Robot Operating System
RS-422	EIA RS-422 standard
RS-485	ANSI/TIA/EIA-485-A-1998 standard
RTK	Real Time Kinematics
RTOS	Real Time Operating System
SDU	Syddansk Universitet / University of Southern Denmark
SLAM	Simultaneous Localization And Mapping
SLOC	Source Lines Of Code (physical)
SMD	Surface Mounted Devices
SNR	Signal-to-noise ratio: $SNR = \frac{P_{signal}}{P_{noise}}$, P is average power
TM	Transverse Mercator
TTL	Transistor-Transistor Logic
UAS	Unmanned Aerial Systems
UH	University of Hohenheim
UHF	Ultra High Frequency
USB	Universal Serial Bus

UTC	Universal Time Coordinated
UTM	Universal Transverse Mercator
UTM32	Universal Transverse Mercator zone 32
UXO	Unexploded ordnance
V	Electrical Potential [Volt]
VHF	Very High Frequency
VRLA	Valve Regulated Lead Acid
WGS-84	World Geographical System datum defined 1984
Wh	A unit of energy equivalent to one watt (1 W) of power expended for one hour (1 h) of time: 3.600 J
yaw	In the *Robot* coordinate system yaw is the vertical axis about which the robot turns.

Contents

Chapter 1

Introduction

This chapter gives an introduction to the background and purpose of this work. The hypothesis and aim of the work are formulated, and a summary of the included scientific papers is presented.

In 1948 W. G. Walter demonstrated the first simple robots exhibiting biological inspired autonomous behaviours known as phototaxis. The two wheeled turtle like mobile platforms named Elsie and Elmer used a light sensor, a touch sensor, motors for steering and propulsion, and a vacuum tube analog computer to follow a light source while avoiding obstacles (Walter, 1950; Holland, 1997). Ten years later a paper titled "A step towards an automatic tractor" was published describing a prototype of a tractor capable of navigating around a farmstead. The navigation was based on a wire laid in or on the ground and energized with an alternating current. The wire induces currents in two search coils mounted at the front of the tractor and based on this the hydraulic valves of the tractor steering are operated to keep the tractor centered above the wire. The paper argues the value of automatic tractors where labour is "unskilled and destructive" or "scarce and expensive" as well as for hazardous tasks (Morgan, 1958).

Today, more than 60 years later, robotics has become one of the most significant technologies and is now used in many different areas such as industrial production, domestic maintenance, health care, entertainment, military tasks and space exploration. Robots are typically allocated to tasks that are considered dirty, dangerous or dull or have high requirements for speed, accuracy and reliability. In mobile robotics the use of simple robot products for vacuum cleaning and mowing are on the rise (Sahin and Guvenc, 2007), and driverless cars (Thrun et al., 2006) and unmanned aerial systems (UAS) (Cai et al., 2010) are emerging technologies as well. Field robotics is an active research area especially within biological production applications such as agriculture and horticulture

because of the economic potential (Slaughter et al., 2008; Pedersen et al., 2008). The arguments presented in (Morgan, 1958) are more applicable than ever. In Denmark like in many other high income industrial countries farmers have great difficulties obtaining manual labor to perform the often monotonous, repetitive and wearing tasks, and the labour costs are too high compared to the possible earnings on the products (Baraldi et al., 2006).

But despite the active research in field robotics for the past decades, the current achievements are somewhat stagnated. The navigation wire has been replaced by high precision Global Navigation Satellite Systems (GNSS) for waypoint navigation (Norremark et al., 2008) and computer vision and Light Detection And Ranging (Lidar) based systems for navigation along crop rows, orchards and swaths etc. (Blas, 2010; Barawid et al., 2007; Åstrand and Baerveldt, 2005) But the functionality of guidance systems currently available on the market is essentially still about following a predefined path in the field while the implement performs the given task, and it still requires an operator to observe the tractor and react when something out of the ordinary happens (Ming Li, 2009).

An important reason is the unstructured, dynamic and weather affected environments that field robots operate in. Figure 1.1 shows examples of semi-structured environments like crop fields and orchards and unstructured open fields. The level of complexity in these environments are very high compared to an industrial robot working in a well defined and controllable production environment or a household robot for vacuum cleaning the living rooms (Kelly and Thorpe, 2007), and field robotics is considered one of the most difficult areas in robotics (Thorpe and Durrant-Whyte, 2001). Safety is also a major concern. Tractors and also the smaller field robots may constitute a serious threat to humans, animals and real estate if errors occur or if static and dynamic objects near the robot are not detected and identified correctly. The main problem is not to detect the objects but to do this in an outdoor environment with the high level of certainty required for a safety system (Bouraine et al., 2012; Griepentrog et al., 2009).

Moving from the current row guidance systems to fully autonomous robots that are capable of performing more complex tasks in the field without human intervention requires a significant higher complexity of the robot. It must support enhanced perception and sophisticated cognitive behaviours to enable a more reliable autonomy in the given environment (Griepentrog et al., 2010). Research groups performing experimental work in field robotics are faced with the challenge of developing and maintaining field robot platforms and software supporting these capabilities, a task that is highly interdisciplinary and requires significant resources. Mobile robots tuned to the indoor laboratory environment are readily available thereby facilitating research (Siegwart et al., 2011), but this is not the case for the field environment. The consequence is that many researchers in precision agriculture and related areas are limited to test-

(a) Row crops (b) Orchard

(c) Grass slopes (d) Mine field

Figure 1.1: *Examples of semi-structured and unstructured environments.*

ing in simulated or laboratory sized environments. Only research groups with substantial resources invest in developing own robots from scratch or based on retrofitting existing field vehicles (Berge et al., 2012; Bakker et al., 2010; Ruckelshausen et al., 2009; Zeitzew, 2007; Jørgensen et al., 2007; H.W. Griepentrog, 2007; Blackmore et al., 2004). This task is made more difficult by the lack of cooperation at the implementation level among research groups. Papers are published on findings and results, but little hardware and software in the domain of experimental field robots have actually been released, published and documented to support use by others (Suprem et al., 2013).

The Biosystems Engineering group at the University of Southern Denmark has set out to develop low-cost field robot platforms (Jensen et al., 2012c). The primary purpose is to support own research within agricultural and horticultural crop production, green area maintenance etc. But to the extent possible the findings and results related to the robot platforms including drawings, schematics etc. are open sourced and shared with the research community and the industry. All activities within this work have been focused on this task with a strong emphasis on innovation and the integration of systems in collaboration with industrial partners.

This thesis addresses the main part of this work focusing on development of robot software solutions that support navigation of field robots performing precision tasks in outdoor environments. The hypothesis of this work is that an application oriented open software platform for multi-purpose field robotics will reduce the resources required for experimental research considerably due to

reuse of existing work across projects and robotic platforms. The aim of this thesis is to establish such a software platform which provides the basis for enhanced perception and sophisticated behaviour based autonomy, and to evaluate the performance of this platform when applied to different precision tasks and robot platforms.

1.1 Summary of papers

The thesis is mainly based on three of the scientific papers produced within this work. Below is a brief summary of the papers.

The first paper *"Towards an open software platform for field robots in precision agriculture"* presents a review of existing work within software solutions for architectures, frameworks, development environments, libraries etc. and proposes the FroboMind software platform for field robots based on Linux and Robot Operating System (ROS). The software platform consists of an operating system, a middleware, an architecture and software components. Only the architecture and components are part of this work. Experiments with the purpose of trying to quantify the portability of FroboMind to new robot platforms and applications are described and the results are presented.

The second paper *"Evaluating the performance of a low-cost GPS in precision agriculture applications"* deals with low-cost global pose estimation, one of the significant challenges with regards to commercialization of field robots. Global Navigation Satellite System (GNSS) position updates based on Precise Point Positioning and Real Time Kinematics (RTK) are evaluated in static tests and field trials, and global pose estimation based on Extended Kalman Filtering is introduced.

In the third paper *"Autonomous precision spraying trials using a novel cell spray implement mounted on an Armadillo robot"* the FroboMind software platform is applicated to experimental work in precision agriculture. The paper presents results from a large scale precision spraying trial where a tracked robot navigates autonomously along rows of maize carrying a cell sprayer implement. Navigation is based on RTK-GNSS. The implement senses the weed density using computer vision and cell spray nozzles are controlled by experimental precision spraying algorithms. The paper focuses on the cell sprayer design including camera setup, sprayer system performance, integration with the Armadillo robot and the FroboMind software platform.

1.2 Contributions

The main contributions of our work are as follows:

- Design of FroboMind, an application-oriented open software platform for multi-purpose field robotics.
- Implementation of major parts of the software platform.
- Development and in-field testing of open-source FroboMind components for field robot global pose estimation and navigation, a prerequisite to many precision agriculture experiments.
- Development of several new small and medium scale prototypes of field robots for use in experimental research.
- Application of FroboMind to novel large scale precision spraying trials seeking to lower the use of herbicides in crop production.
- Application of FroboMind and field robot development to humanitarian demining based on knowledge and experience from precision agriculture.
- Dissemination of knowledge and experience to fellow researchers and industrial partners with regards to FroboMind, development of field robots and experimental research.
- Contributions to the development of industrial field robotic products through participation in innovation projects.

These contributions have been disseminated to the scientific community, students, and the general public: Publication of findings, software and documentation via scientific publications and the FroboMind website; Knowledge about field robotics to engineering students via courses, project supervision and extracurricular activities; General research perspectives to the public via talks, exhibitions, high school workshops, events etc.

Based on these contributions the scientific community has received new tools to facilitate the task of carrying out experimental research in application-oriented field robotics. Industrial companies interested in constructing autonomous field robotic products have received a working prototype which they may use as a basis for their development without concerns about the future cost of licenses etc. Hopefully this will enhance research through extensive field experiments and facilitate the development of new innovative products.

Chapter 2

Towards an open software platform for field robots

This chapter deals with FroboMind, an application-oriented open software platform for multi-purpose field robotics. FroboMind is described and the results of experiments trying to quantify the portability of FroboMind to new robot platforms and applications are presented.

2.1 Introduction

Robotics in precision agriculture has the potential to improve competitiveness and increase sustainability compared to current crop production methods (Pedersen et al., 2006), and has become an increasingly active area of research during the past decades. Tractor guidance using Global Navigation Satellite System (GNSS) based sensor systems for route following and local sensor systems for accurate in-row navigation in row crops and orchards have already reached the market. One example is the John Deere iTEC Pro which supports GNSS based steering in straight and curved rows and at headlands while controlling speed and performing active implement guidance. Another example is the Claas Cam Pilot system which navigates a tractor through row crops using 3D computer vision to detect the location of the crop rows. Early prototypes of smaller field robots performing precision agriculture tasks without human intervention also exist (Green et al., 2014; Bakker et al., 2010; Ruckelshausen et al., 2009; Jørgensen et al., 2007; Blackmore et al., 2004). But research in advanced cognitive (Neisser, 1967) perception and behaviour that are required to enable a more efficient, reliable and safe autonomy becomes increasingly demanding. The level of complexity in an unstructured, dynamic, open-ended and weather influenced environment like a crop field or an orchard is high, and the size and complexity

of the software needed to perform experiments impose ever greater demands on research groups. A lack of collaboration between the research groups contributes to the problem. Scientific publications are published on findings and results in precision agriculture, but little field robot software has actually been released, published and documented for others to use. We hypothesize that a common open software platform tailored to field robots in precision agriculture will significantly decrease the development time and resources required to perform field experiments due to efficient reuse of existing work across projects and robot platforms. The aim of this work is to establish such a software platform and evaluate the performance when applied to different precision agriculture tasks and field robots.

2.1.1 Related work

The literature contains numerous references to relevant proposals and implemented solutions within robot software architectures, frameworks, middlewares, development environments, libraries etc. Below is a brief review of some of the best known and widely used solutions. Table 2.1 compares the middleware specifications.

CARMEN Robot Navigation Toolkit from the Carnegie Mellon University (CMU) (Montemerlo et al., 2003) is a modular software library for mobile robot control. It provides interfaces to a number of robot platforms and sensors, a 2d simulator and algorithms for localization, mapping, path planning etc. The architecture features 3 layers, the lowest layer contains hardware interfaces and collision detection, the middle layer localization and navigation, and the highest layer contains all high level tasks. Inter Process Communication (IPC), another project by CMU, is used for sending data between processes based on TCP/IP sockets.

CLARAty (Coupled Layer Architecture for Robotic Autonomy) (Nesnas et al., 2003)(Pivtoraiko et al., 2008) is a framework for generic and reusable software for heterogeneous robot platforms developed by the Jet Propulsion Laboratory. CLARAty is a two-tiered coupled layer (decision and functional) architecture. The functional layer is a modular software library which provides an interface to the robot system and contains algorithms for low- and mid-level autonomy. The decision layer builds on top of this adding high-level autonomy to achieve mission goals. CLARAty consists of a public and a private repository.

MRDS (Microsoft Robotics Developer Studio) (Cepeda et al., 2010) is a development environment for robot control and simulation. MRDS has support for a number of programming languages including it's own platform specific language Visual Programming Language (VPL). It supports a wide range of robotics hardware platforms and integrates a fully featured simulation environ-

ment. The component interface is based on the .NET Concurrency and Coordination Runtime (CCR) library for managing asynchronous, parallel tasks using message-passing and Decentralized Software Services (DSS), a lightweight .NET-based runtime environment.

Orca (Makarenko et al., 2006) is an open-source software framework for developing component-based robotic systems. The aim of Orca is to promote software reuse through definition of interfaces, providing component libraries and maintaining a public component repository. The intended use is for both commercial applications and research environments. The Orca project is a branch out from Orocos, the main difference is that Orca uses the Internet Communications Engine (ICE) (Henning, 2004).

Orocos (Open Robot Control Software) (Bruyninckx, 2001) contains portable C++ libraries for advanced machine and robot control. Orocos builds upon the open source Common Object Request Broker Architecture (CORBA) middleware. Orocos is in active development and contains extensions that support other frameworks including ROS.

Player (Gerkey et al., 2001) is one of the most used robot control interfaces and supports a wide variety of robots and components. The Player architecture is based on a *network server* which runs on the robot platform and provides a TCP socket interface to the robot. The *client program* is thus able to read data from sensors, write commands to actuators etc. by connecting to the socket. Player coexists with Stage which is a 2d multiple robot simulator and Gazebo which is 3d multiple robot simulator based on a physics engine.

ROS (Robot Operating System) (Quigley et al., 2009) is a flexible framework for writing robot software. It provides services, libraries and tools for building robotics applications. Examples are hardware abstraction and device control, interprocess communication, multi-computer environment support etc. ROS is in active development and is maintained by Open Source Robotics Foundation who provides access to a large repository of available components and maintains a list of external repositories provided by the growing community. The ROS core code and most ROS packages are released under the BSD 3-Clause License which facilitates commercial use of the code.

In terms of robot software architectures the above mentioned *Carmen* and *CLARAty* each use their own architecture, wheres a *MRDS*, *ORCA*, *Orocos* and *ROS* do not specify or endorse a certain architecture. *Player* does not specify an architecture beyond the two-layer *Network server* and *Client program*. Several relevant but lesser known robot software architectures are described in the literature:

(Vázquez-martín et al., 2008) argues that the sense-model-plan-act paradigm used in most architectures is unable to react in a dynamical environment because it depends heavily on the model, and that a behaviour-based approach such as

Name	*Updated*	*Main languages*	*Primary platforms*	*Component interface*	*License*
Carmen	2008	C	Linux	IPC	GPLv2 /BSD
CLARAty	2007	C++	VxWorks Linux Solaris	Unknown	Proprietary /Closed
MRDS	2012	C# VPL	Windows	CCR DSS	Commercial /Academic
ORCA	2009	C++	Linux	ICE	LGPL GPL
Orocos	2014	C++	Linux Windows OSX	CORBA	LGPL
Player	2010	C++ TCL Java Python	Linux Solaris BSD OSX	TCP sockets	GPLv3
ROS	2014	C++ Python Lisp	Linux	XMLRPC	BSD 3-Clause

Table 2.1: *Comparison of middleware specification. The year listed under Update indicates the latest official release or substantial update.*

the subsumption architecture (Brooks, 1986) is limited by the purely reactive behaviours and does not perform well when carrying out complex tasks. A hybrid is thus presented containing three layers: Hardware layer, Reactive layer and Deliberative layer. The reactive layer is grouped into a perception module (localization and mapping) and an action module (navigation and actuation). The deliberative layer contains high level mapping and path planning.

(Patricio Nebot and Martinez, 2011) introduces Agricultural Architecture (**Agriture**), a control architecture designed for teams of agricultural robots. Agriture consists of three layers, the physical layer which is either the robot or the Stage/Gazebo simulators, an Architecture middleware based on *Player*, *Java Agent Development Framework* and *High Level Architecture*, and a Distributed application layer for the application software.

(Garcia-Perez et al., 2008) proposed **Agroamara**, a hybrid agent of behaviour architecture for autonomous farming vehicles. Here *agent of behaviour* describes computational and control processes addressed to reach or to maintain a goal, with perceptual, deliberative and reactive abilities. The architecture groups the agents into perceptual and motor agents and uses layers to describe the information flow and hierarchy of activation of the agents. Data sharing is handled through shared memory and peer to peer message parsing. Multiprocessing is supported using winsocket.

(Kuhnert, 2008) describes the software architecture of the Autonomous Mobile Outdoor Robot (**AMOR**) which won the first price in autonomous driving in the European Land Robot Trial (ELROB) 2007 for urban and non-urban terrain. The robot software is not publicly available, however the paper does provide a good introduction to the design as well as the navigation modules. Intercommunication is handled by a *Virtual Sensor Actor Layer* that use TCP sockets and unifies the access to all sensors and actors. The main modules of the high level software is *global model* (map database), *global path planning* (deciding plans for the indended behaviour), *local model* (based on laser scanner and camera data), *local path planning* and *basic reflexes*.

(Beck et al., 2010) introduces **Mobotware**, a plug-in based framwork for mobile robots. Mobotware is divided into a hard- and a soft realtime section. Each section is decomposed into layers of increasing abstraction: Hardware abstraction, reactive execution, deliberative perception and planning. The framework has three core modules, a *Robot Hardware Daemon* providing hardware abstraction, a *Mobile Robot Controller* handling real-time closed loop controlling, and *Automation Robot Servers* handling sensor processing, mission planning and management.

(Simon Blackmore and Have, 2002)(Fountas et al., 2007) propose a system architecture to enable behavioural control of an autonomous tractor based on the subsumption architecture (Brooks, 1986). This work is related to the Software Architecture for Agricultural Robots (**SAFAR**) project with the purpose to develop a set of designs, tools and resources to promote the development of agricultural robots. SAFAR is based upon MRDS and is closed source but supports a Python scripting engine.

In 2005 the Stanford robot **Stanley** won the DARPA Grand Challenge by driving autonomously for 131 miles along an unrehearsed desert trail (Thrun et al., 2006). The robot software is not publicly available, however the paper provides information on the software architecture and design considerations. Stanley contains approximately 30 software modules organized in 6 layers representing sensor interfaces, perception, planning and control, vehicle interface, user interface and global services. All modules are executed individually without interprocess synchronization and all data are globally time stamped. Interprocess communication is handled using publish-subscribe mechanisms based on the CMU IPC kit.

Table 2.2 compares some properties of the reviewed architectures with respect to this work. The data is based on the reviewed publications and information publicly available on the web. It should be noted that all the reviewed publications contains valuable and relevant knowledge, but at the same time the table underpins the need for collaboration between the research groups in precision agriculture with respect to software reuse as discussed in the problem

	Agricultural applications	**Multiple platforms**	**Multiple users**	**Open source**	**Updated recently**
CARMEN	Yes	Yes	Yes	(Yes)	No
CLARAty	No	Yes	Yes	(Yes)	No
Agriture	No	No	No	No	No
Agroamara	Yes	No	No	No	No
AMOR	No	No	No	No	No
Mobotware	Yes	Yes	Yes	No	Yes
SAFAR	Yes	Yes	No	No	No
Stanley	No	No	No	No	No

Table 2.2: *Comparison of robot software architectures with respect to the problem domain and hypothesis of this work. In the columns Agricultural applications, Multiple platforms and Multiple users Yes means that practical field trials have taken place. (Yes) in Open source means that not all software has been released and/or the license is not permissive free.*

description.

2.2 The FroboMind software platform

2.2.1 Design goals

The software platform presented in this work has been named FroboMind which is a contraction of *Field Robot Mind*. It is based on design goals that promote reuse, support projects of varying size and complexity, and facilitate collaboration:

- **Modularity:** It must provide a proper modular structure to ease the development of new components and facilitate reuse. Interfaces between modules must be well defined, however still allowing researchers and developers to experiment freely.
- **Extensibility:** The design must support a wide variety of applications from a student working with signal processing or route planning algorithms to large scale field experiments conducted by research groups with several robots interacting online with each other and human operators.
- **Scalability:** It must support online computation of high load algorithms such as computer vision and mapping with respect to both memory and processing power requirements. This includes scaling to distributed applications on networked heterogeneous platforms.
- **Software reuse:** There must be a strong focus on minimizing resource requirements in the development of new applications by facilitating reuse

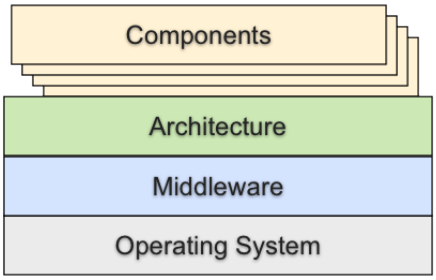


Figure 2.1: *Overview of the FroboMind software platform structure.*

of software. The future workload of maintaining the software platform is also an important consideration. When establishing the software platform, existing work should be reused to the extent possible. To prevent maintenance problems arising due to the existing work, this must be under active development and supported by a community.

- **Open source:** All core components of the software platform must be released under a permissive free software license to keep it free for others to use, change and commercialize upon. This is necessary to facilitate collaboration with industrial partners.

2.2.2 FroboMind

The FroboMind software platform structure consists of four parts as illustrated in figure 2.1. The *Operating System* provides basic facilities for application execution, file handling, hardware interfacing, communication, networking etc. The *Middleware* provides services like timing and communication between software components in a distributed system which simplifies the software development significantly. The *Architecture* organizes the software components into layers and modules with well defined interfaces. The modularity eases the development proces and facilitate efficient software reuse. The *Components* are the actual building blocks used for field robot applications and reused across projects and robot platforms. In the following sections each of the four parts of the software platform structure is described in detail.

2.2.3 Operating system

When choosing an operating system for FroboMind the design goals, in particular the *Software reuse* and *Open source* goals must be taken into consideration.

In robotics software implementations a Real-Time Operating System (RTOS) dedicated to embedded systems is typically used. The reason is that RTOS improves the performance for components that have hard or near real-time re-

quirements, and some of the available systems (eg. VxWorks and QNX) improve the reliability significantly compared to traditional operating systems. However they are usually not open source, and software developed for the RTOS typically runs on the target system only and therefore requires cross compilation, and on-target debugging etc. Linux based open source RTOS exist in different varieties. Examples are dedicated distributions where the entire system run as preemptive processes (eg. RTLinux), add-ons that create a small microkernel where linux runs as a task, and kernel patches or libraries that implement a near real-time environment like the RealTime Application Interface for Linux (RTAI) (Dozio and Mantegazza, 2003) and CONFIG_PREEMPT_RT (Carsten Emde, 2011). Some of these are well functioning and are utilized in commercial products as well as academic projects. However they are trying to match an RTOS with a fully fledged operating system which is inherently a tradeoff between low latency requirements and the overall efficiency of the system (McKenney, 2008). Other typical challenges of the RTOS are small development teams and a limited community supporting the projects. The RTOS distributions therefore often lag behind regarding support for new software and hardware, and support options in case of problems are limited. In applications where the robot is connected to the internet, network security may be an issue as well due to delayed software updates.

Based on the above considerations the linux distribution Ubuntu was chosen as operating system for FroboMind. Ubuntu is one of the largest and most active linux distributions available, and the annual Long-Term Support versions have guaranteed support for five years after release. Choosing Ubuntu has the major advantage that one can effortlessly run the same operating system on the field robot and a standard laptop used for software development. There is also a huge community of linux software developers which is an advantage when support is needed for a particular problem. Ubuntu only supports soft real-time execution, but in field robotics real-time requirements at the order of 50-200 Hz typically only exist in relation to low level actuator controlling, and this task is often distributed to an external embedded controller that communicates via a serial or network interface. The use of Ubuntu in robot systems with near real-time requirements is evaluated in the experimental section.

2.2.4 Middleware

As described in the review of related work a number of middlewares designed for robot control exists already. However for FroboMind most of the frameworks and middlewares listed in table 2.1 and described elsewhere must be disregarded because they do not comply with the *open source* design goal. Considering in addition the *software reuse* design goal stating that the middleware must be in active development this leaves only two attractive contestants: Orocos and

ROS.

Taking a closer look at Orocos it is based upon a few principal components: The Orocos Toolchain which contains the tools required to build Orocos components, the iTaSC framework (generates robot motions using constraints). the rFSM toolkit (Finite State Machines) and the libraries KDL (Kinematics and Dynamics) and BFL (Bayesian Filtering). The Orocos Toolchain includes the Real Time Toolkit (RTT) framework for developing real-time components in C++.

The ROS middleware offers inter-process communication through a set of facilities: Anonymous publish/subscribe message passing, recording and playback of messages, request/response remote procedure calls and a distributed parameter system. ROS also provides a set of robot libraries such as robot geometry, pose estimation, localization, mapping and navigation etc. ROS has a very good support for distributing components between networked computers. ROS is not a realtime framework and is thus not suitable for hard real-time critical software such low level control algorithms. There is, however, support for integration with the Orocos RTT.

Orocos and ROS both fit well with the FroboMind design goals. In favor of Orocos it supports real-time execution and the code base appears to be stable. The requirements for CPU power and memory appear to be lower than for ROS. But Orocos seems more oriented towards low level control than robot application building, and it is not as user friendly as ROS, which also has by far the largest supporting community and the most active development. ROS provides a very detailed documentation using a wiki, videos and examples. Fast prototyping using scripting is available with Python which may speed up application development in many cases. Based on these considerations ROS was chosen as middleware for FroboMind. The requirements with respect to computing power when running ROS and Ubuntu in robot systems with near real-time requirements is evaluated in the experimental section.

2.2.5 Architecture

The purpose of having a unified architecture shared across different projects and research groups is to facilitate software reuse at the application level. This allows the entire software platform to be transferred to new applications and field robots, the user needs only to add new high level behaviours and low level interface drivers needed for a particular application.

ROS does not propose or endorse a certain architecture or structure of the software, instead the different software components named ROS nodes are able to intercommunicate freely without restrictions to structure. As a result different users tend to use different designs and to some extent different interfaces

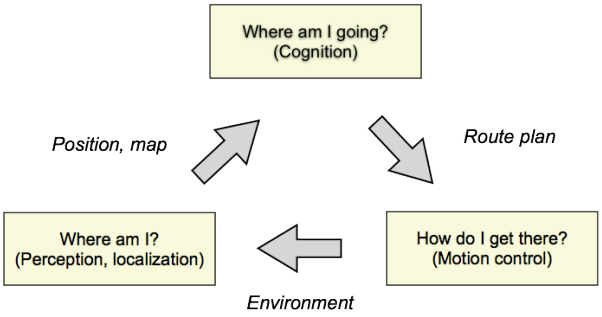


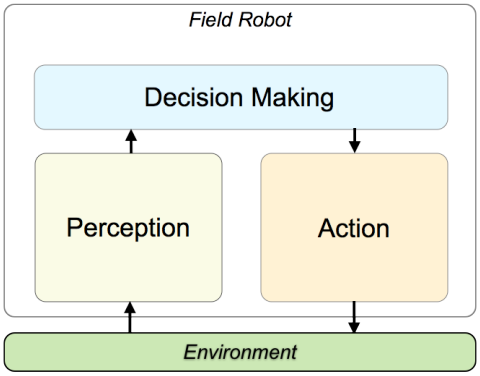
Figure 2.2: *A conceptual basis for the architecture of mobile robot navigation software described in the literature: Where am I? Where am I going? How do I get there?*

between components. The user therefore often has to build a new robotic application from scratch, and reuse mainly exists from component level down to snippets of code.

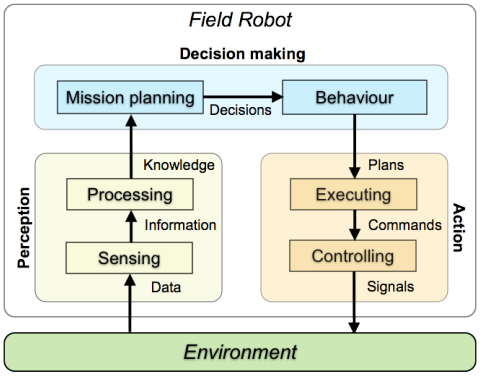
Adding a unified architecture to FroboMind increases the reusability of the components, but at the same time it limits the flexibility that ROS provides. In this section an architecture is proposed with the aim of balancing the tradeoff between reusability and flexibility.

Designing a conceptual architecture that appears intuitive to researchers, engineers and technicians is a challenging task due to their different backgrounds and experience from different projects. A simple basis for mobile robot software has been formulated by the questions: *Where am I?*, *Where am I going?* and *How should I get there?* (Leonard and Durrant-Whyte, 1991). The robot localizes itself based on percepts and prior knowledge, it then plans a route which leads towards the mission goal and navigates the route using motion control (figure 2.2). However for a field robot in precision agriculture the task is the primary objective and navigation is merely a means to fulfil the task. Performing the task typically involves interaction with an attached or trailed implement, and the conceptual architecture in figure 2.2 is therefore insufficient. Instead a more generic approach based on Intelligent Agents (Stuart Russell, 2010) is used for modelling the architecture. An agent is an autonomous entity which perceives its environment through sensors and acts upon that environment through actuators. The action taken by the agent in response to any percept sequence is defined by an agent function. The FroboMind principal architecture consists of perception, decision making and action layers (figure 2.3(a)). The layers have been decomposed to define abstraction levels (figure 2.3(b)).

Perception represents the organization, identification and interpretation of sensory information in order to represent and understand the environment (Schacter, 2011). The perception layer has been decomposed into:



(a)



(b)

Figure 2.3: *Decomposition of an Artificial Intelligence agent. (a) The agent perceives its environment through sensors and acts through actuators. The action taken by the agent in response to any percept sequence is defined by an agent function. (b) Decomposition to define data- and hardware abstraction levels.*

- **Sensing:** The robot perceives the surrounding partially observable environment (external percepts) through its sensors and assess the system interior state (internal percepts) through feedback from the platform and implement systems. Together these percepts constitute the available *information*. In addition this layer constitutes the abstraction between sensor hardware and the other parts of the architecture.
- **Processing:** By combining this *information* with previous, shared and a priori knowledge, the field robot maintains a model of the world and system state which constitute the accumulated *knowledge*. Since observables are governed by a certain amount of uncertainty, probabilistic methods are typically utilized to optimize the model.

Decision Making constitutes the cognitive layer and represents an AI agent function which determines the action taken in response to the accumulated *knowledge*. The implementation of the decision making layer may vary depending on application and users may choose to use other methodologies than the current support for model-based, utility-based AI agents and hierarchically organized finite-state machines to describe robot behaviour.

- **Mission Planning:** The robot mission planner continuously monitors the accumulated knowledge as well as any user interaction, and makes on this basis a *decision* about the optimal behaviour which leads towards the fulfillment of the mission. This corresponds to an AI agent utility function described in (Stuart Russell, 2010).
- **Behaviour:** The optimal behaviour decided by the mission planner origins from a library of possible behaviours available to the robot and implement. The active behaviour continuously monitors the accumulated knowledge and user interaction and updates action *plans* for the robot and implement action accordingly.

Action The action layer carries out the action defined by the action plans and has been decomposed into:

- **Executing:** The *plans* produced by the active behaviour are executed with respect to time and state. Based on this *commands* are sent to the controlling layer.
- **Controlling:** Commands issued by the executing layer are transmitted to low level controllers within the architecture or to external controller interfaces. This layer constitutes the abstraction between the field robot actuator hardware and other parts of the architecture.

The FroboMind architecture (figure 2.4) represents an expansion of the decomposition in figure 2.3. This is indicated by the grouping of modules containing

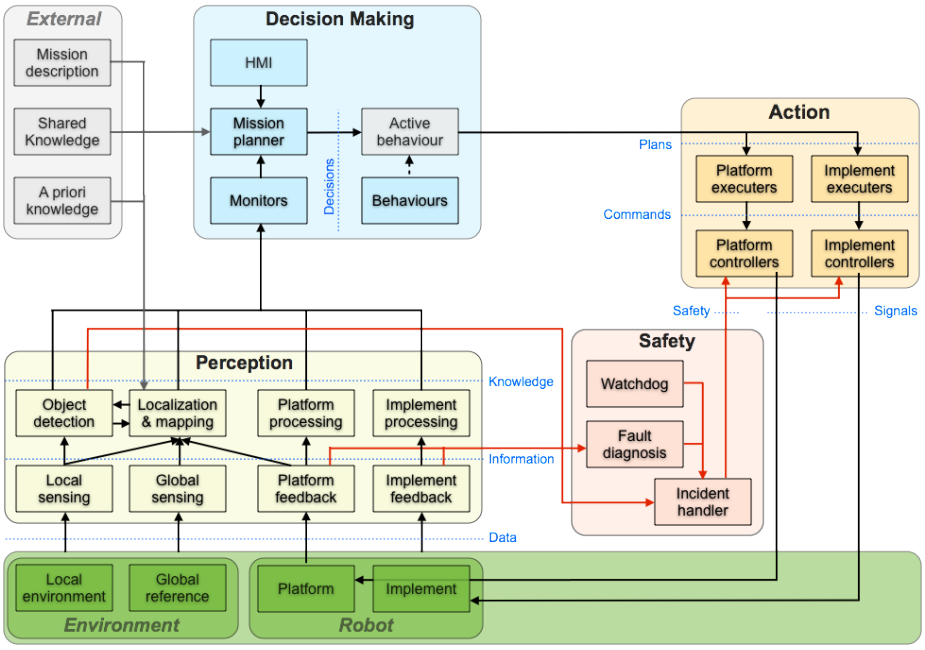


Figure 2.4: *The FroboMind Architecture. The modules containing software components are grouped into the Perception, Decision making, Action and Safety layers. The blue dashed lines indicate the data interfaces between layers and modules.*

software components into the *Perception*, *Decision making*, *Action* and *Safety* layers as well as the data interface between layers and modules indicated by the blue dashed lines. Internal fault diagnosis and incident handling are organized in a separate *Safety* module to minimize potential software errors and thus ensuring a high level of reliability. In order not to clutter the overview it is assumed that any component has access to data accessible by its predecessor. Multiple connections to successors are shown only where relevant to the understanding, and data available globally has not been included.

2.2.6 Components

The FroboMind components are implemented as ROS packages. Most of the current lower level components at the Perception and Action layers are written in C++ while many of the higher layer components are written in Python.

The components are located in a directory structure where each layer in the architecture (figure 2.4) is represented by a directory (table 2.3), each module as a subdirectory herein and the components are located in the module subdirectories. The FroboMind component directory structure and related documentation are located in a repository available through the website `http:`

Directory	Content
/fmSensors	Sensor interfaces and information extraction (hardware abstraction layer)
/fmProcessors	Processing sensor information to obtain knowledge about the robot state.
/fmDecisionMakers	Robot mission, behaviour and HMI components.
/fmExecutors	Executing the current behaviour, e.g. navigation and implement control.
/fmControllers	Low level controllers and actuator interfaces (hardware abstraction layer).
/fmSafety	Watchdog, fault diagnosis and incident handling.
/fmApp	Directories containing components, scripts, launch files etc. related to the application of FroboMind to a particular project.
/fmLib	Interface drivers, software libraries etc. not represented in the architecture.
/fmTools	Various non-ROS tools and utilities.

Table 2.3: *The component (top level) directory structure at the FroboMind repository.*

//www.frobomind.org

2.2.7 Versions

FroboMind has been in active development since 2011 where the first prototype architecture and components were implemented in ROS Electric on Ubuntu 10.04. Since then FroboMind versions have been created each year with an updated architecture, improved components and compatible with updated versions of Ubuntu and ROS. The version numbers corresponds to the farming season. Version 2014 is the current FroboMind software platform described in this work. It is based on ROS Hydro and Ubuntu 12.04 LTS.

2.3 Experimental validation

In this work three experiments have been carried out to evaluate the performance of FroboMind when applied to various precision agriculture tasks and field robots. The first experiment evaluates the performance of FroboMind in robot systems with near real-time requirements including requirements with respect to computing power. The second experiment evaluates whether FroboMind significantly decreases the development time and resources required to perform field experiments. The third experiment uses available data from existing projects and robotic platforms that use FroboMind, to evaluate software reuse across these projects.

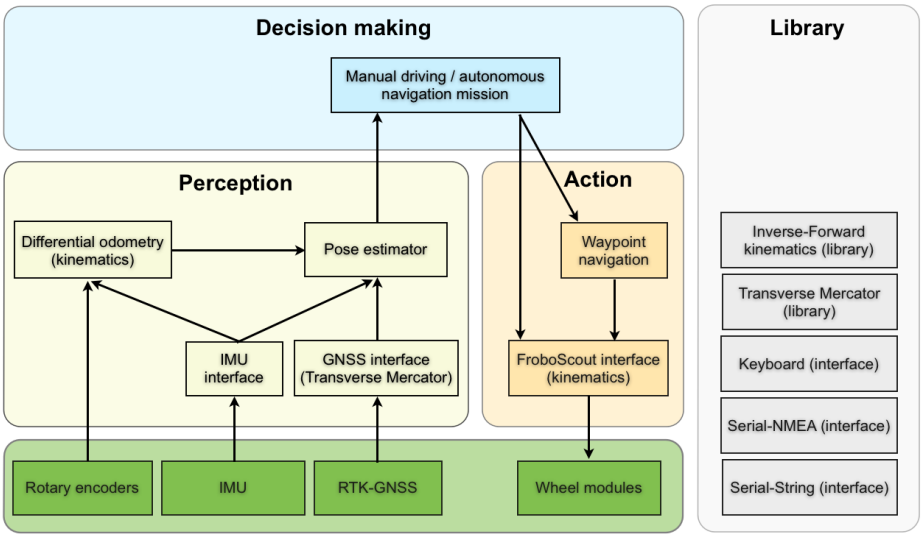


Figure 2.5: *FroboMind example architecture for autonomous navigation of a route plan.*

2.3.1 Evaluating the performance of FroboMind in robot systems with near real-time requirements.

An experiment was defined to evaluate the performance of FroboMind including ROS and Ubuntu in robot systems with near real-time requirements including requirements with respect to computing power. The experiment is based on a typical application for a field robot in precision agriculture: Autonomous navigation of a predefined route while controlling an attached or trailed implement. Figure 2.5 shows an example of the FroboMind architecture listing the components required for the autonomous navigation. Each of the components are described in table 2.4. All the library functions listed in figure 2.5 are implemented in C++.

In the experiment a differentially steered FroboScout robot (table 2.8) was set to autonomously navigate a static route plan containing 5 waypoints using the architecture in figure 2.5. The route length is 47 m. Two auxiliary components were added to monitor the system real time performance:

- A computer load monitor component which averages the CPU and memory load every 0.2 s and publishes the results through ROS.
- A real time performance test component consisting of a Python script and an external Atmel ATmega series microcontroller connected to a serial port. The Python script running at a rate of 100 Hz sends one byte to the serial device at each update. The ATmega firmware continuously counts the number of 100 μs ticks between each received byte and returns the

Layer	Component	Rate	SLOC
/fmSensors	**IMU interface** reads and publishes the yaw axis gyro rate from an IMU.	40 Hz	261c
	GNSS interface reads and publishes the GNSS *GPGGA* information including conversion of the position to Transverse Mercator coordinates.	10 Hz	271c
/fmProcessors	**Differential odometry** estimates the current position and orientation based on information from the wheel encoders and the IMU yaw angle gyro.	50 Hz	527c
	Pose estimator uses pre-processing and an Extended Kalman Filter to estimate the absolute position and orientation based on the differential odometry and GNSS information.	50 Hz	592p
/fmDecisionMakers	**Manual driving/autonomous navigation mission** In the *manual driving* behaviour velocity commands from the keyboard are sent directly to the FroboScout interface. In the *autonomous navigation* behaviour the waypoint navigation component is activated.	20 Hz	199p
/fmExecutors	**Waypoint navigation** navigates the waypoint list by publishing linear and angular velocity commands.	20 Hz	1077p
/fmControllers	**FroboScout interface** converts linear and angular velocity commands to wheel velocities, publishes encoder feedback.	50 Hz	428p

Table 2.4: *FroboMind example components used for autonomous navigation of a route plan. Rate describes the update function rate. Source Lines Of Code (SLOC) include comments and blank lines but not library functions. c refer to C++ and p to Python.*

latest count upon receiving a byte. This count is in turn received by the Python script and published through ROS.

In total 14 ROS nodes were launched. Together they published ROS messages under 28 different ROS topics, hereof 2 at 100 Hz, 9 at 50 Hz, 1 at 40 Hz, 4 at 20 Hz, 3 at 10 Hz, 6 at 5 Hz and 3 at a lower rate. During the experiment trials all messages published by ROS were recorded using the rosbag tool. The computers used in the trials were connected to internet during the trials but FroboMind did not initiate external network connections. All computers had a standard Ubuntu installation, no attempts to optimize for performance by shutting down default services were made.

2.3.2 Evaluating if FroboMind significantly decreases the resources required to perform field experiments

Evaluating whether the FroboMind software platform significantly decreases the development time and resources required to perform field experiments is difficult because no reference data seem to exist in the literature. An experiment was therefore conducted with the purpose of trying to quantify the portability of FroboMind to a new robot platform by analysing the associated work. The experiment was conducted in collaboration with the University of Hohenheim (UH) and is presented in (Jaeger-Hansen et al., 2013). A brief description and the results is included here for completeness.

The experiment was designed so that as many parameters as possible were controllable. The task was defined as field robot navigation through an apple tree orchard using local sensors only which is another typical precision agriculture application (Barawid et al., 2007; Blas, 2010). The Autonomous Mechanisation System (AMS) robot (figure 2.6(a)) owned by UH was used for the experiment. The AMS has been utilized in several previous precision agriculture research projects (Reske-Nielsen et al., 2006)(Griepentrog et al., 2009) using Mobotware (Beck et al., 2010), and both the hardware and low level software is well tested and documented. A lidar and an Inertial Measurement Unit (IMU) were used as navigation sensors. An RTK-GNSS system was included for the purpose of recording reference data.

The experiment was conducted during a 5 day workshop where 4 developers worked full time porting FroboMind to the AMS, implementing the orchard navigation task and documenting the process. Preparations before the workshop included preparing the AMS and related hardware, reading documentation and planning the workshop. The task at the workshop was therefore defined as interfacing FroboMind to the AMS and building an application supporting autonomous navigation along the tree rows of the orchard and perform headland turns at the end.

(a) Autonomous Mechanisation System (AMS)

(b) Apple three orchard used for the trials in this work.

Figure 2.6: *Photos of the Autonomous Mechanisation System (AMS) and the apple three orchard used for the trials.*

In the orchard used for the navigation tests the tree rows were approx. 100 m long and interspaced by 4 m (figure 2.6(b)). In the first trial the AMS mission was to navigate autonomously between two rows of trees. At the end of the row the AMS would make a 90 degree left turn, drive straight, and make a 90 degree left turn which would position the robot in an adjacent row. It was decided to repeat this process continuously 12 times corresponding to 6 full rounds driving the same track. In the second trial the AMS mission was to navigate autonomously through the same orchard alternating between left and right turns. Estimation of position and orientation (pose) relative to the tree rows is handled by FroboMind components from a previous project: A Ransac based algorithm uses the lidar data to detect the tree rows and outputs the angle and offset relative to the rows. The angle is fused with the IMU yaw angle data using an Extended Kalman Filter, and the resulting pose is used to control the AMS steering wheels using a PID controller. When the lidar detects the ends of the tree rows, it switches to a turn state and performs the turn based on the IMU data. When the lidar detects the ajacent row, it switches back to in-row navigation.

2.3.3 Evaluating software reuse across existing projects using FroboMind.

During the past three years FroboMind has been used in various projects in precision agriculture as well as in some projects in similar domains. Examples of tasks are navigation of route plans using GNSS, navigation in row crops and orchards using local sensors, control of passive and active implements and interfacing to autonomous implements. Examples of sensor interfaces implemented in FroboMind are encoders, GNSS, IMU, lidar, 3d lidar, 3d vision row camera, localization using stereo vision, total station and metal detector. Examples of actuator interfaces controlled by FroboMind are relays, brushed and brushless motors, servos, linear actuators, hydraulics propulsion and diesel engines. Together these tasks and interfaces support many of the operations required in precision agriculture and the potential for efficient software reuse is thus high.

The third experiment is based on available data from existing projects and field robots that use FroboMind. The purpose is to evaluate the extent of software reuse by looking at the approximate number of physical Source Lines Of Code (SLOC) shared across these projects. As the projects have been carried out during a period where the FroboMind architecture and core components were in very active development it is difficult to perform an accurate comparison. But it provides an indication of the degree of reusability.

	PC1	PC2	PC3
Laptop:	Acer Aspire One ZG5	IBM ThinkPad X61s	Acer Travelmate B113
RAM:	1 Gb	2 Gb	4 Gb
CPU (Intel):	Atom N270	Core 2 Duo L7300	Core i3-2377M
Clock:	1.60 GHz	1.40 GHz	1.50 GHz
Cores/siblings:	1/2	2/2	2/4
Cache:	512 kb	4096 kb	3072 kb

Table 2.5: *Specifications for the computers used for the 3 trials in the FroboMind real-time performance experiment.*

	mean	variance	95'th percentile	maximum
PC1	0.58 ms	0.41 ms^2	1.52 ms	14.1 ms
PC2	0.26 ms	0.055 ms^2	0.72 ms	4.0 ms
PC3	0.24 ms	0.0074 ms^2	0.36 ms	1.3 ms

Table 2.6: *Statistics for the schedule delays experienced by the 100 Hz component in the three trials of the FroboMind performance experiment.*

2.4 Results

2.4.1 Evaluating the performance of FroboMind in robot systems with near real-time requirements.

Three different trials were conducted, each using one of the computers listed in table 2.5. The Graphical User Interface (GUI) *Ubuntu desktop* was running on PC2 and PC3, but on PC1 it was shut down prior to the trial because of high CPU load.

In all trials did the robot complete the waypoint navigation task successfully in about 100 seconds. Figure 2.7 shows the CPU and memory load averaged at 0.2 s intervals. Figure 2.8 shows the scheduling delays experienced by the 100 Hz FroboMind component during each of the 3 trials, table 2.6 shows statistical data for the delays. Due to lack of an absolute, accurate time source in the experiment setup the delay measurements have been calibrated for offset and skew errors based on statistical calculations, and the data may therefore slightly inaccurate. The skew calibration constants were verified using external frequency measurement.

2.4.2 Evaluating if FroboMind significantly decreases the resources required to perform field experiments

During the workshop a journal was continually updated with information about completed tasks etc. A summary of the estimated time consumption on different

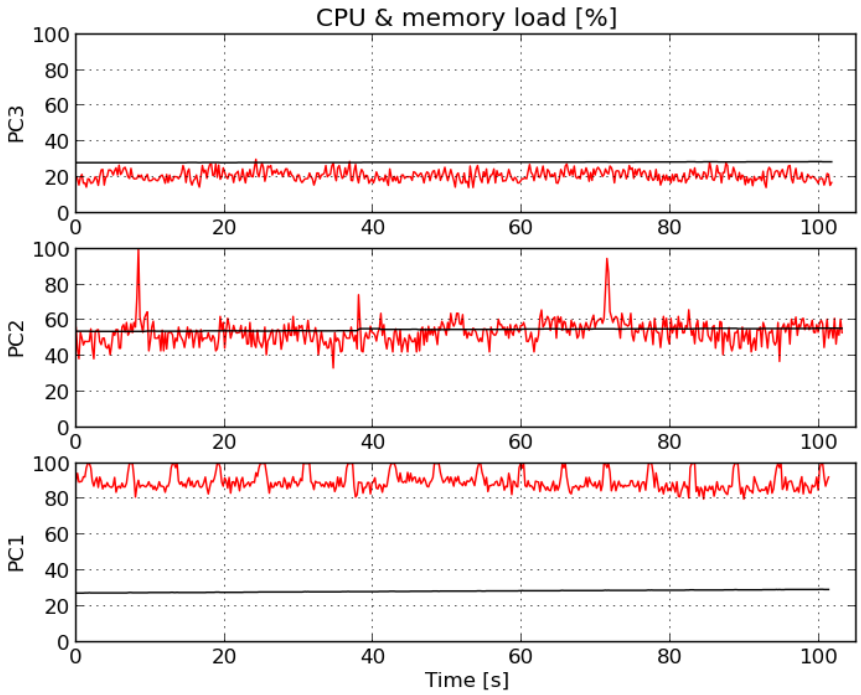


Figure 2.7: *CPU & memory load during autonomous navigation of a route plan for each of the 3 trials. The red graphs shows % of max CPU load. The black graphs show usage % of the total memory.*

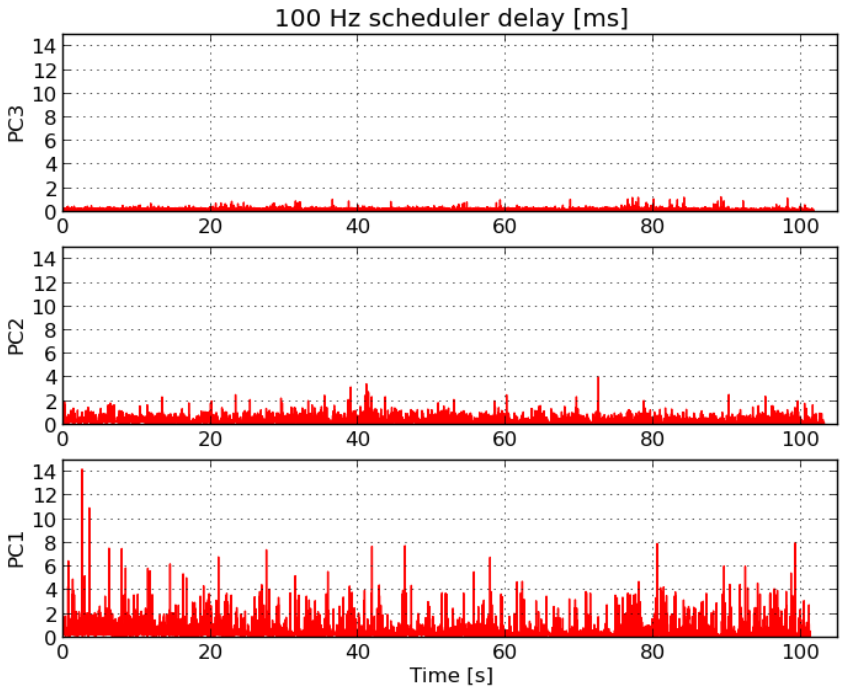


Figure 2.8: *100 Hz scheduler delays during autonomous navigation of a route plan for each of the 3 trials. The y axis represents the delay of each 10 ms schedule.*

Activity	Duration
Adding new components to FroboMind that directly support the AMS platform. This includes obtaining information about the AMS interface from documentation and available source code.	60 hours
Simulating and testing remote controlled and autonomous driving with the AMS positioned in a test stand.	30 hours
Adding and improving the documentation www.frobomind.org when the project work revealed issues not properly documented.	10 hours
Updated documentation for the AMS when the project work revealed issues not properly documented.	10 hours
Testing the implemented behaviours with the AMS on the ground.	20 hours
Collecting data from a manual run in the orchard.	6 hours
Analysing data from a manual run in the orchard to prepare autonomous operation.	20 hours
Creating a FroboMind mission controller and behaviors for the AMS navigating autonomously in an orchard using local sensors.	40 hours
Testing autonomous behaviour in the orchard.	30 hours
Updating project documentation and web pages.	12 hours

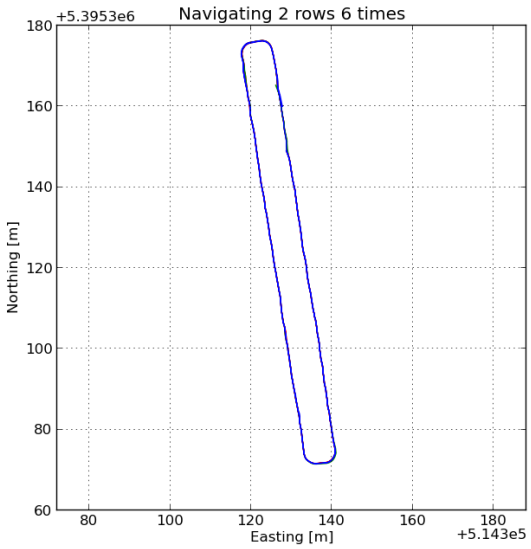
Table 2.7: *Summary of estimated time consumption during the workshop.*

sub tasks is listed in table 2.7.

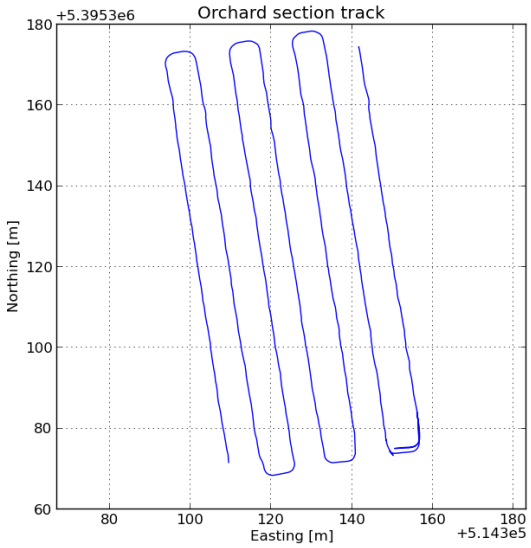
While porting FroboMind to the AMS it was discovered that the AMS exhibited errors when controlling the front wheels. When requesting the AMS to drive straight it would drift slowly to the left. When requesting the AMS to turn left the wheels were positioned correctly but when requesting the AMS to turn right the steering wheels were positioned at an angle significantly lower than the requested angle. Based on the performed tests it is likely that the problem is with the steering hardware, however due to the time constraints it was decided to mitigate the problems by modifying the relevant FroboMind components. The process to identify, understand and mitigate these errors added substantially to the consumed time.

The first trial navigating 6 full rounds driving the same track in two adjacent rows was completed successfully. Three rounds were completed without human intervention. During two of the rounds the navigation failed one time at the same location where the apple trees at one of the rows were replanted by much younger and hence smaller trees. During one round the navigation failed at a location where trees were missing for more than 6 meters at one of the rows. Figure 2.9(a) shows the overlay GNSS track from all 6 rounds.

In the second trial navigating a section of rows in the orchard it was decided



(a) Trial 1.



(b) Trial 2.

Figure 2.9: *Recorded GNSS track from the trials. (a) shows the overlay GNSS track from all 6 rounds.*

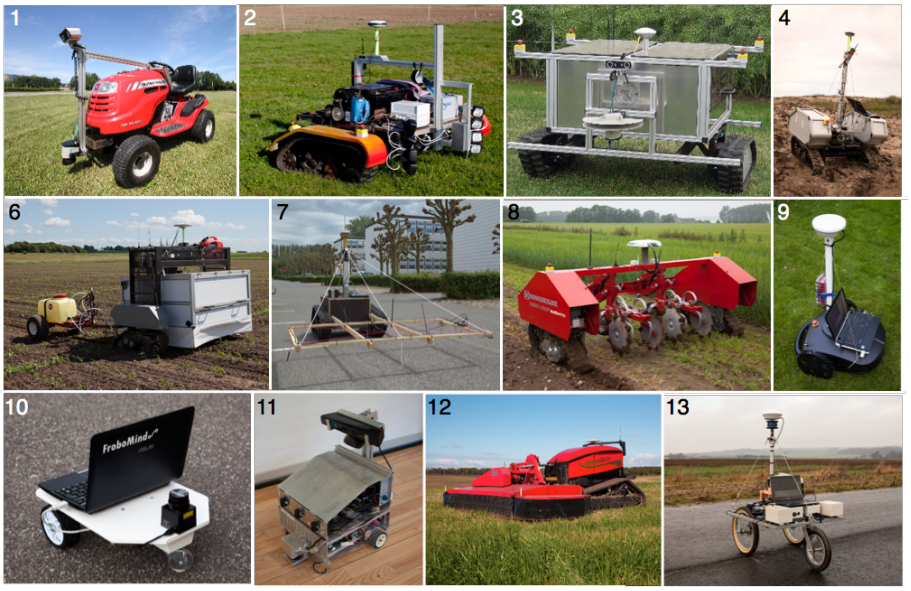


Figure 2.10: *Images of robots using FroboMind. The numbers show the historical order of FroboMind integration and refer to the list of robots in table 2.8.*

that focus would not be on the sensor/controller performance under difficult circumstances, so a person stepped in as "tree" whenever one of the rows had a large hole between the trees. The trial was concluded after completing 7 rows. The trial was completed without human intervention except two times where the AMS failed a turn to the right at a row end due to the described steering problems. Figure 2.9(b) shows the recorded GNSS track from this trial.

2.4.3 Evaluating software reuse across existing projects using FroboMind.

Table 2.8 lists robots that are known to have been interfaced to FroboMind. The table lists the latest known application where FroboMind has been used as well as the latest version of FroboMind known to work on the robot. Based on information obtained from each application, abbreviations for software components shared with other robots are listed next to the version number. The abbreviations refer to table 2.9 which lists the component function along with the physical SLOC including comments and blank lines. Due to different component versions the SLOC may differ between different robot installations and should thus be considered to be approximate. Excluded from the list of components are sensor interfaces and libraries etc. Figure 2.10 shows images of each robot, the image number refer to the list of robots in table 2.8.

	Latest application	**FroboMind**
ASuBot 1	Navigating orchards using lidar.	2013 ASU RDT RNV
Armadillo I 2	Navigating maize fields using lidar.	2012 ARD RDT RNV
Armadillo Scout 3	Crop monitoring.	2012 ARD
Armadillo III 4	Driving and navigation in dunes.	2012 ARD ODO POS WPT
AMS 5 fig. 2.6(a)	Navigating orchards using lidar. (Jaeger-Hansen et al., 2013).	2012 AMS RDT RNV
Armadillo IV 6	Large scale precision spraying trails (Jensen et al., 2013).	2013 ARD ODO POS WPT PMP
Pichi 7	Detection and mapping of anti-tank mines and large unexploded objects.	2013 PIC ODO POS WPT COV MDI HMP
Robotti 8	Mechanical row weeding (Green et al., 2014).	2013 ARD ODO POS WPT RWD
FroboMower 9	Mowing using low cost sensors.	2013 FBT ODO POS WPT
Frobit 10	Education (Larsen et al., 2013) & rapid prototyping.	2013 FBT ODO WPT
SMR 11	Education.	2013 MBW ODO WPT
GrassBots 12	Grass mowing on low lands.	2013 GBT ODO POS WPT COV
FroboScout 13	Highway surveying.	2014 FST ODO POS WPT SUR

Table 2.8: *A list of robots using FroboMind. Column 3 lists the FroboMind version number followed by the reused components detailed in table 2.9.*

Component	Description	SLOC
AMS	Interface to the AMS robot	410c
ARD	Interface to the Armadillo robot	845c
ASU	Interface to the ASuBot robot	435c
COV	Area coverage algorithm	170p
FBT	Interface to the Frobit robot	285c
FST	Interface to the FroboScout robot	428p
GBT	Interface to the Grassbot robot	?
HMP	Hazard mapping algorithm	850c
MBW	Interface to Mobotware (Beck et al., 2010)	367c
MDI	Mine Detection Implement interface	125c
ODO	Differential odometry algorithm	527p
PIC	Interface to the Pichi robot	890c
PMP	Parcel map localization algorithm	330p
POS	Pose estimator algorithm	592p
RDT	Detecting crop rows using Lidar	266c
RNV	Navigation in row crops	255c
RWD	Row weeding implement interface	90c
SUR	Highway surveying application	304p
WPT	Waypoint navigation algorithm	775p

Table 2.9: *List of components referred from table 2.8. Physical Source Lines Of Code (SLOC) for the Grassbot robot interface is unknown as it is closed source. c refer to C++ and p to Python.*

2.5 Discussion

2.5.1 Evaluating the performance of FroboMind in robot systems with near real-time requirements.

Ubuntu was chosen as operating system and ROS as middleware because they complied best with the design goals *extensibility*, *scalability*, *software reuse* and *open source*. This was, however, at the cost of hard real-time capabilities that could have been obtained using e.g. RTAI (Dozio and Mantegazza, 2003) and Orocos (Bruyninckx, 2001). Although hard real-time capabilities are rarely required in field robotics except for low-level control which is typically handled by external embedded systems, it is important to know the system timing uncertainty.

The computing power in the three computers used in the performance evaluation in the first experiment (table 2.5) are at the low end of what is available today, however still above the currently popular ARM based embedded boards such as Raspberry Pi and BeagleBone Black.

The trials show a stable memory usage (figure 2.7) which is consistent with that none of the active FroboMind components perform any significant dynamic

memory allocation. The approximate memory usage was approximately 1/3 Gb for PC1 where the GUI was disabled. For PC2 and PC3 the memory usage was approximately 1 Gb.

The CPU load varies in the different trials which is to be expected considering the available computing power. The CPU load for PC1 exhibits a certain periodic pattern of peaks. A subsequent analysis revealed that the peaks were caused by an installed fan control script which sleeps 5 seconds between each execution, so this is not related to FroboMind.

Comparing the CPU load with the scheduler delays shown in figure 2.8 and table 2.6 it is clear that PC1 is pushed to the limits of its computing power at this load. It has difficulties running a 100 Hz scheduler without overstepping every now and then. PC2 and PC3 manages fine at the current load if the observed delays are acceptable to the application.

In other precision agriculture applications there may be components causing the CPU load to vary significantly with time which may be more or less predictable. Examples are dynamical mapping, route planning, image processing etc. In these cases there are different options: 1) Perform a similar experiment to validate that enough computing power is available to the application. 2) Distribute the components with varying load to another networked platform. 3) Integrate ROS with the Orocos Real Time Toolkit which is documented at the ROS website.

The results from this experiment gives rise to the recommendation that in any application where safety is a concern such as the use of large or heavy machinery, implements with moving mechanical parts, driving at high speed etc., the CPU load and the soft real-time performance should be monitored continuously, and detected anomalies should be used as input to the safety system.

2.5.2 Evaluating if FroboMind significantly decreases the resources required to perform field experiments

During the 5 day workshop the 4 man development team managed to interface FroboMind to the AMS and build the application for autonomous navigation of an orchard as well as testing the application in two trials. The trials were completed, but navigation was not completely reliable. Upon reviewing the observations at the trials and subsequent analysis of logged ROS messages it was concluded that the reliability of the in-row navigation can be increased by a few modifications to the row detecting algorithm and the navigation controller.

The workshop revealed some issues with FroboMind that need to be addressed. 1) Some of the basic components needs further development with respect to reliability, in particular interfacing to a robot platform through low level in-

terface drivers needs to work seamlessly. 2) Interfaces between layers in the architecture need to be reviewed and properly documented. 3) The usability of FroboMind must be enhanced by a better integration with a simulation environment like Stage. These issues have not yet been solved completely, but a number of improvements have been made since the workshop took place.

As shown in table 2.7 the developers worked approximately 12 hours/day each, about half the time was spent on interfacing and the other half on the orchard application. Factors like the described problems with the AMS, transport from the university to the orchard test site and changing weather conditions prolonged the work. Similarly the developer team had collectively an extensive experience in all part of the FroboMind software platform, and the AMS actuators and sensor interfaces were already installed and thoroughly tested using Mobotware, which eased the work. This makes it very difficult to compare with similar experiments, and thus to conclude if FroboMind significantly decreases the resources required to perform field experiments, but experiences from previous projects show that this is a very short time moving from a completely new software implementation on a robot to trials in the field navigating autonomously.

2.5.3 Evaluating software reuse across existing projects using FroboMind.

Table 2.8 lists a total of 13 robots that have been interfaced to FroboMind during the past three years. With the exception of the AMS and SMR robots that are based on Mobotware they have all been constructed within the same timeframe. The use of FroboMind is not widespread though, the listed robots are spread across only 4 universities and 3 companies, most of them working together developing the field robots. In addition to the list a smaller number of robots developed by researchers, companies and students are known to either use FroboMind or have used FroboMind as a starting point for the software development.

The statistical website ohloh.net reports that the FroboMind directory structure contains 104,617 SLOC (April 2014) and the core part of the ROS platform contains 418,977 SLOC (December 2013). In this context the third column of table 2.8 and the associated table 2.9 listing SLOC for the components that are shared between some but not all platforms could be considered an indication of the differences in the robot software rather than the similarities.

It seems reasonable to conclude that the level of software reuse between the listed robots is high which may in part be attributed to the modular structure of the FroboMind architecture and to the flexibility that ROS provides. It is, however, also caused by the fact that the robot platforms and their current

applications are fairly similar with respect to the robot software.

2.5.4 Lessons learned

Choosing Ubuntu as operating system for the software platform has proven to be a great advantage. The limitations caused by not having hard real-time capabilities are counterbalanced by a number of advantages that all relates to simplicity in installation, use and maintenance and hence decreasing the time consumption from development to field trials. The ability to use the same operating system on the robot and the developers laptops speeds up the development process significantly. Ubuntu is the only part of FroboMind not fulfilling the *Open source* design goal of using permissive free open source software. This is not a major problem though, commercial products based on Ubuntu exist in many varieties.

The choice of ROS as middleware has had its advantages as well as drawbacks. ROS has a very steep learning curve in the beginning, but after some struggling with understanding the concepts it becomes a valuable tool. The user feedback indicates that FroboMind decreases the learning curve because it provides full working examples for simulation and execution on small robots rather than the user having to build the first test ROS setup from scratch. The ROS messaging system working across networked platforms, the ROS launch tool that ease launching of multiple ROS nodes locally and on networked computers, and the setting of parameters, as well as the many existing libraries and drivers etc. are highly valuable and supports the FroboMind design goals *Extensibility*, *Scalability* and *Software reuse*. The ROS tool for recording and playback of messages saves a tremendous amount of time when analyzing data and debugging. But ROS is still in a very active state of development. Experience shows that at new version releases, not all core components are fully working with the new version, and some of the new versions include substantial changes that induce unforeseen maintenance tasks. Examples are the introduction of the catkin build system and removal of the Stack concept in the ROS Groovy version. Seen in retrospect ROS was a better choice than Orocos when considering the purpose of FroboMind and the stated design goals. But it comes with the price of lacking hard real-time capabilities which is not a major problem but needs to be addressed in most applications, and the ROS code base appears to be less mature and stable than Orocos.

The FroboMind architecture design builds upon the results of the reviewed publications, especially Stanley (Thrun et al., 2006), CARMEN (Montemerlo et al., 2003) and (Vázquez-martín et al., 2008) provided a significant input to the design. The architecture has undergone several revisions since the first prototype was developed in 2011. The original draft was very detailed but has been simplified significantly because experiences from using FroboMind in research projects

have revealed a relationship between complexity and the willingness to accept and utilize the architecture. If the architecture is too complex, users tends to short circuit it by merging all the remaining functionality into a single or a couple of components, which is not in line with the *Software reuse* design goal. It is debatable if the architecture is still too complex, this should be investigated in the future work.

A well defined specification of data exchanged between the components is an important part of the architecture with respect to the design goals *Modularity* and *Software reuse*. For navigation sensors, localization and low level propulsion control and actuation this is reasonably well defined and where applicable aligned with current ROS standards. Interfacing the decision making layer and implement modules still need to be properly specified. Where applicable this work should be based on existing standards within agriculture and related field work such as OpenGIS, agroXML, GPX, Isobus etc.

Compared to the architectures listed in table 2.2 FroboMind obtains a *Yes* in all parameters which indicates that the aim of this work has been achieved. An overall performance comparison of the FroboMind software platform to solutions described in the related work is impossible though, as data relevant for comparison of the architectural and software performance has not been published.

2.5.5 The future of FroboMind

The primary focus in the development of FroboMind has until now been to establish a common open software platform that promotes software reuse and thus decreases the development time and resources required to perform field experiments. To a large extent this objective has been achieved and the work will continue through the application of FroboMind to new research projects, development of new components as well as improving the existing components in terms of reliability.

In addition to this a project focusing on safe behaviour of agricultural machines has been launched recently. FroboMind will be used in parts of the project, and it is expected that this will contribute significantly to the development of: 1) The safety layer which is currently limited to a deadman signal which enables the actuator controllers. 2) The decision making architecture, which currently is limited to fairly simple behaviours defined in finite state machines. 3) Source code integrity through the implementation of model driven auto-generation of component interfaces and structures.

2.6 Conclusion

This chapter has presented FroboMind, a software platform tailored to field robots in precision agriculture. FroboMind optimizes field robot software development in terms of code reuse between different research projects. At the current stage FroboMind has been ported to 2 different 4-wheeled tractors, 7 different configurations of tracked field robots and 4 differentially steered robots. Examples of current FroboMind applications are autonomous crop scouting, precision spraying, mechanical weeding, grass cutting, humanitarian demining and land surveying.

In an experiment evaluating the performance of FroboMind in robot systems with near real-time requirements it was concluded that FroboMind's soft real-time execution is sufficient for typical field robot tasks such as autonomous navigation of a predefined route. In any application where safety is a concern, the CPU load and the soft real-time performance should be monitored continuously, and detected anomalies should be used as input to the safety system.

To test whether FroboMind decreases the development time and resources required to perform precision agriculture field experiments, an experiment was performed porting FroboMind to a new field robot and applying this robot to autonomous navigation in an orchard using local sensors. This was achieved by 4 developers during a 5 day workshop, which is a very short time moving from a completely new software implementation on a robot to trials in the field navigating autonomously.

The software reuse across projects has been assessed using available data from existing projects and robot platforms. It was concluded that the level of software reuse between the listed robots is high which may in part be attributed to the modular structure of the FroboMind architecture, the flexibility that ROS provides and the fact that the robot platforms and their current applications are fairly similar with respect to the robot software.

It is concluded that FroboMind is usable in practical field robot applications by research groups and other stakeholders. Further development of FroboMind continue through new research projects where the current activities focus on the decision making structure and autonomous safe behaviour. The FroboMind software has been released as open-source at `http://www.frobomind.org` for others to build upon.

Chapter 3

Global pose estimation

This chapter deals with the FroboMind global pose estimation. The architecture and components used for global pose estimation are presented and evaluated in field tests. A substantial part of the chapter is devoted to low-cost positioning which is very important to the commercial potential of field robotics.

3.1 Introduction

Global positioning is a significant challenge when applying field robots to tasks in unstructured outdoor environments. Currently the only feasible solution for estimating the absolute position and orientation (pose) of the robot without installing some means of local infrastructure is using Global Navigation Satellite Systems (GNSS) such as Navstar GPS and GLONASS.

Tractor guidance systems available on the market today typically uses Real Time Kinematics (RTK), which is a technique applied to GNSS to obtain centimeter-level positioning accuracy. Researchers experimenting with field robots for precision agriculture tasks often use RTK systems as well to enable precise navigation and positioning of implements (Noack and Muhr, 2008; Norremark et al., 2008).

Using an RTK system is not ideal though. In a typical field environment there are locations where bushes, trees, buildings, power lines or other objects may shade the satellite signals or induce multipath errors. This causes temporary drops in positioning accuracy and the field robot must thus rely on dead reckoning. The RTK system output is therefore often fused with vehicle encoder feedback as well as inertial sensors to accommodate sporadic satellite signal degradation and outages (Bingbing Liu, 2005; Jones et al., 2006; Larsen, 1998). In semi-structured environments local sensors such as lidar and computer vision are used to aid the navigation as well (Blas, 2010).

Though the prices on RTK systems have fallen through the years, it is still a considerable investment. According to gpsworld.com the 2014 prices for an RTK-GNSS receiver ranges from 4,700 to 18,000 Euro. In addition to the RTK-GNSS receiver installed on the tractor or robot, access to data from a nearby reference station or network is required. This is usually either paid through an annual subscription or provided by a similar RTK-GNSS receiver installed at a static position by the farmer. For large scale crop seeding, harvesting and similar traditional tractor operations the cost is acceptable, but it constitutes a significant share of the cost of smaller field robots with a lower production capacity (Pedersen et al., 2006). For tasks where a low cost is imperative to the product viability such as crop scouting and weeding performed by multiple medium sized robots, grass mowing on smaller areas etc. this is not a feasible solution.

Fusion of the RTK system output with vehicle encoder feedback and inertial sensors to estimate the vehicle pose has been explored in previous research such as the above listed references. The use of low-cost GNSS receivers and RTK systems in field robotics and precision agriculture has also been evaluated in some publications such as (Freitas et al., 2012; Gomez-Gil et al., 2013; Cai et al., 2011; Codol et al., 2011).

This chapter contains a description of the implemented and field tested FroboMind global pose estimation components and an evaluation of the performance of low-cost GNSS in field robot applications. Section 3.2 describes the implementation of the FroboMind global pose estimation components, the sections 3.3 and 3.4 tests and evaluates standalone GNSS and low-cost RTK system applicability to precision agriculture, section 3.5 describes tests of the FroboMind global pose estimation, section 3.6 discusses the results.

3.2 FroboMind global pose estimation

Chapter 2 presented a FroboMind example architecture for autonomous navigation of a route plan by a differential steered robot. The perception layer outlined in figure 2.5 has been used in several FroboMind applications. Figure 3.1 shows a generalized version of the perception layer which fuses frequent relative measurements from robot encoder feedback and IMU with infrequent absolute measurements from GNSS. The purpose is to:

1. Improve the accuracy of the pose estimate by means of a probabilistic approach and a priori knowledge of the system.

2. Extrapolate the absolute pose estimation providing pose updates at a higher frequency than possible using the GNSS itself.

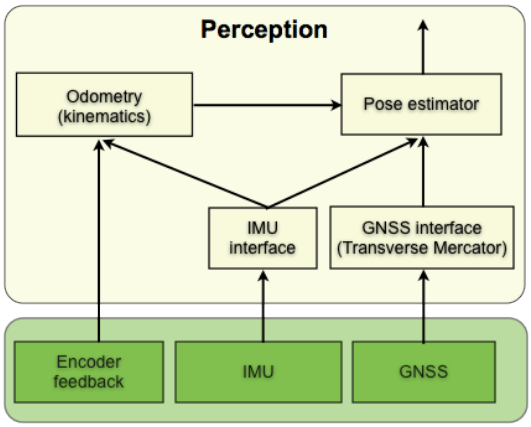


Figure 3.1: *The FroboMind pose estimation architecture for a typical GNSS based application.*

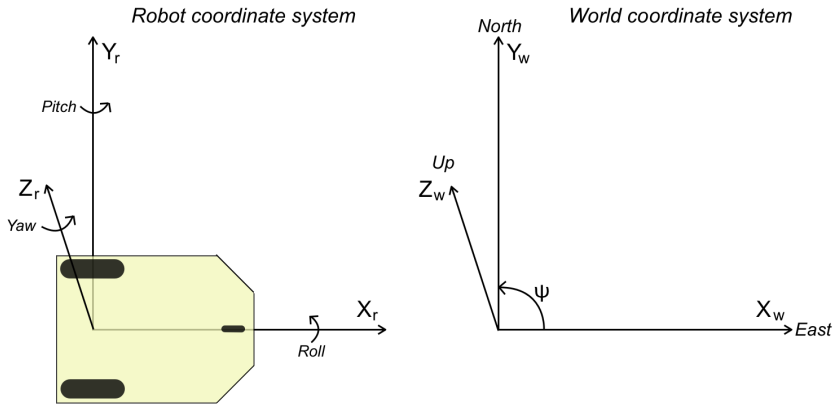


Figure 3.2: *FroboMind reference coordinate system. The Robot and World coordinate systems are aligned as illustrated.*

3. Increase the pose estimation availability during short periods of GNSS satellite signal degradation or outage.

A description of the components follows the introduction of the coordinate systems.

3.2.1 Reference coordinate systems

In this section the *Robot* and *World* reference coordinate systems used by the FroboMind perception components are defined. Figure 3.2 illustrates the relation between these two coordinate systems.

World coordinate system
The *World* coordinate system is based on the East, North, Up (ENU) Cartesian

coordinate system which by convention is used for land vehicles. The X_w-axis is oriented towards East, the Y_w-axis towards North and the Z_w-axis parallel to the Earth radial oriented upwards.

The ENU coordinate system is convenient as it is easily geolocated using a Transverse Mercator (TM) projection. The perhaps best known TM projection is the Universal Transverse Mercator (UTM) projection which is widely used for global localization. Being a a *flat earth* coordinate system ENU is best suited for smaller areas where the distortion due to the Earth curvature may be ignored. As an example UTM has been designed to keep the scale distortion less than 1 part in 1,000 within one UTM zone which spans 6°longitude. This means that the UTM positional error on a 1,000 m stretch is maximum 1 m. When a lower distortion is required such as in the surveying project described in section 7.3.4 other regionally defined TM projections are normally used.

The 2d pose of the robot with respect to the *World* coordinate system is described in (3.1):

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.1}$$

Where x, y represents the position with respect to the X_w-axis and Y_w-axis, θ is the robot orientation about the Z_w-axis oriented Counter Clock Wise (CCW) with respect to the X_w axis.

Robot coordinate system
The *robot* coordinate system is fixed to the robot. The X_r-axis is parallel to the longitudinal (roll) axis of the robot heading forward, the Y_r-axis is parallel to the lateral (pitch) axis of the robot heading to the left, and the Z_r-axis is parallel to the robot vertical (yaw) axis heading upwards. The Origo is located at surface level below the robot geometric center.

When the robot moves in the field, the *Robot* coordinate system is rotated by the angle ψ^w about the Z_w-axis with respect to the *World* coordinate system. The inclination of the robot is thus described by the *pitch* and *roll* angles.

Field coordinate system
In sloped areas a *Field* coordinate system is often introduced to accurately describe the field inclination at the robot location. The coordinate system is aligned with the *World* coordinate system but tilted by the angles ζ_x about the x-axis and ζ_y about the y-axis in the *World* coordinate system to describe the inclination.

In the FroboMind applications developed so far it has not yet been relevant

to use the *Field* coordinate system, so it is not implemented in the perception layer components. The 2d rotation matrix from the *Robot* coordinate system to the *World* coordinate system is thus given by the 3d rotation matrix about the Z_w-axis using the right hand rule:

$$\mathbf{R}^{\mathbf{r}\rightarrow\mathbf{w}} = \begin{bmatrix} cos(\theta^w) & -sin(\theta^w) & 0 \\ sin(\theta^w) & cos(\theta^w) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

3.2.2 Odometry

In this section the functionality of the odometry component is described. Odometry is the use of system feedback to estimate the updated robot pose with respect to the *World* coordinate system over time.

The typical system feedback from a field robot is output from incremental rotary encoders mounted on propulsion wheels and/or tracks, and from absolute encoders mounted on a steering wheel or similar directional control actuators. The encoder update rate is usually in the range of 20-200 Hz.

Most of the robots using FroboMind (table 2.8) are differential steered. The kinematic equations for a differential steered is derived in section 4.2.3.1. By multiplying the kinematic equations (4.10) and (4.11) by the encoder update interval δt it is seen that for each encoder update the robot motion in the *Robot* coordinate system is:

$$\delta d = \frac{d_r + d_l}{2} \tag{3.3}$$

$$\delta\theta = \frac{d_r - d_l}{W} \tag{3.4}$$

Assuming that the motion is linear within the time step interval δt the robot motion from time step $k-1$ to k in the *World* coordinate system is:

$$\mathbf{x_k} = \mathbf{x_{k-1}} + \delta\mathbf{x_{k-1}} \tag{3.5}$$

$$= \mathbf{x_{k-1}} + \mathbf{R^{r\rightarrow w}}\delta\mathbf{x^r_{k-1}} \tag{3.6}$$

$$= \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} cos(\theta^w) & -sin(\theta^w) & 0 \\ sin(\theta^w) & cos(\theta^w) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta d \\ 0 \\ \delta\theta \end{bmatrix} \tag{3.7}$$

$$\Updownarrow$$

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \delta d\ cos(\theta^w_{k-1} + \frac{\delta\theta}{2}) \\ \delta d\ sin(\theta^w_{k-1} + \frac{\delta\theta}{2}) \\ \delta\theta \end{bmatrix}, \tag{3.8}$$

$$\theta^w = \theta^w_{k-1} + \frac{\delta\theta}{2}$$

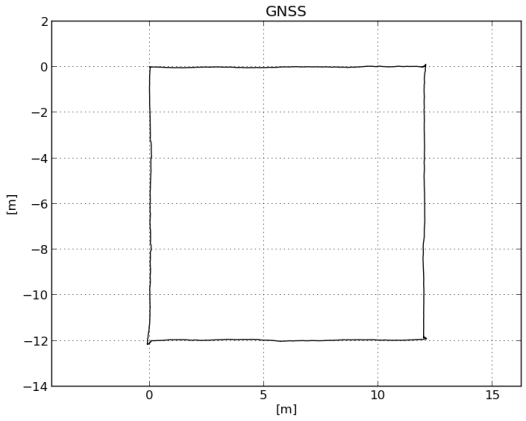
Equation (3.8) computes the updated robot pose in the *World* coordinate system.

Experiments performed within this work have shown that the linear motion for both differential and skid steered vehicles can be estimated surprisingly accurately even when driving in typical field environments such as bare soil, grass, crop fields etc. The accuracy of angular motion based on the odometry however varies with the robot and surface properties.

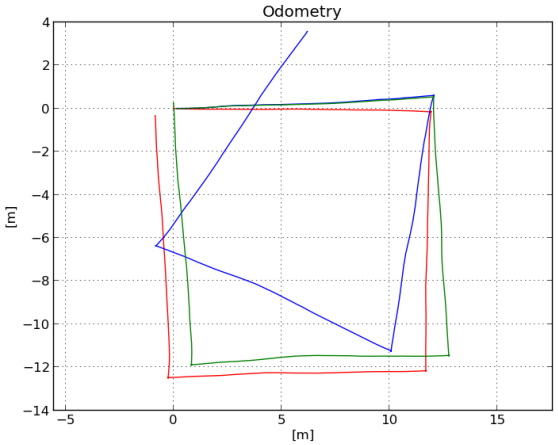
This is exemplified in figure 3.3 showing the Armadillo III robot (described in section 7.1.2) navigating a square sized 12x12 m. Figure 3.3(a) shows the recorded GNSS track and figure 3.3(b) shows the corresponding odometry. The blue track is the odometry with $W = 1.02m$ in (3.4) which corresponds to the distance between the center of the two tracks. The green track is the result of fitting W so the track resembles the GNSS track as much as possible. The resulting $W = 1.17$ indicates that the part of the Armadillo track which does not skid on the ground is not the center of the 20 cm wide track but further away from the Armadillo geometric center. Since it is a substantial work modelling a tracked robot accurately and thereby computing W, the feasible solution is to determine the parameter experimentally. Still this is no guarantee that the parameter is fixed for all conditions.

The red track is another approach calculating δd using (3.3) but using the angular rate output $\dot{\theta}$ from a gyro oriented in parallel with the robot yaw axis:

$$\delta\theta = \int_0^{\delta t} \dot{\theta}_{gyro}\ dt \tag{3.9}$$



(a) GNSS track from a trial navigating a 12x12m square.



(b) Odometry from the trial. The blue track is odometry using the distance 1.02m between the tracks of the Armadillo robot, the green track is odometry using a distance of 1.17m between the tracks, the red track is odometry using the angle rate from a gyro for estimating angular changes.

Figure 3.3: *Odometry accuracy example from the Armadillo III robot navigating a square on a grass field.*

Figure 3.3 shows that though the red track does not end exactly at the starting waypoint it resembles the GNSS track better at the straight parts of the track than the green track does. This is in line with the findings in (Ippoliti et al., 2005) that explores the use of a gyroscope sensor and finds that it produces a huge improvement of the estimated trajectory.

In section 4.3.2 an experiment similar to the one illustrated in figure 3.3 was performed using the Frobit robot described in section 7.1.3. After navigating a path of length 12 m containing 7 90°turns left and right the mean deviation from the true position was 0.118 m corresponding to 0.99%.

3.2.3 IMU interface

The purpose of this component is simply to extract information from the IMU data and publish this to the FroboMind fmProcessing layer. The ROS standard *Imu* message is used.

An IMU normally outputs angular rates, linear accelerations and magnetic field strength across three orthogonal axes. The output rate is in the range of 20-300 Hz. Some IMU's have built-in filters and algorithms for estimating the orientation in 3d space based on these sensor data. To accommodate all options the FroboMind odometry component supports using either the IMU gyro rate, the IMU estimated orientation or the odometry angle as input to the odometry computation.

The gyro data used in figure 3.3(b) was obtained from the z-axis gyro of a VectorNav VN-100 IMU. The gyro rate was read at 40 Hz.

3.2.4 GNSS interface

The purpose of the GNSS interface component is to extract position information from the GNSS data. GNSS receivers normally output data using the National Marine Electronics Association (NMEA) protocol 0183 v2.0. The basic message type is GPGGA which contains time (UTC), position and altitude (WGS-84), solution, number of satellites, Horizontal Dilution of Precision (HDOP) and differential station information.

The NMEA formatted output supports only positions formatted as geodetic coordinates which is inconvenient for field use. The geodetic coordinates are therefore often transformed into the UTM projection which is often used for absolute positioning in precision agriculture. However the pose estimation and navigation software implementations often lack support for transitions between different UTM zones and thus cause singularities when operating across zone borders. This is mitigated in the FroboMind Transverse Mercator (TM) conversion library by passing projection parameters directly. The component also

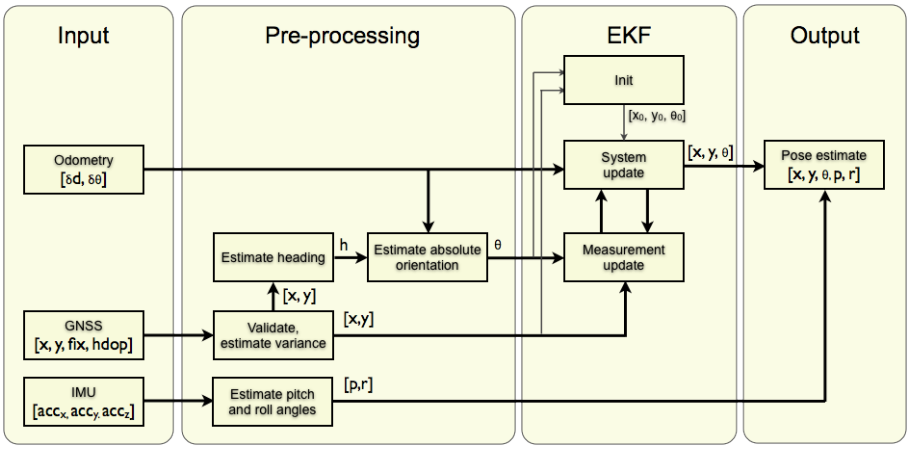


Figure 3.4: *Pose estimator component structure. Input from robot odometry, GNSS and IMU are pre-processed and the result is used as input to an Extended Kalman Filter (EKF) which output the updated odometric pose estimate.*

facilitates direct support for regional or local conventions like in Denmark where UTM zone 32 references are now used in areas physically located in UTM zone 33. The component is based on work documented in (Snyder, 1987; Gloeckler et al., 1996). Newer and more accurate TM conversion implementations exist, but the accuracy of this one is sufficient for the purpose and the computational load is acceptable.

The GPGGA data is published using a FroboMind GPGGA message as the standard ROS messages do not cover all vital information from the GNSS receiver. The FroboMind GPGGA message follows the principle defined in the widely used GPS Exhange format (GPX) standard to include all data published by the GNSS receiver

3.2.5 Pose estimator

Estimation of the robot absolute pose is a key issue in field robotics as inaccurate pose estimates make the robot navigation and implement operation inaccurate and sometimes impossible. An typical pose estimation approach is to implement a Kalman filter using wheel odometry feedback and optional gyro sensor measurements for the system update (prediction), and GNSS receiver data for the measurement update (correction) (Rodríguez and Gómez, 2009; Larsen et al., 1999).

Figure 3.4 shows the structure of the pose estimator developed in this work. It is based on an Extended Kalman Filter (EKF) that is prepended by a pre-processing filter. Both filters are described in the following.

3.2.5.1 Pre-processing

The purpose of the pre-processing is to validate and filter the input based on a priori knowledge of the system and hence mitigate the effects of sensor data errors often experienced in the field. For this purpose the pre-processor maintains an updated buffer of all measurements received the past 2 s.

Data from the RTK system is validated by rejecting sudden position jumps seen on some RTK-GNSS receivers. Position jumps are typically caused by multipath conditions or changes in satellite configuration, and often a jump is followed by a reverse jump after a few measurements. Depending on the RTK-GNSS receiver the jumps are sometimes but not always reflected in a changed solution, changed number of satellites and/or the HDOP values that often accompany the position output. The validation is performed by screening for position changes that would exceed the robot maximum velocity. Invalid jumps cause a rejection of the current and subsequent RTK-GNSS receiver data for a defined period thus forcing the pose estimator to rely solely on the odometry during this period. The issue is also discussed in (Jones et al., 2006) which proposes another method that limits the maximum innovation between consecutive GNSS updates based on constraints defined by the robot orientation. The method used in this work has proven to work well in the field tests though.

For validated positions the variance is estimated using constants which are based on the expected accuracy of GNSS in field conditions (Bevly, 2010; Rovira Mas et al., 2010). The constants are multiplied by the HDOP output from the RTK-GNSS receiver to increase the variance at locations with challenging signal reception:

$$\sigma^2_{rtk\_fixed} = (0.02m * HDOP)^2 \tag{3.10}$$

$$\sigma^2_{rtk\_float} = (2.0m * HDOP)^2 \tag{3.11}$$

$$\sigma^2_{dgps} = (5.0m * HDOP)^2 \tag{3.12}$$

$$\sigma^2_{gps(sps)} = (15.0m * HDOP)^2 \tag{3.13}$$

Estimating the absolute orientation of a field robot about the Z_w-axis of the *World* coordinate system is difficult. Not all RTK-GNSS receivers return a heading angle based on the change of position, and for those that do, the conditions and accuracy of the estimation is in most cases unknown due to the proprietary algorithms used. This is a problem, what would for instance be the heading output if the field robot is driving in hilly terrain and the GNSS antenna mounted on top of the robot moves laterally? The absolute heading available from some IMU's and digital compasses is usually not very accurate,

the electromagnetic noise from actuators and the amount of metal in the robot often contribute significantly to the inaccuracy thus rendering the measurements unusable for navigation.

The absolute orientation estimation is therefore based on position measurements from the GNSS and the robot odometry. A set of conditions are applied to the measurements which is detailed in algorithm 1. It should be noted that the algorithm makes the assumption that when the conditions are met, the robot is not skidding and the orientation is thus equal to the heading. The algorithm has proven to work quite well in the field trials performed until now, however there is room for improvement in future work. Filtering on the shape of the odometry and GNSS track rather than requiring straight line segments would probably increase both the accuracy and robustness of the orientation estimation.

Algorithm 1: Estimate absolute orientation

```
input  : GNSS and Odometry measurement buffers
output: Robot orientation

begin
    orientation = invalid
    if GNSS buffer solution = RTK fixed then
        gd = GNSS buffer distance
        gh = GNSS buffer absolute heading
        if gd ≥ 0.25m then // moved more than 0.25m
            od = odometry buffer relative distance
            oh = odometry buffer relative heading
            if od > 0 then // driving ahead
                if abs(oh) ≤ 5° then // driving straight
                    if abs(od-gd) < od*0.1 then // max 10% diff.
                        orientation = gh // assume no skidding
    return orientation
```

The robot pitch and roll angles are computed directly from the IMU accelerometer values (which normally refer to the North, East, Down (NED) Cartesian coordinate system):

$$p = arctan2(-acc_y, \sqrt{acc_x^2 + acc_z^2}), \quad p \in \left[\frac{-\pi}{2}; \frac{\pi}{2}\right] \tag{3.14}$$

$$r = arctan2(acc_x, acc_z), \quad r \in [-\pi; \pi[\tag{3.15}$$

The accuracy of the output has been found to be sufficient, however a fusion of the linear accelerations and gyro rates may yield a higher accuracy.

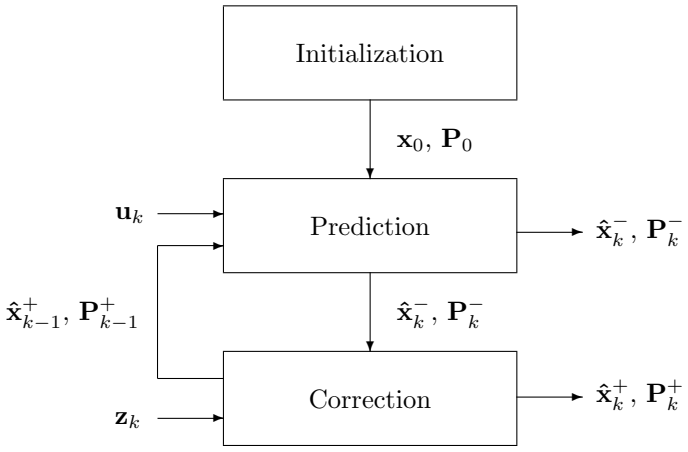


Figure 3.5: *The recursive prediction and correction steps of an Extended Kalman Filter. The inputs to and outputs from the steps are described in details in the text.*

3.2.5.2 Extended Kalman Filter

This section describes the odometric Extended Kalman Filter (EKF) used in the pose estimator component. The EKF equations and the implementation are stated and explained. For derivation and details please refer to (Grewal and Andrews, 2001; Welch and Bishop, 2001; Sebastian Thrun, 2006).

The Kalman Filter (KF) is a recursive algorithm with two steps: A system update (prediction) where the state estimate from the previous step is used to estimate the current state. The second step is a measurement update (correction) where updated measurements are used to update the prediction to obtain a more accurate estimate. The term recursive means that the KF does not need to process historical measurement data during the correction (Maybeck, 1979). The KF requires that the models of the system and the sensors are linear which is not the case as e.g. seen in (3.8). The EKF solves the linearity problem by continuously linearizing the models about the current state estimate.

Prediction step

Figure 3.5 illustrates the steps of the EKF algorithm, which is detailed in the following. The equations for the prediction step of a discrete EKF are listed in

(3.16) to (3.19):

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_k, \mathbf{w_k}) \tag{3.16}$$

$$\hat{\mathbf{z}}_k = \mathbf{h}(\hat{\mathbf{x}}_k^-, \mathbf{v_k}) \tag{3.17}$$

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}^-} \tag{3.18}$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\,\mathbf{P}_{k-1}^+\,\mathbf{F}_{k-1}^T + \mathbf{Q}_k \tag{3.19}$$

The robot state $\mathbf{x}$ is described by (3.1), the 2d pose of the robot with respect to the *World* coordinate system. In (3.16) the system model $\mathbf{f}$ computes the predicted state estimate $\hat{\mathbf{x}}_k^-$ using the corrected state estimate from the previous step $\hat{\mathbf{x}}_{k-1}^+$, and the system input vector $\mathbf{u}_k$ defined by the odometry output. In the FroboMind pose estimator implementation $\mathbf{f}$ is defined by (3.8) and $\mathbf{u}_k$ by:

$$\mathbf{u}_k = \begin{bmatrix} \delta d \\ \delta\theta \end{bmatrix} \tag{3.20}$$

Equation (3.17) computes the predicted measurements $\mathbf{z_k}$ as a function of the estimated robot state $\hat{\mathbf{x}}_k^-$. It is used during the correction step for updating the state estimate. In the FroboMind pose estimator implementation the sensors measure the robot state directly, so $\mathbf{h}$ is trivially defined as a direct mapping:

$$\hat{\mathbf{z}}_\mathbf{k} = \hat{\mathbf{x}}_k^- \tag{3.21}$$

KF assumes that the system and sensor models are perturbed by stochastic white noise vectors $\mathbf{w_k}$ and $\mathbf{v_k}$ which are Normal and Independent Distributed (NID) with mean $\mu = \mathbf{0}$ and covariance matrix $\mathbf{Q}$ and $\mathbf{R}$ respectively. Since the vectors are unknown they are approximated to be equal to their mean:

$$\mathbf{w} \in NID(\mathbf{0}, \mathbf{Q}) \tag{3.22}$$

$$\mathbf{v} \in NID(\mathbf{0}, \mathbf{R}) \tag{3.23}$$

The Jacobian matrix $\mathbf{F_{k-1}}$ in (3.18) linearizes the system model by computing the partial derivatives of the system function $\mathbf{f}$ about the current state estimate

$\hat{\mathbf{x}}_k^-$. In the FroboMind pose estimator implementation $\mathbf{F_{k-1}}$ is defined as:

$$\mathbf{F_{k-1}} = \begin{bmatrix} 1 & 0 & -\delta d\ sin(\theta_{k-1} + \frac{\delta\theta}{2}) \\ 0 & 1 & \delta d\ cos(\theta_{k-1} + \frac{\delta\theta}{2}) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

In (3.19) the a priori error covariance matrix for the system update $\mathbf{P}_k^-$ is computed. In the FroboMind pose estimator implementation the system state covariance matrix $\mathbf{Q}$ is updated at each system state update to represent the increasing variance on the system updates:

$$\mathbf{Q_k} = \mathbf{Q_{k-1}} + \begin{bmatrix} \sigma_d^2 & 0 & 0 \\ 0 & \sigma_d^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{3.25}$$

At each measurement update the diagonal elements of $\mathbf{Q}$ are reset to 0 if new measurements are available.

The output of the prediction step is the predicted state estimate $\hat{\mathbf{x}}_k^-$ and the corresponding error covariance matrix for the system update $\mathbf{P}_k^-$ as shown in figure 3.5.

Correction step

The equations for the correction step of a discrete EKF are listed in (3.26) to (3.29):

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k^-} \tag{3.26}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\ \mathbf{H}_k^T\ (\mathbf{H}_k\ \mathbf{P}_k^-\ \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{3.27}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\ (\mathbf{z}_k - \hat{\mathbf{z}}_k) \tag{3.28}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\ \mathbf{H}_k)\mathbf{P}_k^- \tag{3.29}$$

The Jacobian matrix $\mathbf{H_k}$ in (3.26) linearizes the sensor model by computing the partial derivatives of the measurement function $\mathbf{h}$ about the current state estimate$\hat{\mathbf{x}}_k^-$. As mentioned under the prediction step, the FroboMind pose estimator implementation $\mathbf{h}$ is a direct mapping, so $\mathbf{H_k}$ is defined as a 3x3 identity matrix.

Equation (3.27) computes the Kalman gain $\mathbf{K_k}$ to minimize the a posteriori covariance matrix $\mathbf{P}_k^+$. The measurement covariance matrix $\mathbf{R_k}$ is in the Frobo-

Mind pose estimator implementation defined as:

$$\mathbf{R_k} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{3.30}$$

In (3.28) the corrected state estimate $\hat{\mathbf{x}}_k^+$ is computed from the predicted state estimate $\hat{\mathbf{x}}_k^-$ and the measurement residual $\mathbf{z}_k - \hat{\mathbf{z}}_k$. In the FroboMind pose estimator implementation this is simply the measured state minus the predicted state estimate: $\mathbf{x}_k - \hat{\mathbf{x}}_k^-$.

Finally in (3.29) the a posteriori error covariance matrix for the measurement update $\mathbf{P}_k^+$ is computed based on the Kalman gain $\mathbf{K_k}$ and the a priori covariance matrix $\mathbf{P}_k^-$.

The output of the correction step is the corrected state estimate $\hat{\mathbf{x}}_k^+$ and the corresponding a posteriori covariance matrix $\mathbf{P}_k^+$ as shown in figure 3.5.

Initialization
The assumption that the state estimate can be represented by a Gaussian has the implication that the KF needs to be initialized with a state vector. The Gaussian is unimodal and has only one *best estimate* corresponding to the mean value. The KF is thus unable to localize in the state space without an initial state vector $\mathbf{x_0}$ and a corresponding covariance matrix $\mathbf{P_0}$ (figure 3.5).

The FroboMind pose estimator implementation uses relative coordinates with respect to Origo of the *World* coordinate system until the first absolute measurement is available. At this point the EKF is re-initialized with the GNSS based TM position and orientation, and the corresponding position and yaw variances are set according to user parameters.

3.3 Standalone PPP positioning

With an estimated accuracy of 15 m under ideal conditions the GPS Standard Positioning Service (SPS) has little use in precision agriculture and similar field robot applications. However recently newer GNSS receivers promising higher accuracy by using Precise Point Positioning (PPP) and Satellite Based Augmentation System (SBAS) have entered the market.

One example is the NEO-6P GPS module from u-blox. At a retail price of approximately 150 Euro the specifications claim an increased accuracy in static and slow moving applications. It supports raw data output as well, which allows use in RTK systems using an external computation library. This work evaluates the performance of this GPS module with respect to precision agriculture

applications in semi-structured environments, where local sensors can provide the required pose accuracy relative to crops or other entities. Long term static tests were used to test the absolute accuracy of the module. To test the module in a precision agriculture environment it was installed on a robot driving in a row crop field.

The u-blox NEO-6P GPS module evaluated in this paper contains a 50-channel single frequency (L1) receiver. Two engineering samples were used, however u-blox has confirmed that the PPP algorithms are identical with the production version. To test the NEO-6P a prototype board was designed and a linux program was written to update the NEO-6P configuration after power-up. During some tests the NEO-6P was re-configured to output at a rate of 5 Hz. In all tests in this work the NEO-6P was connected to a GNSSA200 antenna from Gutec AB (GPSWorld, 2012). In the NEO-6P field tests FroboMind was used for data acquisition. A GPS antenna signal splitter was constructed to split the signal between the NEO-6P and a consumer grade Garmin GPS60 receiver. The signal splitter has been designed to block DC voltage from one of the GNSS connectors to ensure that only one of the connected GNSS receivers supply the active GNSS antenna. The antenna signal power loss has been measured to approximately 3 dB which corresponds with the 1:2 power split between the two output ports.

3.3.1 NEO-6P static test

The purpose of this test is to evaluate the absolute accuracy of the NEO-6P in a standalone setup during variations in the satellite constellation and atmospheric disturbances. The antenna was installed at a location with no significant obstacles above the NEO-6P default elevation mask of 5 degrees. The position of the antenna was estimated by averaging a sample of 74869 RTK fixed solution measurements at 10 Hz using a DataGrid MK3 receiver and is assumed to be exact.

Figure 3.6 shows the results of a 95 hour test with the NEO-6P configured for 5 Hz output rate. 99% of the outputs returned DGPS fix indicating that SBAS was used to improve the positioning quality. 95% of the measurement errors were within 1.98 m. For the Garmin GPS60 95% of the measurement errors were within 2.60 m. A number of tests were performed with similar outcome. Tests without the antenna signal splitter resulted in slightly better results from the NEO-6P probably due to the increased signal to noise ratio (approximately 3 dB).

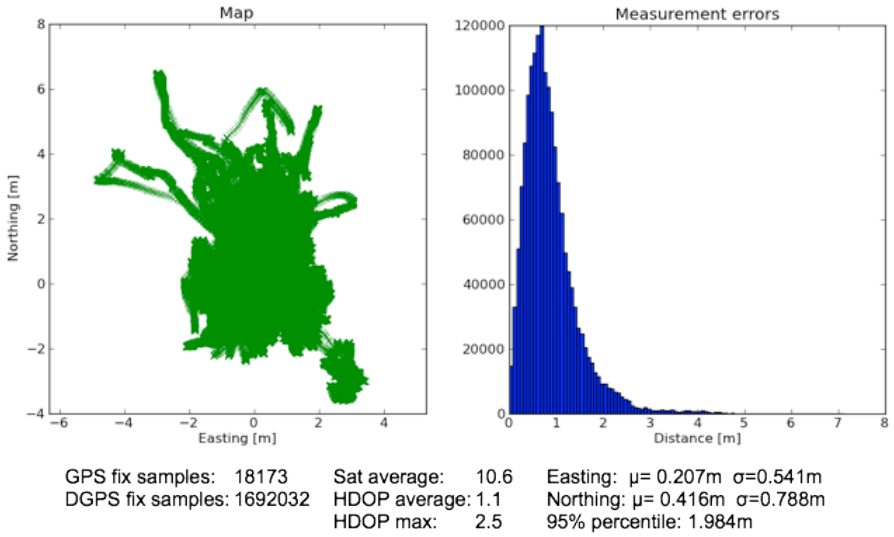


Figure 3.6: *u-blox NEO-6P 95 hour static accuracy test.*

3.3.2 NEO-6P field test

The purpose of this test is to evaluate the NEO-6P in a precision agriculture scenario. During a crop scouting trail in a row crop field performed by the Armadillo III robot (described in section 7.1.2) data was logged from the NEO-6P GPS module, and a Topcon GRS-1 was used to collect reference data. The NEO-6P antenna was mounted 0.3 m in front of the Topcon antenna. The field had no nearby obstacles shading the GNSS antenna view. The tracked distance was 1750 m which was completed in approximately 2 hours corresponding to an average velocity of 0.24 m/s.

Figure 3.7 shows the result of this trial. The green track shows the NEO-6P log and the black track below shows the reference RTK log. The histograms show the distribution of offset between NEO-6P and concurrent GRS-1 measurements. The NEO-6P antenna location 0.3 m in front of the reference antenna has not been compensated in the histograms due to lack of an accurate heading estimation at the time of the experiment. More than 95% of the measurement errors were estimated to be within 1.60 m which means that the 95th percentile is lower than 1.90 m and the maximum offset is lower than 2.75 m.

3.4 Single frequency RTK system positioning

This section evaluates the Low-cost RTK system (LCRS) that has been developed within this work. A set of tests have been performed to evaluate the LCRS

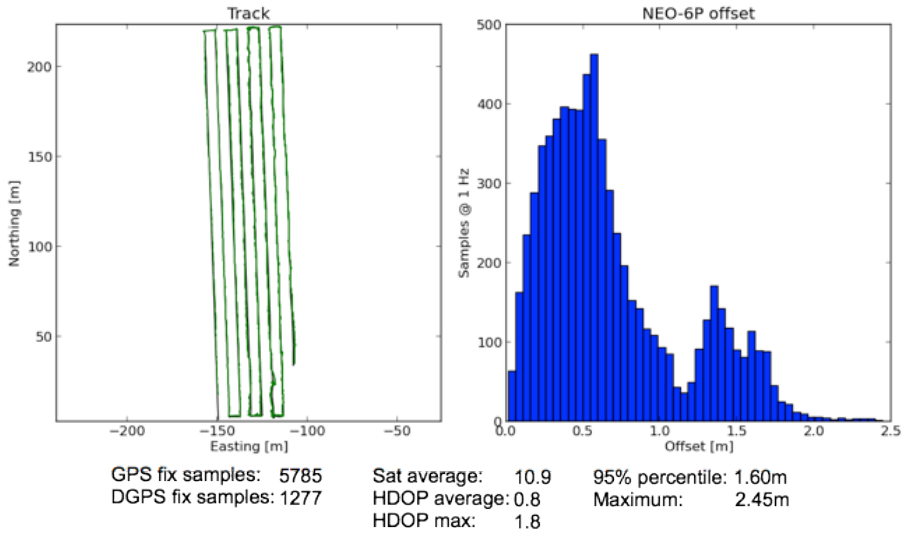


Figure 3.7: *u-blox NEO-6P field accuracy test. The green track is the NEO-6P log, the black track below is the Topcon GRS-1 log. The histogram shows the distance offset to the GRS-1 without compensating for the antenna spacing.*

performance with regards to precision agriculture and similar precision tasks.

RTK is based on measuring the phase of the satellite signal carrier wave rather than reading the content of the signal. It is performed as a relative measurement between two receivers which in this context are named the *rover* and the *reference station*. The output of RTK is the *baseline* defined by the spatial vector between the rover and the reference station which added to the position of the reference station yields the absolute position of the rover. GNSS satellites transmits simultaneously on different frequencies such as GPS L1 and L2. RTK systems available on the market today takes advantage of this to estimate the error contribution caused by delays induced in the ionosphere. This results in a higher accuracy for long baselines where the difference in ionosphere error contributions between the rover and reference station becomes more significant. Multi frequency RTK systems have the ability to maintain centimetre level accuracy at baselines up to 40 km. The maximum practical baseline for single-frequency RTK system is less than 10 km. Another advantage when using multi frequency measurements is a faster positioning. For some multi frequency systems on the market the time to first fix after startup is only a few seconds. Single frequency systems may require 30 minutes or more (Sass, 2007).

Further details about RTK is beyond the scope of this thesis and may be explored in (Bevly, 2010). Important to low-cost field robot applications is the possibility of lowering the price for an RTK system significantly by using carrier phase measurements from low-cost single-frequency GNSS receivers. This idea

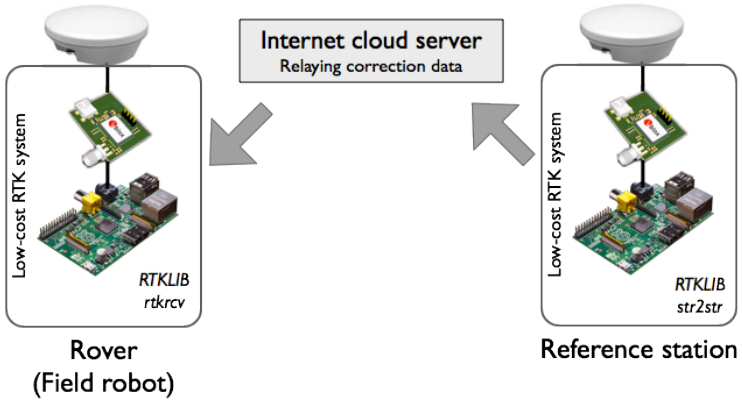


Figure 3.8: *Low-cost RTK system (LCRS). The reference station running the RTKLIB str2str transmits raw correction data through a cloud based server to the rover running RTKLIB rtkrcv.*

has been explored in the work cited in the (Freitas et al., 2012; Gomez-Gil et al., 2013; Cai et al., 2011; Codol et al., 2011). This work, which is a continuation of the work in (Jensen et al., 2012b), contains an evaluation of the performance of low-cost GNSS in field robot applications.

A LCRS designed primarily from off-the-shelf components and open source software has been developed in this work. The LCRS consists of a rover unit installed on the field robot and a reference station installed nearby (figure 3.8). The units are almost identical and each consists of a single frequency GPS receiver connected to a GPS antenna and to an embedded computer. Figure 3.9 shows a prototype of the unit. A communication medium is needed for transmitting correction data from the reference station to the rover. Wireless network (wifi) is a possibility but this limits the maximum distance between rover and reference station (baseline) to the range of the transceivers. VHF/UHF serial modems are another possibility however a license is required in most countries for equipment capable of transmitting data reliably on VHF/UHF frequencies. In this work transmission of correction data from the reference station to the rover was performed using the internet with the rover connecting using a 3G modem. An intermediate cloud based socket server was set up in order to solve the problem of establishing direct access to servers behind firewalled internet connections.

The embedded computer used in the LCRS is the Raspberry Pi (model B) which is marketed as a $35 computer. The computer is based on an ARM11, with floating point unit, running at 700Mhz. It has 512 Mb RAM and uses a SD-card for file storage. In this work Debian Linux is used as operating system. Software for configuring the u-blox modules has been added to FroboMind.

RTK GNSS positioning is performed using the open source RTKLIB v2.4.2

Figure 3.9: *A prototype Low-cost RTK (LCRS) unit consisting of a u-blox LEA-6T GPS module and a Raspberry Pi embedded computer. Position updates are received by the field robot via ethernet or a serial port.*

(T.Takasu and A.Yasuda, 2009). The reference station transmits the raw phase measurements to the rover using the RTKLIB str2str application. The rover performs the actual positioning using the RTKLIB rtkrcv application and outputs this as NMEA formatted data to the field robot.

In these tests single frequency (L1) GPS modules from u-blox are used for the LCRS. At the time of writing the LEA-6T and NEO-6P modules are the newest versions that support raw phase measurements output. NEO-6P was initially used in this work, but field tests revealed that the raw output of the NEO-6P modules was sensitive to vibrations that often occurs when driving off-road. LEA-6T modules are thus used for rover units and NEO-6P modules for reference stations in this work.

3.4.1 LCRS static test

The purpose of this test is to evaluate the ability to maintain an RTK fixed solution as well as the absolute accuracy of the LCRS during variations in the ionosphere. The reference station was equipped with a Gutec GNSSA200 antenna mounted on a location with a clear view of the sky except for a single tree about 20 m away which shades up to about 25°above the horizon. The rover was equipped with a Trimble AG15 antenna mounted few meters away from the reference antenna.

432,000 positions were sampled at a rate of 5 Hz corresponding to a measurement period of 24 hours. 410,951 positions were based on RTK fixed solution and 21,049 were based on RTK float solution. Average number of satellites were 8.6, HDOP was reported to 1.0 for all measurements. 95% of the measurements

were within 0.022 m from the estimated midpoint. Figure 3.10(a) shows a map of the positions. (0,0) is the estimated midpoint, blue represents fixed solution and red represents float solution. Figure 3.10(b) shows the error of the positions with respect to the estimated midpoint. Figure 3.11(a) (Top) shows the solution during the 24 hour measurement period. 4 corresponds to an RTK fixed solution, 5 corresponds to an RTK float solution. (Bottom) shows the number of satellites during the 24 hour measurement period. Figure 3.11(b) shows the duration of the periods with absence of an RTK fixed solution. The test was performed two times with comparable results.

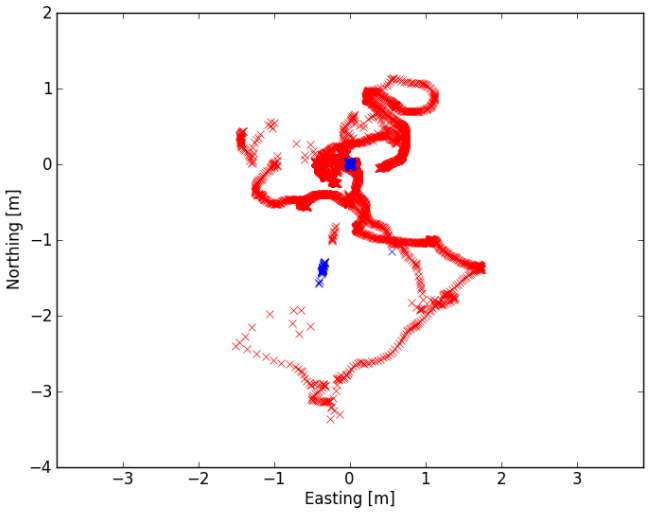
The results show that for RTK fixed solutions the accuracy is high except for a small group of positions with an error of approximately 1.5 m which might be an error caused by RTKLIB. These positions would be very difficult to distinguish due to the RTK fixed solution. One possibility would be to look at each period after a change of satellite configuration, another would be to look at the variance of the RTK algorithm used in RTKLIB. The numerous and quite lengthy periods with degraded accuracy due to RTK float solution presented in figure 3.11(b) are somewhat surprising. Measures were taken to ensure that this was not caused by an erroneous test setup. The building roof below the reference and rover antennas is made of metal, but aside from this there are no apparent sources of error. In field robot applications this would lead to periods in which the robot must stop the task and wait for RTK fixed solution.

3.4.2 LCRS field test with a good view of the sky

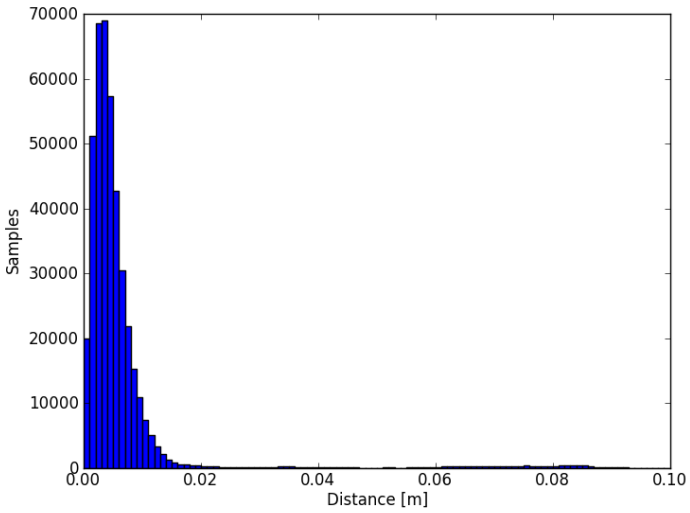
The purpose of this test is to evaluate the absolute accuracy and reliability of the LCRS under field conditions with a good view of the sky. Dynamic tests methods described in the literature typically use a railed test track to ensure an optimal ground truth reference (Stombaugh et al., 2008; Taylor et al., 2004). However this test is performed with the rover unit installed on a field robot, where it is exposed to uneven driving, vibrations etc. The test track is defined by 4 waypoints that forms a square sized 12x12 m similar to the odometry test described in section 3.2.2. The baseline to the farthest corner is 200 m. Figure 3.12 shows the view from the site.

The Pichi robot (described in section 7.1.4) was used in this test. In order to achieve as accurate reference data as possible a Trimble Zephyr Geodetic II antenna is mounted at the geometric center of the robot. A GPS antenna signal splitter developed in this work was used to allow the LCRS and a Trimble BX982 GNSS receiver to receive the same signal. The Trimble output is used as ground truth reference.

The test was performed two times with comparable results. In both tests all positions were based on RTK fixed solution, however the static test in section

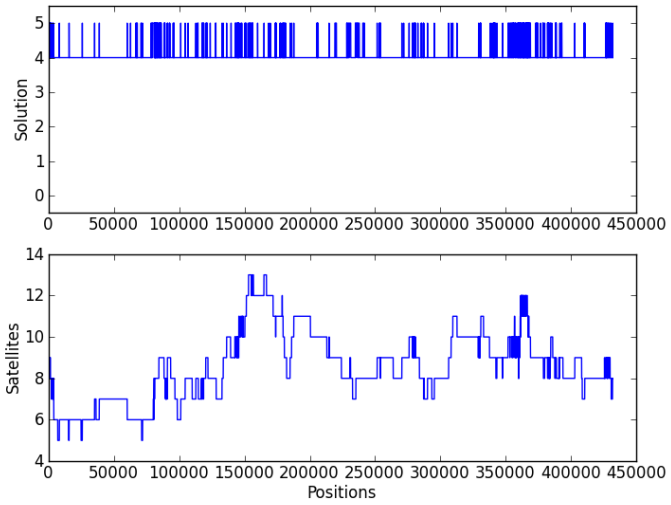


(a) Measured positions: RTK fixed solution (blue), RTK float solution (red). (0,0) is the estimated midpoint

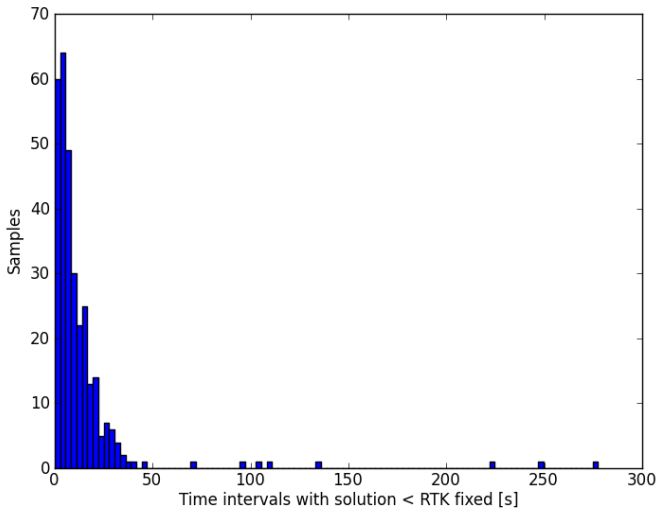


(b) Position error histogram. 7049 measurements (1.6%) have been excluded from the histogram due to an error > 0.10 m.

Figure 3.10: *LCRS static test: Position map and error histogram.*

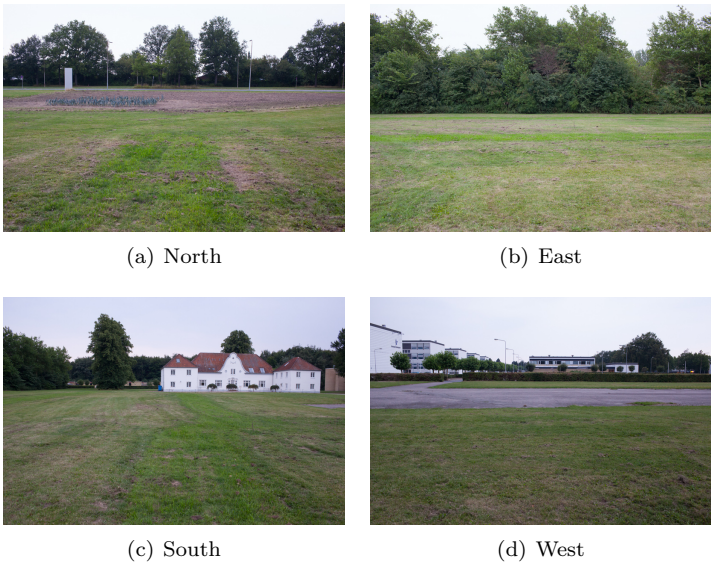


(a) RTK solution and number of satellites. RTK fixed solution = 4, RTK float solution = 5



(b) Number of time intervals with absence of RTK fixed solution.

Figure 3.11: *LCRS static test: RTK solution and time intervals.*



(a) North (b) East

(c) South (d) West

Figure 3.12: *Test field with a good view of the sky. The obstacles closest to the track are the trees to the east which are just below the configured 10° elevation mask. The images are taking using a 35mm lens.*

3.4.1 as well as experience from similar field trials shows that this is not always the case when using the LCRS. Periodic degradation of the accuracy to RTK float solution is to be expected even in areas with an unobstructed view of the sky. Figure 3.13 shows the track from the GPS (blue), the estimated pose based on GPS and odometry (red) and the reference GNSS (black). In this figure they overlay each other, and assuming that the reference GNSS receiver is accurate this indicates a high accuracy for the LCRS as well. Figure 3.14 quantifies the accuracy using the positioning difference between the LCRS and reference GNSS receiver. The histogram, which is generated using algorithm 2, shows an interesting pattern: The differences are grouped in two fractions, and based on a comparison with 3.15 and results from the static test it was assumed that this relates to timing differences when driving ahead vs. turning:

A review of the NMEA data received from the GNSS receivers by the FroboMind serial string component revealed a computing delay $\Delta t \approx 0.19s$ for data from the LCRS compared to data from the reference GNSS receiver. Two simple tests were designed to validate this observation. In a static test data was sampled for 214 s while the Pichi robot stood still. The result is shown in figure 3.15(a). The average $\widehat{\mu}_{static} = 0.0062m$ and the 95th percentile: $P95_{static} = 0.011m$. This corresponds well with the results of the static test. In a dynamic test the Pichi robot was driving 29.9 m in 79.2 s at an average velocity $v_{avg} = 0.38m/s$. The result is shown in figure 3.15(b), the average $\widehat{\mu}_{dynamic} = 0.080m$ and the 95th percentile with respect to distance from $\widehat{\mu}_{dynamic}$: $P95_{dynamic} = 0.034m$.

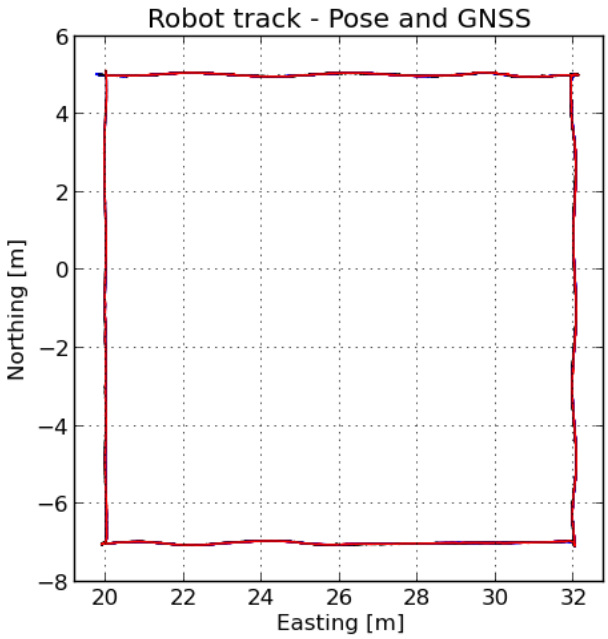


Figure 3.13: *Difference between estimated pose (red), LCRS (blue) and reference GNSS (black). In this test the tracks overlap each other almost exactly.*

The average $\widehat{\mu}_{dynamic} = 0.080m$ represents a measure of the observed position offset when driving. In (3.31) the theoretical position offset Δs is calculated to $0.076m$, and it is reasonable to conclude that the greatest contribution to the position offset is caused by the computing delay Δt.

Based on this it is concluded that the accuracy of the LCRS is comparable to the results from the static test. There is however a significant computing delay Δt on the position output which should be taken into account when fusing the output with other sensors to estimate the robot pose. Support for this has not yet been included in the pose estimator in this work.

Algorithm 2: Generating GNSS difference histogram for figure 3.14

input : LCRS and reference GNSS position buffers (TM)
output: Position difference buffer

begin
  for *pos = each LCRS position update* **do**
    ref = latest received reference GNSS update
    difference = $\sqrt{(e_{pos} - e_{ref})^2 + (n_{pos} - n_{ref})^2}$

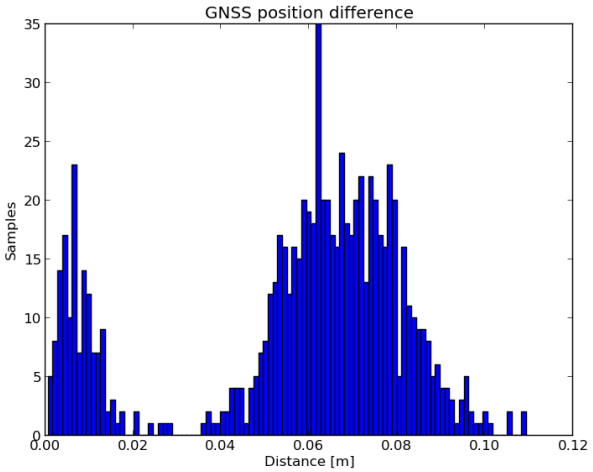


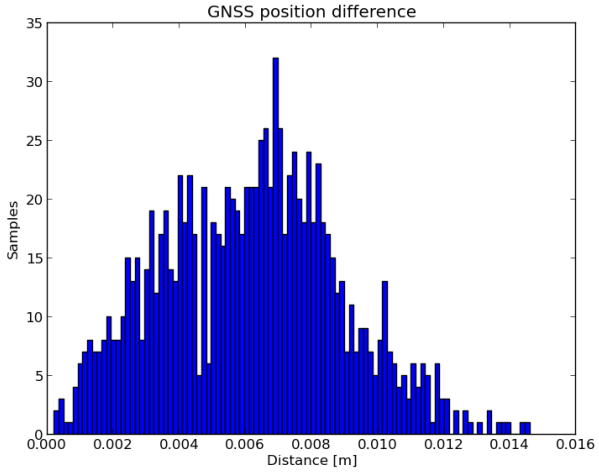
Figure 3.14: *Difference between low-cost RTK GNSS (LCRS) and reference GNSS.*

$$\begin{aligned} \Delta s &= \Delta t * v_{avg} \\ &= 0.20\ [s] * 0.38\ [\frac{m}{s}] \\ &= 0.076\ [m] \end{aligned} \tag{3.31}$$

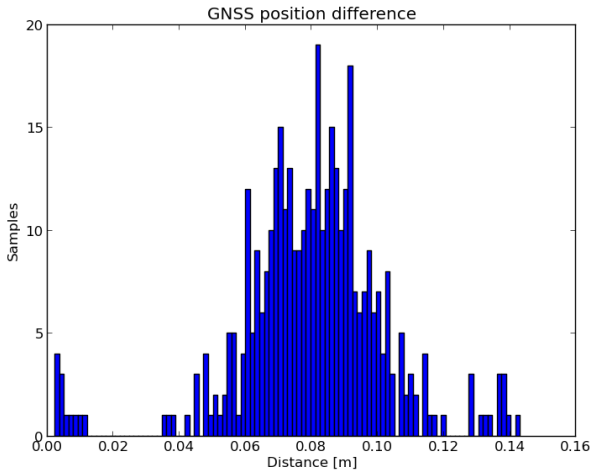
3.4.3 LCRS field test with a limited view of the sky

The purpose of this test is to assess the usefulness of the LCRS under field conditions with a limited view of the sky. The site representing a worst case scenario is a typical detached house garden environment surrounded by dense bushes and high trees and next to the house with metal roof. Figure 3.17 shows the view from the site. For this test the FroboMower robot described in section 7.1.5 is used. The test consisted of 4 trials spread evenly across a period of 24 hours. In each trial a rectangle sized 5x4 m was navigated manually 4 times. The FroboMower was equipped with a Trimble AG15 antenna installed 0.9 m above the ground. Table 3.1 lists the approximate costs of the FroboMower navigation sensors (IMU and odometry results are not used in these tests). The reference station was equipped with a Gutec GNSSA200 antenna.

In this test positions based on RTK float solution were dominant though there were occasional changes to RTK fixed solution. Figure 3.18(a) shows the smallest spread for the four squares in a single trial ($<$ 10 minutes). Figure 3.18(b) shows the largest spread for the four squares in a single trial. Figure 3.18(c)



(a) Static test



(b) Dynamic test

Figure 3.15: *Static and dynamic tests to validate GPS data delay.*

Sensor	**Price**
LEA-T GPS board	$150 (ebay)
Trimble AG15 GPS antenna	$300 (ebay)
Raspberry Pi model B	$35
SparkFun Razor IMU B	$125
2 pcs Yumo rotary encoders	$80
Total	*$690*

Table 3.1: *Approximate cost of the FroboMower navigation sensors.*

Figure 3.16: *The FroboMower robot.*

(a) North (b) East

(c) South (d) West

Figure 3.17: *Photos of the test field with a limited view of the sky taken using a 24mm lens.*

shows the largest spread across the 24 hour test. The highest absolute inaccuracy observed within the 24 hour test is at the order of 2 m. Together the tracks in figure 3.18 indicate that the relative accuracy within short time frames is quite high in most cases.

3.5 Evaluating the FroboMind global pose estimation

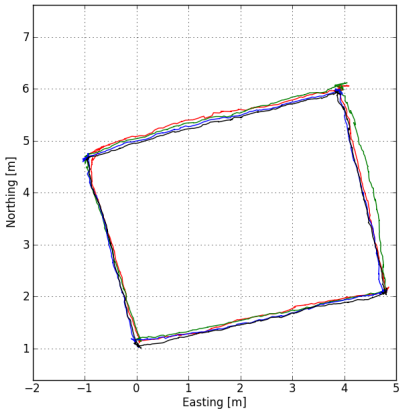
In this section the FroboMind global pose estimation is tested and evaluated with respect to the purpose hereof stated at the beginning of the chapter.

3.5.1 Absolute position estimation

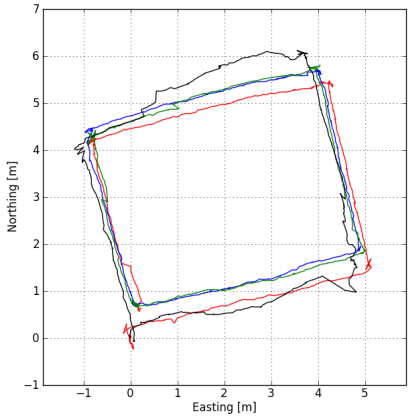
A test was defined to evaluate the accuracy of the estimated absolute positions. The accuracy is a result of two factors: 1) The accuracy of the odometry updates is often higher than the accuracy of the RTK-GNSS updates, so the EKF will to some extent be able to mitigate the effects of the Gaussian noise embedded in the GNSS updates. 2) The higher update rate yields more accurate position information when the robot is in motion. Say the GNSS updates at 5 Hz and the pose estimation updates at 50 Hz. At a velocity of 2 m/s the distance between updates are 0.4 and 0.04 meter respectively. This makes a significant difference in precision agriculture and similar precision tasks.

The test is not exhaustive but it does provide a reasonable assessment of the accuracy. In the test the FroboScout robot described in section 7.1.6 was configured to drive at an approximate speed of 0.9 m/s. The odometry is based on wheel encoders and the yaw rate gyro in a VectorNav VN-100 IMU. The GNSS used is a Trimble BX982 receiver and a Trimble AG25 antenna. The update rates are 50 Hz for the encoders, 40 Hz for the gyro and 10 Hz for the GNSS. The standard deviation for RTK fixed solution position updates was set to $\sigma = 0.02m$.

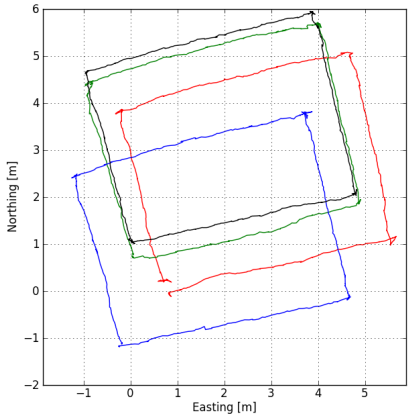
Figure 3.19 shows an example of the temporal updates from the pose estimator and the GNSS. Even though the GNSS updates appear to be very accurate, the plots show that the pose estimator filter behaves as expected. It updates the current pose estimate at a rate of at 50Hz based on odometry updates. Whenever an absolute position update from the GNSS is available, the correction step of the EKF updates the pose estimation towards this position.



(a) The trial with the 4 tracks closest to each other.



(b) The trial with the 4 tracks most distant from each other.



(c) Most distant tracks from all 4 trials.

Figure 3.18: *Tracks from the FroboMower robot navigating a 5x4 m square. The experiment consisted of 4 trials spread evenly across a period of 24 hours. At each trial the track was navigated 4 times.*

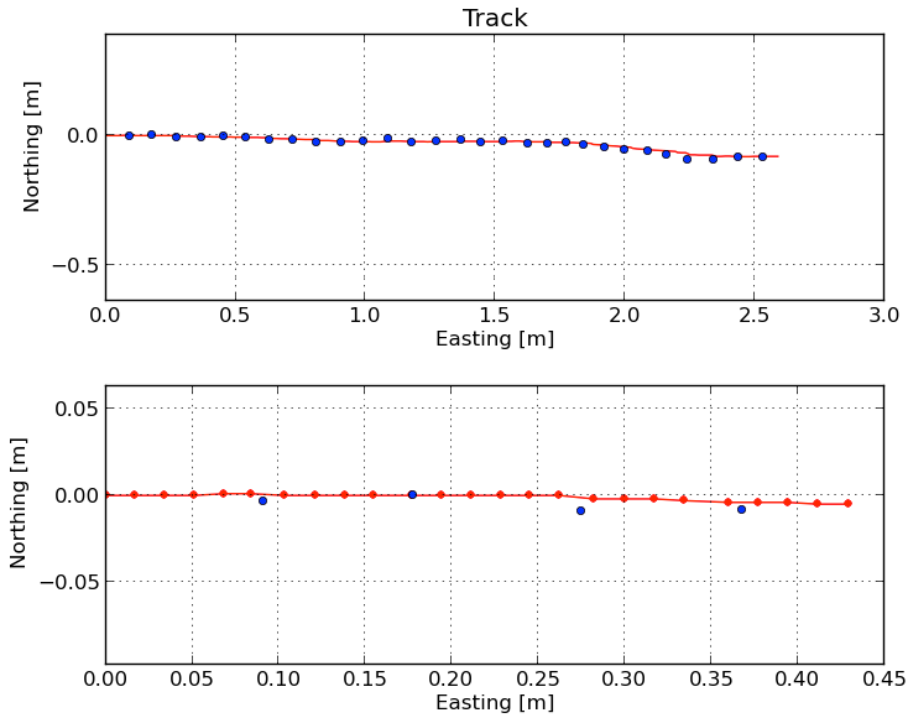


Figure 3.19: *The plots show the temporal progress of the frequent pose estimator extrapolation updates (red dots) and infrequent absolute GNSS updates (blue dots). The top plot shows a 3 s sequence from a track recorded by the FroboScout robot. Relative coordinates are displayed so the track begins in* $x = [0, 0, 0]^T$*. The bottom plot shows a detailed view of the first 0.5 s.*

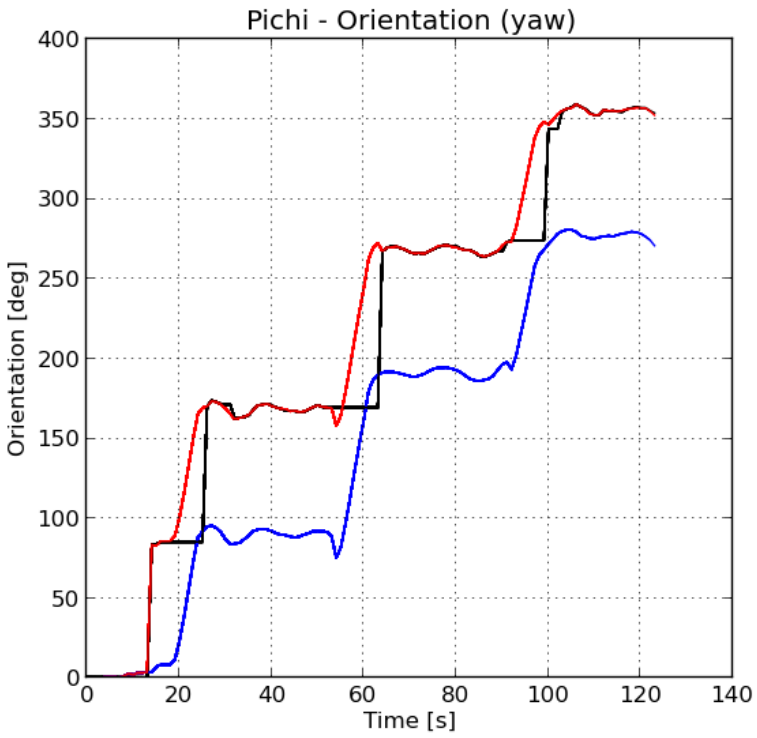


Figure 3.20: *Orientation estimation (red) based on frequent relative updates from an IMU (blue) and infrequent absolute orientation updates based on GNSS measurements (black). The plot shows the Pichi robot making 3 turns approximately 90° each followed by forward motion with some fluctuations in the orientation.*

3.5.2 Absolute orientation estimation

In this test the accuracy of the estimated absolute orientations is evaluated similarly to the estimated absolute positions above. For the test the Pichi robot was used. As described in section 7.1.4 the first prototype of the Pichi robot had difficulties manoeuvring smoothly due to problems with the motors and transmission. This is utilized in the test to stress the pose estimator which requires the robot to drive straight and smoothly ahead in order to update the orientation estimation. The odometry is based on feedback from motor hall elements and the yaw rate gyro in a VectorNav VN-100 IMU. The LCRS was used for this test along with a Trimble AG15 antenna. The update rates are 50 Hz for the encoders, 40 Hz for the gyro and 5 Hz for the GNSS.

Figure 3.20 shows robot orientation as a function of time, the trial duration is approximately 2 minutes. Immediately after initialization no absolute orientation updates is available from the GNSS yet. The pose orientation is therefore

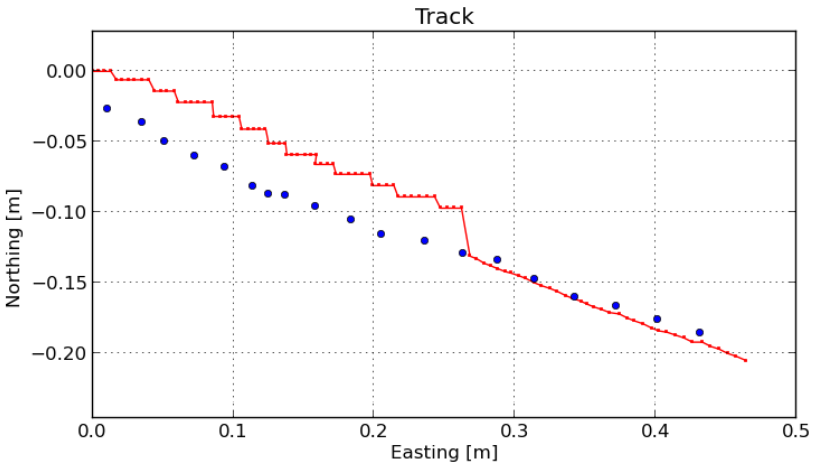


Figure 3.21: *Pose estimator extrapolation (red dots) of infrequent absolute GNSS updates (blue dots). The plot shows approximately 1 s before and after the first absolute orientation update. The corrected orientation improves the subsequent EKF predictions significantly.*

equal to the odometry orientation which assumes that the robot is oriented in the direction of the x-axis in the *World* coordinate system. After approximately 14 s the first absolute orientation update is available, and the pose estimation is therefore corrected by the EKF correction step to approximately 80°. Hereafter follows a 90° turn during which period no absolute orientation updates are available according to algorithm 1. During turning the absolute position therefore remains at approximately 80° while the EKF prediction step continuously updates the pose orientation estimation based on the odometry input. After completion of the turn the robot begins to move forward albeit with some fluctuations in the direction. As soon as the conditions in algorithm 1 are met, another absolute position update is available which the EKF uses in the correction step to adjust the pose orientation output. During the forward motion the absolute orientation updates are frequent, and when making another 90° turn the pattern repeats itself.

The accuracy of the absolute orientation estimation is essential to the accuracy of the absolute position estimation. If the orientation is off, it will cause the position update computed by the EKF prediction steps to diverge away from the true absolute positions. The EKF correction steps will reduce the position errors to some extent but the EKF input variance for the odometry positions is lower than the variance for the GNSS positions so the problem remains. This is examplified in figure 3.21 which origins from the same trial data as figure 3.19. This track shows the first 2 s of motion of the robot after startup. The pose orientation estimation is equal to the odometry orientation estimation until the first absolute orientation update after about 0.3 meter which causes a re-

initialization of the EKF. Hereafter the EKF pose estimation converges.

3.5.3 Pose availability during GNSS signal degradation or outage

To test the pose availability during short periods of GNSS signal outage a test was performed on the dataset presented in figure 3.15. A sequence of 30 s of GNSS data was removed from the dataset forcing the EKF to predict only using the odometry updates. The sequence was removed from the NE corner of the square to observe the performance during both driving straight and turning.

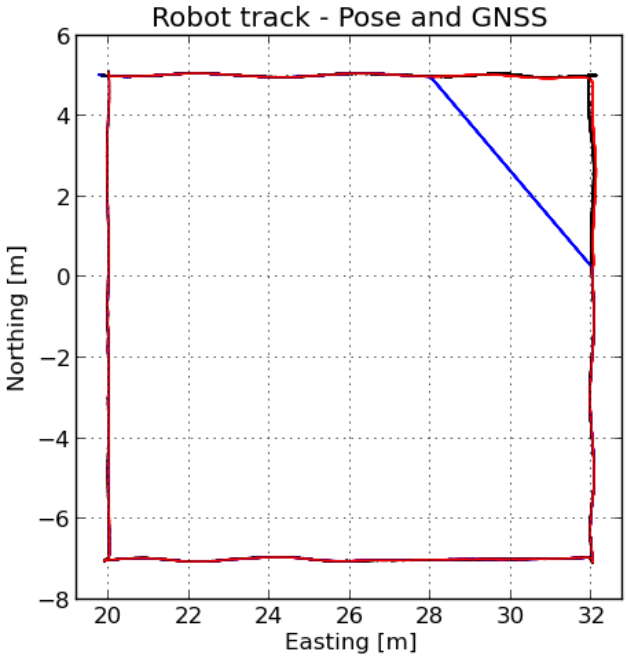
The result is shown in figure 3.22(a). Figure 3.22(b) shows a zoom of the NE corner. It indicates that the pose estimator outputs reasonably good pose updates based on the available odometry updates during the GNSS outage.

3.6 Discussion

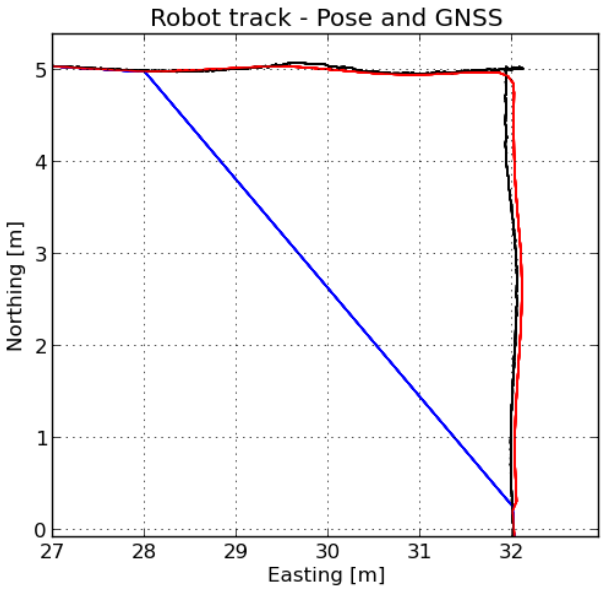
Though standalone PPP positioning appears to perform better than standard SPS positioning, the tests of the NEO-6P GPS module indicate that it has limited use in precision agriculture. If the required positioning accuracy is higher than 2-3 meter then PPP positioning is not a feasible solution yet. There are, however, certain applications where the low price and no need for access to a reference network can offset the high inaccuracy. One example is orchard or vineyard field robots relying on local and inertial sensors for the accurate lateral in-row positioning. For localization of the robot beyond the in-row positioning the PPP positioning could be a valuable sensor input to e.g. a particle filter. In addition to the described trials the NEO-6P GPS module has been exposed to antenna oscillations and vibrations on the ASuBot robot (described in section 7.1.1). These tests gave no reason to review the above conclusion.

Single frequency RTK system positioning seems much more attractive to precision agriculture and similar precision applications for field robots. Specifically in applications with multiple unsupervised robots where the hardware cost is more important than the duration of the task. The LCRS based on u-blox GPS modules and the open source software RTKLIB was designed and constructed in this work, and tests were performed to evaluate the performance of the LCRS.

In a static test the LCRS rover antenna was installed at a fixed position with a good view of the sky. For RTK fixed solutions the accuracy is within the expected except for a small group of positions with an error of approximately 1.5 m. It should be considered how this group may be filtered to avoid undetectable positioning offsets. The test results show numerous and quite lengthy periods with degraded accuracy due to RTK float solution which is somewhat surprising.



(a) Simulated GNSS outage at NE corner



(b) Zoomed in on GNSS outage

Figure 3.22: *Simulating a 30 s GNSS outage. The robot begins driving at the NW corner heading E. The GNSS track is blue, the sloping line indicates the position jump due to lack of updates. The pose estimation track is red and the reference GNSS track is black.*

In field robot applications this would lead to periods in which the robot must stop the task and wait for RTK fixed solution.

In a field test with a good view of the sky the LCRS rover was installed on the Pichi robot and a test track defined by a square sized 12x12 m on a grass field. The baseline to the farthest corner was 200 m. All LCRS output positions were based on the RTK fixed solution, however periodic degradation of the accuracy to RTK float solution should be expected when using the LCRS in field environments. The test results show that the accuracy of the LCRS when installed on a driving field robot is comparable to the accuracy obtained in the static test. However the LCRS has a computing delay on the position output at the order of 0.19 s which depending on the robot velocity may induce a significant higher inaccuracy if this is not supported by the pose estimation.

The field test with a limited view of the sky shows that only RTK float solution can be expected in such environments. This is not sufficient for precision tasks, but like PPP positioning it may be useful for applications like mowing or navigation in orchards or vineyards using local sensors as the primary means of navigation.

The FroboMind global pose estimation was described and evaluated with respect to improving accuracy and providing pose updates at a higher frequency than the GNSS, and increase the availability of pose estimation updates during GNSS signal degradation and outage.

Though not exhaustive the tests indicate that the global pose estimation performs well under the described conditions. There is, however, not yet support for precise positioning in hilly or sloping fields. The tests provide a good basis for assessing the performance of the global pose estimation for a particular implementation of FroboMind on any given field robot. In practice, they have been performed for all the robots described in this work which navigate autonomously. For instance the position test will reveal any discrepancy between the odometry estimation as the pose estimator

The accuracy of the pose estimation during periodic GNSS signal outage was evaluated by removing a sequence of 30 s GNSS positions from one a trial log and then recomputing the pose estimation. The pose estimator outputs reasonably good pose updates based on the available odometry updates during the simulated GNSS outage.

The accuracy of the pose estimation during periodic degradation of the GNSS signal accuracy due to RTK float solution state was not evaluated. The reason is that the field trials performed in this work have revealed that it makes no sense using the RTK float solution measurements for the estimation in applications requiring cm level precision. In principle EKF can utilize any measurement as long as the variance is known, but the problem with RTK float solution positions is that the noise does not appear to be Gaussian and therefore it

does not comply with the requirements stated in (3.23). RTK float solution measurements tend to "wander" around the true position while the GNSS is trying to resolve the integer ambiguities, so the measurements can thus not be considered stochastically independent either. Figure 3.10 and 3.18 illustrate this well: Even though the RTK float solution causes deviations between the trials of up to 2 m, the individual trials appear quite similar because GNSS updates are close to each other with regards to time.

In applications where an accuracy of the pose estimation updates within a few meters are adequate to the task it makes good sense to include the RTK float solution measurements. To accomodate this the FroboMind pose estimator component supports a launch parameter specifying the required solution quality of the GNSS updates.

Compared to the related work described in section 3.1 the design and algorithms of the FroboMind global pose estimation components described in this chapter are not novel in itself. It is, however, a good example of the problem stated in section 2.1: As fundamental as global pose estimation is in many aspects of outdoor mobile robotics, the published work focus on findings and results only, the pose estimation software is not published. The main contribution from this work is thus the publication of the implemented and field tested global pose estimation components and an evaluation of the performance of low-cost GNSS in field robot applications.

3.7 Conclusion

In this work FroboMind components for global pose estimation have been developed. The pose estimation is based on output from robot odometry, an Inertial Measurement Unit, and GNSS positions. The current implementation can be improved in some respects but until now focus has been to provide a reliable pose of a reasonable quality for the current experiments using FroboMind.

Global pose estimation using low-cost, single-frequency GPS receivers and the open-source RTKLIB has been tested under field conditions. Standalone Precise Point Positioning algorithm has limited use in field robotics. Single-frequency RTK provides accuracies comparable to much more RTK GNSS solutions and has a huge potential because of the low cost. However a problem with frequent drops to float solution needs to be resolved.

The main contribution from this work is thus the publication of the implemented and field tested global pose estimation FroboMind components and an evaluation of the performance of low-cost GNSS in field robot applications.

Chapter 4

Autonomous navigation

This chapter deals with autonomous navigation of field robots. The architecture and components used for autonomous path following are presented and evaluated in field tests.

4.1 Introduction

Similar to the global pose estimation described in chapter 3 autonomous navigation in a field environment is a challenging task: The navigation rely on accurate pose estimations which are based on noisy sensor measurements, the low-level controllers and actuators are not ideal meaning that commanded velocities are not executed quickly and accurately. Finally the field environment introduces inaccuracies to both systems.

In this work navigation concerns following a path (route) which is either static and planned prior to the mission or updated dynamically. The path may be either a continuous function consisting of segments of lines and arcs, or it may be discrete consisting of a list of waypoints.

Figure 4.1 illustrates the principle a path following (tracking) controller. Path following controllers are based on either kinematic or dynamic models of the robot which usually has non-holonomic constraints such as differential or Ackermann steering. The majority of the path following controllers are based on *pure pursuit*, a path following algorithm that works by calculating the curvature that will move the robot from it's current position to some goal position (Coulter, 1992). By choosing the goal position to be at a distance ahead of the robot on the desired path, the robot will follow the path to the best of it's ability.

Most path following controllers based on pure pursuit that are described in the literature, focus exclusively on ensuring that the robot reaches the goal without

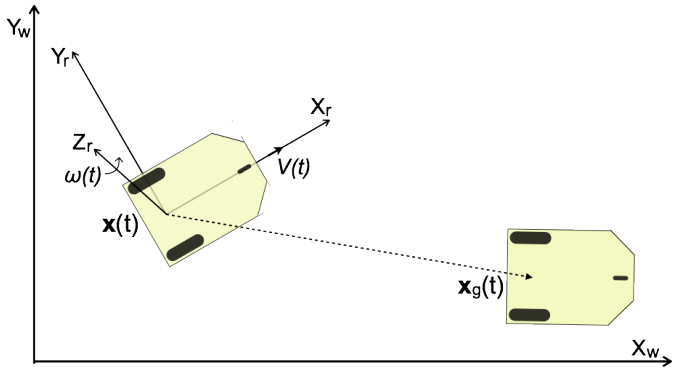


Figure 4.1: *The principle of path following: Let* $\mathbf{e}(t) = \mathbf{x_g}(t) - \mathbf{x}(t)$ *be the robot pose error with respect to the desired goal* $\mathbf{x_g}(t)$ *at the time* t*. The objective of the path following controller is then to command velocities* $V(t)$ *and* $\omega(t)$ *such that* $\lim_{t \to \text{inf}} \mathbf{e}(t) = \mathbf{0}$.

concerns about how it gets there. Examples of such path following controllers are described in (Siegwart et al., 2011, section 3.6), (Fahimi, 2009, chapter 6) and (Kozlowski, 2011, chapter 5).

Many of the tasks in precision agriculture involve navigation in row crop fields and orchards where the path consists of long straight segments along the crops connected by tight turns at the headland. Path following along the rows are constrained by the robot to be within a maximum lateral distance to the path and at the same time be oriented along the path at a maximum angle error. The literature contains some references to autonomous path following along crops and on headlands. Examples are (Jingtao and Taochang, 2014) who designed a navigation controller for a tractor consisting of a Proportional-Derivative (PD) path following controller and a PD steering controller based on a kinematic model. Similarly (Bakker et al., 2011) designed a path following controller for an agricultural research robot based on double Ackermann steering. The lateral error and heading to the path were controlled by two Proportional-Integral (PI) controllers resulting in low level control set points for wheel angles and wheel speeds. (Li et al., 2009) lists other examples of autonomous guidance systems for agricultural vehicles and (Noack and Muhr, 2008) discusses the available commercial products.

The FroboMind architecture supports the implementation of different path following controllers and motion controllers but inherently the implementation and testing of these controllers is a major task. The scope of this work is therefore limited to a path following controller for differential steered robots navigating a list of waypoints. Collision avoidance based on local obstacle detection is not included in this work.

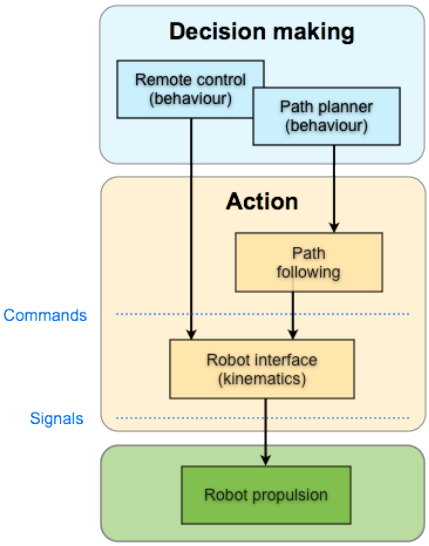


Figure 4.2: *FroboMind navigation architecture for a typical GNSS-based application. The architecture illustrates two different behaviours of which only one is active at any time.*

4.2 FroboMind navigation

In this section the FroboMind action layer architecture with regards to autonomous navigation is described.

Figure 4.2 shows a generalized version of the FroboMind example architecture presented in section 2.3 figure 2.5. The mission planner ensures that only one motion behaviour is active at a time, so in this case the robot will be either commanded directly from the HMI remote control or by the path planner though the path follower.

The command interface for robot motion is defined as:

$$\mathbf{V_{cmd}} = \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{4.1}$$

where V and ω are the linear and angular velocities of the robot respectively. This is in line with the ROS standard for commanding velocity messages, however in FroboMind the ROS *TwistStamped* message is used to ensure time stamps on all messages.

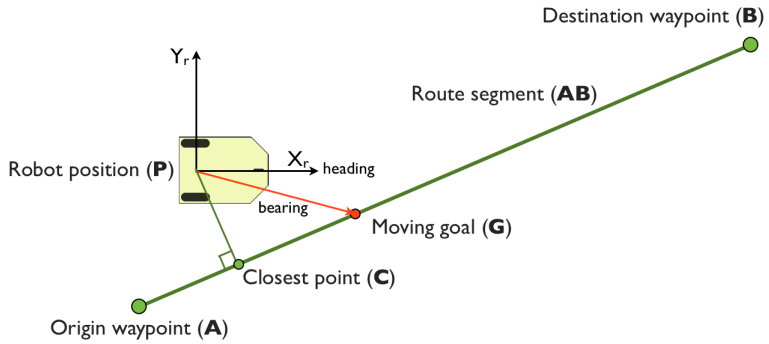


Figure 4.3: *The robot aims towards the moving goal* **G** *which is pushed ahead of the robot along the line* **AB**.

4.2.1 Path following

In this section the FroboMind path following component is described. This component navigates a path plan defined as a list of waypoints. The output of the path following component is a commanding velocity $\mathbf{V_{cmd}}$ defined in equation (4.1). Computation of the angular and linear velocity as well as waypoint arrival is described below.

4.2.1.1 Angular velocity

The robot seeks to navigate along a straight line between the destination waypoint **B** and origin waypoint **A** of the particular path segment (figure 4.3) rather than heading directly towards the destination waypoint. To keep as close to the **AB** line as possible, a moving goal **G** is introduced. **G** is located on **AB** ahead of **C** which is the point on **AB** that is closest to the robot position **P**.

G is defined by a piecewise function depending on **P** (equation 4.2): If the robot is behind the **AB** line it first navigates towards **A**. While parallel to the line it navigates towards the moving goal **G**. The goal distance from **C** to **G** is defined by a constant g until the distance from **C** to **B** becomes less than $2g$. The goal distance then becomes half the distance to **B**. If the robot somehow ends up passing **B** outside the acceptable waypoint arrival threshold, it will navigate directly towards **B**. This is implemented as a slightly modified version of the *pure pursuit* algorithm proposed in (Coulter, 1992). The path following update algorithm for determining the angular velocity ω at each time step is detailed in algorithm 3.

Algorithm 3: Estimating the robot angular velocity

input : Current robot pose $\mathbf{x}$, waypoint_list
output: Angular velocity ω

begin
- $\mathbf{P}$ = get_position($\mathbf{x}$)
- *heading* = get_heading($\mathbf{x}$)
- **if** *waypoint_arrival(*$\mathbf{B}$*) == True* **then**
  - $\mathbf{A} = \mathbf{B}$
  - $\mathbf{B}$ = get_next_waypoint(waypoint_list)
- Find $\mathbf{C}$ along $\mathbf{AB}$ closest to $\mathbf{P}$
- Compute goal point $\mathbf{G}$
- Compute *bearing* from $\mathbf{P}$ to $\mathbf{G}$
- heading_error = signed_angle_difference(*heading*, *bearing*)
- ω = compute_pid(heading_error)
- **return** ω

$$\mathbf{G} = \begin{cases} \mathbf{A} & d \leq 0 \\ \mathbf{A} + (d+g)\widehat{\mathbf{AB}} & 0 < d < ||\mathbf{AB}|| - 2g \\ \mathbf{P} + (d + \frac{||\mathbf{PB}||}{2})\widehat{\mathbf{AB}} & ||\mathbf{AB}|| - 2g \leq d < ||\mathbf{AB}|| \\ \mathbf{B} & d \geq ||\mathbf{AB}|| \end{cases}, \tag{4.2}$$

$$\mathbf{AB} = \mathbf{B} - \mathbf{A},$$

$$\widehat{\mathbf{AB}} = \frac{\mathbf{AB}}{||\mathbf{AB}||},$$

$$d = \mathbf{P} \cdot \widehat{\mathbf{AB}} - \mathbf{A} \cdot \widehat{\mathbf{AB}}$$

4.2.1.2 Linear velocity

The commanding linear velocity V is defined by parameters: The waypoint list may define a velocity $V_{desired}$ for each waypoint. If this waypoint velocity is undefined, $V_{desired}$ is set according to a default parameter velocity.

The velocity $V_{desired}$ is maintained during navigation, however at the distance d_{ramp} to the origin waypoint $\mathbf{A}$ and the destination waypoint $\mathbf{B}$ the velocity is ramped:

$$V = V_{desired} - (1 - \frac{d_{robot}}{d_{ramp}})\ (V_{desired} - V_{min}) \tag{4.3}$$

The parameter V_{min} specify the minimum velocity of the robot. d_{robot} is the smallest distance from the robot to the origin waypoint $\mathbf{A}$ and the destination

waypoint **B**.

In future work this linear ramp should be replaced by a sigmoid curve to allow smooth transitions. The ramping close to **A** should be based on time instead of distance. The current solution may cause the robot to remain stopped if the parameter V_{min} is set below the lowest possible velocity of the robot.

4.2.1.3 Waypoint arrival conditions

At each update of the algorithm 3 it is tested whether the robot has arrived at the waypoint. In the current implementation the robot has arrived at the destination waypoint **B** if the robot pose **x** is within a defined threshold distance of the waypoint and one of the following conditions are met:

1. The distance from the robot pose **x** to the waypoint **B** is ≤ 0.02 m which corresponds to the approximate inaccuracy of the RTK GNSS fixed solution.
2. The distance from the robot pose **x** to the waypoint **B** has increased since the previous time step meaning the robot will never get closer without turning.
3. The angle difference between the current and previous bearing towards the destination waypoint **B** is greater than $\pi/8$ meaning that the robot is just passing the waypoint at a close proximity.

4.2.2 Robot interface

With regards to navigation the purpose of the *Robot interface* is to execute the commanded velocity (4.1) as quickly and accurately a possible. The translation of (4.1) into actuator controlling and hence robot motion is based on the kinematics of the robot.

Most of the current field robots using FroboMind (described in chapter 7) are either differential or skid steered. In the following the differential steered Frobit robot described in section 7.1.3 is therefore used as an example for the derivation of the forward and inverse kinematic equations.

4.2.3 Kinematics

The derivation of the kinematic equations is based on the following references which also include kinematic equations for robots using other steering principles (Dudek and Jenkin, 2010; Siegwart et al., 2011). The *Bicycle model* often used for traditional tractors with Ackermann steering is available in (Rovira Mas et al., 2010, section 2.2).

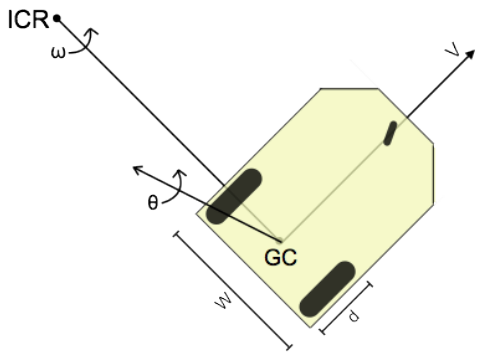


Figure 4.4: *Frobit robot with wheel diameter d and wheel distance W. The robot is turning about the Instantaneous Center of Rotation (ICR) at linear velocity V and yaw axis angular velocity ω*

4.2.3.1 Forward kinematics

In the following the *forward kinematic* equations for a differential steered vehicle are derived. They describe how the velocity of the two wheels v_l and v_r translate into a forward velocity V and an angular speed ω of the robot.

Figure 4.4 shows the Frobit robot with wheel diameter d and wheel distance W between the center of the left and right propulsion wheels. The Frobit geometric center (GC) is located on the wheel center axis at the distance $W/2$ from each wheel. A third castor wheel supports the robot, it is passive and does not contribute to the robot motion. The Frobit system feedback is incremental ticks from the left and right propulsion wheel. The ticks relate to the wheel velocity as described in equation (4.4) for the left wheel:

$$v_l = \frac{d\ \pi\ ticks_l}{ticks\_per\_rotation\ \Delta t} \tag{4.4}$$

In real life implementations the discrete nature of the velocity calculation may be inaccurate at low velocities and high encoder update rates where the number of ticks per update becomes very low. Extrapolation and/or low pass filtering are therefore often applied.

The robot direction of movement is determined by the difference between v_l and v_r: The two wheels follow circular trajectories about the same point located on the wheel center axis. This point is called the Instantaneous Center of Rotation (ICR) and its distance R from the robot Geometrical Center (GC) varies with the difference between v_l and v_r.

To derive the relation between the forward velocity V of the robot GC and the angular velocity of the robot $\dot{\theta}$ a relation between angular velocity ω and linear

velocity V from the classical mechanics is used:

$$\omega = V/R \tag{4.5}$$

Using figure 4.4 this implies that:

$$v_l = \omega(R - \frac{W}{2}) \tag{4.6}$$
$$v_r = \omega(R + \frac{W}{2}) \tag{4.7}$$

Solving for ω and R yields:

$$v_r - v_l = (\omega R + \omega W/2) - (\omega R - \omega W/2) \Updownarrow$$
$$\omega = \frac{v_r - v_l}{W} \tag{4.8}$$
$$R = \frac{W(v_r + v_l)}{2(v_r - v_l)} = \frac{v_r + v_l}{2\omega} \tag{4.9}$$

Assuming the robot follows the ideal trajectory without slip, the yaw axis angular velocity of the robot in the *Robot* coordinate system $\dot{\theta}$ is equal to the robot angular velocity ω about ICR and may thus be expressed as:

$$\dot{\theta} = \omega = \frac{v_r - v_l}{W} \tag{4.10}$$

The forward velocity of the robot geometric center can be found by using equation (4.5):

$$V = \frac{v_r + v_l}{2} \tag{4.11}$$

There are different cases to consider:

1. $v_r = v_l \implies R \to \text{inf}$, $\omega \to 0$, $V = v_r = v_l$ The robot moves along a straight line
2. $v_r = -v_l \implies R = 0$, $V = 0$ The robot turns about GC.
3. $v_r \neq v_l$ The two wheels follow circular trajectories i.e. the robot turns about GC (figure 4.4).

The equations (4.11) and (4.10) constitute the forward kinematic equations of a differential steered robot and is used by the FroboMind odometry component to estimate the updated robot pose with respect to the *World* coordinate system.

4.2.3.2 Inverse kinematics

The *inverse kinematic* equations for a differential steered vehicle describe how the velocity of the wheels v_l, v_r must be set for the robot to obtain the forward velocity V and the angular velocity ω.

They are derived by substituting ωR in the equations (4.6) and (4.7) with (4.5):

$$v_l = V - \omega\frac{W}{2} \tag{4.12}$$

$$v_r = V + \omega\frac{W}{2} \tag{4.13}$$

4.2.4 Robot propulsion

A differential drive robot is very sensitive to the relative velocity between the left and right wheels. As seen in equation (4.10) even small errors in the velocities result in a different trajectory. This is especially prevailing for robots with a small distance between the left and right wheel i.e. when W is small.

This places great demands on the wheel velocity controller. The commanded velocity must be executed quickly and accurately for the robot to exhibit a smooth driving.

The prototype robots described in section 7 use different means to control the wheel or track velocity. The often used solution is some means of PID control based on feedback from motor hall elements or an external encoder. Where possible the velocity control is handled by an external embedded controller. As discussed in section 2.5.1 FroboMind only support near real-time, and velocity control should ideally be executed in hard real-time. For the Armadillo robot the velocity control is performed by FroboMind though. This is not ideal but as described in section 7.1.2 there have been several problems with the current motor controller.

4.3 Evaluating the FroboMind waypoint navigation

In this section the current implementation of the FroboMind waypoint navigation is tested and evaluated with respect navigation accuracy.

4.3.1 Navigating along a straight line

This experiment is based on the results from a trial presented in section 5.3. In the trial the Armadillo IV robot described in section 7.1.2 mounted with a cell spray implement navigated a straight line in a crop field. The path following algorithm described in section 4.2.1 was used, the moving goal **G** was fixed 0.9 m ahead of the closest point **C** (figure 4.3). The velocity was approximately 0.5 m/s. The test results shows that over a distance of approximately 145 m the 95th percentile of the absolute distance to the AB line is 0.025 m and the 95th percentile of the absolute heading error is 1.17 degrees. These results are discussed in section 4.4.

4.3.2 Waypoint arrival accuracy

The ability to stop at an exact spot is another measure of the navigation performance which is relevant to a number of precision tasks. The best available data material concerning stopping at waypoints is from the surveying project described in section 7.3.4. In this work the FroboScout robot described in section 7.1.6 has performed numerous trials navigating a certain pattern on the asphalt surface of newly constructed highways performing height measurements.

The results presented here are based on the trial where the largest number of waypoint stops were performed in a single trial. The robot navigated a path of length 1031 m in 78 minutes. Along the path 259 waypoints were visited and at each waypoint the robot stopped for 6 seconds while measuring. The desired navigation velocity was set to 0.9 m/s which is the maximum feasible velocity of the current FroboScout prototype. Due to the many stops the ramping of the velocity has lowered the average velocity though.

For this highway surveying trial the mean of the waypoint arrival distance was 0.035 m, the standard deviation was 0.016 m, the 95th percentile was 0.064 m and the maximum distance was 0.083 m. Figure 4.5 shows a histogram of the result.

Given the somewhat high inaccuracies in the highway surveying trial another experiment was performed to assess the cause. In the experiment a simulated trial was defined for the Frobit robot described in section 7.1.3. The trial resembles navigation along straight rows in a row crop field combined with sharp headland turns. The path consists of 8 waypoints and is 12 m long. The moving goal **G** was fixed 0.6 m ahead of the closest point **C** (figure 4.3). Given the lack of an absolute pose the experiment is based on odometry from the Frobit wheel feedback and a VectorNav VN-100 IMU. The trial was repeated 5 times and the deviation between Start and End for each trial was measured. The mean deviation of the odometry was 0.118 m which corresponds to 0.99% of the path length.

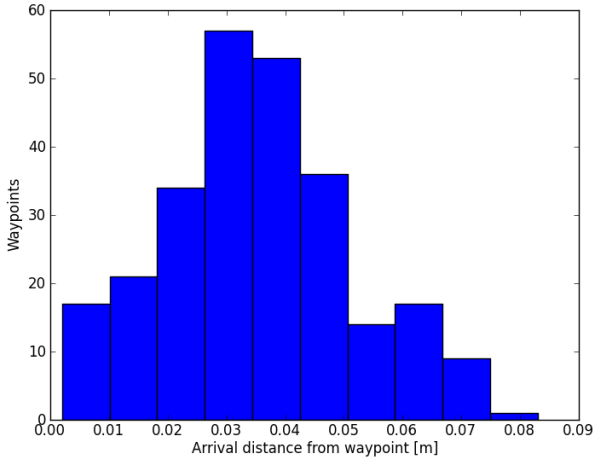
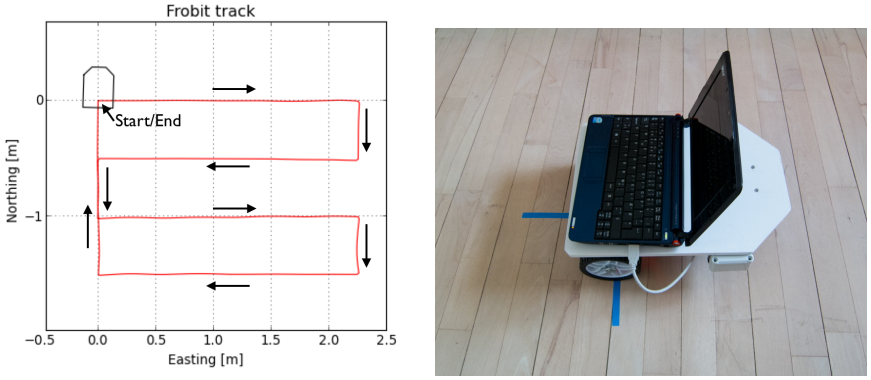


Figure 4.5: *Histogram of the arrival distance for 259 stops at waypoints during a highway survey performed by the FroboScout robot.*



(a) Frobit track. The orientation of the coordinates are relative, but the lengths are absolute

(b) The geometric center of the Frobit placed at the Start/End position.

Figure 4.6: *Frobit experiment setup. Figure 4.6(a) shows the Frobit track resembling navigation along straight rows in a row crop field combined with sharp headland turns. After completion of the trial the distance from the Frobit geometric center to the blue marking in figure 4.6(b) was measured to estimate the inaccuracy of the odometry.*

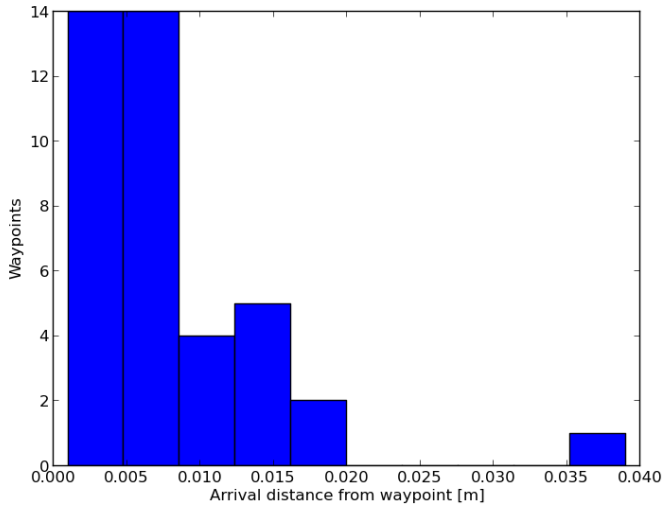


Figure 4.7: *Histogram of the arrival distance for 40 stops at waypoints during 5 simulated trials performed by the Frobit robot. Except for a single outlier all deviations from the waypoints are within 0.02 m.*

For the Frobit experiment the mean of the waypoint arrival distance was 0.0078 m, the standard deviation was 0.0068 m, the 95th percentile was 0.018 m and the maximum distance was 0.039 m. Figure 4.7 shows a histogram of the result.

4.4 Discussion

The results of the experiment where Armadillo IV with the cell spray implement navigated a straight path in a row crop field are fully adequate for most precision agriculture tasks.

Comparing to similar experiments (Bakker et al., 2011) presents results from a large 4 wheel double Ackermann steered field robot driving at 0.3 m/s: Mean, standard deviation, minimum and maximum lateral error were respectively 0.1, 1.6, -4.5 and 3.4 cm. Mean, standard deviation, minimum and maximum of the heading error over the same distance were respectively 0.0°, 0.46°, -1.26°and 1.32°.

(Jingtao and Taochang, 2014) presents results from a 3.1 ton 59 kW rear-wheel drive tractor: The mean value of the lateral position deviation was 0.02 m and standard deviation of lateral position deviation was less than 0.04 m. The results also reveal that mean value of steering angle tracking errors is 0.5°

According to (Li et al., 2009) Stanford University, USA has presented results from a John Deere 7800 Tractor with a line tracking deviation of 2.5 cm and a

1°heading accuracy.

Although the presented references have used significantly different vehicle platforms and the results are thus not directly comparable it is evident that the results presented in this work is on par with the referred results.

The presented waypoint arrival accuracies from the highway surveying trial are sufficient for the task but leaves some room for improvement. A 95th percentile of 0.064 m is not impressive considering that the surface is newly laid asphalt and the FroboScout robot is differential steered using bicycle wheels.

To assess the cause of this a simulated trial was performed using the Frobit robot. This trial produced a 95th percentile of 0.018 m and only one outlier above 0.02 m. This indicates that the Frobomind waypoint navigation works well under normal conditions. Some accuracy may probably be gained by optimizing the approach of the waypoint in the FroboMind path following component. But the major contributor to the inaccuracies in the highway surveying experiment is probably the FroboScout robot platform. As described in section 7.1.6 the drivetrain is rather rudimentary based on windshield wiper motors, the robot was never intended to perform large scale trials like this.

As described above the FroboMind navigation system and its performance in the field is not a novelty in itself. Autonomous waypoint navigation is however a prerequisite to performing many autonomous tasks in precision agriculture. Similarly to the global pose estimation discussed in section 3.1 the software behind the published results referred above is hard if not impossible to access.

4.5 Conclusion

In this chapter the FroboMind architecture and components used for autonomous path following for differential steerede robots are presented and evaluated in field tests. The main contribution from this work is the publication of the implemented and field tested FroboMind components.

The performance of the navigation was evaluated regarding the ability to navigate along a straight path and to stop at an exact spot. In both experiments the FroboScout autonomous navigation software has delivered satisfactory results.

Chapter 5

Case study: Precision agriculture

This chapter contains a case study on using FroboMind within precision agriculture. It is based on the conference paper (Jensen et al., 2013). Appendix C contains the submitted paper in the original formatting.

In precision agriculture the desire for an increased efficiency has resulted in a trend towards larger implements for crop treatment. The advantages are obvious, as the efficiency increases and labor hours are saved. However when driving in the field these heavy machines cause a significant compaction of the upper soil layers (Kroulik et al., 2009). This reduces the energy absorption ability of the crop plants and it is estimated that 80-90% of the energy going into traditional cultivation is there to repair the damage done by heavy vehicles driving in the field (B.S. Blackmore, 2007).

The use of field robots makes it possible to move from the current principle of a uniform treatment of the crop field towards a focus on the growth conditions for the individual crop plant and hereby make the production more efficient and sustainable. This requires accurate positioning and driving in the field which can be achieved by the autonomous navigation described in chapter 4. It also saves labor costs to the often monotonous and tedious work, reducing the need to use large implements.

This work presents a precision agriculture example application aiming to reduce the amount of herbicide in crop production. It describes an autonomous precision spraying trial using a novel cell spray implement mounted on the Armadillo robot. This is one of the latest examples of FroboMind used in experimental research applications.

The trials defining the experimental part of the project have been completed

successfully, but results from the crop treatment are not yet available due to the workload concerning processing of the substantial amount of recorded images. The conference paper focuses on the system integration and the robotic properties required by the cell spray implement. It is expected that a journal paper will be published later based on the full results of the experiment. Large scale autonomous cell spraying of fields is a novelty not found in the existing literature (Slaughter et al., 2008).

5.1 Introduction

Precision weeding is one of the most promising applications for autonomous service robots in biological production (Pedersen et al., 2006; Sørensen et al., 2007). The extensive use of herbicides during the past decades (Kurstjens, 2007) is being challenged by a growing concern among consumers and political decision makers about the potential environmental impact on drinking water reservoirs, fauna in watercourses etc. In 2010 pesticide residues were found in 44% of the drinking water monitoring points in Denmark, the requirement of a concentration of maximum 0.1 μg/l was exceeded in 15% of the monitoring points. (Thorling et al., 2011).

Using computer vision technology to detect the location of individual weed plants and precision spraying technology to accurately dispense only the required amount of herbicide given the detected weed concentration makes it possible to significantly reduce the consumption of herbicides (Midtiby, 2012). However both computer vision and precision spraying are technologies that require significant research and development efforts. At the same time they make heavy demands on the robot platform with respect to performance parameters such as precise navigation and vibration avoidance as well as infrastructure in terms of carrying capacity, power, protection against rain and dust, etc. To date only few complete robotic weed control systems have been tested under field conditions (Slaughter et al., 2008)

The work presented here is part of a project with the purpose of performing large scale autonomous precision spraying trials using a novel cell sprayer. This work focus on the cell sprayer implement design including camera system, sprayer module and integration with the service robot and the robot software. The Armadillo robot described in section 7.1.2 is used for the cell sprayer and we hypothesize that using FroboMind the cell sprayer can drive smoothly through a test field with a lateral positioning accuracy better than 50 mm.

The first task of the cell sprayer is to perform a precision spraying trial in maize using different treatment methods while at the same time registering the impact of the treatments on the crops and weed. This trial will be used for testing our hypothesis.

Figure 5.1: *The test field with maize crops. The periodic horizontal transverse rows are patches of sown weed plants.*

5.2 Materials and methods

5.2.1 Trial description

The trial site is located at Flakkebjerg, Denmark, where a 1.4 ha field of maize was sown. During sowing the crop row lines were stored using a RTK-GNSS. The field is divided into 566 test parcels with a size of 3 m along the crop rows and 1.5 m across. Between each parcel a buffer zone is placed to avoid boundary effects. The location of each parcel is described by its four corner coordinates. Based on the robot position estimation it is possible to determine the parcel number and the associated treatment of the parcel. In about 30% of the parcels specific weed plants were sown in lines (figure 5.1).

12 different treatments are applied each to a random subset of the test parcels. The treatments are based on two different weed detection algorithms, each with 5 different thresholds of when to spray a cell. The last two treatments are *spray the entire parcel* and *no spraying*, both to build a reference for evaluation of the results. In the first algorithm weed plants are detected using a monocot/dicot separation algorithm (Laursen et al., 2012), the second detects weed plants by analyzing the plant location with respect to the crop plant grid structure (Midtiby, 2012).

A visualization of the parcel placement in one half of the field is shown in Figure 5.2. Each parcel is shown as a small rectangle. The planned route which covers all the parcels is shown in red, green and blue. The total length is 2520 m. Due to physical constraints on the attachment of the trailed tank which carried

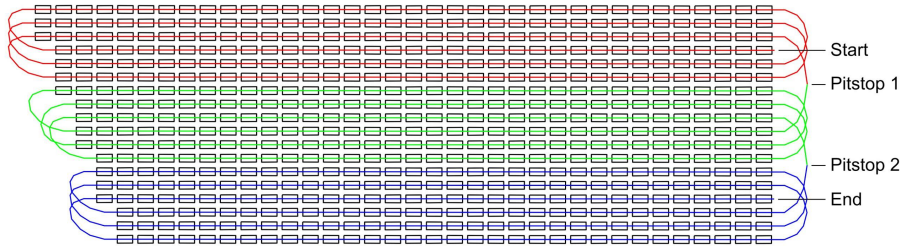


Figure 5.2: *Overview of the test field parcels and route plan.*

Figure 5.3: *The image acquisition chamber with cameras. The illumination tubes are installed above the white plate.*

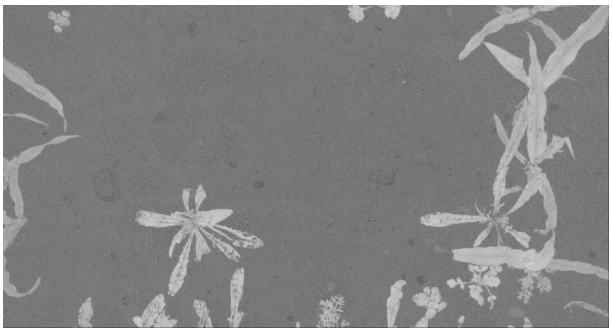
the spray liquid, it was not feasible to make left turns, and right turns had to use small turn angles. The shown route consists only of right turns and during each turn at least two rows are skipped which increases the turning radius to a feasible level. The trailer are capable of carrying enough spray liquid to cover a distance of 900 m with all the spray nozzles turned on. Two pit stops for trailer refill are placed on the route such that each part of the route between two stops has a length of approximately 900 m.

After completion of the sowing each parcel is treated with the cell sprayer twice. The first treatment is scheduled 21 days and the second treatment 31 days after sowing. At each treatment and a scouting drive scheduled 41 days after sowing, images from all the parcels are acquired to evaluate the weed infestation rate of each parcel.

5.2.2 Cell spray implement

The cell spray implement consists of two chambers, one for image acquisition and one for spraying. The image acquisition chamber (figure 5.3) measuring 1.8 x 1.5 x 1.1 m is made of alubond with a matt white finish that shields

(a) Example image acquired by the camera.

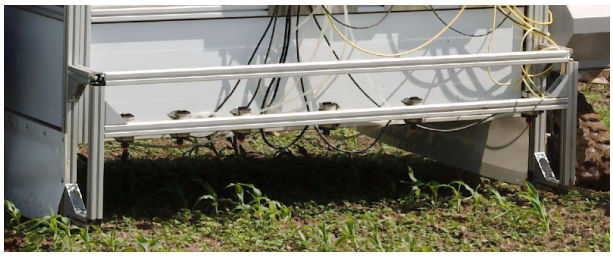


(b) Image after filtering.

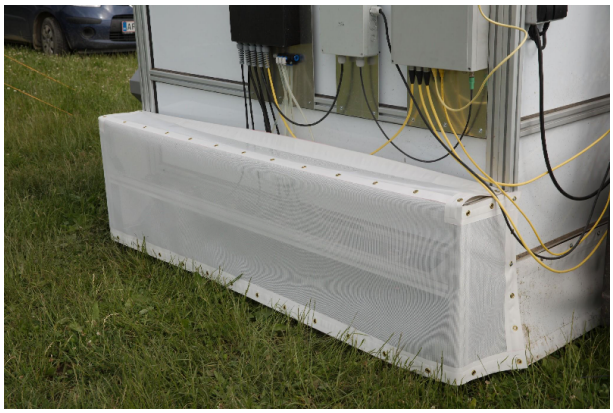
Figure 5.4: *Example image before and after processing.*

the outside illumination and results in diffuse reflections. Inside the acquisition chamber the illumination consists of 36 Philips TL-D 90 Graphica 58 W fluorescent tubes, providing a 5000 kelvin illumination with a colour rendition index of 97%. In front of the fluorescent tubes a sheet translucent polycarbonate is placed (Makrolon GP white) in order to ensure diffuse illumination. This corresponds to an illumination of 78840 Lumen, which generates an intensity of approximately 30,000 Lux. Inside the chamber two Basler acA2000-50gc cameras mounted with Azure 1214M5M lenses take care of the image capturing, the cameras are global shutter CMOS cameras with 5.5 μm pixels at a resolution of 2048 x 1088 each having a 60 dB SNR. The camera focal points are 920 mm above the ground and the lenses adjusted to F/4.0, resulting in an airy disc of 5.3 μm. The lenses are focused at 820 mm, resulting in circle of confusion less than 5.5 μm from 737 mm to 924 mm, resulting in in-focus images with object heights from - 4 - 183 mm (figure 5.4).

The spray chamber (Laursen et al., 2011) (figure 5.6) is a WxHxD 1800x500x300 mm chamber mounted with windbreaks in order to reduce wind drift. Inside the chamber 6 high speed solenoid Weedseeker VC-01 valves are mounted for turning the spraying on/off. Each valve is mounted with a Hardi LD-01 110^0 nozzle. The nozzle is mounted 250 mm above the ground. The valves are pressurized



(a) The row of spray nozzles behind the windbreaks.



(b) Spray chamber with windbreaks installed.

Figure 5.5: *The spray chamber mounted at the rear of the image acquisition chamber.*

using a Hardi ATV spray tank with pressure release valve. The pressure was set at 2.8 bar with the valves closed, which resulted in 2.2 bar with all the valves open.

5.2.3 Robot tool carrier

The cell spray implement is carried by the Armadillo IV robot in a shock absorbing frame (figure 5.6). The Armadillo IV is described in section 7.1.2. In this work a new version track module drivetrain was developed and tested. The drivetrain is described in section 7.1.2 as well. The Armadillo IV robot supports odometry feedback originating from the brushless motor hall sensors and is in this work equipped with a VectorNav VN-100 IMU and a Trimble BX982 RTK-GNSS receiver connected to the GPSnet.dk reference network.

(a) Front view.

(b) Rear view with the windbreaks removed from the spray chamber.

Figure 5.6: *The cell sprayer based on the Armadillo IV robot. The GNSS antenna is above the geometrical center. The other antennas are for Wifi, Bluetooth and a wireless emergency stop. The red generator supply the illumination in the image acquisition chamber.*

5.2.4 FroboMind

The Armadillo robot is controlled by a industrial PC (dual core, 2.x GHz) running FroboMind. In this work the components for TM projection conversion, global pose estimation, waypoint navigation and parcel mapping was developed and tested in field experiments for the first time. The global pose estimation is described in chapter 3 and the waypoint navigation is described in chapter 4.

The new FroboMind parcel mapping component is described in the following: Information about the robot's entry and exit of the test parcels is communicated from the robot to the cell spray implement during route plan navigation. This is handled by a mapping component that loads a list of polygon coordinates describing the 566 test parcel locations. At each pose update the new pose is checked against the polygons and changes are published.

To lower the computational load without delaying the published parcel changes, the update algorithm seeks to minimize the number of required calculations. For each polygon a rectangular bounding box, a timeout value and a nearby flag is assigned at startup. At each robot pose update the polygon map update (algorithm 4) iterates through a subset of polygons determining if the robot distance to the polygon bounding box is within a certain threshold distance. For polygons outside the threshold a timeout value based on distance and the robot maximum speed determines the next time this polygon should be checked. All polygons in the list inside the threshold are checked at each update using a point in polygon algorithm. In this trial the pose update rate is 20 Hz, the subset is 100 polygons and the threshold distance is 5 m.

Algorithm 4: Polygon map update

input : Current pose, list of polygons
output: Polygons with changed status

```
begin
    for polygons in current subset do
        if polygonNearbyFlag or polygonTimeout then
            distance = min(pose distance to corners of bounding box)
            if distance < nearbyThreshold then
                polygonNearbyFlag = true
            else
                polygonNearbyFlag = false
                polygonTimeout = now + distance/maxSpeed - mapUpdateTime
    for all polygons with polygonNearbyFlag = true do
        Check if pose is inside polygon
        if pose inside polygon status just changed then
            Publish change
```

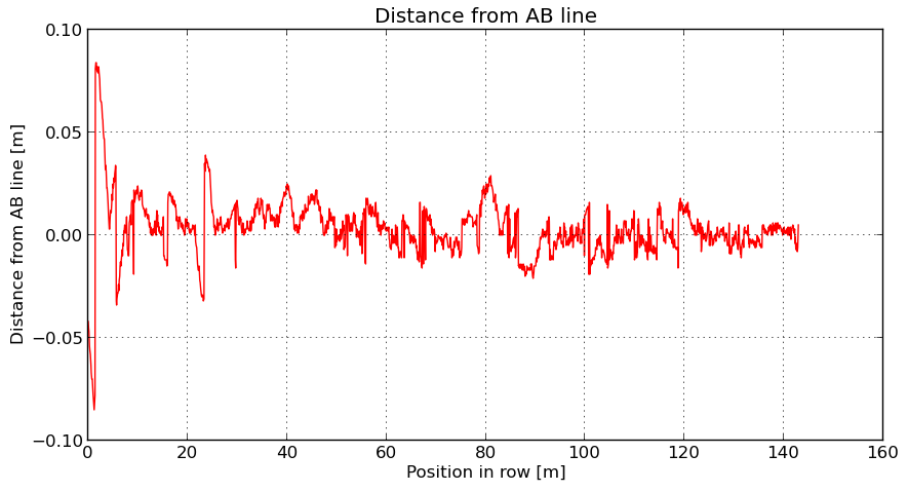


Figure 5.7: *Results showing distance from AB line. Except for the first measurements the distance is well within 50 mm.*

5.3 Results

The results are kept in this chapter to maintain the full description of the case study. They are however evaluated and discussed in greater details in section 4.4.

In preparing for the first spray task a series of test runs were conducted to configure and parameterize the robot and cell spray implement. The following figures are based on data from driving along the first crop row in one of these test runs. The spray tank was attached and half filled with water during the test run. The data has been filtered to exclude headland turns and subsequent rows. No filtering and no down sampling has been applied to the data shown in the figures. The row length was approximately 145 m, the robot was driving at a speed of approximately 0.5 m/s and data was sampled at a rate of 5 Hz. Samples from subsequent rows and other test runs showed similar results.

Figure 5.7 shows the robot lateral position accuracy with respect to the navigated **AB** line. The 95th percentile of the absolute distance to the **AB** line is 0.0250 m. The estimated robot pose is used as reference because it is assumed to be the most accurate. The 95th percentile on the difference between raw GNSS data and the estimated pose is 0.0104 m for the trial. The data has not been compensated for robot tilting about the roll and pitch axis. Figure 5.8 shows an example of the robot track along on of the **AB** lines.

Figure 5.9 shows the robot heading error with respect to the destination **B** waypoint. The 95th percentile of the absolute heading error is 1.17 degrees. The estimated pose is used as reference as there exist no raw sensor data describing

Figure 5.8: *Visible track after AB line navigation.*

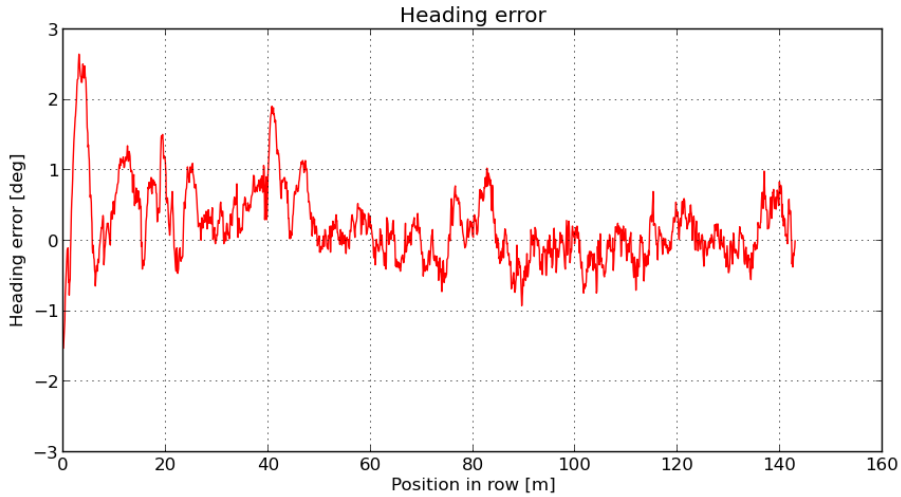


Figure 5.9: *Results showing heading error.*

the absolute orientation about the yaw axis.

5.4 Discussion

In this work the spray implement was developed, a new Armadillo IV module was developed, and the first versions of the FroboMind components for Transverse Mercator coordinate conversion, pose estimation, waypoint navigation and polygon mapping were developed.

Data analyzed from test runs performed in conjunction with the trials shows that the Armadillo robot with the cell sprayer implement attached is capable of navigating the test field smoothly with a lateral positioning accuracy better than the anticipated 50 mm. The test results shows that over a distance of approximately 145 m the 95th percentile of the absolute distance to the AB line is 0.025 m and the 95th percentile of the absolute heading error is 1.17 degrees which is within the requirements of the spray implement. The analyzed data substantiates that the Armadillo robot with the cell sprayer implement attached is capable of navigating the test field with a lateral positioning accuracy better than the anticipated 50 mm. These results are evaluated and discussed in greater details in section 4.4.

The FroboMind components developed in this work performed as expected after having undergone a thorough testing and debugging process in the field. Some problems with the underlying Armadillo track velocity controller component were discovered and was not solved within these trials. The Armadillo track velocity controller was not capable of correctly controlling the desired velocity, instead it varied depending on the load and hence the surface conditions, implement weight etc. It was therefore not worth the effort to fine tune the navigation controller, the test results presented here were thus obtained after a few test runs with manual parameter calibration. The reason that the results are satisfactory anyway is that a skid steer vehicle driving at low velocities with the tracks spaced far apart is relatively easy to control as discussed in section 4.2.4, the navigation controller was therefore capable of compensating the track velocity errors. The problems with the Armadillo track velocity controller have been resolved in subsequent work and the Armadillo now responds correctly to velocity commands.

The first treatment has been completed successfully, even though it took place shortly after a heavy rainfall that turned the top layer of the clay soil into mud (figure 7.5(b)). The resulting higher load on the motors caused the RoboteQ motor controllers to trip frequently which has made the navigation data time consuming to analyze. The motor controller problem has subsequently been analyzed and it is believed that the internal current measurement is too sensitive and responds to short term peak values rather than average measurements.

Adjusting the maximum current from 75A to 125A and setting stall detection to its highest setting has somewhat mitigated the problem, but the average current consumption for each track module is much lower than 75A. The entire treatment was completed without requiring a battery recharge before completion, and the theoretical capacity for each of the two battery packs is 2640 Wh.

The treatment was performed at a speed of 0.3 m/s. The limiting factor was the computation speed of the imaging analysis algorithms.

Testing the new Armadillo IV drivetrain design has revealed a few issues in the transmission where the shafts caused a lateral displacement of the tooth wheels. These issues have been resolved in subsequent work.

The results from the weed detection algorithms and the resulting weeding performance will be published in a later publication and the cell sprayer is expected to be used in further precision spraying experiments.

5.5 Conclusion

This work has presented a FroboMind use case based on a precision agriculture application aiming to reduce the amount of herbicide in crop production. A novel cell sprayer based on the Armadillo robot and a precision spraying implement were developed and used in large scale precision spraying trials.

New FroboMind components including global pose estimation and waypoint navigation were developed as part of this work, and navigation performance in the field was evaluated. The cell sprayer completed three trials where it navigated a 2.5 km route through the rows of a 1.4 ha maize field autonomously. It was concluded that the Armadillo robot is capable of navigating the test field within the requirements of the spray implement.

Chapter 6

Case study: Humanitarian demining

This chapter contains a case study on using FroboMind for autonomous mapping of areas contaminated by landmines and unexploded ordnance.

Humanitarian demining is a very challenging application area for field robots, and as indicated in section 6.2 progress in deploying humanitarian demining robots to mine action campaigns is rather slow.

From a robotics perspective there are many similarities between weed detection in precision agriculture and the detection of hazards buried below the ground. Much of the knowledge and experience from precision agriculture applications may therefore be applied advantageously to humanitarian demining.

Two papers based on this work have previously been submitted to conferences: In (Jensen et al., 2011) the concept of a field robot performing mapping of landmine and Unexploded Ordnance (UXO) contaminated areas is outlined. In (Jensen et al., 2012a) the first results of mapping based on remote control of the robot was presented. A publication is currently being drafted where parts of the work presented here will be included.

6.1 Introduction

Landmines and explosive remnants of war (ERW) are a threat to the life and livelihood of many thousands of people in many parts of the world. ERW include UXO, which are explosives like grenades, mortars, cluster munition etc. that have failed to detonate as intended. Aside from the killing and injury of people, the landmines and UXO have a significant impact on the local economy due to hindered access to water points, schools etc. and loss of fertile agricultural

areas.

No one knows how many landmines and UXO remain uncleared, however according to (LandMineMonitor, 2013) the statistics for 2012 reports the following figures: 3682 registered casualties caused by mines, improvised explosive devices (IED), cluster munition remnants, and other ERW. The real number is expected to be significantly higher due to lack of registration. 281 km^2 of mined areas were released through clearance or survey during which 240,000 antipersonnel mines and 9,300 anti-vehicle mines were destroyed. In addition, some 245 km^2 of battle areas were cleared including 78 km^2 of area contamination by cluster munitions, destroying just over 300,000 items of unexploded ordnance. Despite the Mine Ban Treaty (Ottawa Convention) landmines are still used actively. As examples antipersonnel landmines were laid in large numbers in Yemen in 2011, while Syria and Myanmar laid antipersonnel mines in 2012 and 2013.

Humanitarian demining is activities leading to clearance of landmine and UXO hazards that poses a threat after the conflict has ended. The aim is the identification and removal or destruction of all hazards, from a specified area to a specified depth to ensure the land is safe for land users (UNMAS, 2003).

The removal or destruction of a landmine or UXO hazard is a relatively simple process once the location is known, however the critical problem is detecting the precise location of a hazard and ensuring that all hazards within the area have been identified. In a typical scenario the mine clearing is performed by segmenting the area into marked lanes. Deminers are then working their way along the lanes using a metal detector or prodder. When a possible hazard is detected, the deminer excavates it and in case of a landmine or UXO it is either removed or destructed on site.

The mine clearance speed appears to be 3 to 30 square meters per deminer per day depending on the terrain and level of metal contamination (GICHD, 2005; Lardner, 2005). The cost is estimated to about US$ 300 to 1000 per mine (Walsh and Walsh, 2003). Newer mine types contains little metal and hence require a much more sensitive metal detector. This leads to a lot of false positives which slows down the speed significantly as all potential hazards need to be inspected carefully (GICHD, 2007).

The demining tasks require that deminers work at or in close proximity to the minefield. They wear protective gear and it is often unbearably hot. The task of searching for mines is slow and monotonous, it also requires both systematics, precision and endurance. It is therefore understandable that the concentration sometimes fails, but unfortunately this may result in hazards not being detected and that the deminer is exposed to great danger.

On contaminated areas accessible by small vehicles a robot may be applied to the process of localizing potential hazards. This method allows the deminer to operate from a safe distance and mitigates the problems with fatigue and

failing concentration. The robot can perform online analysis of measured data and thus support dynamic changes in robotic behaviour to obtain more detailed measurements at a close proximity to potential hazards to improve the reliability.

The aim of this work is to design and construct a field robot platform for humanitarian demining tasks that allows operation from a safe distance. The robot will be tested in a trial consisting of autonomous mapping of an area contaminated by dummy landmines. The results of this trial will be compared to a similar human operated mapping task.

6.2 Related work

The idea of using a field robot fitted with a mine detection sensor to achieve a faster and more reliable detection of landmines and UXO is not new. Different robot prototypes have been developed and tested the past years, and even though none of them seem to have been implemented in mine action campaigns on a larger scale, valuable lessons may be learned from the projects. (Baudoin and Habib, 2011; Baudoin, 2005) gives a thorough description and state of the art of robotics for humanitarian demining and related risky interventions and includes many references to related work within this field. (Rajasekharan and Kambhampati, 2003) surveyed the robots and search methods for landmine detection over the last decade and describes the problems involved, some of the issues that have been overlooked, and stresses certain guidelines for future robot design. Some important recommendations are to involve the end users in the design of equipment, to design for low-cost, lightweight, high mobility, simplicity, fault tolerance and easy maintenance.

(Habib, 2008) describes the challenges of robotics applied to demining and gives a number of recommendations based on lessons learned. The major challenge is pointed out as being the ability to discriminate landmines from metal debris, natural clutters and other objects (false positives) without the need for vegetation cutting. The paper recommends using multi sensor systems to improve the discrimination ability. (Havlik, 2005) analyses some most important characteristics that should be taken into consideration in building the robotic demining vehicle. Some of the mentioned conditions are: Reliable detection and localization, the harsh working conditions, end user acceptance and usability concerning educational level, cost, maintenance and transport to minefields. (GICHD, 2006) gives a review and status summary of detection technologies that could be applied to humanitarian demining operations. Of particular importance is the section on vehicle based multi sensor systems. (Herman et al., 2010) made a performance comparison between manual sweeping and a teleoperated robotic system and concluded that remotely operating a mine detector

is technically very feasible, and does not affect the detection rate negatively. (Freese et al., 2007) introduces Gryphon, a mine detection semi-autonomous robot based upon an ATV retrofitted with a 3 DoF counter balanced pantographic arm. Gryphon can be controlled either manually or using a remote control. (Hemapala et al., 2009; Belotti et al., 2009) explore the idea of using common agricultural machines as demining robots.

6.3 Design considerations

The landmine and UXO contaminated areas vary from reasonable flat open areas to quite rough terrain with slopes, rocks, trees, bushes and other obstacles. The surface vary between soil, various types of sand, gravel and stones which depending on the climate may be either dry and dusty, muddy or even partially covered with water. This terrain puts high demands on any vehicle operating in the area, and some areas will be almost impassable by a vehicle.

Many contaminated areas are located in countries having a poor infrastructure with respect to logistics, availability of materials, machine shops etc. Therefore challenges such as transportation of the vehicle to the area of operation and availability of fuel and spare parts needs to be taken into account. It is also of great importance that any technology introduced is accepted by both the authorities and the local residents. The people and hence available labor often have only minimal formal education, and skilled technicians may be quite difficult to find.

Based on the above and the review of related work a humanitarian demining robot must be rugged and durable yet very simple in design to allow repair in the field with limited tools and parts. It must be small and lightweight to allow transport on a 4wd pickup truck or a regular sized trailer. Making the mine detection module dismountable from the tool carrier will allow the module to be utilized for hand carried operation at locations inoperable by the robot. It is important that the robot operation is very simple and intuitive allowing local deminers to perform the operation with a minimum of training. To sum up the requirements, the design should aim to optimize for reliability, efficiency, usability, low-cost, easy maintenance, rugged, modularity, multi sensor support and easy transport. This is a complex task requiring a lot of resources. In this work the main focus will therefore be on the first four design parameters: Reliability, efficiency, usability, low-cost. These requirements resembles what is seen in agricultural machinery production, so it makes sense to seek knowledge and inspiration from this domain.

Hardware parts	**Estimated cost**
Robot frame	$ 2,500
Motors & gear boxes	$ 7,000
Motor controllers	$ 1,500
Battery pack	$ 1,500
Robot computer	$ 4,000
Navigation sensors	$ 6,000
Total	*$ 22,500*

Table 6.1: *Estimated hardware cost of the Pichi demining robot.*

6.4 Demining robot platform

The *Pichi* demining robot (figure 7.12) was designed and constructed in an innovation project together with the industrial partner Lynex. It is based on the Armadillo robot described in section 7.1.2 and has been optimized towards the design parameters. The modularity has been given a lower priority in favor of a smaller size, lighter weight and lower costs as well as a simpler design to ease maintenance in the field. The technical description of the Pichi robot is found in section 7.1.4.

The Pichi robot has been constructed using low-cost materials. The most expensive parts are the motors, gear boxes, navigation sensors and the computer. Table 6.1 shows the estimated costs for hardware excluding mine detection sensors. The calculation is based on estimated retailer prices.

The Pichi robot is equipped with the LCRS described in section 3.4. Together with a VectorNav VN-100 IMU and feedback from the motor controller this constitutes the basis for estimation of the global robot pose. The odometry is updated at a rate of 50 Hz, the IMU at 40 Hz and the LCRS at 5 Hz.

6.5 Mine Detection Implement

The Pichi demining robot may utilize various landmine and UXO detection implements using different detection technologies like metal detectors, magnetometers, ground penetrating radar, explosive vapour detection (GICHD, 2007), mine raking (Silva, 2010) etc. Due to the ability to carry multiple implements the robot is also capable of utilizing multi-sensor technologies (GICHD, 2006; Sato, 2008). However the theory of UXO detection methods and principles is outside the scope of this work. To test the robot performance an implement based on Wide Area Detection System (WADS) developed by the organization DanChurchAid (DCA) has been fitted to the robot.

DCA became involved in humanitarian mine action in the mid 1980s. In 1999

Figure 6.1: *WADS used for area clearance by DCA in Sudan. The operator in the middle operates the UPEX 740 electronic box and data logging PC, the two assistants on each side carries the search loop.*

DCA started running operational demining projects and has carried out demining projects in a number of countries since then. DCA typically uses WADS and has implemented a setup where measurements are geopositioned using GPS precise positioning (Durocher and Jansson, 2006). In 2010 the DCA implementation of WADS was accredited as compliant with IMAS 03.40 Test and Evaluation of Mine Action Equipment by The United Nations Mine Action Office (UNMAO) in Sudan and can thus be used to mark areas as free of landmines and UXO (Persson, 2010).

The DCA implementation of WADS has been used as a portable setup (figure 6.1) as well as mounted on a vehicle and uses commercially available metal detectors. Output from the metal detector is continuously sampled through an analog interface and saved on a laptop along with the current position measured using precise positioning GPS (figure 6.2).

The sensor used is an Ebinger UPEX 740 Large Loop UXO Detector which has been used in humanitarian demining projects in a number of countries. The UPEX 740 uses a pulse induction principle to detect metal components in the targets. The detector transmits short electromagnetic pulses of low field strength through its search loop. This induces eddy currents and a secondary electromagnetic field into conductive objects within the detectors range. Between each transmission pulse the transmitter is switched off and a receiver is activated to detect secondary fields from conductive objects (Ebinger, 2005). Pulse induction metal detection allows detection of deeper and larger targets but has less discrimination between different types of metal. The UPEX 740 is intended for fast search of large areas, it allows an adjustable search loop diameter.

The UPEX 740M-S version used in this work outputs an analog voltage cor-

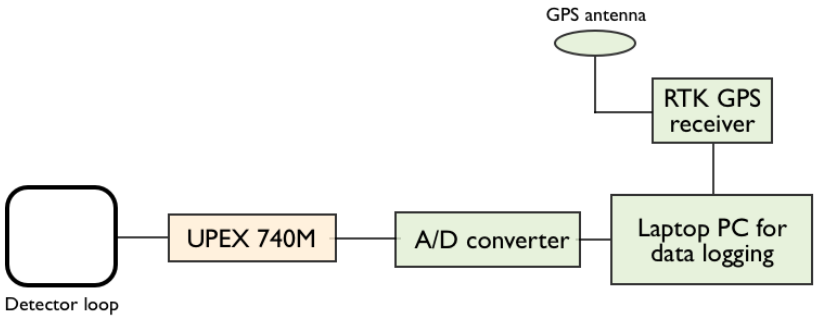


Figure 6.2: *WADS setup implemented by DCA*

Figure 6.3: *The Upex 740M-S electronic box*

responding to secondary fields. Figure 6.3 shows the UPEX740M-S *electronic box*. The *Mode* selector supports *Lin* which produces a linear signal increase upon approach of metal objects, *Log* produces a logarithmic indication of signal strength, which helps to pinpoint target center in case of strong signals, and *Slow* which increases the integration producing a slower response. The *Delay* adjuster excludes short response signals up to 250 μs. This allows to user to discriminate objects close to the search loop such as small metal scrap laying on the ground. The sensitivity of the search loop needs to be calibrated before use by setting the *Sensitivity* adjuster.

Adapting the UPEX 740M-S detector for use as a Mine Detection Implement (MDI) on the Pichi demining robot requires only a few modifications. The search loop is very sensitive to nearby metal and electromagnetic interference, hence it must be placed at a distance from the robot. Practical tests showed that the loop must be at least 0.6 m away from the robot. A 2x1 m metal-free frame supporting the search loop constructed with brackets for mounting in front of the robot was developed. The frame was made of wooden laths since this is easy to repair almost anywhere. It is possible to dismount the frame which allows hand carried operation of the WADS in terrain inaccessible by the demining robot. A computer interface was developed for the analog output of the UPEX 740M-S. The output Voltage range is -4V to +3V which is converted and measured using the RoboCard micro controller board described in section 7.1.3. Measurements with a resolution of 10 bits are transmitted at a rate of 50 Hz through a USB serial port. According to Ebinger the UPEX 740M-S needs

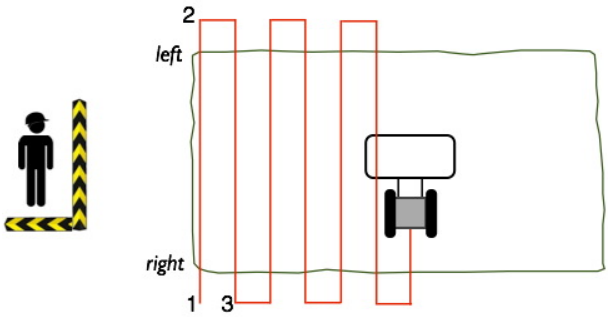


Figure 6.4: *Operator assisted autonomous coverage*

an independent power supply due to the unit grounding design, so a 12V VRLA battery therefore supplies the detector.

6.6 FroboMind

The Pichi robot is controlled by a PC running FroboMind. In the current implementation the navigation system is similar to the example architecture shown in figure 2.5. The autonomous navigation behaviour dynamically updates a list of waypoints based on the area coverage described in the following.

6.6.1 Area coverage behaviour

Mapping of a contaminated area is often difficult to plan ahead. The geographic positions of the area corners are usually unknown, and measurement of the positions may expose the operator to unnecessary risks. In this work a FroboMind area coverage behaviour for operator assisted autonomous coverage for open areas was therefore developed. It is based on previous work by the author in another project focusing on industrial mowing of large grass areas. An example of the coverage principle for a rectangular shaped area is shown in figure 6.4. The method of operation is as follows:

1. The operator located in a safe position places the robot at his right hand corner of the area facing towards his left hand corner. He activates the algorithm and the robot navigates ahead following a straight path.

2. When reaching the left hand corner, the operator presses the *Turn right* button which makes the robot turn 90°clockwise, drive a distance corresponding to the search loop width, make another 90°turn and follow a straight path parallel to and in the opposite direction of the previous path.

3. When reaching the right hand area limit the operator presses the *Turn left* button which makes the robot perform a similar turning operation counter-clockwise. The robot now knows the limits and will continue to cover the area shaped as a growing rectangle.

To decrease either the left or right limit the operator presses the corresponding turn button at the new limit. The robot will then use this limit. To increase the left or right limit the user presses the *Reset limit* button which makes the robot ignore the previous limit and continue until the corresponding turn button is pressed. Algorithm 5 presents the area coverage algorithm. The robot navigates

the line defined by **AB**. The *rot* function represents a standard rotation matrix.

Algorithm 5: Operator assisted area coverage algorithm.

input : Robot pose, user input
output: Dynamical waypoints
begin
  $l = default\_coverage\_length$; $w = implement\_width$
  $\mathbf{L} = \mathbf{R} = False$
  $\widehat{\mathbf{b}} = rot([1,0]^T, bearing)$
  $\mathbf{A} = \mathbf{P}$
  $\mathbf{B} = \mathbf{P} + \widehat{\mathbf{b}} * l$
  while *autonomous mode enabled* **do**
    if *User pressed 'Turn left' while moving right* **then**
      $\mathbf{B} = \mathbf{A} + \widehat{\mathbf{b}}(\mathbf{P} \cdot \widehat{\mathbf{b}} - \mathbf{A} \cdot \widehat{\mathbf{b}})$
      if $\mathbf{L} == False$ **then** $l = dist(\mathbf{L}, \mathbf{B})$;
      State = *Moving right*
    if *User pressed 'Turn right' while moving left* **then**
      $\mathbf{B} = \mathbf{A} + \widehat{\mathbf{b}}(\mathbf{P} \cdot \widehat{\mathbf{b}} - \mathbf{A} \cdot \widehat{\mathbf{b}})$
      if $\mathbf{R} == False$ **then** $l = dist(\mathbf{R}, \mathbf{B})$;
      State = *Moving left*
    if *User pressed 'Reset length' while moving* **then**
      $l = default\_coverage\_length$
      $\mathbf{B} = \mathbf{A} + \widehat{\mathbf{b}} * l$
      if *State == Moving left* **then** $\mathbf{L} = False$;
      else $\mathbf{R} = False$;
    if *Arrived at* **B** **then**
      switch *Current state* **do**
        case *Turning left*
          $\mathbf{R} = \mathbf{A} = \mathbf{B}$; $\widehat{\mathbf{b}} = rot(\widehat{\mathbf{b}}, \frac{\pi}{2})$; $\mathbf{L} = \mathbf{B} = \mathbf{A} + \widehat{\mathbf{b}} * l$
          State = *Moving left*
        case *Moving left*
          $\mathbf{A} = \mathbf{B}$; $\widehat{\mathbf{b}} = rot(\widehat{\mathbf{b}}, \frac{-\pi}{2})$; $\mathbf{L} = \mathbf{B} = \mathbf{A} + \widehat{\mathbf{b}} * w$
          State = *Turning right*
        case *Turning right*
          $\mathbf{L} = \mathbf{A} = \mathbf{B}$; $\widehat{\mathbf{b}} = rot(\widehat{\mathbf{b}}, \frac{-\pi}{2})$;
          $\mathbf{R} = \mathbf{B} = \mathbf{A} + \widehat{\mathbf{b}} * l$
          State = *Moving right*
        case *Moving right*
          $\mathbf{A} = \mathbf{B}$; $\widehat{\mathbf{b}} = rot(\widehat{\mathbf{b}}, \frac{\pi}{2})$; $\mathbf{R} = \mathbf{B} = \mathbf{A} + \widehat{\mathbf{b}} * w$
          State = *Turning left*

6.6.2 Wriggle behaviour

A *wriggle* behaviour was implemented to improve the quality of the map. Whenever the output from the MDI exceeds a defined threshold, the area coverage behaviour is suspended and the wriggle behaviour is activated. The wriggle behaviour makes the robot stop, then make a left turn, then make a right turn before terminating and thus reactivating the area coverage behaviour. The turning will wriggle the frame laterally in a circular motion, thereby increasing the number of measurements in the vicinity of a probable target.

6.6.3 Area search mapping

While navigating the contaminated area scalar measurements are continuously sampled from the MDI. The scalar measurements represent the field strength of the received response signal to the transmission pulse as described in section 6.5. They are thus a function of metal, conducting objects, conductive soil and electromagnetic fields in the vicinity of the search loop as well as the settings of the UPEX 740M-S. The settings of the *mode selector*, *delay adjuster* and *sensitivity adjuster* affects the measurements significantly and require calibration before use. Using signal processing directly on the response signal it would be possible to extract more detailed information about the measurements. But the response signal is not accessible from the UPEX 740M-S.

In the reference field trial described in section 6.7 the scalar measurements were approximated to a single point in the geometrical center of the search loop. This approach is not ideal because much information is discarded and thereby lowering the quality of the information. The source of a certain response might be located off-center and will thus be offset to the position associated with the measurement.

In a student project an attempt was made to design a FroboMind area search mapping component which optimizes the mapping accuracy based on the scalar measurements and hence increases the probability of detecting potential hazards. If static conditions of the vicinity is assumed and the sensor noise is disregarded any change in the scalar measurement output is a function of a physical movement of the search loop and the sensitivity of the search loop. The sensitivity model in figure 6.5 illustrates that the sensitivity is nonlinear and asymmetric across the search loop.

Under the assumption that the most significant change in the scalar measurements is observed when the search loop approaches or leaves a target it was decided to focus on the area gained by and area left by the search loop. Figure 6.6 illustrates the principle (the sensitivity model is not taken into account in the figure). The mapping algorithm implements the heuristic that if the implement output is increased, there is a high probability of the area gained by the

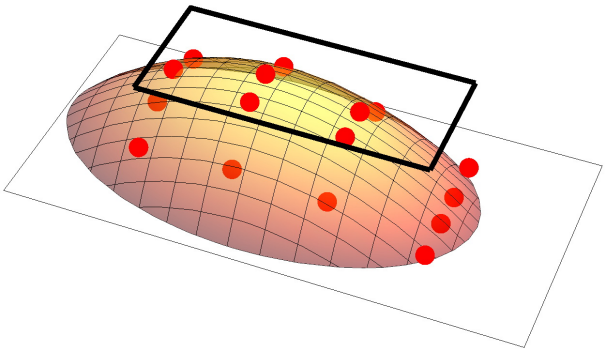


Figure 6.5: *Sensitivity model for the mine detection implement based on the scalar measurements in response to an iron plate buried 0.3 m below ground with the implement placed 0.3 m above ground. The height of each red dot above the plane corresponds to the measurements recorded while the geometric center of the search loop was located above this red dot.*

search loop being contaminated, whereas a decrease in implement output means that the area left has a high probability of being contaminated.

A thorough evaluation of the mapping algorithm based on results from field trials has revealed that it does not work well. One major issue is that the sensitivity model illustrated in figure 6.5 vary significantly with the UPEX 740M-S settings. A subsequent calibration of the sensitivity model after each change of settings is a laborious task but is required for the mapping algorithm to work.

In the presented experimental results the scalar measurements are therefore mapped into a grid map similar to the approach used ind the manual tests described in section 6.7. The *LinearNDInterpolator* function available in the Python library SciPy (Oliphant, 2007) is used for the grid mapping.

6.7 Field trial

A field trial was designed in order to compare the Pichi demining robot to manual demining. The equipment required for manual operation was not available so a direct comparison was not possible. Instead the manual tests performed by deminers described in (Persson, 2010) are used as basis for comparison. In one of the trials documented in this report 9 simulated hazards (targets) were buried at 0.3 m depth in a trial area sized 20x40 m. The area was level and flat with no visible obstacles above the ground. The trial procedure was described as:

1. Initial calibration of the WADS (3 operators, 5 minutes).
2. Search of the trial area (3 operators, 10 minutes).

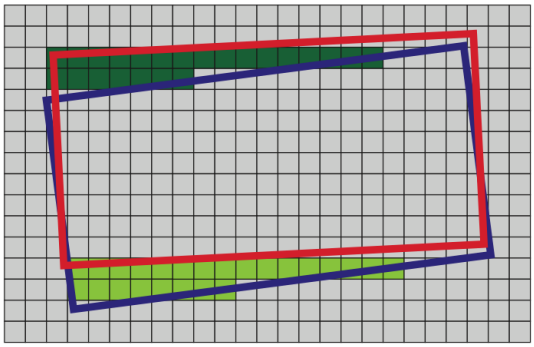


Figure 6.6: *Search loop difference map showing two discrete measurements while the robot is in motion. The red frame represents the search area during a measurement at time step T_k and the blue frame represents the search area at time step T_{k-1}. The collection of dark green cells represents area gained and light green cells represent the area left.*

3. Acquired search data transferred from the WADS computer to a laptop (1 technician, duration unknown).

4. Processing search data to generate map and identify targets (1 technician, duration unknown).

5. Transfer waypoints for identified potential hazards to a PDA (1 technician, duration unknown).

6. Relocate targets using PDA equipped with precise positioning GPS and precise marking of these using the large loop detector (3 operators, duration unknown).

For the tests in this work a grass field sized 10x20 m at the university campus was used. 6 targets consisting of an iron plate measuring 80x80x6 mm were placed in a depth of approximately 0.1 m and their positions were recorded for reference. The trial procedure was:

1. Initial calibration of the MDI.

2. Search of the trial area.

3. Processing search data to generate a search map.

The robot is capable of generating a map and identify targets online so task 3. of the manual trial is omitted. The purpose is to detect and map the targets only, so task 5. and 6. of the manual trial are omitted as well.

Figure 6.7: *The Armadillo III robot fitted with the mine detection implement.*

6.8 Results and discussion

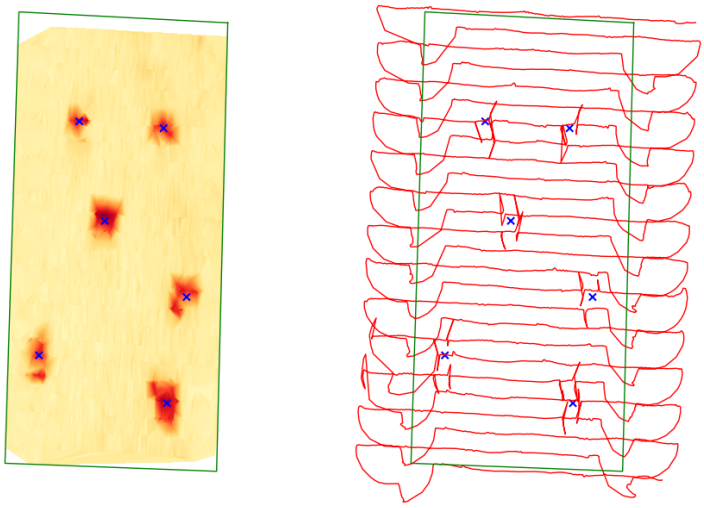
For the initial trials the Pichi deming robot was equipped with a Trimble BX982 RTK GNSS for use as ground truth reference of the robot pose. The GNSS was connected through a GPS antenna signal splitter to the same Trimble Zephyr geodetic II antenna as the LCRS installed on the robot. The results from tests concerning the accuracy of the LCRS are described in section 3.4.2 and is therefore omitted here.

The trial described in section 6.7 was performed several times without success because of problems with the robot propulsion system. It did not perform as expected and caused a slow and inaccurate driving at a high power consumption. It was possible to complete the trial procedure with a somewhat uneven motion though. The area search took 16 minutes to complete and the distance travelled by the robot was 210 m.

During the trials it was concluded that this first prototype of the Pichi robot has some fundamental flaws which need to be corrected. The flaws are detailed in section 7.1.4. It was decided to perform the trials using the Armadillo III robot (figure 6.7), however due to the vibration problems with this robot described in section 7.1.2 the subsequent trials were conducted at a lower velocity.

Before the first search trial the MDI was calibrated using one of the targets. The *Log* mode was used to achieve a clear identification of the targets, the *Delay* was adjusted to the minimum because of the target location only approximately 0.1 m below the ground, and the *Sensitivity* was adjusted to a rather low value. Figure 6.8 shows the result of a completed area search. The area search map in figure 6.8(a) clearly identifies the targets by the red contours.

After this first area search trial the target at the top left corner was examined



(a) Area search map. (b) Search loop track.

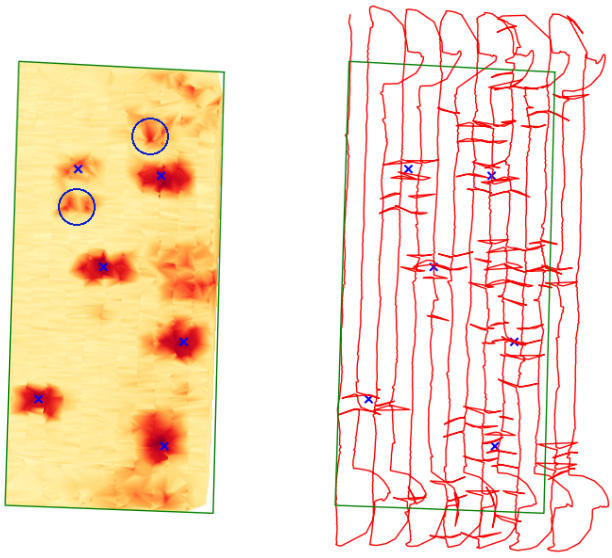
Figure 6.8: *Search map and search loop track for a full area search of the test field. The blue x marks the true locations of the targets. The wriggle behaviours are easily identifiable in the search loop track at a close proximity to the targets. This area search was performed using a low sensitivity.*

to look for any obvious reason for the somewhat weak response. There was no apparent cause but hereafter the target gave an even weaker response in the subsequent trials.

In another area search trial a higher sensitivity was used. Figure 6.9 shows the results which are quite different. The targets are still clearly identified by red contours but the search map shows other apparant identifications as well. To investigate the cause two contours marked by a blue circle in figure 6.9(a) were examined. The leftmost circle turned out to be an old beer can buried at a depth of approximately 0.15 m. The rightmost circle was a larger piece of metal buried at a depth of approximately 0.3 m. Figure 6.10 shows the result of an area search trial performed after removing the two pieces of scrap metal.

There is no doubt that all the contours in figure 6.10 aside from the targets represent metal scrap buried in the ground. When using only a metal detector there is no way to discriminate the scrap from targets of the same size, so all contours would need to be examined in a real demining situation. The results of these area search trials clearly shows that the MDI used in this work requires careful calibration and subsequent interpretation of the search map which only a skilled operator can perform.

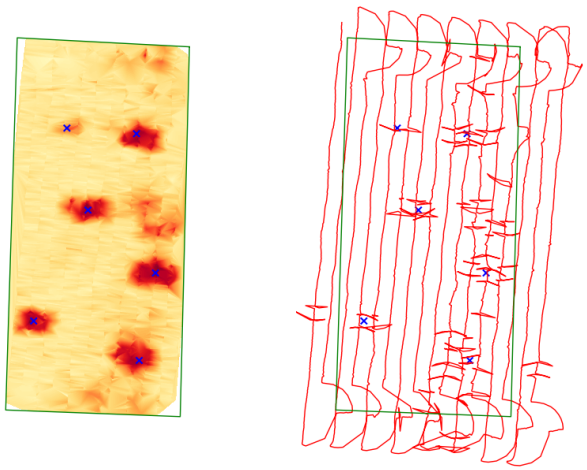
The calibration concerns the MDI, the robot behaviour and the mapping algorithm. Regarding the MDI the *Mode* selector, the *Delay* adjuster and the



(a) The area search map.

(b) Track for the center of the WADS search antenna.

Figure 6.9: *Search map and search loop track for a full area search using a higher sensitivity. Two of the now emerging red contours marked by blue rings in figure 6.9(a) were examinated and scap metal was found in the ground. Figure 6.9(b) shows that the higher sensitivity caused more wriggle behaviours.*



(a) The area search map.

(b) Track for the center of the WADS search antenna.

Figure 6.10: *Search map and search loop track for a full area search using a higher sensitivity. Before the trial the two pieces of scrap metal were removed and these contours have now disappeared.*

Sensitivity adjuster of the UPEX 740M-S detector all have significant impact on the resulting search map. This is no different than during the manual search though. Concerning the robot behaviour there are two parameters which need to be set. One is the distance between neighboring tracks, in these experiments it was set to 0.8 m. This is a rather conservative setting however in one of the performed trials using 1.2 m and a low sensitivity setting the top left target was not recognized at all. The other is the minimum distance between two wriggle behaviours. In these experiments it was set to 0.7 m. The parameters of the mapping algorithm does not seem to affect the resulting map significantly except for the grid cell size which affects the map resolution. Changes in cell size down to about 0.02 m is visible in the resulting map.

The somewhat oval shape for some of the target detections is a result of the MDI scalar measurements. The many scalar measurements along the track clearly defines the extent of the target whereas the closest lateral measurements are from the wriggle behaviour or the adjacent track. This causes the contour function to plot a more gentle gradient. Taking this into account the locations of the targets may be identified from the data with a reasonable high accuracy, but this has to be investigated further. In comparison the manual test described in (Persson, 2010) where the largest deviation from identified target to real target (buried 0.3 m below ground) was 1.4 m.

It should be noted that to increase the reliability of the map and target identifications in real applications, the described procedure should be repeated in two or more independent trials preferably with varying starting points, coverage directions and sensitivity. A comparison of the produced maps will then provide a measure of the area search quality.

Based on the time required to complete the described search using the Pichi robot it would take approximately one hour to complete a 20x40 area. A new prototype of the Pichi will allow the robot to complete the search faster, however the trials have revealed that vibrations from the robot locomotion are transferred to the MDI and this will limit the maximum speed depending on the ground unevenness. The manual search requires three deminers while the robot requires only one operator. Since both tests use the same detector, the mentioned initial calibration should take about the same time. It is possible to reduce the time required for task 6 in the manual test by enabling the robot to visually mark the ground above a target. The processing of search data and map generation can be computed online, so this is where the robot saves significant time compared to the manual search.

In terms of usability the operator assisted autonomous area coverage works well. It is easy to cover large areas will little effort from the operator, and the search loop track documents that the entire area has been covered properly. However it might be a problem that the coverage algorithm requires free space for turning at

each end of the search area which might not be possible everywhere. In addition the robot platform will exceed the maximum ground pressure allowed for safe overpass of some types of hazards. This requires an update of the area coverage planning each time a new potential hazard is identified. Approaches to area coverage which take overpass restrictions into account such as the Crab and the Hybrid techniques should therefore be investigated in future work (MACCA, 2011; GICHD, 2005; Garcia and de Santos, 2004; Acar et al., 2003).

The humanitarian demining application described in this work is detecting anti-tank mines and larger UXO like mortars etc. This particular application was chosen to demonstrate the potential of utilizing robots in humanitarian demining based on recommendations by the organization DanChurchAid who has extensive knowledge and practical experience from the field. However the perspectives are greater than this. An even more relevant application is the ability to sense anti-personnel mines and smaller UXO like cluster munition etc. Here the deminer is at a much higher risk while operating in the area, and sensing the hazards is much more challenging as well. This should be considered in future work.

6.9 Conclusion

In this work the Pichi humanitarian demining robot has been developed based on knowledge and experience precision agriculture. The robot design aims to optimize the design parameters: Reliability, efficiency, usability and low-cost. The first prototype of the Pichi robot has been found to have a number of problems relating to design choices and the quality of the some of the parts used. It has been decided to disassemble the prototype and construct a new version.

A Wide Area Detection System (WADS) developed by the organization DanChurchAid has been retrofitted for use as a mine detection implement. Low-cost pose estimation based on a single frequency RTK system was partly developed in this work.

FroboMind components for area coverage and wriggle behaviours as well as area search mapping have been developed. The robot operation is based on a combination of user assisted autonomous navigation and behavioural response to the measured data.

Field trials have been conducted and the results have been compared to a similar but manual field trial performed by DanChurchAid in Sudan. The main conclusions from this trial are: The trial area was mapped and all simulated hazards were detected. Comparing to data from DanChurchAid the robot is slower at the data acquisition process than manual searching, however only one

operator is required. The operator assisted autonomous area coverage worked well, but overall usability needs to be matured. Calibration and operation of the demining robot and mine detection implement as well as interpretation of the produced area search map is a complex tasks that requires a skilled operator.

Future work is to develop a new prototype of the Pichi demining robot solving the identified problems. Improved methods for area coverage should be investigated. The reliability of the search area mapping should be evaluated more thoroughly and algorithms for identifying target locations and the probability of detection should be developed. Finally application towards anti-personnel mines and smaller UXO such as cluster munition should be considered.

Chapter 7

Field robot prototyping

This chapter describes some of the field robot prototypes and projects to which this work has contributed. All robots run FroboMind and the adaptation hereof was part of this work. For each robot is listed a brief description including applications where the robot is used, the current status, technical specifications etc.

7.1 Research platforms

This section focuses on the field robot prototypes developed for research purposes. Table 2.8 in section 2.4.3 includes the robots described in this section focusing on shared software components and the latest application. Table 7.1 lists some of the technical specifications.

7.1.1 ASuBot

ASuBot (figure 7.1) is an abbreviation for *AU and SDU university Robot*, a joint project between Aarhus University and SDU. The project was launched in parallel with the PhD project with the aim to establish a low-cost autonomous tractor platform for research in precision agriculture tasks.

The first prototype of FroboMind was tested on the ASuBot robot in a research project *FruitGrowth* focusing on weeding in organic orchards. Currently the robot is used by Aarhus University in other research projects.

ASuBot is based upon a Massey Fergusson 38-15 garden tractor retrofitted with a Topcon AES-25 steering system.

	ASuBot	Armadillo	Frobit	Pichi	Frobo-Mower	Frobo-Scout
$L \cdot W \cdot H$ [cm]	225·99·100	105·var·90	35 · 26 · 12	110·100 ·60	45 · 40 · 95	70 · 60 · 100
Weight [kg]	195	450	5	215	10	40
Motor [W]	8,900	10,000	8	4,000	30	140
Batt. [Wh]		5,280	25	2640	50	576
Steering	Ackerm.	Skid	Diff.	Skid	Diff.	Diff.
Payload max [kg]	200	500	2	50	2	5

Table 7.1: *Specifications for the prototypes of research robots to which this work has contributed. Ackerm. means Ackermann, Diff. means Differential. Some of the listed values are based on estimates and they should not be considered accurate.*

Figure 7.1: *The ASuBot robot mounted with sensors for surface contour scanning.*

Figure 7.2: *The Armadillo I robot.*

7.1.2 Armadillo

Armadillo (Jensen et al., 2012c) is a low-cost robotics tool carrier for precision agriculture research. The first Armadillo prototype (figure 7.2) was built in the Spring 2011 and demonstrated at the CIGR section V & NJF section VII conference *Automation and System Technology in Plant Production* July 2011 in Herning, Denmark. The project has now evolved into a series of field robots, all designed and developed at SDU in collaboration with academic and industrial partners.

The Armadillo has a modular design which makes the robot configurable and adaptable to a wide range of precision agriculture research projects. During the design phase the three-point hitch used for attaching implements to agricultural tractors since 1926, was considered. But given the limited footprint and low weight of the platform this could create balancing problems when using heavy or soil manipulating implements, and it would also limit the flexibility of the platform. Instead a more modular design was chosen: The implement is carried by two identical (however mirrored) track modules each self-contained with batteries, motor controller, electric motor and transmission. The width and clearance height are determined by the geometry of the implement which facilitates adaptation to many different tasks within biological production. The track modules are controlled by a robot computer integrated in the implement.

The first prototype is now disassembled. The second version named Armadillo Scout was built for the University of Hohenheim. Armadillo III (figure 7.3) and Armadillo IV (figure 7.4) are both in use by SDU. The fifth version is documented in section 7.3.1.

The specifications vary slightly for the different versions, but overall each track

Figure 7.3: *Armadillo III with a row cleaning implement.*

Figure 7.4: *Armadillo IV with precision weeding frame.*

module weigh about 200-250 kg, the footprint for each track is approximately 0.2x0.8 m, they are propelled by a 5 kW brushless DC motor through a two step drivetrain. The standard battery pack mounted on top of each track contains 4x12V 55Ah AGM VRLA batteries. Driving on flat terrain with no extra load consumes about 500 W.

The motor controller used are the RoboteQ VBL1650 and RoboteQ HBL1650 depending on the version of the Armadillo. There have been many problems with regards to these motor controllers. The firmware as well as the documentation is buggy and it seems very sensible to electromagnetic interference (EMI). Despite its specifications the RoboteQ seems to be unable to perform closed loop velocity control based on feedback from the motor hall elements, the velocity control is therefore currently handled by FroboMind. It has not yet been possible to find a better solution.

The batteries in the standard battery are charged individually using a CTEK MXS 10.0 A charger configured to AGM charging. This setup works very well, the back current drain is less than 1 Ah/month when disconnected and a multi step charging program allows the robot to be plugged in at all times while the robot is not in use.

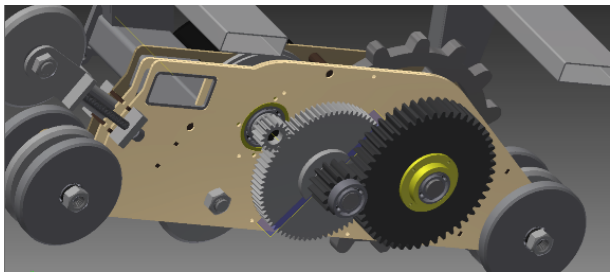
For Armadillo IV a new drivetrain was designed as part of the cell sprayer project described in chapter 5. Figure 7.5 shows the track module. The drivetrain has been redesigned to optimize towards higher efficiency and durability. The frame consists of two metal plates supporting the sprockets for a two step transmission with a combined ratio of 1:19.5. The total weight of a track module excluding the battery case and its content is 97 kg.

All current versions of the Armadillo robot vibrates significantly when driving on hard surfaces due to the mechanical design. A probable cause has been identified: The free running wheels supporting the bottom of the track creates an uneven motion due to the track design. A potential solution is to place two supporting wheels beside the track center instead of the current wheel placed at the track center. Modifications will be performed on one of the Armadillo platforms in the near future to verify the assumption.

The Armadillo robots were developed alongside with FroboMind which has resulted in a close interaction between the two projects. The Armadillo robots and FroboMind have been applied to various research projects within crop monitoring, mechanical weeding and precision spraying. The latest precision spraying project is documented in chapter 5.

7.1.3 Frobit

The Frobit robot (figure 7.6) is a small robot platform developed in this work. It is designed for rapid prototyping of FroboMind field robot applications as



(a) The new drivetrain design.

(b) Image from driving in mud.

Figure 7.5: *Drivetrain of the Armadillo IV track module.*

well as for educational purposes. The Frobit is used as a reference platform for FroboMind which means that FroboMind will always contain updated working applications for the Frobit.

Simulation is a very useful tool when testing new software for a field robot application. But software testing in a simulated environment is only possible up to a certain level, after which it is necessary to perform the tests using a field robot. This is often very time consuming and cumbersome, as the tests typically take place in the field where it is difficult to work efficiently.

A possible solution to this problem is to use small lab-based indoor robots to prototype large outdoor vehicles in an agricultural environment (Edwards et al., 2012; Hagras et al., 2001). This allows the developer to perform rapid prototyping and intermediate functional testing of the software in a physical environment before moving to testing in the field.

The advantage of using the Frobit for rapid prototyping and intermediate functional testing is that however small and simply constructed as it is, the fundamental design is similar to the majority of differential and skid steered field robots using wheeled and tracked propulsion. This means that the workflow of migrating from simulation to Frobit to field robot (figure 7.7) during the process of testing a new application is possible simply by switching between the lower level FroboMind components that serve as interface to the robot hardware. In

Figure 7.6: *The Frobit V1 robot with a small Lidar mounted in front.*

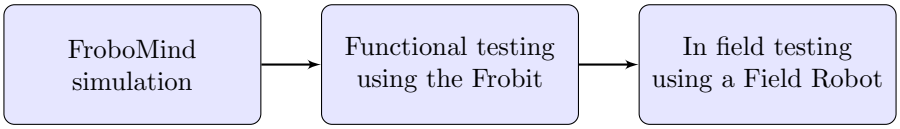


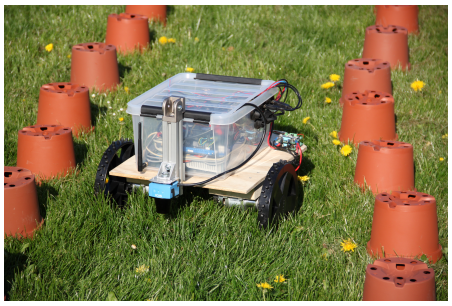
Figure 7.7: *FroboMind workflow for software testing.*

addition it is capable of carrying navigation sensors often used in the field such as cameras, Lidar or RTK-GNSS.

One example is the waypoint arrival accuracy experiment described in section 4.3.2. Here a Frobit was used to assess if unexpected inaccuracies in the results from a field trial were caused by the FroboMind navigation algorithms or by the FroboScout robot performing the navigation. An experiment was defined where the Frobit performed a similar navigation trial using the odometry as pose estimation. The trial was performed with 5 repetitions indoor using the Frobit. The data was post-processed and the documentation added to this thesis. This entire process took less than 3 hours which is a very short time compared to performing a similar trial in the field.

The Frobit has also been used in the education of engineering students including thesis projects and courses. In one example it was used in an extracurricular activity for first year engineering students. The activity was named *Practical experience from day one*. Its primary objective was to improve the participants' practical experience and give them a broader view of technology applied in agriculture. The results of this activity are published in (Larsen et al., 2013).

The Frobit robot has been constructed in different variants and sizes. Figure 7.8(a) shows an example, a larger *Frobyte* developed by undergraduate robotics students at SDU for local navigation tasks in maize rows. The Frobyte is ex-



(a) The Frobyte robot for navigation in maize rows developed by robotics students at SDU.

(b) A Frobit robot for SLAM and RTK-GNSS experiments developed by Conpleks Innovation.

Figure 7.8: *Examples of other Frobit variants.*

pected to participate in the FieldRobotEvent 2014 (Henten and Müller, 2007). Figure 7.8(b) shows another example developed by the industrial partner Conpleks Innovation for experiments and rapid prototyping within Simultaneous Localization And Mapping (SLAM) and RTK-GNSS applications.

Description

The Frobit consists of a wooden plate that serves as a stand for the laptop and sensors. It has two parallel wheels and motion is based on differential drive. A third castor wheel supports the robot, it is passive and does not contribute to the robot motion. The kinematics is derived in 4.2.3.1.

The design goals are: *Keep it simple, low-cost, small and light weight, open source, capable of carrying a laptop and sensors.* It is simple and easy to construct even with limited practical skills, and parts are available through webshops. The cost of hardware for the Frobit V1 is approximately 150 Euro.

Figure 7.9 shows the Frobit upside down. The wheels are mounted directly on the shaft of 12V DC brushed motors with a built in 30:1 reduction gearbox and a rotary encoder giving 180 quadrature pulses per drive shaft turn. The motors are interfaced through a dual H-bridge motor controller. Communication with the laptop takes place through a USB or Bluetooth serial port using simple messages based on the NMEA 0183 protocol. The robot is powered by a 12V 2.1Ah VRLA battery.

The robot is controlled by a RoboCard (figure 7.10) which was developed within this work. RoboCard is a micro controller board designed for educational and rapid prototyping purposes which is very much in line with the Frobit robot. RoboCard is based on the Atmel ATmega 328 series microcontroller. It differs from other microcontroller boards by not using any Surface Mounted Devices (SMD) components and by facilitating an on-board prototyping area. Recently the *FroboMind Controller* (figure 7.11) was developed in this work as well. The

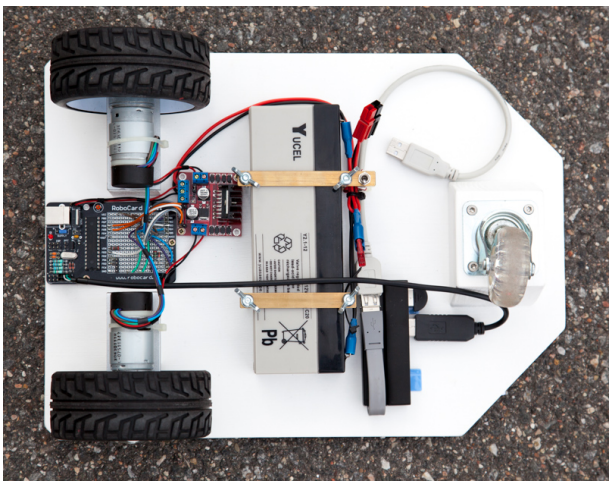


Figure 7.9: *The Frobit upside down. The RoboCard is mounted in between the two motors. The black cap on each motor contain the encoder. The red PCB is the dual H-bridge motorcontroller. The black box at the lower right is a 4 port USB hub.*

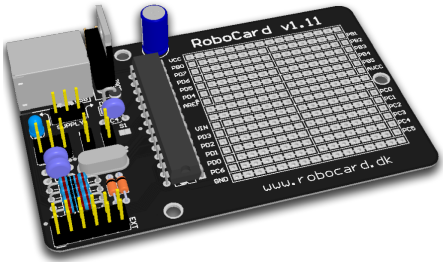
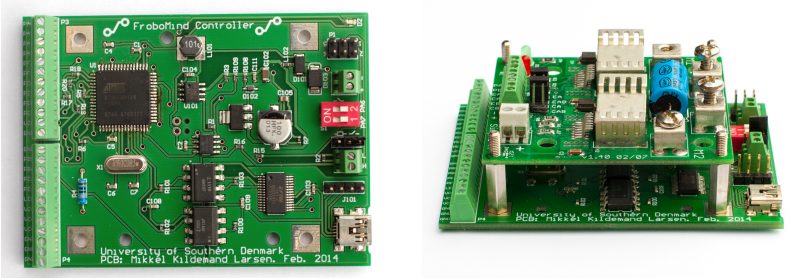


Figure 7.10: *The RoboCard used for controlling the Frobit.*



(a) FroboMind Controller PCB.

(b) FroboMind Controller with a Simple-H motor controller installed.

Figure 7.11: *FroboMind Controller*

FroboMind Controller is designed specifically for field robot applications and will likely replace the RoboCard for controlling larger Frobits. The overall specifications are Atmel AT90CAN128 microcontroller, Controller Area Network (CAN) interface, opto-isolated USB serial port, easy direct access to relevant I/O , 6.5-42V DC in and battery voltage monitor. Both boards are released using a Creative Commons license and documentation is available through the FroboMind website.

The Frobit firmware is written in C for the Atmel ATmega 328 series microcontroller. It supports open loop as well as closed loop PID control of the wheel velocity. It takes left and right velocity as input and returns ticks from the wheel encoders, the battery voltage and propulsion status parameters. The firmware and hardware documentation are open source and are available at the FroboMind website.

7.1.4 Pichi

The Pichi robot (figure 7.12) is a small all-terrain robot developed for humanitarian demining tasks in collaboration with the industrial partner Lynex. The mechanical frame is designed by Lynex and the majority of the installation of electric wiring and electronics was carried out as part of a humanitarian demining student project.

In many regards the Pichi is a scaled down version of the Armadillo robot, however it is not modular. Instead it is based on a fixed size frame which makes the mechanical construction more light weight and much simpler to repair in the field with limited access to tools and spare parts.

The Pichi robot is skid steered using rubber tracks designed for a mini excavator. The frame measures 1.1x1.0 m and each track footprint is 0.15x0.80 m. The base is 0.6 m above ground, the highest point on the robot is the mast 1.2 m above ground. Except for the aluminium mast the frame is made of squared iron tubes plated with 20 mm waterproof plywood. The total weight of the platform is 215 kg including the battery pack which weighs 75 kg. It can easily be transported on a standard trailer or pickup.

Each track module (figure 7.13) is powered by a 2 kW brushless DC motor. The motor shaft is attached to a sealed 15:1 reduction planetary gearbox. The top speed is approximately 7 km/h. The two track modules are controlled by a 2x75A RoboteQ motor controller connected to a battery pack of 4 x 12V, 55Ah AGM VRLA batteries. Another 12V battery supplies the emergency stop system, the motor controller logic and the sensors.

The first prototype of the Pichi robot has been found to have a number of problems with the motors and motor controllers. They need to be upgraded to a better quality to increase reliability and lower the power consumption.

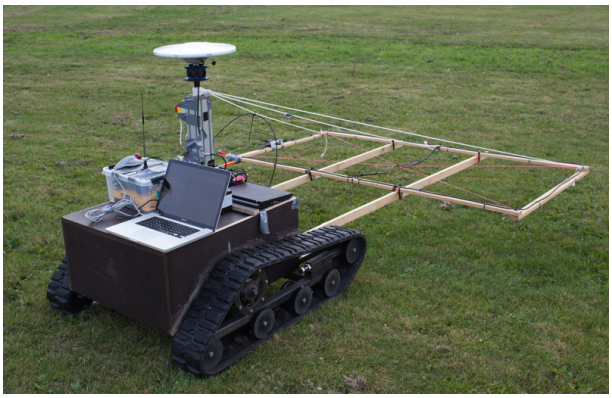


Figure 7.12: *The Pichi demining robot.*

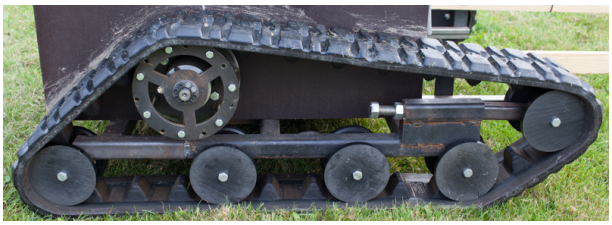


Figure 7.13: *The Pichi track module seen from the side. The motor shaft is connected to the top wheel through the gearbox.*

The track configuration needs to be redesigned to support smoother turning using less power. When driving on hard surfaces the part of the track touching the ground is too long. The AGM VRLA batteries contributes to the turning problem by making the robot quite heavy. However they are still considered the best solution since they are maintenance free and easy to handle in terms of recharging. More advanced batteries such as Lithium ion phosphate technologies have a much higher capacity/weight ratio and will improve the operation time per charge considerably. But at the same time this will also make the robot significantly more expensive, and it will be more difficult to repair and obtain spare parts in the field.

Based on an evaluation of the above described problems it has been decided to disassemble the prototype and construct a new version.

7.1.5 FroboMower

The FroboMower robot (figure 7.14) is a small lawn moving robot developed within this work. The purpose of FroboMower is to perform field experiments related to grass cutting and precision weeding. Examples of topics are area coverage, state estimation based on low-cost sensors other than the typical wire

Figure 7.14: *The FroboMower robot.*

buried along the edge of the lawn, robust autonomous behaviour etc. The FroboMower robot is operational, it was used for the trials described in section 3.4.3.

The FroboMower platform origins from a scrapped low-cost commercial robotic mower. Everything except the chassis, cutters, motors and transmission have been removed. Figure 7.15 shows the interior of the FroboMower robot. The original electronics has been replaced by a RoboCard described in section 7.1.3 running the Frobit firmware. FroboMower is thus supported by FroboMind through the Frobit interface component. The RoboCard controls the propulsion motors (24V 15W) through a H-bridge motor controller, and low-cost optical rotary encoders have been installed on the wheel shafts to provide odometry feedback. The three cutter motors are controlled by the RoboCard as well. An IMU, a mast for a GNSS antenna and a laptop platform have been installed to support the experimental work. The robot is powered by two 12V 2.1Ah VRLA batteries.

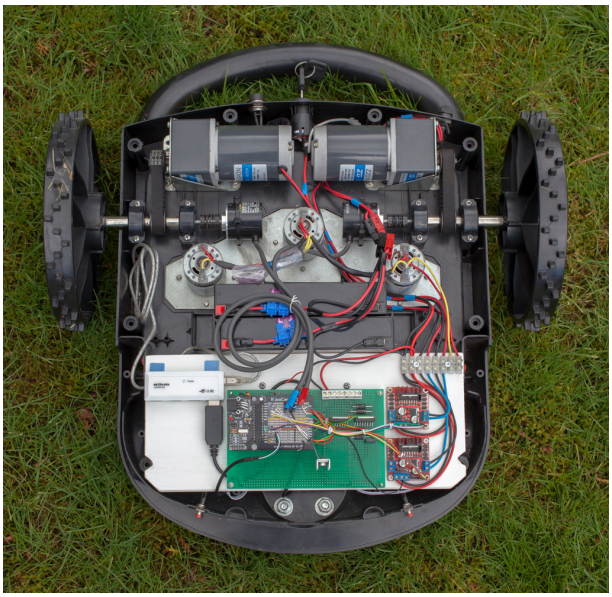


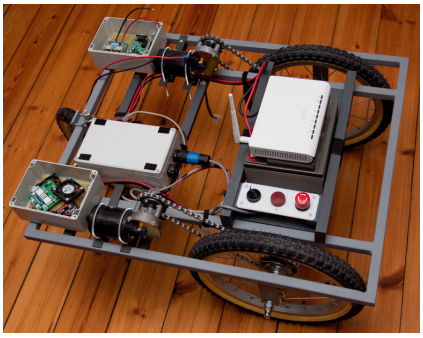
Figure 7.15: *The interior FroboMower robot. The green PCB holds the RoboCard which interfaces to the laptop. The two red PCB's are the H-bridge motorcontrollers.*

7.1.6 FroboScout

FroboScout (figure 7.21) is a small differential steered field robot designed for various scouting tasks as well as test platform for field robot software. The first hardware prototype was developed in previous work by the author. In this work FroboScout has been modified to support FroboMind, and a FroboMind interface component has been developed.

The first hardware prototype (figure 7.16(a)) is based on a simple iron frame with two 16" bicycle propulsion wheels and a free running 12" bicycle castor wheel. The wheels are powered by windscreen wiper motors designed for the Wolksvagen Golf car and the transmission is made of bicycle sprockets and chains. Since the windscreen wiper motors have one of the power terminals connected to the chassis, the FroboScout is running two independent 12V DC circuits and the motor controllers based on Atmel AT90CAN128 microprocessors are opto-isolated from the laptop controlling the robot. The battery used for each of the two circuits are 12V 26Ah AGM VRLA. During trials the robot has been able to navigate more than 1.5 km during 3 hours of operation without the battery voltage at rest falling below 12.0V.

In a current project FroboScout is used for surveying tasks that are part of the quality assurance when constructing new roads. A description of this project is found in section 7.3.4.



(a) The FroboScout robot under development. The drivetrain consists of the two WV Golf windscreen wiper motors connected to the wheels using bicycle sprockets and chains.

(b) New 16" wheel module with bicycle hub motor for the next FroboScout version.

Figure 7.16: *Development of the FroboScout robot hardware.*

Given the intensive use of the robot caused by the surveying project, the drivetrain has begun to show its limitations. The motor has a built-in worm gear made of plastic, so the efficiency is low and the motor velocity is difficult to control accurately. The current motor bracket is of low quality so the bicycle chain becomes loose during operation. This was to be expected though, the robot has definitely outperformed its intended use.

A new and more capable version of the FroboScout is currently being developed. The most significant upgrade from the first prototype is that the drivetrain will be replaced by wheel modules containing hub motors made for electric bicycles (figure 7.16(b)).

7.2 Student robots

Each year in August SDU hosts a 4 week university summer course for engineering students. The course is titled *Design and development of field robots for plant nursing* and is inspired by the annual FieldRobotEvent competition. The students work in interdisciplinary project groups comprising areas of specialties such as design, mechanics, mechatronics, robotics etc. Each group builds their own field robot from scratch and compete in the competition at the last day of the course (figure 7.18).

The planning and execution of the summer course is part of this work and FroboMind has been included in the curriculum for the past couple of years. This has sparked several student robots running FroboMind. Figure 7.17 shows some of the latest student field robots.

Figure 7.17: *Examples of student robots from the SDU field robot summer course.*

Figure 7.18: *Field robot competition at the last day of the summer course.*

7.3 Towards industrial applications

Though FroboMind is designed for use in experimental research the industry shows interest in FroboMind as well. FroboMind is not yet mature for industrial applications in terms of reliability, but it serves as a good starting point for companies producing machinery for agriculture, horticulture, forestry etc. FroboMind benefits from the industrial collaborations as well, as it enforces a focus on reliability and stability of both the software platform as a whole and of the different components. Examples of FroboMind used in industrial applications will be presented in the following.

7.3.1 Kongskilde Robotti

The first example of the industry using FroboMind is the *Robotti* by Kongskilde Industries (figure 7.19). Robotti consists of tracked modules attachable with different forms of working implements. It is currently fitted with a Kongskilde Industries implement section for mechanical weed control. The Robotti track modules are a further development of the Armadillo track modules presented in section 7.1.2. SDU contributed to the Robotti project with the FroboMind software platform and design of the track modules. These contributions were coordinated within this work. Robotti was presented for the first time to the Global market at AgriTechnica, 2013 in Hannover, Germany.

Figure 7.19: *Robotti by Kongskilde Industries.*

7.3.2 Conpleks Innovation

Conpleks Innovation, a Research and Development company in Denmark, develops advanced robotic software solutions for machinery manufacturers. Conpleks Innovation uses FroboMind as a basis for some of their robotics products to the agricultural industry and has recently established two PhD student positions in collaboration with Kongskilde Industries, Aarhus University and SDU. The positions build upon this work and are titled "Model-driven Software Development for Agricultural Robotics" and "Towards a generic software platform for agricultural robotics based on autopoietic separation of safety and functionality".

7.3.3 Grassbots

GrassBots is a current project concerning grassland harvesting operations for bio gas and bio refinery plants. Within the project a novel lightweight, autonomous machine concept for economically and environmentally sound harvest of grass on lowland is being developed (figure 7.20) .

The coordination of the SDU contributions to this project is part of this work. This includes porting the FroboMind autonomous navigation of route plans documented in chapter 4 to the Grassbots robot in collaboration with the participating industrial partners. At the current stage the Grassbots robot is capable of performing the autonomous navigation, but it is not yet well tested. Extensive tests will be performed in the fall 2014.

Grassbot consists of a Lynex tracked tool carrier and a grass cutting implement from Kongskilde Industries. The cutting width is 3 m, the propulsion is hydraulic driven tracks powered by a 100 hp generator. The robot weighs about 2000 kg.

Figure 7.20: *The Grassbots robot.*

7.3.4 Surveying robot

FroboMind is also used in a surveying project in collaboration with a land surveying company. The project focuses on quality assurance measurements during highway construction (figure 7.21 and 7.22). The construction includes adding three layers of asphalt. At each layer a surveyor performs quality assurance in the form of accurate height measurements typically each 5-20 meters. The task is monotonous and tedious and fits perfectly to a small field robot. One of the main challenges in this project is the required high accuracy which is achieved by combining the robot measurements with data from a surveyor total station.

The SDU contributions to this project is part of this work. This includes application of the FroboScout robot described in section 7.1.6 and FroboMind to the task. At the current stage the FroboScout robot is capable of performing the surveying task at a level which makes the robot useful to the surveyor. Several tests have been performed using manual surveying measurements as ground truth reference, and to test the commercial feasibility the FroboScout has been used by the surveying company for production work for a month. The results hereof are positive, the robot is capable of delivering measurement results at an acceptable accuracy, and the time used for the measurements is approximately half of the time used by an employed surveyor under similar conditions. The collaboration is expected to be extended in order to establish a permanent solution for the surveying company.

7.4 Lessons learned

This section describes some of the lessons learned from the work designing and developing the field robot prototypes described in this chapter:

- Developing track modules for the Armadillo robots has been a learning

Figure 7.21: *The FroboScout robot.*

Figure 7.22: *The FroboScout robot used for surveying tasks during highway construction.*

experience in several regards. The Armadillo robot differ from most other tracked vehicles this size by being driven by an electrical motor powered by a battery pack. The use of the Armadillo robot for precision tasks also places other demands than the typical use case, and expensive and sensitive equipment is often mounted on the robot. In terms of vibrations the mechanical design must therefore seek to minimize vibrations caused by the drivetrain itself to the extent possible. As mentioned in section 7.1.2 the current versions of the Armadillo robot suffers from vibrations when driving on hard surfaces and this limits the maximum operational velocity and potential applications of the robot significantly.

- A differential steered vehicle requires considerable more power when turning about its own center compared to driving forward. This is very significant for skid steered tracked platforms such as the Armadillo and Pichi robots. The new version of the Pichi robot described in section 7.1.4 will therefore have a slightly modified track design. Instead of a completely flat track surface against the ground, only the middle 0.5 m will be flat. Towards the front and rear end the tracks will be elevated slightly to make turning on flat surfaces easier without affecting the driving abilities on rough surfaces. Another experience is that when designing velocity controllers it is a good idea to use a separate set of control parameters for driving and for turning operations. This makes parameter tuning much easier.

- Small and medium sized field robots used in research projects rarely exceed velocities of 2 m/s during operation. Often the forward velocity is between 0.25 and 1 m/s depending on the task. In the planning phase there often is a desire to perform the given task at a higher velocity, but at the time the experiment moves to the field the reality tends to change the perspective. When designing the drivetrain for a field robot this should be taken into account. Drivetrains designed for higher velocities often provide less torque and may complicate the velocity control at these low velocities.

- It is a common misconception that vehicle odometry for a tracked vehicle is very inaccurate. The example described in section 3.2.2 shows an odometry error of approximately 1 m after navigating 48 m which is comparable to the other experiments performed in this work. This makes the odometry supported by an IMU very useable when fused with local or global sensors outputting infrequent absolute pose updates.

- EMI is a common problems in experimental field robotics. The relatively powerful motors often induce noise, and the small size of the robot means that power cables are located close to small signal cables such as USB and TTL-level serial cables that make perfect 'antennas'. The following recommendations based on experiences may seem trivial to the expert,

but nevertheless these problems are often found in field robotics: Keep a common ground using a cable of adequate size. Use a star topology for grounding, do not create a ring topology which may result in ground loops. Keep signal wires away from power wires, make sure they cross rather than follow each other in parallel. Use optical isolation of the signals whenever possible. Use shielded cables with one end of the shield connected to the common ground. Use twisted cable for balanced signals such as ethernet, CAN, RS-485, RS-422 etc.

- Most of the field robots described in this chapter are using AGM VRLA batteries as power source. The only issue with this battery type is the poor capacity vs. weight ratio which makes the robots relatively heavy and/or shortens the operation time. The Armadillo Scout described in section 7.1.2 was fitted with Lithium Iron Phosphate (LiFePO4) batteries that packs about 3 times the capacity pr. weight unit. The downside is that they are definitely not as forgiving as VRLA batteries. Voltages and currents need to be monitored strictly while in use as well as during charging. In one instance one of the cells on the Armadillo Scout exploded during charging despite best efforts to perform a correct installation. For a robot product where the voltages and currents can be controlled well, LiFePO4 might be the right solution, but for a research robot this is not recommended. The VRLA batteries are very good at absorbing Back-EMF, and the charging process is seamless as described in section 7.1.2.

- Field robots used in research experiments often have to carry significant amounts of extra equipment such as sensors, cameras, laptops etc. As trivial as it may sound it is a very good idea to take this into account when designing the robot. There should be enough space for the equipment at places where it is protected from the environment. Laptops should be easily accessible while operating in the field. There has to be external power available for the laptops, otherwise the experiment has to stop when the first laptop battery is empty. Also it is a good idea to make room for a separate battery. The above described problems with grounding potentials and ground loops are common, and having a battery that floats with respect to the common ground may in some situations prove to be useful.

7.5 Discussion

The various field robot projects described in this chapter have been a significant part of this work. Even though the direct contributions to the larger field robots has been primarily FroboMind software components, conducting field experiments, coordination etc. it does require in-depth knowledge and experience

with the robot platform hardware which is very time consuming.

The perhaps most interesting part of this chapter is the industrial applications that have emerged during the past year. FroboMind was intended for research experiments, so the industrial interest has been a positive surprise. Based on statements from industrial partners the main reason is that companies aiming towards commercial field robotic products experiences a significant challenge in acquiring the knowledge and experience to design and write the robot software. The working examples provided by FroboMind facilitate this process and the permissive free license allows the companies to use the software as a basis for their own projects without concerns for subsequent license limitations or fees. In a masters thesis related to a collaboration between the Norwegian University of Science and Technology and the company Adigo AS about developing a field robot, the conclusion included: *"The FroboMind platform has proven to be a good choice for the system, with a logical and easy to understand hierarchy, and several built in functions, easing the work of implementation of sensors and control nodes"* (Lien, 2013).

At the same time this impose several challenges. The FroboMind design goals did not include safety and reliability which is paramount to the industrial applications, and until now little effort has been put into developing a proper safety system within FroboMind. Fortunately the companies are addressing this issue, and this is expected to reflect back on FroboMind. One example is a newly launched innovation project named Safer Autonomous Farming Equipment (SAFE) which will be one of the direct continuations of this work. Part of this project focus on autonomous proactive and reactive responses to internal and external safety threats, and it is expected that FroboMind will be used during the development process.

The application of field robots to quality assurance surveying tasks during highway construction has not been found elsewhere in the literature and is considered novel. A paper is currently being drafted based on the findings in this project.

7.6 Conclusion

In this chapter the field robots and related projects to which this work has contributed are described briefly. All robots run FroboMind and the adaptation hereof was part of this work. The list is not exhaustive, smaller projects focusing on implements and sensors have been left out, and a few projects are not mentioned due to agreements with industrial partners about confidentiality.

FroboMind has proven to serve as a good basis for companies constructing new autonomous machines for the field environment. Some emerging industrial applications are presented, namely the Kongskilde Robotti, the Grassbots project

and the Surveying robot. Dissemination of knowledge to these industrial applications is part of this work.

Chapter 8

Conclusion

This chapter contains the overall conclusions of the project. Future work is embedded in the last part of the conclusion.

The activities in this work have focused on the development and integration of low-cost field robot systems to support experimental research and prototyping in biological production and similar applications. The activities include dissemination of knowledge to researchers and the industry with the objective of enhancing research through extensive field experiments and facilitate the development of new innovative products.

This thesis addresses the main part of this work, which is robot software solutions to control field robots performing precision tasks in outdoor environments. The hypothesis is that an application oriented open software platform for multi-purpose field robotics will reduce the resources required for experimental research considerably due to reuse of existing work across projects and robotic platforms.

FroboMind
The FroboMind software platform developed in this work consists of an operating system, a middleware, an architecture and software components. Ubuntu Linux was chosen as operating system and ROS as middleware based upon an evaluation of available solutions against the defined design goals. This work adds thereto an architecture and a number of components designed for field robotic navigation and implement operation. The architecture is based on the representation of intelligent agents and consist of the principal *Perception*, *Decision making* and *Action* layers. The layers have been decomposed and expanded, and data interfaces have been defined to facilitate efficient reuse of software components across robot platforms and implements.

Providing metrics to assess the performance of FroboMind has proven difficult. In an experiment evaluating the performance of FroboMind in robot systems

with near real-time requirements it was concluded that FroboMind's soft real-time execution is sufficient for typical field robot tasks. Another experiment was performed in which FroboMind was ported to a new field robot and applied to a task involving autonomous navigation in an orchard. This was achieved by 4 developers during a 5 day workshop, which is a very short time moving from a completely new software implementation on a robot to trials in the field navigating autonomously. In another experiment the software reuse across projects has been assessed based on the number of *Source Lines of Code* for some of the reused FroboMind components. It was concluded that the level of software reuse between the projects is high

FroboMind components
FroboMind components for global pose estimation based on low-cost sensors have been described in this thesis. The estimated absolute position and orientation were tested in field experiments where RTK-GNSS data was recorded as ground truth reference. A subsequent qualitative analysis shows that the estimated pose accurately extrapolate the infrequent GNSS updates. In another experiment the pose estimation performance was qualitatively evaluated by simulating a 30 s GNSS satellite signal outage. It was concluded that the pose estimator outputs reasonably good pose updates based on the available odometry updates during the outage.

The work concerning global pose estimation included a study of the applicability of low-cost single frequency GNSS receivers. In one experiment a standalone GPS module utilizing Precise Point Positioning (PPP) algorithms and Satellite Based Augmention System (SBAS) was tested in a field trial. The 95th percentile of the position errors was 1.9 m. This indicate that it has limited use in precision agriculture except for overall localization in orchards etc. In a similar experiment a Low-cost RTK System (LCRS) based on a single-frequency GPS module and the open source RTKLIB software was tested in a field trial. The 95th percentile of the position errors was 0.034 m. The mean was 0.08 m corresponding approximately to the theoretical longitudinal error induced by a delay on the GNSS data. The measured accuracy of the LCRS indicates a high potential for use in field robot applications, however it should be noted that the LCRS does experience varying periods of RTK float solutions where the accuracy is significantly lower. During these periods the field robot may have to stop navigating and wait for the RTK fixed solution to reoccur.

The FroboMind components for autonomous navigation of differential steered robots have been described in this thesis. In one experiment the navigation along straight paths in row crops has been evaluated. The 95th percentile of the lateral distance from the path was 0.025 m which is fully adequate for most precision agriculture tasks and on par with results published about similar work. In another experiment the ability to stop at an exact spot was evaluated. The resulting 95th percentile of the error was 0.064 m which is sufficient, however

there is room for improvement. Results from another experiment indicate that part of the inaccuracy origins from the FroboScout robot used in the trials. It was concluded that the FroboMind autonomous navigation components deliver results that are adequate for field robot applications.

The FroboMind navigation components implemented in this work is not a novelty as such. Similar systems are described in other work and also exist as commercial products for agricultural tractors. However only the results and findings are available, the software is not available to researchers and the industry, which argues in favour of the underlying rationale behind FroboMind.

Case studies
Two different FroboMind use cases are presented in this thesis. The first concerns navigation of a novel cell sprayer designed for large scale tests of weed detection algorithms and spraying strategies. The integration of the systems and the navigation performance in the field are discussed. The cell sprayer consists of a spray implement carried by an Armadillo field robot. In this work new FroboMind components including the global pose estimation and waypoint navigation mentioned above as well as polygon mapping for implement control were developed. The cell sprayer completed three trials where it navigated a 2.5 km route through the rows of a 1.4 ha maize field autonomously. Precision spraying of the weed was performed within the crop rows in the first two trials according to a spray plan defining 12 different treatments applied to 566 parcels. In an experiment the navigation accuracy was measured. Over a distance of approximately 145 m the 95th percentile of the absolute distance to the AB line is 0.025 m and the 95th percentile of the absolute heading error is 1.17 degrees. It was concluded that the Armadillo robot is capable of navigating the test field within the requirements of the spray implement.

The second use case concerns development of a low-cost robot for humanitarian demining tasks. The task considered in this work is autonomous mapping of areas contaminated by landmines and unexploded ordnance. The advantages of using robotics in humanitarian demining are obvious. Searching for landmines and unexploded ordnance is slow and monotonous task. It often takes place under extreme conditions wearing heavy protection suits in hot climates, and the deminers are exposed to great danger while working in the mine field This work focus on utilizing the knowledge and experience from robotics in precision agriculture for humanitarian demining tasks. A low-cost field robot was developed, and a Wide Area Detection System (WADS) designed to be carried by three deminers was retrofitted as a mine detection implement for the robot. FroboMind components for area coverage and wriggle behaviours as well as area search mapping have been developed. The robot operation is based on a combination of user assisted autonomous navigation and behavioural response to the measured data. Field trials have been conducted and the results have been compared to a similar but manual field trial performed by the organization

DanChurchAid in Sudan. The main conclusions are: The trial area was mapped and all simulated hazards were detected, however calibration and operation of the demining robot and mine detection implement as well as interpretation of the produced area search map is a complex tasks that requires a skilled operator. The first prototype of the Pichi robot has been found to have a number of problems relating to design choices and the quality of the some of the parts used. This requires construction of a new prototype. The operator assisted autonomous area coverage worked well, but improved methods for area coverage in mine fields should be investigated. The reliability of the search area mapping should be evaluated more thoroughly and algorithms for identifying target locations and the probability of detection should be developed. Application towards anti-personnel mines and smaller UXO such as cluster munition should be considered.

Usability
FroboMind has evolved through use in research projects since 2011. The software matures gradually as new components, bug fixes and enhancements are continuously added, and many of the approximately 100 software components are fully functional. Some of the components however are characterized by the fact that they were developed to fill a gap in the architecture rather than to investigate the optimal solution to the given problem. They can be improved, for example by using more efficient algorithms or filters. Hopefully this will happen over time, now that FroboMind provides a functioning software platform. The overall experience is that FroboMind has reached a level where users only have to invest few development resources in migrating to a new robot or application.

Though FroboMind is designed for use in experimental research, the industry has started to show interest in FroboMind as well. FroboMind is not considered mature for industrial applications as fundamental safety measures are not yet properly implemented. But it serves as a good basis for companies constructing new autonomous machines for the field environment. Examples of industrial use of FroboMind as a starting point for commercial products are the *Kongskilde Robotti* lightweight tool carrier, the *Grassbots* autonomous machine concept for harvest of grass on lowland, and the *FroboScout survey robot* for quality assurance measurements during highway construction. Dissemination of knowledge to these industrial applications is part of this work.

Future work
At the University of Southern Denmark the continuing work with FroboMind will include further research in sensing and perception to build a better knowledge of the local environment, and in complex behaviours to respond appropriately to this knowledge. This will include online collaboration with external entities such as unmanned aerial systems for enhancing sensing capabilities and performing collaborative tasks.

The growing interest in FroboMind by industrial collaborators as well as interfacing to larger and more powerful robots will require more focus on software reliability and safety. This will among other be addressed in the newly launched innovation project *"Safer Autonomous Farming Equipment"*.

The development of the FroboMind software platform is to some extent defined by the research and innovation projects through which FroboMind is continually improved and extended. Some of the projects described in this thesis are ongoing and will thus continue to contribute to FroboMind throughout the project periods. In addition new innovation projects have been financed as a result of this work. These projects deal with applications within mechanical weeding, grass harvesting, mowing and transportation.

The field robot prototypes discussed in this thesis will continue to be developed. Specifically the work in the near future will concern development of new prototypes of the Pichi and FroboScout robots, a larger version of the Frobit for outdoor use and a study of the possibilities of solving the problem of vibrations on the Armadillo robot.

Conclusion

We conclude that the aim of this work has been achieved with the development of the FroboMind software platform. The application of FroboMind to the projects described in this thesis has shown that reuse across projects and robotic platforms saves considerable amounts of time and resources. FroboMind facilitates research in enhanced perception and sophisticated behaviour based autonomy which will hopefully take us far beyond the automatic tractor described in (Morgan, 1958).

The FroboMind software architecture and components have been released as permissive free open-source software for others to build upon. It is available at `http://www.frobomind.org`

List of Tables

List of Figures

Bibliography

E. U. Acar, H. Choset, Y. G. Zhang, and M. Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *International Journal of Robotics Research*, 22(7-8): 441–466, July 2003.

T. Bakker, K. van Asselt, J. Bontsema, J. Müller, and G. van Straten. Systematic design of an autonomous platform for robotic weeding. *Journal of Terramechanics*, 47(2):63–73, Apr. 2010.

T. Bakker, K. van Asselt, J. Bontsema, J. Müller, and G. van Straten. Autonomous navigation using a robot platform in a sugar beet field. *Biosystems Engineering*, 109(4):357 – 368, 2011. ISSN 1537-5110.

F. Baraldi, A. Castellini, R. Ghelfi, A. Palmieri, C. Pirazzoli, and S. Rivaroli. The labour factor in agriculture: Comparison, analysis and actions introduced in some eu countries to boost competitiveness in the primary sector. the 10th Joint Conference on Food, Agriculture and the Environment, Duluth, Minnesota, August 2006.

O. C. Barawid, A. Mizushima, K. Ishii, and N. Noguchi. Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application. *Biosystems Engineering*, 96(2):139–149, Feb. 2007.

Y. Baudoin. Mobile robotic systems facing the humanitarian demining problem state of the art (sota). Technical report, Royal Military Academy (RMA), 30 Av de la Renaissance, B 1000 Brussels, Belgium, December 2005.

Y. Baudoin and M. K. Habib. *Using robots in hazardous environments*, volume 1. Woodhead Publishing, 2011.

A. B. Beck, N. A. Andersen, J. C. Andersen, and O. Ravn. Mobotware—a plug-in based framework for mobile robots. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, volume 7, pages 127–132, 2010.

V. Belotti, M. U. Hemapala, R. C. Michelini, and R. P. Razzoli. Lean robotics for humanitarian mine sweeping, 2009.

T. W. Berge, T. Utstumo, and J. Netland. Field robots for research and developments in site-specific weed management - norwegian activities. In *First RHEA International Conference on Robotics and associated High-technologies and Equipment for Agriculture*, 2012.

D. M. Bevly. *GNSS for vehicle control*. Artech House, 2010.

J. I.-G. Bingbing Liu, Martin Adams. Multi-aided inertial navigation for ground vehicles in outdoor uneven environments. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation Barcelona, Spain, April 2005*, 2005.

B. S. Blackmore, H. W. Griepentrog, H. Nielsen, M. Nørremark, and J. Resting-Jeppersen. Development of a deterministic autonomous tractor, 2004.

R. Blas. *Fault-Tolerant Vision for Vehicle Guidance in Agriculture*. PhD thesis, DTU, 2010.

S. Bouraine, T. Fraichard, and H. Salhi. Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 174–179, 2012.

R. A. Brooks. A robust layered control-system for a mobile robot. *Ieee Journal of Robotics and Automation*, 2(1):14–23, Mar. 1986.

H. Bruyninckx. Open robot control software: the orocos project, 2001.

S. F.-T. G. B.S. Blackmore, H.W. Griepentrog. A specification for an autonomous crop production mechanization system. *CIGR Ejournal*, IX: Manuscript PM 06 032, 2007.

G. Cai, B. Chen, and T. Lee. An overview on development of miniature unmanned rotorcraft systems. *Frontiers of Electrical and Electronic Engineering in China*, 5(1):1–14, 2010. ISSN 1673-3460.

Y. Cai, P. Cheng, X. Meng, W. Tang, and C. Shi. Using network rtk corrections and low-cost gps receiver for precise mass market positioning and navigation applications. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 345–349, 2011.

T. G. Carsten Emde. Quality assessment of real-time linux. Technical report, Open Source Automation Development Lab, 2011.

J. Cepeda, L. Chaimowicz, and R. Soto. Exploring microsoft robotics studio as a mechanism for service-oriented robotics. In *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*, pages 7 –12, oct. 2010.

J.-M. Codol, M. Poncelet, A. Monin, and M. Devy. Safety robotic lawnmower with precise and low-cost l1-only rtk-gps positioning. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS Workshop on Perception and Navigation for Autonomous Vehicles in Human Environment., September 2011.

R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Robotics Institute, Pittsburgh, PA, January 1992.

L. Dozio and P. Mantegazza. Linux real time application interface (rtai) in low cost high performance motion control. In *Motion Control 2003, a conference of ANIPLA, Associazione Nazionale Italiana per l'Automazione (National Italian Association for Automation)*, Milano, Italy, March 2003.

G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2 edition, 2010.

M. E. Durocher and A. Jansson. Road verification and clearance in eastern angola: The dca approach. Journal of Mine Action, Issue 10.1, August 2006.

Ebinger. *UPEX 740 Large Loop Metal Detector Operation Manual*. Ebinger Prüf- und Ortungstechnik GmbH, Hansestr. 13, 51149 Köln, Deutschland, 2005.

G. Edwards, D. D. Bochtis, and C. G. Sørensen. Developing a field robotic test platform using lego mindstorm nxt. International Conference of Agricultural Engineering CIGR-AgEng2012, July 2012.

F. Fahimi. *Autonomous Robots Modelling, Path planning, and Controls*. Springer, 2009.

S. Fountas, B. S. Blackmore, S. Vougioukas, L. Tang, C. G. Sørensen, and R. Jørgensen. Decomposition of agricultural tasks into robotic behaviours. *Agricultural Engineering International: the CIGR Ejournal*, IX, 2007.

M. Freese, T. Matsuzawa, Y. Oishi, P. Debenest, K. Takita, E. F. Fukushima, and S. Hirose. Development of a practical landmine searching robot. Proceedings of the l3th IASTED International Conference ROBOTICS AND APPLICATIONS, August 2007, Würzburg, Germany, August 2007.

G. Freitas, J. Zhang, B. Hamner, M. Bergerman, and G. Kantor. A low-cost, practical localization system for agricultural vehicles. In *Proceedings of the 5th International Conference on Intelligent Robotics and Applications - Volume Part III*, ICIRA'12, pages 365–375, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33502-0.

E. Garcia and P. G. de Santos. Mobile-robot navigation with complete coverage of unstructured environments. *Robotics and Autonomous Systems*, 46(4):195–204, Apr. 2004.

L. Garcia-Perez, M. C. Garcia-Alegre, A. Ribeiro, and D. Guinea. An agent of behaviour architecture for unmanned control of a farming vehicle. *Computers and Electronics In Agriculture*, 60(1):39–48, Jan. 2008.

B. P. Gerkey, R. T. Vaughan, G. S. Sukhatme, K. Stoy, M. J. Mataric, and A. Howard. Most valuable player: A robot device server for distributed control, 2001.

GICHD. History, summary and conclusions of a study of manual mine clearance. Geneva International Centre for Humanitarian Demining (GICHD), August 2005.

GICHD. Guidebook on detection technologies and systems for humanitarian demining. Geneva International Centre for Humanitarian Demining (GICHD), March 2006.

GICHD. A guide to mine action and explosive remnants of war, third edition. Geneva International Centre for Humanitarian Demining (GICHD), April 2007.

F. Gloeckler, R. Joy, J. Simpson, and D. Specht. *Handbook for Transformation of Datums, Projections, Grids and Common Coordinate Systems*. U.S. Army Topographic Engineering Center, 1996.

J. Gomez-Gil, S. Alonso-Garcia, F. J. Gomez-Gil, and T. Stombaugh. A kalman filter implementation for precision improvement in low-cost gps positioning of tractors. *Sensors*, 13(11):15307–15323, 2013. ISSN 1424-8220.

GPSWorld. Gps world antenna survey 2012. www.GPSworld.com, 2012.

O. Green, T. Schmidt, R. P. Pietrzkowski, K. Jensen, M. Larsen, and R. N. Jørgensen. Commercial autonomous agricultural platform - kongskilde robotti. In *Robotics, Associated High-Technologies and Equipment for Agriculture and Forestry*, 2014.

M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley & Sons, Inc., 2. edition, 2001.

H. Griepentrog, N. Andersen, J. Andersen, M. Blanke, O. Heinemann, T. Madsen, J. Nielsen, S. Pedersen, O. Ravn, and D. Wulfsohn. Safe and reliable: further development of a field robot. In *Precision agriculture 09*, 2009.

H. Griepentrog, A. Ruckelshausen, R. Jørgensen, and I. Lund. Autonomous systems for plant protection. In E.-C. Oerke, R. Gerhards, G. Menz, and R. A. Sikora, editors, *Precision Crop Protection - the Challenge and Use of Heterogeneity*, pages 323–334. Springer Netherlands, 2010. ISBN 978-90-481-9276-2.

M. K. Habib. *Humanitarian Demining: Innovative Solutions and the Challenges of Technolog*. I-Tech Education and Publishing, Vienna, Austria, 2008.

H. Hagras, V. Callaghan, and M. Colley. Outdoor mobile robot learning and adaptation. 54 IEEE Robotics & Automation Magazine, September 2001.

S. Havlik. A modular concept of the robotic vehicle for demining operations. *Autonomous Robots*, 18(3):253–262, May 2005.

M. Hemapala, V. Belotti, R. Michelini, and R. Razzoli. Humanitarian demining: path planning and remote robotic sweeping. *Industrial Robot-an International Journal*, 36(2):146–156, 2009.

M. Henning. A new approach to object-oriented middleware. IEEE Internet Computing, January 2004.

E. v. Henten and J. Müller. The field robot event - an international design contest in agricultural engineering. In *VDI-Berichte*, volume 2001, pages 169–174, Düsseldorf, Germany, 2007. VDI-Max-Eyth-Gesellschaft, VDI Verlag GmbH.

H. Herman, T. Higgins, O. Falmier, J. S. Valois, and J. McMahill. Mine detection performance comparison between manual sweeping and teleoperated robotic system, 2010.

O. E. Holland. Grey walter: The pioneer of real artificial life. In C. Langton, editor, *Proceedings of the 5th International Workshop on Artificial Life*, pages 34–44. MIT Press, Cambridge, 1997.

B. B. H.W. Griepentrog. Autonomous crop establishment and control system. VDI-Berichte Nr. 2001, 2007.

G. Ippoliti, L. Jetto, and S. Longhi. Localization of mobile robots: Development and comparative evaluation of algorithms based on odometric and inertial sensors. *Journal of Robotic Systems*, 22(12):725–735, Dec. 2005.

C. Jaeger-Hansen, K. Jensen, M. Larsen, S. Nielsen, H. Griepentrog, and R. Jørgensen. Evaluating the portability of the frobomind architecture to new field robot platforms. 9th Europaean Confernce on Precision Agriculture, Lleida, Spain, July 2013.

K. Jensen, R. N. Jørgensen, A. Bøgild, O. J. Jørgensen, S. H. Nielsen, and R. B. Persson. Work in progress: Robotics mapping of landmine and uxo contaminated areas. In *IARP WS RISE 20-24 June 2011, Leuven, Belgium.* International Advanced Robotics Programme, June 2011.

K. Jensen, L. B. Larsen, K. S. Olsen, J. Hansen, and R. N. Jørgensen. First results: Robot mapping of areas contaminated by landmines and unexploded ordnance. In *HUDEM Symposium, 24 April - 26 April 2012 Sibenik, Croatia.*, 2012a.

K. Jensen, M. Larsen, T. Simonsen, and R. N. Jørgensen. Evaluating the accuracy and reliability of a new low-cost gps - supplementary material. In *First RHEA International Conference on Robotics and associated High-technologies and Equipment for Agriculture*, volume 1, University of Pisa, Italy, September 2012b. University of Pisa.

K. Jensen, S. Nielsen, A. Bøgild, O. Jørgensen, N. Jacobsen, C. Jaeger-Hansen, and R. Jørgensen. A low cost modular robotics tool carrier for precision agriculture research. In *11th International Conference on Precision Agriculture, July 2012, Indianapolis, USA.*, 2012c.

K. Jensen, M. S. Laursen, H. Midtiby, and R. N. Jørgensen. Autonomous precision spraying trials using a novel cell spray implement mounted on an armadillo tool carrier. XXXV CIOSTA & CIGR V Conference, Billund, Denmark, July 2013.

H. Jingtao and L. Taochang. Cascaded navigation control for agricultural vehicles tracking straight paths. *International Journal of Agricultural and Biological Engineering*, 7:36–44, 2014.

E. Jones, B. Fulkerson, E. Frazzoli, D. Kumar, R. Walters, J. Radford, , and R. Mason. Autonomous off-road driving in the darpa grand challenge. In *Proceedings of IEEE/ION PLANS 2006*, pages 366 – 371, San Diego, CA, April 2006. IEEE/ION, IEEE/ION.

R. Jørgensen, C. Sørensen, J. Pedersen, I. Havn, H. Jensen, K.and Søgaard, and S. L.B. Hortibot: A system design of a robotic tool carrier for high-tech plant nursing, 2007.

A. Kelly and C. Thorpe. Session overview field robotics. In S. Thrun, R. Brooks, and H. Durrant-Whyte, editors, *Robotics Research*, volume 28 of *Springer Tracts in Advanced Robotics*, pages 237–238. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-48110-2.

K. Kozlowski. *Robot motion and control 2011*. Springer, 2011.

M. Kroulik, T. Loch, Z. Kviz, and V. Prosek. Procedures of soil farming allowing to reduce compaction. In *Precision agriculture 09*, 2009.

K. D. Kuhnert. Software architecture of the autonomous mobile outdoor robot amor, 2008.

D. A. Kurstjens. Precise tillage systems for enhanced non-chemical weed management. *Soil and Tillage Research*, 97(2):293 – 305, 2007. ISSN 0167-1987.

LandMineMonitor. Land mine monitor 2013. International Campaign to Ban Landmines – Cluster Munition Coalition (ICBL-CMC), November 2013.

T. Lardner. *A study of manual mine clearance*. Geneva International Centre for Humanitarian Demining (GICHD), 2005.

L. B. Larsen, K. S. Olsen, L. Ahrenkiel, and K. Jensen. Extracurricular activities targeted towards increasing the number of engineers working in the field of precision agriculture. XXXV CIOSTA & CIGR V Conference, Billund, Denmark, July 2013.

T. Larsen, K. Hansen, N. Andersen, and O. Ravn. Design of kalman filters for mobile robots; evaluation of the kinematic and odometric approach. In *Control Applications, 1999. Proceedings of the 1999 IEEE International Conference on*, volume 2, pages 1021–1026 vol. 2, 1999.

T. D. Larsen. *Optimal fursion of sensors - Chapter 3.1 Kalman filtering in mobile robotics*. PhD thesis, The Technical University of Denmark, 1998.

M. Laursen, H. Midtiby, R. Jørgensen, and N. Krüger. Spray boom for selectively spraying a herbicidal composition onto dicots. Patent number: WO 2012/122988 A1, March 2011.

M. Laursen, H. Midtiby, R. Jørgensen, and N. Krüger. Validation of modicovi - monocot and dicot coverage ration vision based method for real time estimation of canopy coverage ration between cereal crops and dicotyledon weeds. International Conference on Precision Agriculture, Indianapolis, US., July 2012.

J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3): 376–382, 1991. ISSN 1042-296X.

M. Li, K. Imou, K. Wakabayashi, and S. Yokoyama. Review of research on agricultural vehicle autonomous guidance. *International Journal of Agricultural and Biological Engineering*, 2:1–16, 2009.

T. A. Lien. Mobile robotics in precision agriculture. Master's thesis, Norwegian University of Science and Technology, 2013.

MACCA. *AMAS 06.01 Mine Clearance Techniques*. Mine Action Coordination Centre of Afghanistan (MACCA), Post Box : 520 Kabul – Afghanistan, 3 edition, March 2011.

A. Makarenko, A. Brooks, and T. Kaupp. Orca: Components for robotics. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006). Workshop on Robotic Standardization., 2006.

P. S. Maybeck. Stochastic models estimation and control, 1979.

P. E. McKenney. 'real time' vs. 'real fast': How to choose? Technical report, IBM Linux Technology Center, 2008.

H. S. Midtiby. *Real time computer vision technique for robust plant seedling tracking in field environment*. PhD thesis, University of Southern Denmark, 2012.

K. W.-S. Y. Ming Li, Kenji Imou. Review of research on agricultural vehicle autonomous guidance. *Int J Agric & Biol Eng*, Vol. 2 No.3:1–16, 2009.

M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit, 2003.

K. Morgan. A step towards an automatic tractor. *Farm mech.*, 10 (13):440–441, 1958.

U. Neisser. *Cognitive Psychology*. Meredith Publishing, 1967.

I. A. D. Nesnas, A. Wright, M. Bajracharya, R. Simmons, and T. Estlin. Claraty and challenges of developing interoperable robotic software. *Iros 2003: Proceedings of the 2003 Ieee/rsj International Conference On Intelligent Robots and Systems, Vols 1-4*, pages 2428–2435, 2003.

P. O. Noack and T. Muhr. Integrated controls for agricultural applications – gnss enabling a new level in precision farming. In *1st International Conference on Machine Control & Guidance 2008*, 2008.

M. Norremark, H. W. Griepentrog, J. Nielsen, and H. T. Sogaard. The development and assessment of the accuracy of an autonomous gps-based system for intra-row mechanical weed control in row crops. *Biosystems Engineering*, 101(4):396–410, Dec. 2008.

T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

J. T.-S. Patricio Nebot and R. J. Martinez. A new hla-based distributed control architecture for agricultural teams of robots in hybrid applications with real and simulated devices or environments. *Sensors*, 11 (ISSN 1424-8220):., April 2011.

S. M. Pedersen, S. Fountas, H. Have, and B. S. Blackmore. Agricultural robots - system analysis and economic feasibility. *Precision Agriculture*, 7(4):295–308, Sept. 2006.

S. M. Pedersen, S. Fountas, and S. Blackmore. *Agricultural Robots — Applications and Economic Perspectives*. InTech, 2008.

R. B. Persson. Trial report handheld wide area detection system danchurchaid sudan. Technical report, UNMAO, NMAC Sudan, February 2010.

M. Pivtoraiko, I. A. Nesnas, and H. D. Nayar. A reusable software framework for rover motion controls, Februarry 2008.

M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

S. Rajasekharan and C. Kambhampati. The current opinion on the use of robots for landmine detection. In *Proceedings of the 2003 IEEE lnternational Conference on Robotics & Automation*. Proceedings of the 2003 IEEE lnternational Conference on Robotics & Automation, Taipei, Taiwan, September 14-19. 2003, September 2003.

A. Reske-Nielsen, A. Mejnertsen, N. A. Andersen, O. Ravn, M. Nørremark, and H. W. Griepentrog. Multilayer controller for outdoor vehicle. In *In Proceedings Automation Technology for Off-Road Equipment (ATOE)*, pages 41 – 49, September 2006.

M. Rodríguez and J. Gómez. Analysis of three different kalman filter implementations for agricultural vehicle positioning. *The Open Agriculture Journal*, 3: 13–19, 2009.

F. Rovira Mas, Q. Zhang, and A. C. Hansen. *Mechatronics and Intelligent Systems for Off-road Vehicles*. Springer, 2010.

A. Ruckelshausen, P. Biber, M. Dorna, H. Gremmes, R. Klose, A. Linz, R. Rahe, R. Resch, M. Thiel, D. Trautz, and U. Weiss. Bonirob: an autonomous field robot platform for individual plant phenotyping. In *Precision agriculture 09*, 2009.

H. Sahin and L. Guvenc. Household robotics: autonomous devices for vacuuming and lawn mowing [applications of control]. *Control Systems, IEEE*, 27(2): 20–96, 2007. ISSN 1066-033X.

J. Sass. Low cost, high accuracy, gnss survey and mapping. Coastal Areas and Land Administration – Building the Capacity, 6th FIG Regional Conference, November 2007.

M. Sato. Dual sensor alis for humanitarian demining and its evaluation test in mine fields in croatia. In *IGARSS 2008*, 2008.

D. Schacter. *Psychology.* Worth Publishers, 2011.

D. F. Sebastian Thrun, Wolfram Burgard. *Probabilistic Robotics.* MIT Press, 2006.

R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots - Second Edition.* The MIT Press, 2011.

J. D. A. L. Silva. Proposition and development of a robot manipulator for humanitarian demining, 2010.

S. F. Simon Blackmore and H. Have. Proposed system architecture to enable behavioral control of an autonomous tractor. In *Automation Technology for Off-Road Equipment*, volume Proceedings of the July 26-27, 2002 Conference, pages 13–23, Chicago, Illinois, USA, July 2002. American Society of Agricultural and Biological Engineers (ASABE).

D. C. Slaughter, D. K. Giles, and D. Downey. Autonomous robotic weed control systems: A review. *Computers and Electronics In Agriculture*, 61(1):63–78, Apr. 2008.

J. P. Snyder. *Map Projections: A Working Manual.* Professional Paper 1395. Geological Survey (U.S.), 1987.

T. Stombaugh, M. Sama, R. Zandonadi, S. Shearer, and B. Koostra. Standardized evaluation of dynamic gps performance. 2008 ASABE Annual International Meeting. Paper Number: 084728, July 2008.

P. N. Stuart Russell. *Artificial Intelligence: A Modern Approach (Third Edition).* Prentice Hall, 2010.

A. Suprem, N. Mahalik, and K. Kim. A review on application of technology systems, standards and interfaces for agriculture and food sector. *Computer Standards & Interfaces*, 35(4):355 – 364, 2013. ISSN 0920-5489.

C. A. G. Sørensen, O. Green, M. Nørremark, R. N. Jørgensen, K. Jensen, J. Maagaard, and L. A. Jensen. Hortibot: Feasibility study of a plant nursing robot performing weeding operations – part iv. 2007 ASABE Annual International Meeting, Paper Number: 077019, June 2007.

R. K. Taylor, M. D. Schrock, J. Bloomfield, G. Bora, G. Brockmeier, W. Burton, B. Carlson, J. Gattis, R. Groening, J. Kopriva, N. Oleen, J. Ney, C. Simmelink, and J. Vondracek. Dynamic testing of gps receivers. In *Transactions of the ASAE*, volume 47, page 1017−1025. American Society of Agricultural Engineers, 2004.

l. Thorling, B. Hansen, C. Langtofte, W. Brüsch, R. Møller, S. Mielby, and A. Højberg. Ground water monitoring (grundvandsovervågning) 2011. Technical report, GEUS, 2011.

C. Thorpe and H. Durrant-Whyte. Field robots. In *Proceedings of the 10th International Symposium of Robotics Research, Lorne, Victoria, Australia*, London, November 2001. Springer-Verlag.

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23 (9):661–692, Sept. 2006.

T.Takasu and A.Yasuda. Development of the low-cost rtk-gps receiver with an open source program package rtklib. International Symposium on GPS/GNSS, International Convention Center Jeju, Korea, November 2009.

UNMAS. Imas 09.10 clearance requirements second edition. United Nations Mine Action Service (UNMAS), January 2003.

R. Vázquez-martín, J. Martinez, J. C. Toro, and P. Núñez. A software control architecture based on active perception for mobile robotics 1, 2008.

N. E. Walsh and W. S. Walsh. Rehabilitation of landmine victims - the ultimate challenge. *Bulletin of the World Health Organization*, 81:665–670, 2003.

W. G. Walter. An imitation of life. *Scientific American*, 182:42–45, 1950.

G. Welch and G. Bishop. An introduction to the kalman filter. SIGGRAPH 2001 Course 8, 2001.

M. Zeitzew. Autonomous utility mower. *ATOE 07 016. Vol. IX. July, 2007.*, Vol IX:x, 2007.

B. Åstrand and A.-J. Baerveldt. A vision based row-following system for agricultural field machinery. *Mechatronics*, 15(2):251 – 269, 2005. ISSN 0957-4158.

Appendix A

Paper 1

Towards an open software platform for field robots in precision agriculture.

Kjeld Jensen, Morten Larsen, Søren H. Nielsen, Leon B. Larsen, Kent S. Olsen, Rasmus N. Jørgensen

Accepted for publication May 2014 by:
Robotics — Open Access Journal
ISSN 2218-6581
http://www.mdpi.com/journal/robotics

Submitted to *Robotics*. Pages 1 - 29.

OPEN ACCESS

Article

Towards an open software platform for field robots in precision agriculture

Kjeld Jensen [1,*], Morten Larsen [2], Søren H. Nielsen [1], Leon B. Larsen [1], Kent S. Olsen [1] and Rasmus N. Jørgensen [3]

[1] Faculty of Engineering, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
[2] Conpleks Innovation, Fælledvej 17, 7600 Struer, Denmark
[3] Institute of Engineering, Aarhus University, Nordre Ringgade 1, 8000 Aarhus, Denmark

* Author to whom correspondence should be addressed; e-mail, telephone and fax number of the corresponding author.

Version April 15, 2014 submitted to Robotics. *Typeset by LaTeX using class file mdpi.cls*

Abstract: Robotics in precision agriculture has the potential to improve competitiveness and increase sustainability compared to current crop production methods and has become an increasingly active area of research. Tractor guidance systems for supervised navigation and implement control have reached the market, and prototypes of field robots performing precision agriculture tasks without human intervention also exist. But research in advanced cognitive perception and behaviour that is required to enable a more efficient, reliable and safe autonomy becomes increasingly demanding due to the growing software complexity. A lack of collaboration between research groups contributes to the problem. Scientific publications describe methods and results from the work, but little field robot software is released and documented for others to use. We hypothesize that a common open software platform tailored to field robots in precision agriculture will significantly decrease development time and resources required to perform experiments due to efficient reuse of existing work across projects and robot platforms. In this work we present the FroboMind software platform and evaluate the performance when applied to precision agriculture tasks.

Keywords: Field Robots; Precision Agriculture; FroboMind; ROS

1. Introduction

Robotics in precision agriculture has the potential to improve competitiveness and increase sustainability compared to current crop production methods [1], and has become an increasingly active area of research during the past decades. Tractor guidance using Global Navigation Satellite System (GNSS) based sensor systems for route following and local sensor systems for accurate in-row navigation in row crops and orchards have already reached the market. One example is the John Deere iTEC Pro which supports GNSS based steering in straight and curved rows and at headlands while controlling speed and performing active implement guidance. Another example is the Claas Cam Pilot system which navigates a tractor through row crops using 3D computer vision to detect the location of the crop rows. Early prototypes of smaller field robots performing precision agriculture tasks without human intervention also exist [2][3][4][5][6]. But research in advanced cognitive[7] perception and behaviour that are required to enable a more efficient, reliable and safe autonomy becomes increasingly demanding. The level of complexity in an unstructured, dynamic, open-ended and weather influenced environment like a crop field or an orchard is high, and the size and complexity of the software needed to perform experiments impose ever greater demands on research groups. A lack of collaboration between the research groups contributes to the problem. Scientific publications are published on findings and results in precision agriculture, but little field robot software has actually been released, published and documented for others to use. We hypothesize that a common open software platform tailored to field robots in precision agriculture will significantly decrease the development time and resources required to perform field experiments due to efficient reuse of existing work across projects and robot platforms. The aim of this work is to establish such a software platform and evaluate the performance when applied to different precision agriculture tasks and field robots.

1.1. Related work

The literature contains numerous references to relevant proposals and implemented solutions within robot software architectures, frameworks, middlewares, development environments, libraries etc. Below is a brief review of some of the best known and widely used solutions. Table 1 compares the middleware specifications.

CARMEN Robot Navigation Toolkit from the Carnegie Mellon University (CMU) [8] is a modular software library for mobile robot control. It provides interfaces to a number of robot platforms and sensors, a 2d simulator and algorithms for localization, mapping, path planning etc. The architecture features 3 layers, the lowest layer contains hardware interfaces and collision detection, the middle layer localization and navigation, and the highest layer contains all high level tasks. Inter Process Communication (IPC), another project by CMU, is used for sending data between processes based on TCP/IP sockets.

CLARAty (Coupled Layer Architecture for Robotic Autonomy) [9][10] is a framework for generic and reusable software for heterogeneous robot platforms developed by the Jet Propulsion Laboratory. CLARAty is a two-tiered coupled layer (decision and functional) architecture. The functional layer is a modular software library which provides an interface to the robot system and contains algorithms for low- and mid-level autonomy. The decision layer builds on top of this adding high-level autonomy to achieve mission goals. CLARAty consists of a public and a private repository.

MRDS (Microsoft Robotics Developer Studio) [11] is a development environment for robot control and simulation. MRDS has support for a number of programming languages including it's own platform specific language Visual Programming Language (VPL). It supports a wide range of robotics hardware platforms and integrates a fully featured simulation environment. The component interface is based on the .NET Concurrency and Coordination Runtime (CCR) library for managing asynchronous, parallel tasks using message-passing and Decentralized Software Services (DSS), a lightweight .NET-based runtime environment.

Orca [12] is an open-source software framework for developing component-based robotic systems. The aim of Orca is to promote software reuse through definition of interfaces, providing component libraries and maintaining a public component repository. The intended use is for both commercial applications and research environments. The Orca project is a branch out from Orocos, the main difference is that Orca uses the Internet Communications Engine (ICE) [13].

Orocos (Open Robot Control Software) [14] contains portable C++ libraries for advanced machine and robot control. Orocos builds upon the open source Common Object Request Broker Architecture (CORBA) middleware. Orocos is in active development and contains extensions that support other frameworks including ROS.

Player [15] is one of the most used robot control interfaces and supports a wide variety of robots and components. The Player architecture is based on a *network server* which runs on the robot platform and provides a TCP socket interface to the robot. The *client program* is thus able to read data from sensors, write commands to actuators etc. by connecting to the socket. Player coexists with Stage which is a 2d multiple robot simulator and Gazebo which is 3d multiple robot simulator based on a physics engine.

ROS (Robot Operating System) [16] is a flexible framework for writing robot software. It provides services, libraries and tools for building robotics applications. Examples are hardware abstraction and device control, interprocess communication, multi-computer environment support etc. ROS is in active development and is maintained by Open Source Robotics Foundation who provides access to a large repository of available components and maintains a list of external repositories provided by the growing community. The ROS core code and most ROS packages are released under the BSD 3-Clause License which facilitate commercial use of the code.

In terms of robot software architectures the above mentioned *Carmen* and *CLARAty* each use their own architecture, wheres a *MRDS*, *ORCA*, *Orocos* and *ROS* don't specify or endorse a certain architecture. *Player* does not specify an architecture beyond the two-layer *Network server* and *Client program*. Several relevant but lesser known robot software architectures are described in the literature:

[17] argues that the sense-model-plan-act paradigm used in most architectures is unable to react in a dynamical environment because it depends heavily on the model, and that a behaviour-based approach such as the subsumption architecture [18] is limited by the purely reactive behaviours and does not perform well when carrying out complex tasks. A hybrid is thus presented containing three layers: Hardware layer, Reactive layer and Deliberative layer. The reactive layer is grouped into a perception module (localization and mapping) and an action module (navigation and actuation). The deliberative layer contains high level mapping and path planning.

[19] introduces Agricultural Architecture (**Agriture**), a control architecture designed for teams of agricultural robots. Agriture consists of three layers, the physical layer which is either the robot or

Table 1. Comparison of middleware specification. The year listed under *Update* indicates the latest official release or substantial update.

	CARMEN	**CLARAty**	**MRDS**	**ORCA**	**Orocos**	**Player**	**ROS**
Updated	2008	2007	2012	2009	2014	2010	2014
Main languages	C	C++	C# VPL	C++	C++	C++ TCL Java Python	C++ Python Lisp
Primary platforms	Linux	VxWorks Linux Solaris	Windows	Linux	Linux Windows OSX	Linux Solaris BSD OSX	Linux
Component interface	IPC	Unknown	CCR, DSS	ICE	CORBA	TCP sockets	XMLRPC
License	GPLv2 /BSD	Proprietary /Closed	Commercial /Academic	LGPL/ GPL	LGPL	GPLv3	BSD 3-Clause

the Stage/Gazebo simulators, an Architecture middleware based on *Player*, *Java Agent Development Framework* and *High Level Architecture*, and a Distributed application layer for the application software.

[20] proposed **Agroamara**, a hybrid agent of behaviour architecture for autonomous farming vehicles. Here *agent of behaviour* describes computational and control processes addressed to reach or to maintain a goal, with perceptual, deliberative and reactive abilities. The architecture groups the agents into perceptual and motor agents and uses layers to describe the information flow and hierarchy of activation of the agents. Data sharing is handled through shared memory and peer to peer message parsing. Multiprocessing is supported using winsocket.

[21] describes the software architecture of the Autonomous Mobile Outdoor Robot (**AMOR**) which won the first price in autonomous driving in the European Land Robot Trial (ELROB) 2007 for urban and non-urban terrain. The robot software is not publicly available, however the paper does provide a good introduction to the design as well as the navigation modules. Intercommunication is handled by a *Virtual Sensor Actor Layer* that use TCP sockets and unifies the access to all sensors and actors. The main modules of the high level software is *global model* (map database), *global path planning* (deciding plans for the indended behaviour), *local model* (based on laser scanner and camera data), *local path planning* and *basic reflexes*.

[22] introduces **Mobotware**, a plug-in based framwork for mobile robots. Mobotware is divided into a hard- and a soft realtime section. Each section is decomposed into layers of increasing abstraction: Hardware abstraction, reactive execution, deliberative perception and planning. The framework has three core modules, a *Robot Hardware Daemon* providing hardware abstraction, a *Mobile Robot Controller* handling real-time closed loop controlling, and *Automation Robot Servers* handling sensor processing, mission planning and management.

[23][24] propose a system architecture to enable behavioural control of an autonomous tractor based on the subsumption architecture [18]. This work is related to the Software Architecture for Agricultural Robots (**SAFAR**) project with the purpose to develop a set of designs, tools and resources to promote

Table 2. Comparison of robot software architectures with respect to the problem domain and hypothesis of this work. In the columns Agricultural applications, Multiple platforms and Multiple users *Yes* means that practical field trials have taken place. *(Yes)* in Open source means that not all software has been released and/or the license is not permissive free.

	Agricultural applications	**Multiple platforms**	**Multiple users**	**Open source**	**Updated recently**
CARMEN	Yes	Yes	Yes	(Yes)	No
CLARAty	No	Yes	Yes	(Yes)	No
Agriture	No	No	No	No	No
Agroamara	Yes	No	No	No	No
AMOR	No	No	No	No	No
Mobotware	Yes	Yes	Yes	No	Yes
SAFAR	Yes	Yes	No	No	No
Stanley	No	No	No	No	No

the development of agricultural robots. SAFAR is based upon MRDS and is closed source but supports a Python scripting engine.

In 2005 the Stanford robot **Stanley** won the DARPA Grand Challenge by driving autonomously for 131 miles along an unrehearsed desert trail [25]. The robot software is not publicly available, however the paper provides information on the software architecture and design considerations. Stanley contains approximately 30 software modules organized in 6 layers representing sensor interfaces, perception, planning and control, vehicle interface, user interface and global services. All modules are executed individually without interprocess synchronization and all data are globally time stamped. Interprocess communication is handled using publish-subscribe mechanisms based on the CMU IPC kit.

Table 2 compares some properties of the reviewed architectures with respect to this work. The data is based on the reviewed publications and information publicly available on the web. It should be noted that all the reviewed publications contains valuable and relevant knowledge, but at the same time the table underpins the need for collaboration between the research groups in precision agriculture with respect to software reuse as discussed in the problem description.

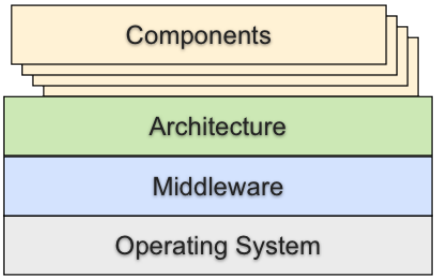
2. The FroboMind software platform

2.1. Design goals

The software platform presented in this work has been named FroboMind which is a contraction of *Field Robot Mind*. It is based on design goals that promote reuse, support projects of varying size and complexity, and facilitate collaboration:

- **Modularity:** It must provide a proper modular structure to ease the development of new components and facilitate reuse. Interfaces between modules must be well defined, however still allowing researchers and developers to experiment freely.

Figure 1. Overview of the FroboMind software platform structure.



- **Extensibility:** The design must support a wide variety of applications from a student working with signal processing or route planning algorithms to large scale field experiments conducted by research groups with several robots interacting online with each other and human operators.
- **Scalability:** It must support online computation of high load algorithms such as computer vision and mapping with respect to both memory and processing power requirements. This includes scaling to distributed applications on networked heterogeneous platforms.
- **Software reuse:** There must be a strong focus on minimizing resource requirements in the development of new applications by facilitating reuse of software. The future workload of maintaining the software platform is also an important consideration. When establishing the software platform, existing work should be reused to the extent possible. To prevent maintenance problems arising due to the existing work, this must be under active development and supported by a community.
- **Open source:** All core components of the software platform must be released under a permissive free software license to keep it free for others to use, change and commercialize upon. This is necessary to facilitate collaboration with industrial partners.

2.2. FroboMind

The FroboMind software platform structure consists of four parts as illustrated in fig.1. The *Operating System* provides basic facilities for application execution, file handling, hardware interfacing, communication, networking etc. The *Middleware* provides services like timing and communication between software components in a distributed system which simplifies the software development significantly. The *Architecture* organizes the software components into layers and modules with well defined interfaces. The modularity eases the development proces and facilitate efficient software reuse. The *Components* are the actual building blocks used for field robot applications and reused across projects and robot platforms. In the following sections each of the four parts of the software platform structure is described in detail.

2.3. Operating system

When choosing an operating system for FroboMind the design goals, in particular the *Software reuse* and *Open source* goals must be taken into consideration.

In robotics software implementations a Real-Time Operating System (RTOS) dedicated to embedded systems is typically used. The reason is that RTOS improves the performance for components that have hard or near real-time requirements, and some of the available systems (eg. VxWorks and QNX) improve the reliability significantly compared to traditional operating systems. However they are usually not open source, and software developed for the RTOS typically runs on the target system only and therefore requires cross compilation, and on-target debugging etc. Linux based open source RTOS exist in different varieties. Examples are dedicated distributions where the entire system run as preemptive processes (eg. RTLinux), add-ons that create a small microkernel where linux runs as a task, and kernel patches or libraries that implement a near real-time environment like the RealTime Application Interface for Linux (RTAI) [26] and CONFIG_PREEMPT_RT [27]. Some of these are well functioning and are utilized in commercial products as well as academic projects. However they are trying to match an RTOS with a fully fledged operating system which is inherently a tradeoff between low latency requirements and the overall efficiency of the system [28]. Other typical challenges of the RTOS are small development teams and a limited community supporting the projects. The RTOS distributions therefore often lag behind regarding support for new software and hardware, and support options in case of problems are limited. In applications where the robot is connected to the internet, network security may be an issue as well due to delayed software updates.

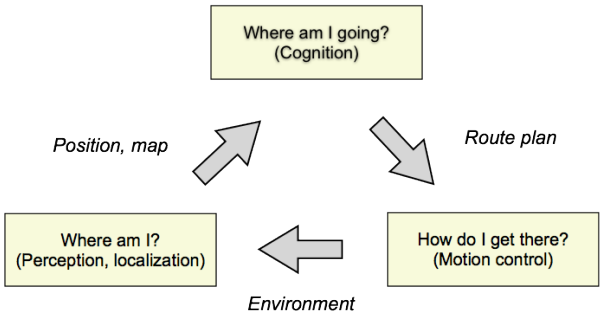
Based on the above considerations the linux distribution Ubuntu was chosen as operating system for FroboMind. Ubuntu is one of the largest and most active linux distributions available, and the annual Long-Term Support versions have guaranteed support for five years after release. Choosing Ubuntu has the major advantage that one can effortlessly run the same operating system on the field robot and a standard laptop used for software development. There is also a huge community of linux software developers which is an advantage when support is needed for a particular problem. Ubuntu only supports soft real-time execution, but in field robotics real-time requirements at the order of 50-200 Hz typically only exist in relation to low level actuator controlling, and this task is often distributed to an external embedded controller that communicates via a serial or network interface. The use of Ubuntu in robot systems with near real-time requirements is evaluated in the experimental section.

2.4. Middleware

As described in the review of related work a number of middlewares designed for robot control exists already. However for FroboMind most of the frameworks and middlewares listed in table 1 and described elsewhere must be disregarded because they do not comply with the *open source* design goal. Considering in addition the *software reuse* design goal stating that the middleware must be in active development this leaves only two attractive contestants: Orocos and ROS.

Taking a closer look at Orocos it is based upon a few principal components: The Orocos Toolchain which contains the tools required to build Orocos components, the iTaSC framework (generates robot motions using constraints). the rFSM toolkit (Finite State Machines) and the libraries KDL (Kinematics and Dynamics) and BFL (Bayesian Filtering). The Orocos Toolchain includes the Real Time Toolkit (RTT) framework for developing real-time components in C++.

Figure 2. A conceptual basis for the architecture of mobile robot navigation software described in the literature: *Where am I? Where am I going? How do I get there?*



The ROS middleware offers inter-process communication through a set of facilities: Anonymous publish/subscribe message passing, recording and playback of messages, request/response remote procedure calls and a distributed parameter system. ROS also provides a set of robot libraries such as robot geometry, pose estimation, localization, mapping and navigation etc. ROS has a very good support for distributing components between networked computers. ROS is not a realtime framework and is thus not suitable for hard real-time critical software such low level control algorithms. There is, however, support for integration with the Orocos RTT.

Orocos and ROS both fit well with the FroboMind design goals. In favor of Orocos it supports real-time execution and the code base appears to be stable. The requirements for CPU power and memory appear to be lower than for ROS. But Orocos seems more oriented towards low level control than robot application building, and it is not as user friendly as ROS, which also has by far the largest supporting community and the most active development. ROS provides a very detailed documentation using a wiki, videos and examples. Fast prototyping using scripting is available with Python which may speed up application development in many cases. Based on these considerations ROS was chosen as middleware for FroboMind. The requirements with respect to computing power when running ROS and Ubuntu in robot systems with near real-time requirements is evaluated in the experimental section.

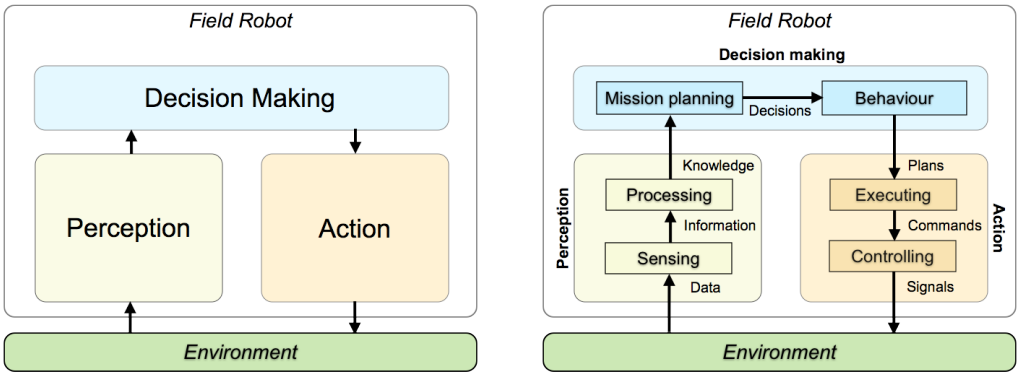
2.5. Architecture

The purpose of having a unified architecture shared across different projects and research groups is to facilitate software reuse at the application level. This allows the entire software platform to be transferred to new applications and field robots, the user needs only to add new high level behaviours and low level interface drivers needed for a particular application.

ROS does not propose or endorse a certain architecture or structure of the software, instead the different software components named ROS nodes are able to intercommunicate freely without restrictions to structure. As a result different users tend to use different designs and to some extent different interfaces between components. The user therefore often has to build a new robotic application from scratch, and reuse mainly exists from component level down to snippets of code.

Adding a unified architecture to FroboMind increases the reusability of the components, but at the same time it limits the flexibility that ROS provides. In this section an architecture is proposed with the aim of balancing the tradeoff between reusability and flexibility.

Figure 3. Decomposition of an Artificial Intelligence agent. (a) The agent perceives its environment through sensors and acts through actuators. The action taken by the agent in response to any percept sequence is defined by an agent function. (b) Decomposition to define data- and hardware abstraction levels.



(a) (b)

Designing a conceptual architecture that appears intuitive to researchers, engineers and technicians is a challenging task due to their different backgrounds and experience from different projects. A simple basis for mobile robot software has been formulated by the questions: *Where am I?*, *Where am I going?* and *How should I get there?* [29]. The robot localizes itself based on percepts and prior knowledge, it then plans a route which leads towards the mission goal and navigates the route using motion control (fig.2). However for a field robot in precision agriculture the task is the primary objective and navigation is merely a means to fulfil the task. Performing the task typically involves interaction with an attached or trailed implement, and the conceptual architecture in fig.2 is therefore insufficient. Instead a more generic approach based on Intelligent Agents [30] is used for modelling the architecture. An agent is an autonomous entity which perceives its environment through sensors and acts upon that environment through actuators. The action taken by the agent in response to any percept sequence is defined by an agent function. The FroboMind principal architecture consists of perception, decision making and action layers (fig.3(a)). The layers have been decomposed to define abstraction levels (fig.3(b)).

Perception represents the organization, identification and interpretation of sensory information in order to represent and understand the environment [31]. The perception layer has been decomposed into:

- **Sensing:** The robot perceives the surrounding partially observable environment (external percepts) through its sensors and assess the system interior state (internal percepts) through feedback from the platform and implement systems. Together these percepts constitute the available *information*. In addition this layer constitutes the abstraction between sensor hardware and the other parts of the architecture.

- **Processing:** By combining this *information* with previous, shared and a priori knowledge, the field robot maintains a model of the world and system state which constitute the accumulated *knowledge*. Since observables are governed by a certain amount of uncertainty, probabilistic methods are typically utilized to optimize the model.

Decision Making constitutes the cognitive layer and represents an AI agent function which determines the action taken in response to the accumulated *knowledge*. The implementation of the decision making layer may vary depending on application and users may choose to use other methodologies than the current support for model-based, utility-based AI agents and hierarchically organized finite-state machines to describe robot behaviour.

- **Mission Planning:** The robot mission planner continuously monitors the accumulated knowledge as well as any user interaction, and makes on this basis a *decision* about the optimal behaviour which leads towards the fulfillment of the mission. This corresponds to an AI agent utility function described in [30].

- **Behaviour:** The optimal behaviour decided by the mission planner origins from a library of possible behaviours available to the robot and implement. The active behaviour continuously monitors the accumulated knowledge and user interaction and updates action *plans* for the robot and implement action accordingly.

Action The action layer carries out the action defined by the action plans and has been decomposed into:

- **Executing:** The *plans* produced by the active behaviour are executed with respect to time and state. Based on this *commands* are sent to the controlling layer.

- **Controlling:** Commands issued by the executing layer are transmitted to low level controllers within the architecture or to external controller interfaces. This layer constitutes the abstraction between the field robot actuator hardware and other parts of the architecture.

The FroboMind architecture (fig.4) represents an expansion of the decomposition in fig.3. This is indicated by the grouping of modules containing software components into the *Perception*, *Decision making*, *Action* and *Safety* layers as well as the data interface between layers and modules indicated by the blue dashed lines. Internal fault diagnosis and incident handling are organized in a separate *Safety* module to minimize potential software errors and thus ensuring a high level of reliability. In order not to clutter the overview it is assumed that any component has access to data accessible by its predecessor. Multiple connections to successors are shown only where relevant to the understanding, and data available globally has not been included.

2.6. Components

The FroboMind components are implemented as ROS packages. Most of the current lower level components at the Perception and Action layers are written in C++ while many of the higher layer components are written in Python.

The components are located in a directory structure where each layer in the architecture (fig.4) is represented by a directory (table 3), each module as a subdirectory herein and the components are located in the module subdirectories. The FroboMind component directory structure and related documentation are located in a repository available through the website http://www.frobomind.org

Figure 4. The FroboMind Architecture. The modules containing software components are grouped into the *Perception*, *Decision making*, *Action* and *Safety* layers. The blue dashed lines indicate the data interfaces between layers and modules.

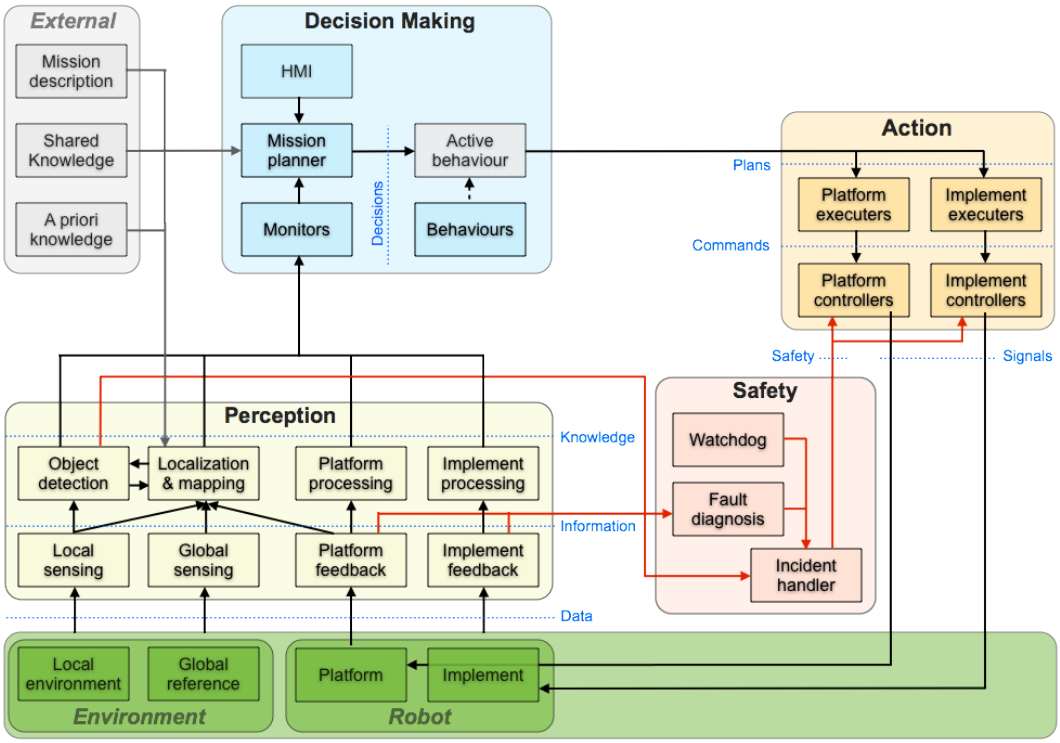


Table 3. The component (top level) directory structure at the FroboMind repository.

Directory	**Content**
/fmSensors	Sensor interfaces and information extraction (hardware abstraction layer)
/fmProcessors	Processing sensor information to obtain knowledge about the robot state.
/fmDecisionMakers	Robot mission, behaviour and HMI components.
/fmExecutors	Executing the current behaviour, e.g. navigation and implement control.
/fmControllers	Low level controllers and actuator interfaces (hardware abstraction layer).
/fmSafety	Watchdog, fault diagnosis and incident handling.
/fmApp	Directories containing components, scripts, launch files etc. related to the application of FroboMind to a particular project.
/fmLib	Interface drivers, software libraries etc. not represented in the architecture.
/fmTools	Various non-ROS tools and utilities.

2.7. Versions

FroboMind has been in active development since 2011 where the first prototype architecture and components were implemented in ROS Electric on Ubuntu 10.04. Since then FroboMind versions have been created each year with an updated architecture, improved components and compatible with updated versions of Ubuntu and ROS. The version numbers corresponds to the farming season. Version 2014 is the current FroboMind software platform described in this work. It is based on ROS Hydro and Ubuntu 12.04 LTS.

3. Experimental Section

In this work three experiments have been carried out to evaluate the performance of FroboMind when applied to various precision agriculture tasks and field robots. The first experiment evaluates the performance of FroboMind in robot systems with near real-time requirements including requirements with respect to computing power. The second experiment evaluates whether FroboMind significantly decreases the development time and resources required to perform field experiments. The third experiment uses available data from existing projects and robotic platforms that use FroboMind, to evaluate software reuse across these projects.

3.1. Evaluating the performance of FroboMind in robot systems with near real-time requirements.

An experiment was defined to evaluate the performance of FroboMind including ROS and Ubuntu in robot systems with near real-time requirements including requirements with respect to computing power. The experiment is based on a typical application for a field robot in precision agriculture: Autonomous navigation of a predefined route while controlling an attached or trailed implement. Fig.5 shows an example of the FroboMind architecture listing the components required for the autonomous navigation. Each of the components are described in table 4. All the library functions listed in fig.5 are implemented in C++.

In the experiment a differentially steered FroboScout robot (table 8) was set to autonomously navigate a static route plan containing 5 waypoints using the architecture in fig.5. The route length is 47 m. Two auxiliary components were added to monitor the system real time performance:

- A computer load monitor component which averages the CPU and memory load every 0.2 s and publishes the results through ROS.
- A real time performance test component consisting of a Python script and an external Atmel ATmega series microcontroller connected to a serial port. The Python script running at at rate of 100 Hz sends one byte to the serial device at each update. The ATmega firmware continuously counts the number of 100 μs ticks between each received byte and returns the latest count upon receiving a byte. This count is in turn received by the Python script and published through ROS.

In total 14 ROS nodes were launched. Together they published ROS messages under 28 different ROS topics, hereof 2 at 100 Hz, 9 at 50 Hz, 1 at 40 Hz, 4 at 20 Hz, 3 at 10 Hz, 6 at 5 Hz and 3 at a lower rate. During the experiment trials all messages published by ROS were recorded using the rosbag

Table 4. FroboMind example components used for autonomous navigation of a route plan. Rate describes the update function rate. Source Lines Of Code (SLOC) include comments and blank lines but not library functions. c refer to C++ and p to Python.

Layer	**Component**	**Rate**	**SLOC**
/fmSensors	**VectorNav interface** reads and publishes the yaw axis gyro rate from a VectorNav VN-100 IMU.	40 Hz	261c
	GNSS interface reads and publishes the GNSS *GPGGA* information including conversion of the position to Transverse Mercator coordinates.	10 Hz	271c
/fmProcessors	**Differential odometry** estimates the current position and orientation based on information from the wheel encoders and the IMU yaw angle gyro.	50 Hz	527c
	Pose estimator uses pre-processing and an Extended Kalman Filter to estimate the absolute position and orientation based on the differential odometry and GNSS information.	50 Hz	592p
/fmDecisionMakers	**Manual driving / autonomous navigation mission** In the *manual driving* behaviour velocity commands from the keyboard are sent directly to the FroboScout interface. In the *autonomous navigation* behaviour the waypoint navigation component is activated.	20 Hz	199p
/fmExecutors	**Waypoint navigation** navigates the waypoint list by publishing linear and angular velocity commands.	20 Hz	1077p
/fmControllers	**FroboScout interface** converts linear and angular velocity commands to wheel velocities, publishes encoder feedback.	50 Hz	428p

Figure 5. FroboMind example architecture for autonomous navigation of a route plan.

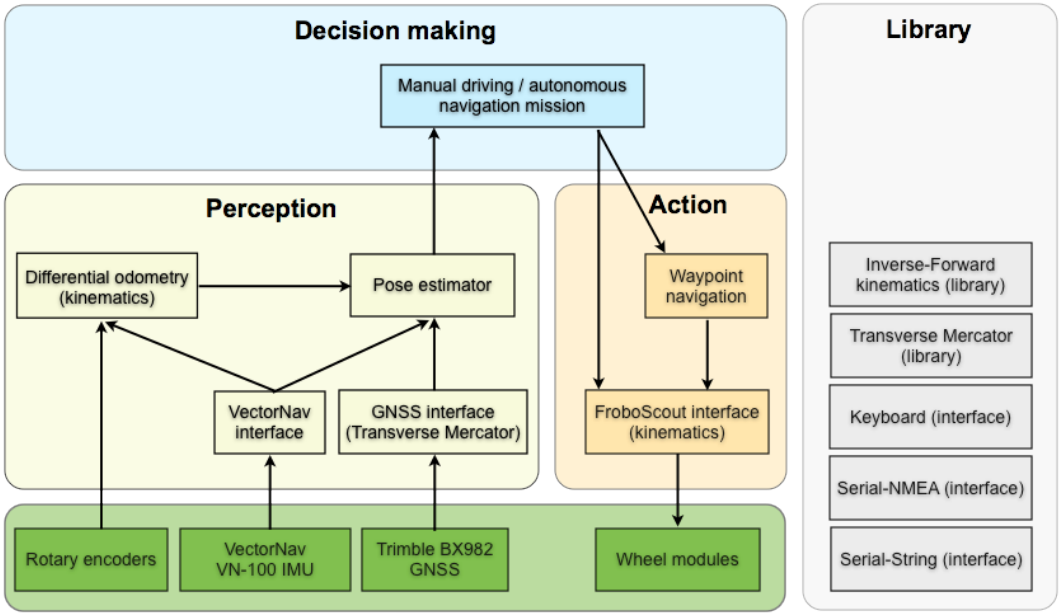


Figure 6. Autonomous Mechanisation System (AMS) and orchard used for the test.

(a) AMS.

(b) Apple three orchard.

tool. The computers used in the trials were connected to internet during the trials but FroboMind did not initiate external network connections. All computers had a standard Ubuntu installation, no attempts to optimize for performance by shutting down default services were made.

3.2. Evaluating if FroboMind significantly decreases the resources required to perform field experiments

Evaluating whether the FroboMind software platform significantly decreases the development time and resources required to perform field experiments is difficult because no reference data seem to exist in the literature. An experiment was therefore conducted with the purpose of trying to quantify the portability of FroboMind to a new robot platform by analysing the associated work. The experiment was conducted in collaboration with the University of Hohenheim (UH) and is presented in [32]. A brief description and the results is included here for completeness.

The experiment was designed so that as many parameters as possible were controllable. The task was defined as field robot navigation through an apple tree orchard using local sensors only which is another typical precision agriculture application [33][34]. The Autonomous Mechanisation System (AMS) robot (fig.6(a)) owned by UH was used for the experiment. The AMS has been utilized in several previous precision agriculture research projects [35][36] using Mobotware [22], and both the hardware and low level software is well tested and documented. A lidar and an Inertial Measurement Unit (IMU) were used as navigation sensors. An RTK-GNSS system was included for the purpose of recording reference data.

The experiment was conducted during a 5 day workshop where 4 developers worked full time porting FroboMind to the AMS, implementing the orchard navigation task and documenting the process. Preparations before the workshop included preparing the AMS and related hardware, reading documentation and planning the workshop. The task at the workshop was therefore defined as interfacing FroboMind to the AMS and building an application supporting autonomous navigation along the tree rows of the orchard and perform headland turns at the end.

In the orchard used for the navigation tests the tree rows were approx. 100 m long and interspaced by 4 m (fig.6(b)). In the first trial the AMS mission was to navigate autonomously between two rows of trees. At the end of the row the AMS would make a 90 degree left turn, drive straight, and make a 90 degree left turn which would position the robot in an adjacent row. It was decided to repeat this process continuously 12 times corresponding to 6 full rounds driving the same track. In the second trial the AMS mission was to navigate autonomously through the same orchard alternating between left and right turns. Estimation of position and orientation (pose) relative to the tree rows is handled by FroboMind components from a previous project: A Ransac based algorithm uses the lidar data to detect the tree rows and outputs the angle and offset relative to the rows. The angle is fused with the IMU yaw angle data using an Extended Kalman Filter, and the resulting pose is used to control the AMS steering wheels using a PID controller. When the lidar detects the ends of the tree rows, it switches to a turn state and performs the turn based on the IMU data. When the lidar detects the ajacent row, it switches back to in-row navigation.

3.3. Evaluating software reuse across existing projects using FroboMind.

During the past three years FroboMind has been used in various projects in precision agriculture as well as in some projects in similar domains. Examples of tasks are navigation of route plans using GNSS, navigation in row crops and orchards using local sensors, control of passive and active implements and interfacing to autonomous implements. Examples of sensor interfaces implemented in FroboMind are encoders, GNSS, IMU, lidar, 3d lidar, 3d vision row camera, localization using stereo vision, total station and metal detector. Examples of actuator interfaces controlled by FroboMind are relays, brushed and brushless motors, servos, linear actuators, hydraulics propulsion and diesel engines. Together these tasks and interfaces support many of the operations required in precision agriculture and the potential for efficient software reuse is thus high.

The third experiment is based on available data from existing projects and field robots that use FroboMind. The purpose is to evaluate the extent of software reuse by looking at the approximate

number of physical Source Lines Of Code (SLOC) shared across these projects. As the projects have been carried out during a period where the FroboMind architecture and core components were in very active development it is difficult to perform an accurate comparison. But it provides an indication of the degree of reusability.

4. Results

4.1. Evaluating the performance of FroboMind in robot systems with near real-time requirements.

Three different trials were conducted, each using one of the computers listed in table 5. The Graphical User Interface (GUI) *Ubuntu desktop* was running on PC2 and PC3, but on PC1 it was shut down prior to the trial because of high CPU load.

In all trials did the robot complete the waypoint navigation task successfully in about 100 seconds. Fig.7 shows the CPU and memory load averaged at 0.2 s intervals. Fig.8 shows the scheduling delays experienced by the 100 Hz FroboMind component during each of the 3 trials, table 6 shows statistical data for the delays. Due to lack of an absolute, accurate time source in the experiment setup the delay measurements have been calibrated for offset and skew errors based on statistical calculations and, the data may therefore slightly inaccurate. The skew calibration constants were verified using external frequency measurement.

4.2. Evaluating if FroboMind significantly decreases the resources required to perform field experiments

During the workshop a journal was continually updated with information about completed tasks etc. A summary of the estimated time consumption on different sub tasks is listed in table 7.

While porting FroboMind to the AMS it was discovered that the AMS exhibited errors when controlling the front wheels. When requesting the AMS to drive straight it would drift slowly to the left. When requesting the AMS to turn left the wheels were positioned correctly but when requesting the AMS to turn right the steering wheels were positioned at an angle significantly lower than the requested angle. Based on the performed tests it is likely that the problem is with the steering hardware, however

Table 5. Specifications for the computers used for the 3 trials in the FroboMind real-time performance experiment.

	PC1	**PC2**	**PC3**
Laptop:	Acer Aspire One ZG5	IBM ThinkPad X61s	Acer Travelmate B113
RAM:	1 Gb	2 Gb	4 Gb
CPU (Intel):	Atom N270	Core 2 Duo L7300	Core i3-2377M
Clock:	1.60 GHz	1.40 GHz	1.50 GHz
Cores/siblings:	1/2	2/2	2/4
Cache:	512 kb	4096 kb	3072 kb

Table 6. Statistics for the schedule delays experienced by the 100 Hz component in the three trials of the FroboMind performance experiment.

	mean	**variance**	**95'th percentile**	**maximum**
PC1	0.58 ms	0.41 ms^2	1.52 ms	14.1 ms
PC2	0.26 ms	0.055 ms^2	0.72 ms	4.0 ms
PC3	0.24 ms	0.0074 ms^2	0.36 ms	1.3 ms

Figure 7. CPU & memory load during autonomous navigation of a route plan for each of the 3 trials. The red graphs shows % of max CPU load. The black graphs show usage % of the total memory.

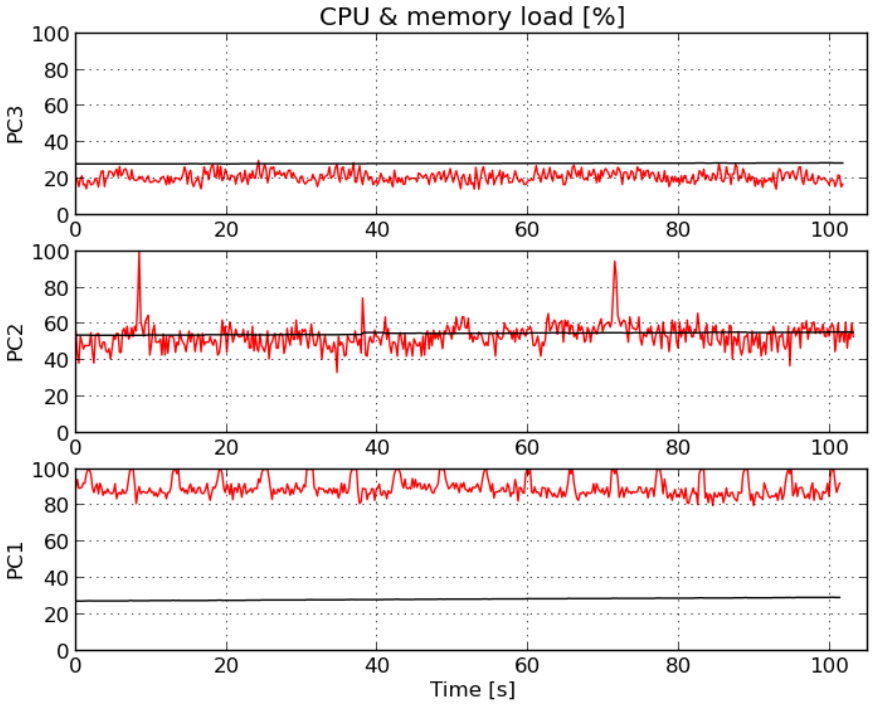


Figure 8. 100 Hz scheduler delays during autonomous navigation of a route plan for each of the 3 trials. The y axis represents the delay of each 10 ms schedule.

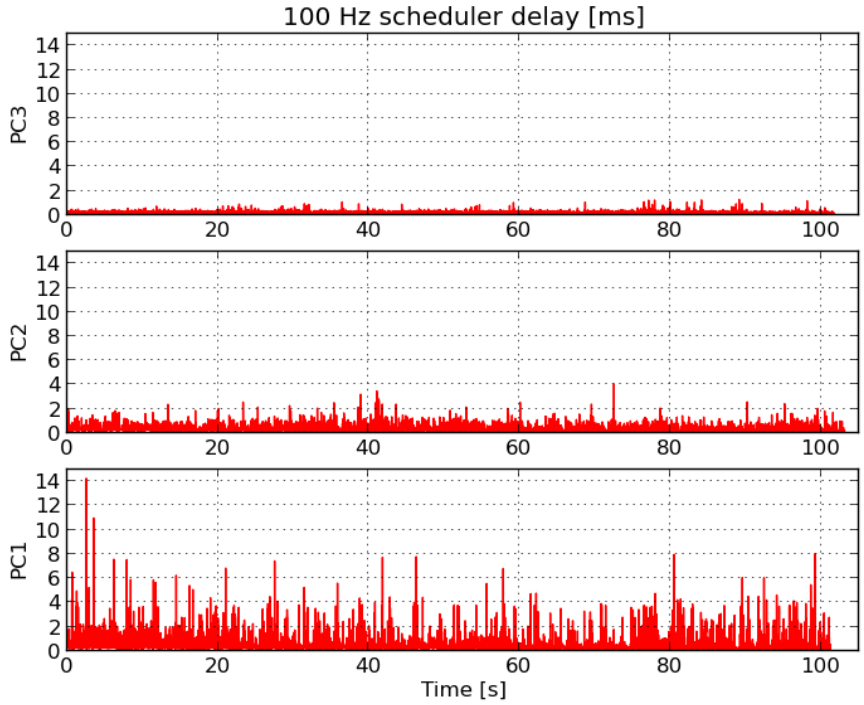
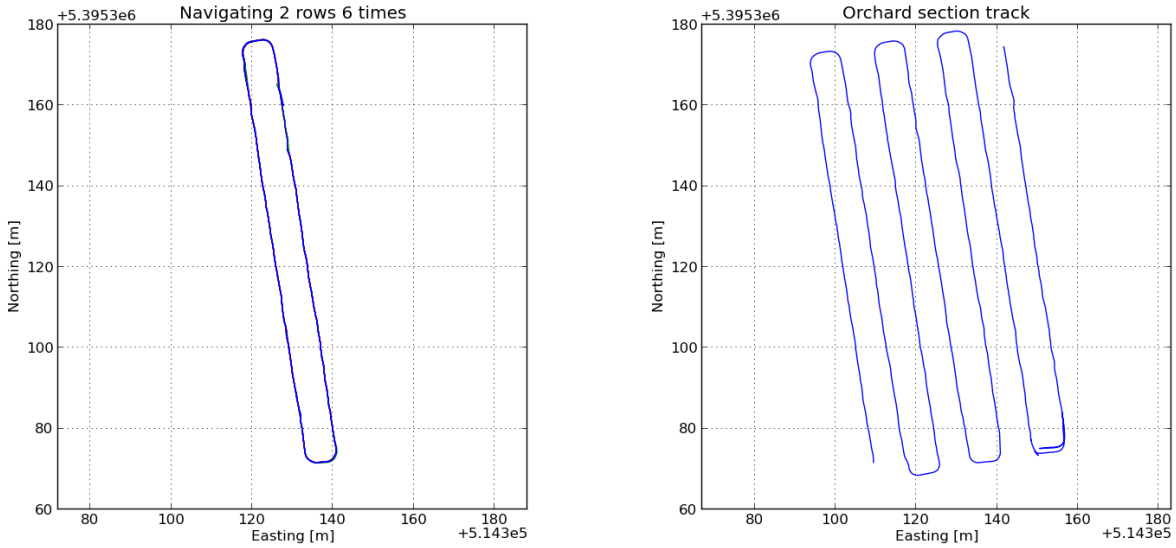


Table 7. Summary of estimated time consumption during the workshop.

Activity	**Duration**
Adding new components to FroboMind that directly support the AMS platform. This includes obtaining information about the AMS interface from documentation and available source code.	60 hours
Simulating and testing remote controlled and autonomous driving with the AMS positioned in a test stand.	30 hours
Adding and improving the documentation www.frobomind.org when the project work revealed issues not properly documented.	10 hours
Updated documentation for the AMS when the project work revealed issues not properly documented.	10 hours
Testing the implemented behaviours with the AMS on the ground.	20 hours
Collecting data from a manual run in the orchard.	6 hours
Analysing data from a manual run in the orchard to prepare autonomous operation.	20 hours
Creating a FroboMind mission controller and behaviors for the AMS navigating autonomously in an orchard using local sensors.	40 hours
Testing autonomous behaviour in the orchard.	30 hours
Updating project documentation and web pages.	12 hours

Figure 9. Recorded GNSS track from the trials. (a) shows the overlay GNSS track from all 6 rounds.



(a) Trial 1.

(b) Trial 2.

due to the time constraints it was decided to mitigate the problems by modifying the relevant FroboMind components. The process to identify, understand and mitigate these errors added substantially to the consumed time.

The first trial navigating 6 full rounds driving the same track in two adjacent rows was completed successfully. Three rounds were completed without human intervention. During two of the rounds the navigation failed one time at the same location where the apple trees at one of the rows were replanted by much younger and hence smaller trees. During one round the navigation failed at a location where trees were missing for more than 6 meters at one of the rows. Fig.9(a) shows the overlay GNSS track from all 6 rounds.

In the second trial navigating a section of rows in the orchard it was decided that focus would not be on the sensor/controller performance under difficult circumstances, so a person stepped in as ”tree” whenever one of the rows had a large hole between the trees. The trial was concluded after completing 7 rows. The trial was completed without human intervention except two times where the AMS failed a turn to the right at a row end due to the described steering problems. Fig.9(b) shows the recorded GNSS track from this trial.

4.3. Evaluating software reuse across existing projects using FroboMind.

Table 8 lists robots that are known to have been interfaced to FroboMind. The table lists the latest known application where FroboMind has been used as well as the latest version of FroboMind known to work on the robot. Based on information obtained from each application, abbreviations for software components shared with other robots are listed next to the version number. The abbreviations refer to table 9 which lists the component function along with the physical SLOC including comments and blank

Table 8. A list of robots using FroboMind. Column 3 lists the FroboMind version number followed by the reused components detailed in table 9.

	Latest application	**FroboMind**
ASuBot 1	Navigating orchards using lidar.	2013 ASU RDT RNV
Armadillo I 2	Navigating maize fields using lidar.	2012 ARD RDT RNV
Armadillo Scout 3	Crop monitoring.	2012 ARD
Armadillo III 4	Driving and navigation in dunes.	2012 ARD ODO POS WPT
AMS 5 fig.6(a)	Navigating orchards using lidar. [32].	2012 AMS RDT RNV
Armadillo IV 6	Large scale precision spraying trails [37].	2013 ARD ODO POS WPT PMP
Pichi 7	Detection and mapping of anti-tank mines and large unexploded objects.	2013 PIC ODO POS WPT COV MDI HMP
Robotti 8	Mechanical row weeding [2].	2013 ARD ODO POS WPT RWD
FroboMower 9	Mowing using low cost sensors.	2013 FBT ODO POS WPT
Frobit 10	Education [38] & rapid prototyping.	2013 FBT ODO WPT
SMR 11	Education.	2013 MBW ODO WPT
GrassBots 12	Grass mowing on low lands.	2013 GBT ODO POS WPT COV
FroboScout 13	Highway surveying.	2014 FST ODO POS WPT SUR

lines. Due to different component versions the SLOC may differ between different robot installations and should thus be considered to be approximate. Excluded from the list of components are sensor interfaces and libraries etc. Fig. 10 shows images of each robot, the image number refer to the list of robots in table 8.

Figure 10. Images of robots using FroboMind. The numbers show the historical order of FroboMind integration and refer to the list of robots in table 8.

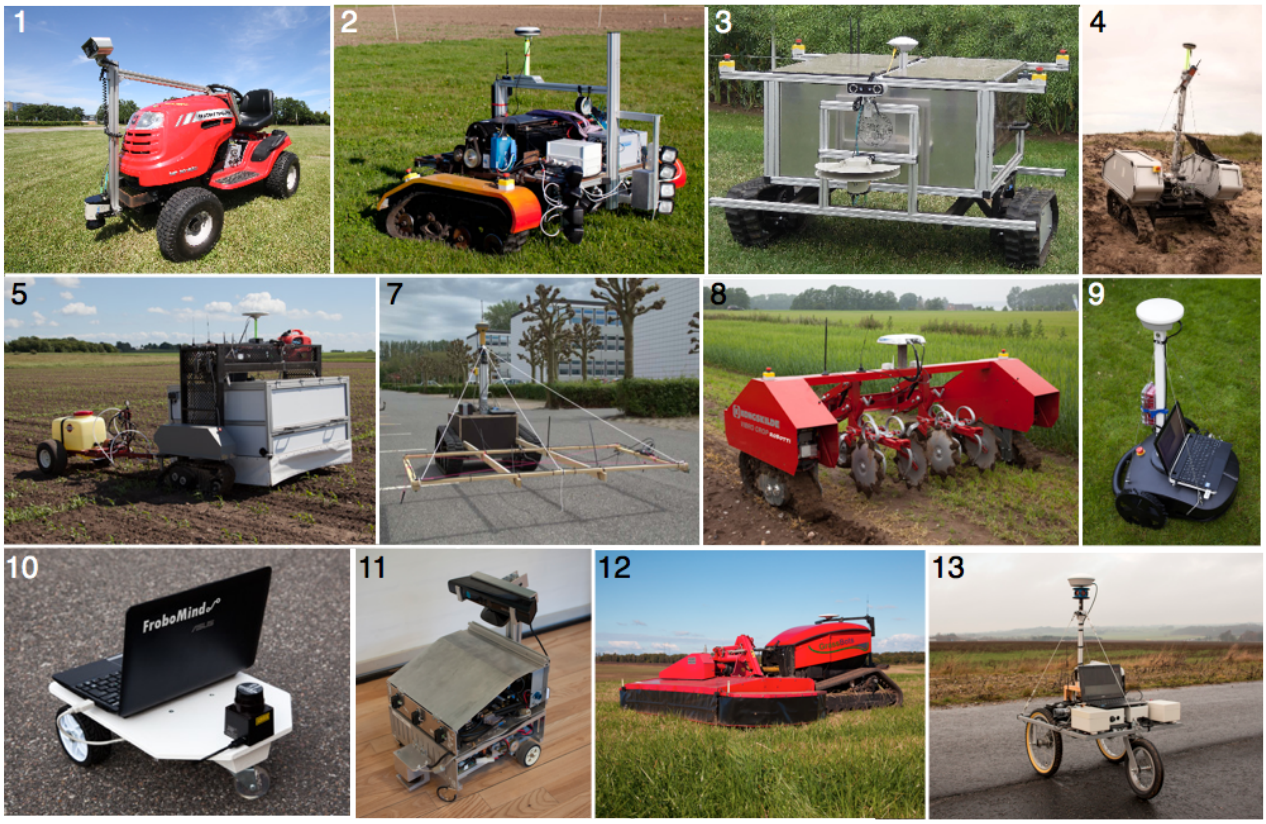


Table 9. List of components referred from table 8. Physical Source Lines Of Code (SLOC) for the Grassbot robot interface is unknown as it is closed source. c refer to C++ and p to Python.

Component	**Description**	**SLOC**
AMS	Interface to the AMS robot	410c
ARD	Interface to the Armadillo robot	845c
ASU	Interface to the ASuBot robot	435c
COV	Area coverage algorithm	170p
FBT	Interface to the Frobit robot	285c
FST	Interface to the FroboScout robot	428p
GBT	Interface to the Grassbot robot	?
HMP	Hazard mapping algorithm	850c
MBW	Interface to Mobotware [22]	367c
MDI	Mine Detection Implement interface	125c
ODO	Differential odometry algorithm	527p
PIC	Interface to the Pichi robot	890c
PMP	Parcel map localization algorithm	330p
POS	Pose estimator algorithm	592p
RDT	Detecting crop rows using LIDAR	266c
RNV	Navigation in row crops	255c
RWD	Row weeding implement interface	90c
SUR	Highway surveying application	304p
WPT	Waypoint navigation algorithm	775p

5. Discussion

5.1. Evaluating the performance of FroboMind in robot systems with near real-time requirements.

Ubuntu was chosen as operating system and ROS as middleware because they complied best with the design goals *extensibility*, *scalability*, *software reuse* and *open source*. This was, however, at the cost of hard real-time capabilities that could have been obtained using e.g. RTAI [26] and Orocos [14]. Although hard real-time capabilities are rarely required in field robotics except for low-level control which is typically handled by external embedded systems, it is important to know the system timing uncertainty.

The computing power in the three computers used in the performance evaluation in the first experiment (table 5) are at the low end of what is available today, however still above the currently popular ARM based embedded boards such as Raspberry Pi and BeagleBone Black.

The trials show a stable memory usage (fig.7) which is consistent with that none of the active FroboMind components perform any significant dynamic memory allocation. The approximate memory usage was approximately 1/3 Gb for PC1 where the GUI was disabled. For PC2 and PC3 the memory use was approximately 1 Gb.

The CPU load varies in the different trials which is to be expected considering the available computing power. The CPU load for PC1 exhibits a certain periodic pattern of peaks. A subsequent analysis revealed that the peaks were caused by an installed fan control script which sleeps 5 seconds between each execution, so this is not related to FroboMind.

Comparing the CPU load with the scheduler delays shown in fig.8 and table 6 it is clear that PC1 is pushed to the limits at this load and has difficulties running a 100 Hz scheduler without overstepping every now and then. PC2 and PC3 manages fine at the current load if the observed delays are acceptable to the application.

In other precision agriculture applications there may be components causing the CPU load to vary significantly with time which may be more or less predictable. Examples are dynamical mapping, route planning, image processing etc. In these cases there are different options: 1) Perform a similar experiment to validate that enough computing power is available to the application. 2) Distribute the components with varying load to another networked platform. 3) Integrate ROS with the Orocos Real Time Toolkit which is documented at the ROS website.

The results from this experiment gives rise to the recommendation that in any application where safety is a concern such as the use of large or heavy machinery, implements with moving mechanical parts, driving at high speed etc., the CPU load and the soft real-time performance should be monitored continuously, and detected anomalies should be used as input to the safety system.

5.2. Evaluating if FroboMind significantly decreases the resources required to perform field experiments

During the 5 day workshop the 4 man development team managed to interface FroboMind to the AMS and build the application for autonomous navigation of an orchard as well as testing the application in

two trials. The trials were completed, but navigation was not completely reliable. Upon reviewing the observations at the trials and subsequent analysis of logged ROS messages it was concluded that the reliability of the in-row navigation can be increased by a few modifications to the row detecting algorithm and the navigation controller.

The workshop revealed some issues with FroboMind that need to be addressed. 1) Some of the basic components needs further development with respect to reliability, in particular interfacing to a robot platform through low level interface drivers needs to work seamlessly. 2) Interfaces between layers in the architecture need to be reviewed and properly documented. 3) The usability of FroboMind must be enhanced by a better integration with a simulation environment like Stage. These issues have not yet been solved completely, but a number of improvements have been made since the workshop took place.

As shown in table 7 the developers worked approximately 12 hours/day each, about half the time was spent on interfacing and the other half on the orchard application. Factors like the described problems with the AMS, transport from the university to the orchard test site and changing weather conditions prolonged the work. Similarly the developer team had collectively an extensive experience in all part of the FroboMind software platform, and the AMS actuators and sensor interfaces were already installed and thoroughly tested using Mobotware, which eased the work. This makes it very difficult to compare with similar experiments, and thus to conclude if FroboMind significantly decreases the resources required to perform field experiments, but experiences from previous projects show that this is a very short time moving from a completely new software implementation on a robot to trials in the field navigating autonomously.

5.3. Evaluating software reuse across existing projects using FroboMind.

Table 8 lists a total of 13 robots that have been interfaced to FroboMind during the past three years. With the exception of the AMS and SMR robots that are based on Mobotware they have all been constructed within the same timeframe. The use of FroboMind is not widespread though, the listed robots are spread across only 4 universities and 3 companies, most of them working together developing the field robots. In addition to the list a smaller number of robots developed by researchers, companies and students are known to either use FroboMind or have used FroboMind as a starting point for the software development.

The statistical website ohloh.net reports that the FroboMind directory structure contains 104,617 SLOC (April 2014) and the core part of the ROS platform contains 418,977 SLOC (December 2013). In this context the third column of table 8 and the associated table 9 listing SLOC for the components that are shared between some but not all platforms could be considered an indication of the differences in the robot software rather than the similarities.

It seems reasonable to conclude that the level of software reuse between the listed robots is high which may in part be attributed to the modular structure of the FroboMind architecture and to the flexibility that ROS provides. It is, however, also caused by the fact that the robot platforms and their current applications are fairly similar with respect to the robot software.

5.4. Lessons learned

Choosing Ubuntu as operating system for the software platform has proven to be a great advantage. The limitations caused by not having hard real-time capabilities are counterbalanced by a number of advantages that all relates to simplicity in installation, use and maintenance and hence decreasing the time consumption from development to field trials. The ability to use the same operating system on the robot and the developers laptops speeds up the development process significantly.

The choice of ROS as middleware has had its advantages as well as drawbacks. ROS has a very steep learning curve in the beginning, but after some struggling with understanding the concepts it becomes a valuable tool. The user feedback indicates that FroboMind decreases the learning curve because it provides full working examples for simulation and execution on small robots rather than the user having to build the first test ROS setup from scratch. The ROS messaging system working across networked platforms, the ROS launch tool that ease launching of multiple ROS nodes locally and on networked computers, and the setting of parameters, as well as the many existing libraries and drivers etc. are highly valuable. The ROS tool for recording and playback of messages saves a tremendous amount of time when analyzing data and debugging. But ROS is still in a very active state of development. Experience shows that at new version releases, not all core components are fully working with the new version, and some of the new versions include substantial changes that induce unforeseen maintenance tasks. Examples are the introduction of the catkin build system and removal of the Stack concept in the ROS Groovy version. Seen in retrospect ROS was a better choice than Orocos when considering the purpose of FroboMind and the stated design goals. But it comes with the price of lacking hard real-time capabilities which is not a major problem but needs to be addressed in most applications, and the ROS code base appears to be less mature and stable than Orocos.

The FroboMind architecture design builds upon the results of the reviewed publications, especially Stanley [25], CARMEN [8] and [17] provided a significant input to the design. The architecture has undergone several revisions since the first prototype was developed in 2011. The original draft was very detailed but has been simplified significantly because experiences from using FroboMind in research projects have revealed a relationship between complexity and the willingness to accept and utilize the architecture. If the architecture is too complex, users tends to short circuit it by merging all the remaining functionality into a single or a couple of components, which makes them less suitable for reuse. It is debatable if the architecture is still too complex, this should be investigated in the future work.

An important part of the architecture is the specification of data exchanged between the components. For navigation sensors, localization and low level propulsion control and actuation this is reasonably well defined and where applicable aligned with current ROS standards. Interfacing the decision making layer and implement modules still need to be properly specified. This work should where applicable be based on existing standards within agriculture and related field work such as OpenGIS, agroXML, GPX, Isobus etc.

Compared to the architectures listed in table 2 FroboMind obtains a *Yes* in all parameters which indicates that the aim of this work has been achieved. An overall performance comparison of the FroboMind software platform to solutions described in the related work is impossible though, as data relevant for comparison of the architectural and software performance has not been published.

5.5. The future of FroboMind

549 The primary focus in the development of FroboMind has until now been to establish a common
550 open software platform that promotes software reuse and thus decreases the development time and
551 resources required to perform field experiments. This objective has to a large extent been achieved
552 and the work will continue though the application of FroboMind to new research projects, development
553 of new components as well as improving the existing components in terms of reliability.
554 In addition to this a project focusing on safe behaviour of agricultural machines has been launched
555 recently. FroboMind will be used in parts of the project, and it is expected that this will contribute
556 significantly to the development of: 1) The safety layer which is currently limited to a deadman
557 signal which enables the actuator controllers. 2) The decision making architecture, which currently
558 is limited to fairly simple behaviours defined in finite state machines. 3) source code integrity through
559 the implementation of model driven auto-generation of component interfaces and structures.

560 6. Conclusions

561 In this paper we have presented FroboMind, a software platform tailored to field robots in precision
562 agriculture. FroboMind optimizes field robot software development in terms of code reuse between
563 different research projects. At the current stage FroboMind has been ported to 2 different 4-wheeled
564 tractors, 7 different configurations of tracked field robots and 4 differentially steered robots. Examples of
565 current FroboMind applications are autonomous crop scouting, precision spraying, mechanical weeding,
566 grass cutting, humanitarian demining and land surveying.
567 In an experiment evaluating the performance of FroboMind in robot systems with near real-time
568 requirements it was concluded that FroboMind's soft real-time execution is sufficient for typical
569 precision agriculture tasks such as autonomous navigation of a predefined route. In any application where
570 safety is a concern, the CPU load and the soft real-time performance should be monitored continuously,
571 and detected anomalies should be used as input to the safety system.
572 To test whether FroboMind decreases the development time and resources required to perform
573 precision agriculture field experiments, an experiment was performed porting FroboMind to a new field
574 robot and applying this robot to autonomous navigation in an orchard using local sensors. This was
575 achieved by 4 developers during a 5 day workshop, which is a very short time moving from a completely
576 new software implementation on a robot to trials in the field navigating autonomously.
577 The software reuse across projects has been assessed using available data from existing projects and
578 robot platforms. It was concluded that the level of software reuse between the listed robots is high which
579 may in part be attributed to the modular structure of the FroboMind architecture, the flexibility that ROS
580 provides and the fact that the robot platforms and their current applications are fairly similar with respect
581 to the robot software.
582 It is concluded that FroboMind is usable in practical field robot applications by research groups and
583 other stakeholders. Further development of FroboMind continue through new research projects where the
584 current activities focus on the decision making structure and autonomous safe behaviour. The FroboMind
585 software has been released as open-source at http://www.frobomind.org for others to build upon.

586 Acknowledgements

This work is linked to and partially funded by the ERA-Net ICT-AGRI project: Grassbots, the Danish Ministry for Food, Agriculture and Fisheries project: FruitGrowth, and the Danish National Advanced Technology Foundation project: The Intelligent Sprayer Boom. The authors wish to thank Claes D. Jaeger and Hans W. Griepentrog, Ulrik P. Schultz and Mikkel K. Larsen for supporting this work.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Pedersen, S.M.; Fountas, S.; Have, H.; Blackmore, B.S. Agricultural robots - system analysis and economic feasibility. *Precision Agriculture* **2006**, *7*, 295–308.
2. Green, O.; Schmidt, T.; Pietrzkowski, R.P.; Jensen, K.; Larsen, M.; Jørgensen, R.N. Commercial autonomous agricultural platform - Kongskilde Robotti. Robotics, Associated High-Technologies and Equipment for Agriculture and Forestry, 2014.
3. Bakker, T.; van Asselt, K.; Bontsema, J.; Muller, J.; van Straten, G. Systematic design of an autonomous platform for robotic weeding. *Journal of Terramechanics* **2010**, *47*, 63–73.
4. Ruckelshausen, A.; Biber, P.; Dorna, M.; Gremmes, H.; Klose, R.; Linz, A.; Rahe, R.; Resch, R.; Thiel, M.; Trautz, D.; Weiss, U. BoniRob: an autonomous field robot platform for individual plant phenotyping. Precision agriculture 09, 2009.
5. Jørgensen, R.; Sørensen, C.; Pedersen, J.; Havn, I.; Jensen, K.and Søgaard, H.; L.B., S. Hortibot: A system design of a robotic tool carrier for high-tech plant nursing, 2007.
6. Blackmore, B.S.; Griepentrog, H.W.; Nielsen, H.; Nørremark, M.; Resting-Jeppersen, J. Development of a deterministic autonomous tractor, 2004.
7. Neisser, U. *Cognitive Psychology*; Meredith Publishing, 1967.
8. Montemerlo, M.; Roy, N.; Thrun, S. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit, 2003.
9. Nesnas, I.A.D.; Wright, A.; Bajracharya, M.; Simmons, R.; Estlin, T. CLARAty and challenges of developing interoperable robotic software. *Iros 2003: Proceedings of the 2003 Ieee/rsj International Conference On Intelligent Robots and Systems, Vols 1-4* **2003**, pp. 2428–2435.
10. Pivtoraiko, M.; Nesnas, I.A.; Nayar, H.D. A Reusable Software Framework for Rover Motion Controls, 2008.
11. Cepeda, J.; Chaimowicz, L.; Soto, R. Exploring Microsoft Robotics Studio as a Mechanism for Service-Oriented Robotics. Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American, 2010, pp. 7 –12.
12. Makarenko, A.; Brooks, A.; Kaupp, T. Orca: Components for Robotics. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006). Workshop on Robotic Standardization., 2006.
13. Henning, M. A New Approach to Object-Oriented Middleware. IEEE Internet Computing, 2004.
14. Bruyninckx, H. Open robot control software: the OROCOS project, 2001.

15. Gerkey, B.P.; Vaughan, R.T.; Sukhatme, G.S.; Stoy, K.; Mataric, M.J.; Howard, A. Most valuable player: A robot device server for distributed control, 2001.
16. Quigley, M.; Conley, K.; Gerkey, B.P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software, 2009.
17. Vázquez-martín, R.; Martinez, J.; Toro, J.C.; Núñez, P. A Software Control Architecture based on Active Perception for Mobile Robotics 1, 2008.
18. Brooks, R.A. A Robust Layered Control-system For A Mobile Robot. *Ieee Journal of Robotics and Automation* **1986**, *2*, 14–23.
19. Patricio Nebot, J.T.S.; Martinez, R.J. A New HLA-Based Distributed Control Architecture for Agricultural Teams of Robots in Hybrid Applications with Real and Simulated Devices or Environments. *Sensors* **2011**, *11 (ISSN 1424-8220)*, .
20. Garcia-Perez, L.; Garcia-Alegre, M.C.; Ribeiro, A.; Guinea, D. An agent of behaviour architecture for unmanned control of a farming vehicle. *Computers and Electronics In Agriculture* **2008**, *60*, 39–48.
21. Kuhnert, K.D. Software Architecture of the Autonomous Mobile Outdoor Robot AMOR, 2008.
22. Beck, A.B.; Andersen, N.A.; Andersen, J.C.; Ravn, O. MobotWare—A plug-in based framework for mobile robots. 7th IFAC Symposium on Intelligent Autonomous Vehicles, 2010, Vol. 7, pp. 127–132.
23. Simon Blackmore, S.F.; Have, H. Proposed System Architecture to Enable Behavioral Control of an Autonomous Tractor. Automation Technology for Off-Road Equipment; American Society of Agricultural and Biological Engineers (ASABE), , 2002; Vol. Proceedings of the July 26-27, 2002 Conference, pp. 13–23.
24. Fountas, S.; Blackmore, B.S.; Vougioukas, S.; Tang, L.; Sørensen, C.G.; Jørgensen, R. Decomposition of Agricultural tasks into Robotic Behaviours. *Agricultural Engineering International: the CIGR Ejournal* **2007**, *IX*.
25. Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; Lau, K.; Oakley, C.; Palatucci, M.; Pratt, V.; Stang, P.; Strohband, S.; Dupont, C.; Jendrossek, L.E.; Koelen, C.; Markey, C.; Rummel, C.; van Niekerk, J.; Jensen, E.; Alessandrini, P.; Bradski, G.; Davies, B.; Ettinger, S.; Kaehler, A.; Nefian, A.; Mahoney, P. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics* **2006**, *23*, 661–692.
26. Dozio, L.; Mantegazza, P. Linux Real Time Application Interface (RTAI) in low cost high performance motion control. Motion Control 2003, a conference of ANIPLA, Associazione Nazionale Italiana per l'Automazione (National Italian Association for Automation); , 2003.
27. Carsten Emde, T.G. Quality assessment of real-time Linux. Technical report, Open Source Automation Development Lab, 2011.
28. McKenney, P.E. 'Real Time' vs. 'Real Fast': How to Choose? Technical report, IBM Linux Technology Center, 2008.
29. Leonard, J.; Durrant-Whyte, H. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on* **1991**, *7*, 376–382.

30. Stuart Russell, P.N. *Artificial Intelligence: A Modern Approach (Third Edition)*; Prentice Hall, 2010.
31. Schacter, D. *Psychology*; Worth Publishers, 2011.
32. Jaeger-Hansen, C.; Jensen, K.; Larsen, M.; Nielsen, S.; Griepentrog, H.; Jørgensen, R. Evaluating the portability of the FroboMind architecture to new field robot platforms. 9th Europaean Confernce on Precision Agriculture, Lleida, Spain, 2013.
33. Barawid, O.C.; Mizushima, A.; Ishii, K.; Noguchi, N. Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application. *Biosystems Engineering* **2007**, *96*, 139–149.
34. Blas, R. Fault-Tolerant Vision for Vehicle Guidance in Agriculture. PhD thesis, DTU, 2010.
35. Reske-Nielsen, A.; Mejnertsen, A.; Andersen, N.A.; Ravn, O.; Nørremark, M.; Griepentrog, H.W. Multilayer Controller for Outdoor Vehicle. In Proceedings Automation Technology for Off-Road Equipment (ATOE), 2006, pp. 41 – 49.
36. Griepentrog, H.; Andersen, N.; Andersen, J.; Blanke, M.; Heinemann, O.; Madsen, T.; Nielsen, J.; Pedersen, S.; Ravn, O.; Wulfsohn, D. Safe and reliable: further development of a field robot. Precision agriculture 09, 2009.
37. Jensen, K.; Laursen, M.S.; Midtiby, H.; Jørgensen, R.N. Autonomous Precision Spraying Trials Using a Novel Cell Spray Implement Mounted on an Armadillo Tool Carrier. XXXV CIOSTA & CIGR V Conference, Billund, Denmark, 2013.
38. Larsen, L.B.; Olsen, K.S.; Ahrenkiel, L.; Jensen, K. Extracurricular Activities Targeted towards Increasing the Number of Engineers Working in the Field of Precision Agriculture. XXXV CIOSTA & CIGR V Conference, Billund, Denmark, 2013.

Appendix B

Paper 2

Evaluating the accuracy and reliability of a new low-cost GPS.

Kjeld Jensen, Morten Larsen, Tom Simonsen and Rasmus N. Jørgensen

Presented at the *First RHEA International Conference on Robotics and associated High-technologies and Equipment for Agriculture. September 19-21, 2012, Pisa, Italy.*

First RHEA International Conference on Robotics and associated High-technologies and Equipment for Agriculture
Hosted by the University of Pisa
Pisa, Italy, September 19-20-21, 2012

TOPIC n° 2.1

Evaluating the performance of a low-cost GPS in precision agriculture applications

Kjeld Jensen[1], Morten Larsen[1], Tom Simonsen[2], Rasmus N. Jørgensen[1]

[1]*Institute of Chemical Engineering, Biotechnology and Environmental Technology, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark*

[2]*Conpleks Innovation ApS, BusinessPark Struer, Fælledvej 17, DK-7600 Struer*

Keywords: precision agriculture, gps, field robot

Abstract: Field Robots are often equipped with a Real Time Kinematic (RTK) GPS to obtain precise positioning. In many precision agriculture applications, however, the robot operates in semi-structured environments like orchards and row crops, where local sensors such as computer vision and laser range scanners can produce accurate positioning relative to the crops. GPS is then primarily needed for robust inter-row navigation.

This work evaluates a new low-cost GPS. Static tests were used to test the absolute accuracy. To test the GPS in a precision agriculture environment it was installed on a robot driving in a simulated row crop field. The GPS supports raw data output as well and similar experiments were performed to evaluate the GPS when used in a RTK setup.

In field tests more than 95% of the position errors were estimated to be within 2.6 m. In RTK field tests more than 95% of the position errors were estimated to be within 0.2 m. It was concluded that the GPS can be applied to selected applications in row crops and orchards if augmented by local sensors and mapping techniques. Using the GPS in a RTK setup applies to general applications where position errors of 0.2 m are acceptable.

1. Introduction

Field robots often use a Global Positioning System (GPS) based on Real Time Kinematic (RTK) for precise navigation. However in a typical agricultural environment there are areas where trees, buildings, power lines etc. may shade or reflect the satellite signals. This causes temporary drops in positioning accuracy, and some means of dead reckoning system is therefore needed as well. RTK equipment retails at USD 15,000 to 60,000 and significantly increases the cost of field robots (Grisso et al., 2009).

In precision agriculture the robot often operates in semi-structured environments like orchards and row crops where local sensors like computer vision and laser range scanners can produce accurate positioning relative to the crops. GPS is then primarily needed for inter-row navigation and mapping, and inaccuracies at the order up to a few meters may be acceptable in some applications.

Recently a new low cost GPS has been introduced. At a retail price of USD 180 the specifications claim an increased accuracy in static and slow moving applications. It supports raw data output as well, which allows use in RTK setups using an external computation library. It is hypothesized that this GPS is useful in some precision agriculture applications in semi-structured environments, and the aim of this work is to evaluate the GPS performance with respect to those applications.

2. Materials and methods

In this work a u-blox NEO-6P GPS engineering sample was evaluated. For RTK two LEA-6T GPS were used since the engineering sample did not have raw data output. u-blox has confirmed that the engineering sample algorithms are identical with the current version and that the raw data available on the NEO-6P is equal to that on the LEA-6T.

To test the NEO-6P a prototype board was designed and linux software was written to configure the NEO-6P. In all tests the NEO-6P was connected to a GNSSA200 antenna from Gutec

AB. In RTK tests the LEA-6T were connected to Trimble Patch antennas, and the RTKLIB v2.4.1 (Takasu and Jasuda, 2010) was used for precise positioning. Post-processing was performed using Python.

3. Results and conclusions

3.1 Static tests

These tests evaluate the absolute accuracy during variations in the satellite constellation and atmospheric disturbances. The antenna was installed at a location with no significant obstacles above the NEO-6P default elevation mask of 5 degrees. The position of the antenna was estimated by averaging RTK fixed solution measurements using a DataGrid MK3 receiver and is assumed to be exact. A GPS antenna signal splitter was constructed to split the signal between the NEO-6P and a Garmin GPS60 receiver. Fig.1 (left) shows the results of a 95 hour test with the NEO-6P configured for 5 Hz output rate. 99% of the outputs returned DGPS fix indicating that SBAS was used to improve the positioning quality. 95% of the measurement errors were within 1.98 m. For the Garmin GPS60 95% of the measurement errors were within 2.60 m. A number of tests were performed with similar outcome.

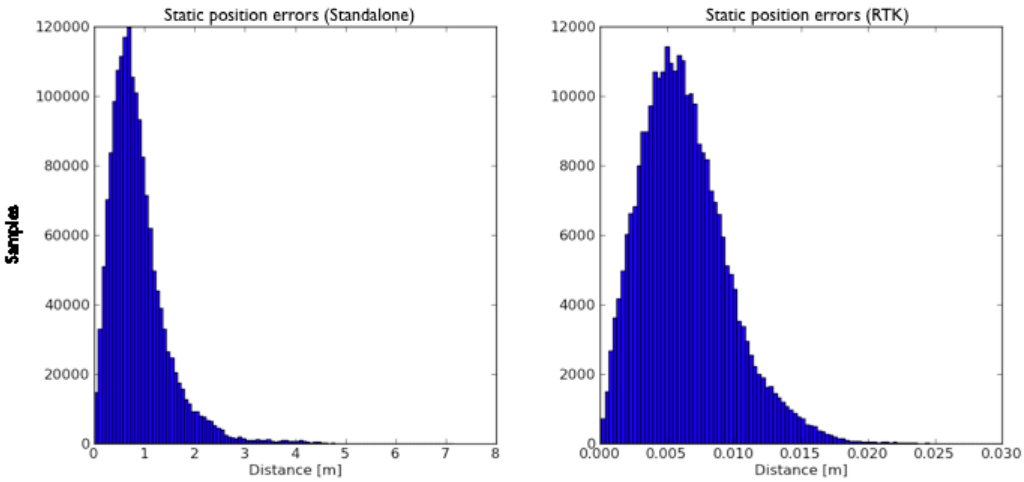


Fig.1. **Static tests**

In the RTK test a LEA-6T was configured to output raw data at 5 Hz, and a PC with RTKLIB performed the precise position

estimation. Another LEA-6T with same configuration was used as reference station less than 50 meters away. Fig.1 (right) shows the results of an 8 hour test where 95% of the measurement errors were within 0.013 m. 0.12% of the measurements with an error less than 0.75 m has been excluded from the histogram.

3.2 Field tests

These tests evaluate the performance in a precision agriculture scenario. A route that simulates driving in row crops was laid out on a 15x15 m grass field using an inter-row spacing of 3 m. Close to the grass field were scattered groups of trees and a barn. The NEO-6P was installed on an Armadillo field robot. A Topcon GRS-1 unit was used to collect reference data which is assumed to be exact. The route was driven three times with 8 and 3 hours between. Only in test 3 did the NEO-6P report DGPS quality in 14% of the position outputs. Fig. 2 (left) shows the result of test 3 which had the largest deviation from the reference track. The black line is the GRS-1 reference track and the green line is the NEO-6P track. The NEO-6P antenna location 0.3 m in front of the GRS-1 reference antenna has not been compensated due to lack of accurate yaw angle estimation. In all tests more than 95% of the measurement errors were estimated to be within 2.6 m.

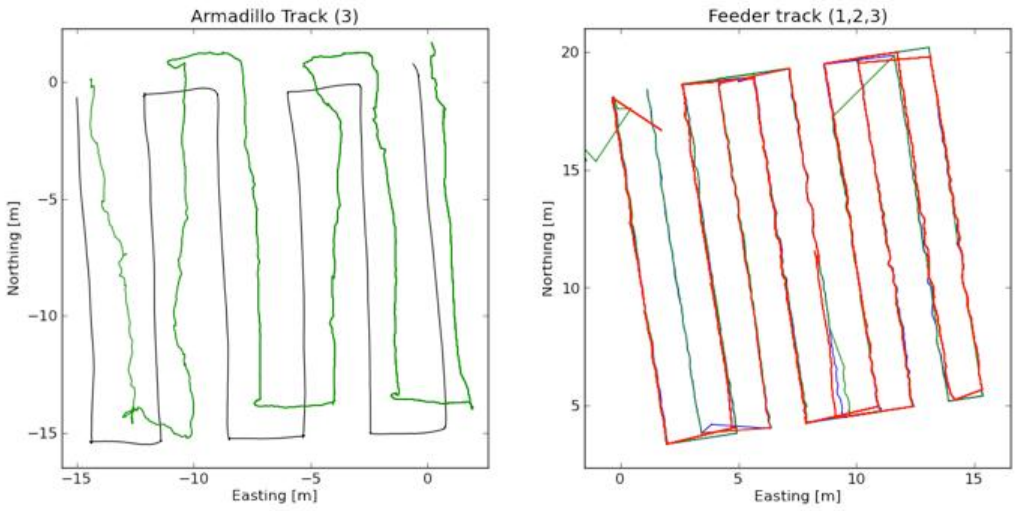


Fig.2. **Field test results**

In the RTK test a route was laid out on a 15x15 m grass field using an inter-row spacing of 1.5 m. Close to the grass field were one storey buildings. The vehicle used for this test is a diesel-driven feeder driving at 0.25 m/s. The LEA-6T setup is the same as described in the static RTK test. The route was driven three times with 25 and 1 hours between. The lateral driving accuracy is estimated to be within +/- 5 cm. Fig.2 (right) shows the resulting tracks measured by the LEA-6T and filtered to exclude the headland turns. Row 6 from the left seems to be affected by a convergence error in the southern end. This may have been caused by signal blocking or multipath conditions caused by buildings just south of the track. The offset between the three individual tests in this worst case situation shows an estimated accuracy of approx. +/- 0.2 m.

In order to improve the state estimation accuracy and reliability an Extended Kalman Filter (EKF) (Larsen 1998) was created. Odometry from the Armadillo motor hall sensors was used as system input and the NEO-6P positions as measurement updates. The NEO-6P gave quite scattered position outputs in the field test 2 so data from this test was used for evaluation.

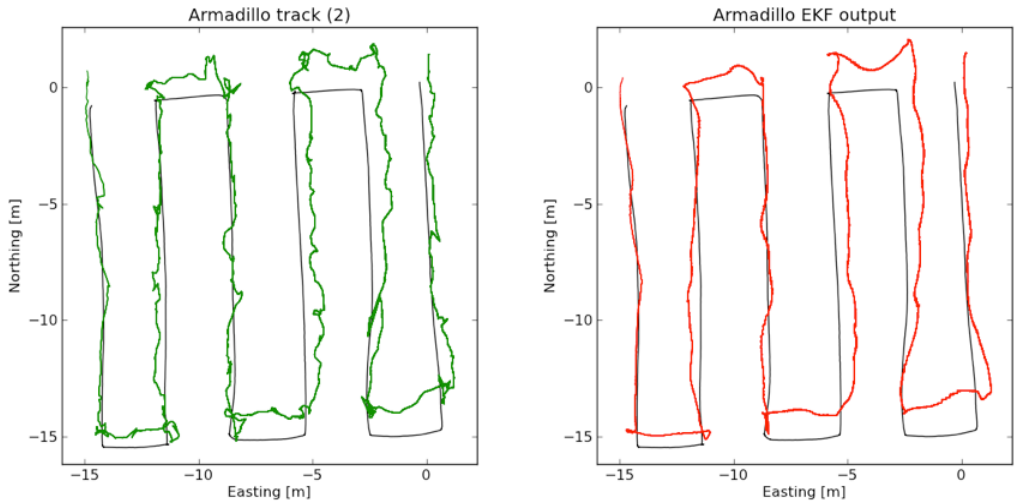


Fig.3. **Armadillo track and EKF output**

Fig.3. shows the result. The black line is the GRS-1 reference track, (left) is the NEO-6P track and (right) is the EKF state

estimate. The NEO-6P antenna offset from the robot geometric center has not been compensated, but even so it is evident that the EKF improves the position estimation accuracy significantly. Similar results though not as distinct were obtained from the other tracks. To test the reliability of the EKF several experiments were carried out using the same data but with GPS fix reset for periods of 5 to 20 seconds. In all tests the EKF managed to produce sensible output relying on odometry data until the next GPS fix corrected the absolute position.

3.3 Conclusion

This work has evaluated the u-blox NEO-6P GPS performance in precision agriculture applications. In standalone static tests 95% of the position errors were within 1.98 m. In RTK static tests 95% of the position errors were within 0.013 m. In all standalone field tests more than 95% of the position errors were estimated to be within 2.6 m and in all RTK field tests more than 95% of the position errors were estimated to be within 0.2 m. It is concluded that the NEO-6P can be applied to selected row crop and orchard applications if augmented by local sensors and mapping techniques. The NEO-6P (LEA-6T) in RTK setup applies to general applications where 0.2 m position errors are acceptable. Prolonged tests and antenna vibration tests were performed in addition to the described tests. They gave no reason to review the above conclusion. An EKF was implemented to improve the robot state estimation. Work ahead is to add local sensors to enable robust navigation in orchards.

Acknowledgements

The authors wish to thank More Electronics, Thorbjørn Jørgensen, Anders G. Pedersen, Hjalte Nygaard and Carsten Albertsen for supporting this work.

References

Grisso, R., Alley, M. and Heatwole, C. *Precision Farming Tools: Global Positioning System (GPS)* Virginia Cooperative Extension Publication 442-503. 2009.

Takasu, T. and Yasuda, A., *Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB*, International Symposium on GPS/GNSS, International Convention Center Jeju, Korea, November 4-6, 2009

Larsen, T. *Optimal fusion of sensors*. PhD Thesis, The Technical University of Denmark. 1998.

Appendix C

Paper 3

Autonomous Precision Spraying Trials Using a Novel Cell Spray Implement Mounted on an Armadillo Tool Carrier.

Kjeld Jensen, Morten S. Laursen, Henrik Midtiby and Rasmus N. Jørgensen

Presented at the *International Commission of Agricultural and Biological Engineers, Section V. CIOSTA XXXV Conference "From Effective to Intelligent Agriculture and Forestry", Billund, Denmark, 3-5 July 2013.*

Autonomous Precision Spraying Trials Using a Novel Cell Spray Implement Mounted on an Armadillo Tool Carrier.

Kjeld Jensen [1], Morten S. Laursen [1], Henrik Midtiby [1], Rasmus N. Jørgensen [2]
[1]Faculty of Engineering, University of Southern Denmark, Campusvej 55, 5230 Odense.
[2]Institute of Engineering, Aarhus University, Nordre Ringgade 1, 8000 Aarhus, Denmark.
Corresponding author: kjen@mmmi.sdu.dk

ABSTRACT

Precision weeding is one of the most promising applications for autonomous service robots in biological production. Herbicides have been the default weeding solution during the past decades, but there is a growing concern about the environmental impact on drinking water reservoirs etc. The use of computer vision and precision spraying technology makes it possible to significantly reduce the consumption of herbicides.
The work presented here is part of a project with the purpose of performing autonomous precision spraying trials. In this work a novel cell sprayer designed for large scale tests of weed detection algorithms and spraying strategies is developed. The front part of the cell sprayer implement is a camera module mounted in a shock absorbing frame. The camera module is followed by a spraying module containing nozzles with individual valve control. The implement is designed for use with an Armadillo robotic tool carrier consisting of two battery powered track modules mounted on each side of the implement.
This paper focus on the cell sprayer implement design including camera system, sprayer module and integration with the service robot and the robot software. The FroboMind software platform and Armadillo robot is used and it is hypothesized that utilizing FroboMind the cell sprayer can drive smoothly through a test field with a lateral positioning accuracy better than 50 mm.
A precision spraying trial in a 1 Ha maize field using different treatment methods was used for testing the hypothesis. The results substantiates that the cell sprayer is capable of navigating the test field smoothly with a lateral positioning accuracy better than 50 mm. The test results shows that over a distance of approximately 145 m the 95% percentile of the absolute distance to the AB line is 0.0250 m and the 95% percentile of the absolute heading error is 1.17 degrees which is within the requirements of the cell spray implement.
The results from the precision spraying trail with respect to weeding performance will be published in a later publication and the cell sprayer is expected to be utilized in further precision spraying experiments.

Keywords: Precision weeding, cell sprayer, Modicovi, Armadillo robot, FroboMind, Denmark

---

Jensen, K., Laursen, M.S., Midtiby, H., Jørgensen, R.N. “Autonomous precision spraying trials using a novel cell spray implement mounted on an Armadillo tool carrier”. International Commission of Agricultural and Biological Engineers, Section V. CIOSTA XXXV Conference “From Effective to Intelligent Agriculture and Forestry”, Billund, Denmark, 3-5 July 2013.

1. INTRODUCTION

Precision weeding is one of the most promising applications for autonomous service robots in biological production (Pedersen et. al., 2006, Sørensen et. al., 2007). The extensive use of herbicides during the past decades (Kurstjens 2007) is being challenged by a growing concern among consumers and political decision makers about the potential environmental impact on drinking water reservoirs, fauna in watercourses etc. In 2010 pesticide residues were found in 44% of the drinking water monitoring points in Denmark, the requirement of a concentration of maximum 0.1 µg/l was exceeded in 15% of the monitoring points. (Thorling et. al., 2011).

Using computer vision technology to detect the location of individual weed plants and precision spraying technology to accurately dispense only the required amount of herbicide given the detected weed concentration makes it possible to significantly reduce the consumption of herbicides (Midtiby 2012). However both computer vision and precision spraying are technologies that require significant research and development efforts. At the same time they make heavy demands on the robot platform with respect to performance parameters such as precise navigation and vibration avoidance as well as infrastructure in terms of carrying capacity, power, protection against rain and dust, etc. To date only few complete robotic weed control systems have been tested under field conditions (Slaughter et. al., 2008)

The work presented in this paper is part of a project with the purpose of performing large scale autonomous precision spraying trials using a novel cell sprayer. This work focus on the cell sprayer implement design including camera system, sprayer module and integration with the service robot and the robot software. The open source FroboMind software platform and Armadillo robot developed at the University of Southern Denmark is used and we hypothesize that utilizing FroboMind the cell sprayer can drive smoothly through a test field with a lateral positioning accuracy better than 50 mm.

The first task of the cell sprayer is to perform a precision spraying trial in maize using different treatment methods while at the same time registering the impact of the treatments on the crops and weed. This trial will be used for testing our hypothesis.

2. MATERIALS AND METHODS

2.1 Trial description

The trial site is located at Flakkebjerg, Denmark, where a 1 hectare field of maize was sown (55.326462° N, 11.382475° E). During sowing the crop row lines were stored using a Real Time Kinematics (RTK) - Global Navigation Satellite System (GNSS). The field is divided into 1177 test parcels with a size of 3 m along the crop rows and 1.5m across. Between each parcel a buffer zone is placed to avoid boundary effects. The location of each parcel is described by its four corner coordinates. Based on the robot position estimation it is possible to determine the parcel number and the associated treatment of the parcel. In about 30% of the parcels specific weed plants were sown in lines (figure 2).

Jensen, K., Laursen, M.S., Midtiby, H., Jørgensen, R.N. “Autonomous precision spraying trials using a novel cell spray implement mounted on an Armadillo tool carrier”. International Commission of Agricultural and Biological Engineers, Section V. CIOSTA XXXV Conference “From Effective to Intelligent Agriculture and Forestry”, Billund, Denmark, 3-5 July 2013.

12 different treatments are applied each to a random subset of the test parcels. The treatments are based on two different weed detection algorithms, each with 5 different thresholds of when to spray a cell. The last two treatments are *spray the entire parcel* and *no spraying*, both to build a reference for evaluation of the results. In the first algorithm weed plants are detected using a monocot/dicot separation algorithm (Laursen et. al., 2012), the second detects weed plants by analyzing the plant location with respect to the crop plant grid structure (Midtiby 2012).

A visualization of the parcel placement in one half of the field is shown in figure 1. Each parcel is shown as a small black rectangle. The planned route which covers all the parcels is shown in red, green and blue. The total length is 2520 m. Due to physical constraints on the attachment of the trailed tank which carried the spray liquid, it was not feasible to make left turns, and right turns had to use small turn angles. The shown route consists only of right turns and during each turn at least two rows are skipped which increases the turning radius to a feasible level. The trailer are capable of carrying enough spray liquid to cover a distance of 900 m with all the spray nozzles turned on. Two pit stops for trailer refill are placed on the route such that each part of the route between two stops has a length of approximately 900 m.

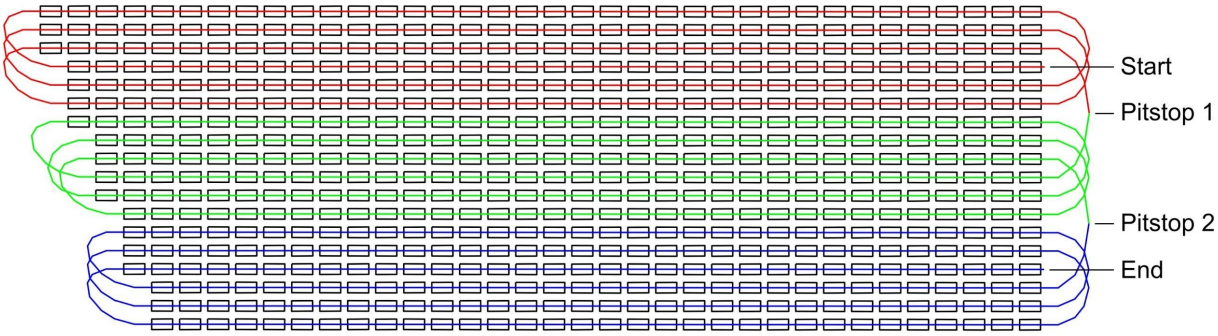


Figure 1. Overview of the test field parcels and route plan.

Figure 2. The test field with visible lines of weed plants

After completion of the sowing each parcel is treated with the cell sprayer twice. The first treatment is scheduled 21 days and the second treatment 31 days after sowing. At each treatment and a scouting drive scheduled 41 days after sowing, images from all the parcels are acquired to evaluate the weed infestation rate of each parcel.

2.2 Cell spray implement

The cell spray implement consists of two chambers, one for image acquisition and one for spraying. The image acquisition chamber (figure 3) measuring 1.8 x 1.5 x 1.1 m is made of alubond with a matt white finish that shields the outside illumination and results in diffuse reflections. Inside the acquisition chamber the illumination consists of 36 Philips TL-D 90 Graphica 58W fluorescent tubes, providing a 5000 kelvin illumination with a colour rendition index of 97%. In front of the fluorescent tubes a sheet translucent polycarbonate is placed (Makrolon GP white) in order to ensure diffuse illumination. This corresponds to an illumination of 78840 Lumen, which generates an intensity of approximately 30,000 Lux. Inside the chamber two Basler acA2000-50gc cameras mounted with Azure 1214M5M lenses take care of the image capturing, the cameras are global shutter CMOS cameras with 5.5 µm pixels at a resolution of 2048 x 1088 each having a 60 dB snr. The camera focal points are 920 mm above the ground and the lenses adjusted to F/4.0, resulting in an airy disc of 5.3 µm. The lenses are focused at 820 mm, resulting in circle of confusion less than 5.5 µm from 737 mm to 924 mm, resulting in in-focus images with object heights from - 4 - 183 mm (figure 4).

Figure 3. The image acquisition chamber with cameras and illumination tubes

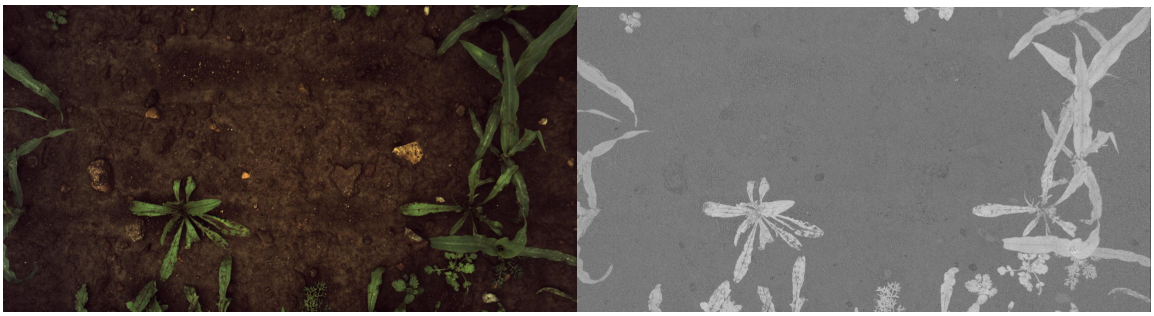


Figure 4. Left: Example image acquired by the camera. Right: Image after filtering.

Figure 5. Left: Row of spray nozzles. Right:Spray chamber with windbreaks installed.

The spray chamber (Laursen et. al., 2012a) (figure 5) is a WxHxD 1800x500x300 mm chamber mounted with windbreaks in order to reduce wind drift. Inside the chamber 6 high speed solenoid Weedseeker VC-01 valves are mounted for turning the spraying on/off. Each valve is mounted with a Hardi LD-01 110^{0} nozzle. The nozzle is mounted 250 mm above the ground. The valves are pressurized using a Hardi ATV spray tank with pressure release valve. The pressure was set at 2.8 bar with the valves closed, which resulted in 2.2 bar with all the valves open.

2.2 Robot tool carrier

The cell spray implement is carried by the Armadillo robot in a shock absorbing frame (figure 6,7) (Jensen et al., 2012). The Armadillo robot consists of two independent track modules each containing a 48V, 5kW brushless DC motor, transmission and a RoboteQ HBL1650 motor controller. The module is supplied by four 12V 65Ah AGM batteries each mounted with a CTEK MSX 10A charger.

Figure 6. Cell sprayer front view.

Figure 7. Cell sprayer rear view.

In this work a new version of the Armadillo track module (figure 8) is tested. The track module has been redesigned to optimize towards higher efficiency and durability. The frame consists of two metal plates supporting the sprockets for a two step transmission with a combined ratio of 1:19.5. The total weight of a track module excluding the battery case and its content is 97 kg.

Figure 8. The new Armadillo track module design and image from driving in mud.

The Armadillo robot supports odometry feedback originating from the brushless motor hall sensors (625 ticks/meter) and is in this work equipped with a VectorNav VN-100 Inertial Measurement Unit (IMU) and a Trimble BX982 RTK-GNSS receiver connected to the GPSnet.dk reference network.

2.4 Robot software

The Armadillo robot is controlled by a industrial PC (dual core, 2.x GHz) running the open source ROS (Quigley et. al., 2009) based FroboMind software platform www.frobomind.org (Jensen et al., 2012a). In this work new components for Transverse Mercator projection

conversion, robot pose estimation, waypoint navigation and parcel map support have been developed and added to the FroboMind repository.

2.4.1 Transverse Mercator Projection

In precision agriculture the Universal Transverse Mercator (UTM) projection is often used as reference for absolute positioning. However the pose estimation and navigation software implementations often lack support for transitions between different UTM zones and thus cause singularities when operating across zone borders. This is mitigated in the Transverse Mercator (TM) conversion component by passing projection parameters directly. The component also facilitates direct support for regional or local conventions like in Denmark where UTM zone 32 references are now used in areas physically located in UTM zone 33. The component is based on work documented in (Snyder 1987, Gloeckler et al., 1996). Newer and even more accurate TM conversion implementations exist but the accuracy of this one is sufficient for the purpose and the computational load is acceptable.

2.4.2 Robot pose estimator

Estimation of the robot absolute position and orientation (pose) is a key issue in field robotics as inaccurate pose estimates make the robot navigation and implement operation inaccurate and sometimes impossible. An typical pose estimation approach is to implement a Kalman filter (Thrun et. al., 2006) using wheel odometry feedback and optional gyro sensor measurements for the system update (prediction), and GNSS receiver data for the measurement update (correction) (Rodrigues et al. 2009, Larsen et al. 1999).

The pose estimator developed in this work is based on an Extended Kalman Filter (EKF) that is prepended by a preprocessing layer (figure 9). The purpose of preprocessing is to validate and filter the input based on a priori knowledge of the system and hence mitigate the effects of sensor data errors often experienced in the field.

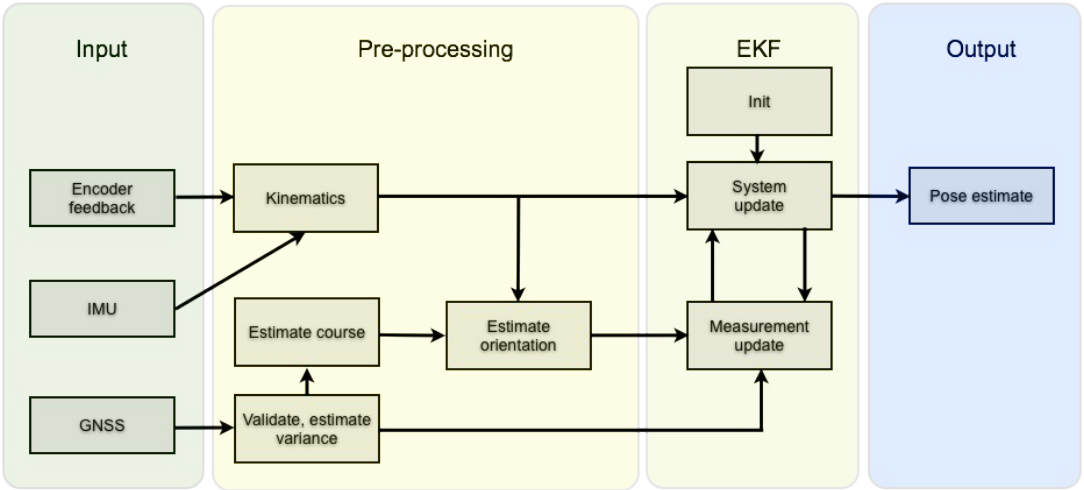


Figure 9. Pose estimator diagram

The kinematics module uses odometry feedback to estimate the robot relative linear motion and the yaw axis gyro embedded in an IMU to estimate the robot relative angular motion.

Data from the RTK-GNSS is validated by rejecting sudden position jumps seen on some receivers. Position jumps are typically caused by multipath conditions or changes in satellite configuration, and often a jump is followed by a reverse jump after a few measurements. The jumps are sometimes but not always reflected in a changed solution, changed number of satellites and/or the horizontal dilution of precision (HDOP) values that often accompany the position output. The validation module screens for position changes that would exceed the robot maximum speed. Invalid jumps cause a reject of the current and subsequent RTK-GNSS data for a defined period. If the RTK-GNSS data is valid, a variance is calculated based on the solution and HDOP value.

The robot course is calculated using the current RTK-GNSS position and the newest RTK-GNSS position at a distance of more than a defined threshold. In this work the threshold was set to 0.25 m. The course is only calculated if all RTK-GNSS data received within this interval are valid and are based on a RTK fixed solution. The course is used as input to the orientation estimation where the orientation is set equal to the course under a set of conditions: Both odometry and RTK-GNSS course data are valid during the entire interval by which the GNSS course was calculated, the robot speed is positive (driving forward), the difference between the odometry distance and GNSS course distance is below a small threshold, and the change in odometry angle is small (driving straight ahead).

Field tests performed so far on various robot platforms and using different IMU's and GNSS receivers have shown that the pose estimator including the absolute orientation estimation works accurate and reliable.

2.4.3 Waypoint navigation

Navigating the planned route through the parcels is handled by the waypoint navigation component. The route plan is described as a list of subsequent waypoints. Rather than heading directly towards the next destination waypoint in the list the robot seeks to navigate along a straight line between the destination waypoint **B** and origin waypoint **A** of the particular route segment (figure 10).

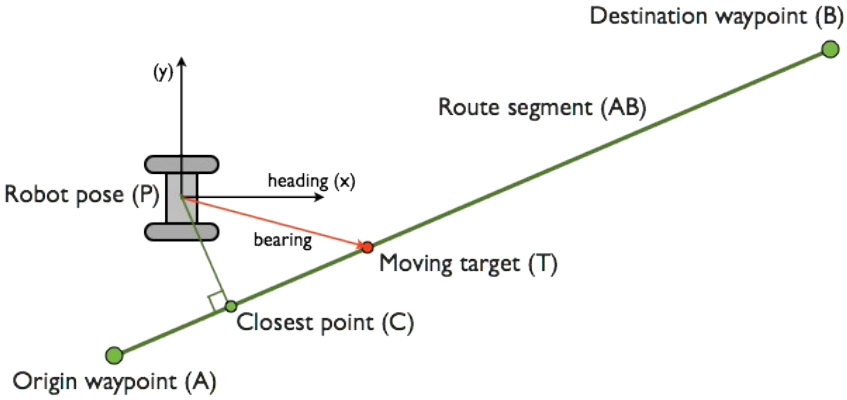


Figure 10. AB line navigator

To keep as close to the **AB** line as possible, a moving target **T** is introduced. **T** is located on **AB** ahead of **C** which is the point on **AB** that is closest to the robot pose **P**. **T** is defined by a piecewise function depending on **P** (equation 1): If the robot is behind the **AB** line it first navigates towards **A**. While parallel to the line it navigates towards the moving target **T**. The target distance from **C** to **T** is defined by a constant *t* until the distance from **C** to **B** becomes less than *2t*. The target distance then becomes half the distance to **B**. If the robot somehow ends up passing **B** outside the acceptable range threshold, it will navigate directly towards **B**. In this work *t* was set to 0.9 meter.

$$\mathbf{T} = \begin{cases} \mathbf{A} & d \leq 0 \\ \mathbf{A} + (d+t)\widehat{\mathbf{AB}} & 0 < d < ||\mathbf{AB}|| - 2t \\ \mathbf{P} + (d + \frac{||\mathbf{PB}||}{2})\widehat{\mathbf{AB}} & ||\mathbf{AB}|| - 2t \leq d < ||\mathbf{AB}|| \\ \mathbf{B} & d \geq ||\mathbf{AB}|| \end{cases},$$

$$\mathbf{AB} = \mathbf{B} - \mathbf{A},$$

$$\widehat{\mathbf{AB}} = \frac{\mathbf{AB}}{||\mathbf{AB}||},$$

$$d = \mathbf{P} \cdot \mathbf{AB} - \mathbf{A} \cdot \mathbf{AB}$$

Equation 1. Moving target function

Navigation towards the moving target is handled by a PID controller which takes the angle difference between the heading and target bearing as input and outputs the robot angular velocity about the robot z-axis (yaw).

2.4.4 Parcel map

Information about the robot's entry and exit of the test parcels is communicated from the robot to the cell spray implement during route plan navigation. This is handled by a mapping component that loads a list of polygon coordinates describing the 1177 test parcel locations. At each pose update the new pose is checked against the polygons and changes are published.

To lower the computational load without delaying the published parcel changes, the update algorithm seeks to minimize the required number of required calculations. For each polygon a rectangular bounding box, a timeout value and a nearby flag is assigned at startup. At each robot pose update the polygon map update (algorithm 1) iterates through a subset of polygons determining if the robot distance to the polygon bounding box is within a certain threshold distance. For polygons outside the threshold a timeout value based on distance and the robot maximum speed determines the next time this polygon should be checked. All polygons in the list inside the threshold are checked at each update using a point in polygon algorithm. In this trial the pose update rate is 20 Hz, the subset is 100 polygons and the threshold distance is 5 meter.

```
begin
    for polygons in current subset do
        if polygonNearbyFlag or polygonTimeout then
            distance = min(pose distance to corners of bounding box)
            if distance < nearbyThreshold then
                polygonNearbyFlag = true
            else
                polygonNearbyFlag = false
                polygonTimeout = now + distance/maxSpeed - mapUpdateTime
            end
        end
    end
    for all polygons with polygonNearbyFlag = true do
        Check if pose is inside polygon
        if pose inside polygon status just changed then
            Publish change
        end
    end
end
```

Algorithm 1: Polygon map update

3. RESULTS

In preparing for the first spray task a series of test runs were conducted to configure and parameterize the robot and cell spray implement. The following figures are based on data from driving along the first crop row in one of these test runs. The spray tank was attached and half filled with water during the test run. The data has been filtered to exclude headland turns and subsequent rows. No filtering and no down sampling has been applied to the data shown in the figures. The row length was approximately 145 m, the robot was driving at a speed of approximately 0.5 m/s and data was sampled at a rate of 5 Hz. Samples from subsequent rows and other test runs showed similar results.

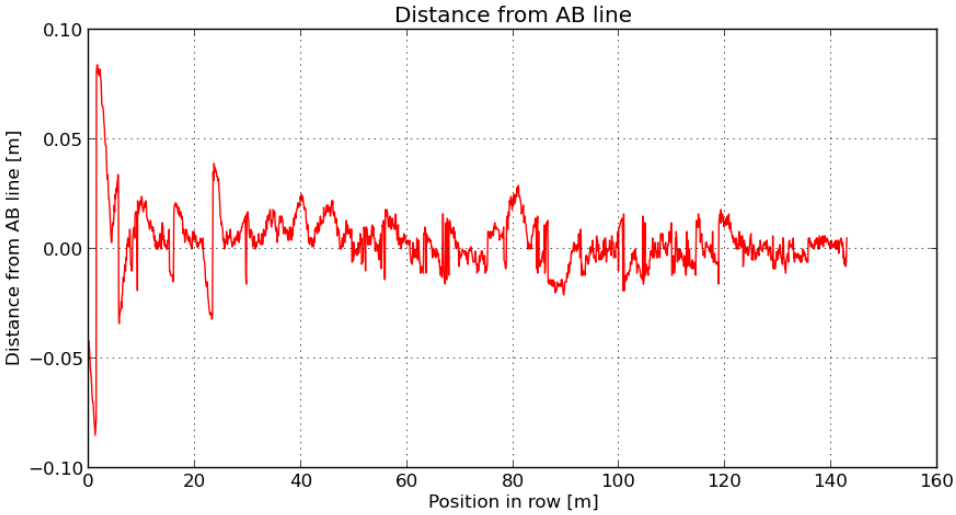


Figure 11. Results showing distance from AB line

Figure 11 shows the robot lateral position accuracy with respect to the navigated AB line. The 95% percentile of the absolute distance to the AB line is 0.0250 m. The estimated robot pose is used as reference because it is assumed to be the most accurate. The 95% percentile on the difference between raw GNSS data and the estimated pose is 0,0104 meter for the trial. The data has not been compensated for robot tilting about the roll and pitch axis. Figure 12 shows an example of the robot track along on of the other AB lines.

Figure 12. Visible track after AB line navigation

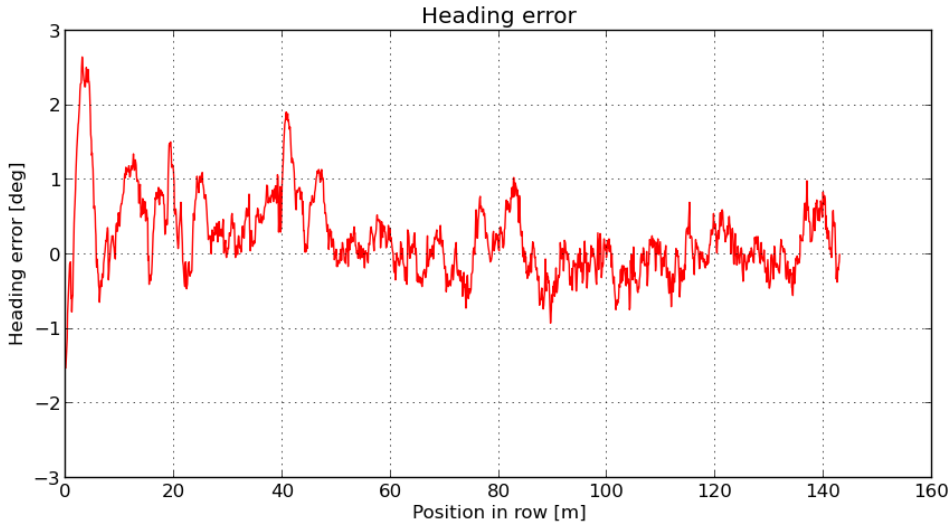


Figure 13. Results showing heading error

Figure 13 shows the robot heading error with respect to the destination (B) waypoint. The 95% percentile of the absolute heading error is 1.17 degrees. The estimated pose is used as reference as there exist no raw sensor data describing the absolute orientation about the yaw axis.

4. DISCUSSION

The results presented here are from a partially completed experiment, and it has therefore not been possible to analyze navigation data recorded during the actual trials. However the analyzed data substantiates that the Armadillo robot with the cell sprayer implement attached is capable of navigating the test field with a lateral positioning accuracy better than the anticipated 50 mm.

The FroboMind software components developed in this work performed as expected after having undergone a thorough testing and debugging process in the field. Some problems with the underlying track speed controller component were discovered however and they have not yet been solved. The speed controller is not capable of correctly controlling the desired speed, instead it vary depending on the load and hence the surface conditions, implement weight etc. It is therefore not worth the effort to fine tune the navigation controller, the test results presented here were obtained after only a few test runs with manual parameter calibration. The reason that the results are satisfactory anyway is that a tracked vehicle with the tracks spaced far apart is quite easy to control, and the navigation controller is therefore capable of compensating the track speed errors.

The first treatment has been completed successfully, even though it took place shortly after a heavy rainfall that turned the top layer of the clay soil into mud (figure 8). The resulting higher load on the motors caused the RoboteQ motor controllers to trip frequently which has made the navigation data time consuming to analyze. The motor controller problem has subsequently been

analyzed and it is believed that the internal current measurement is too sensitive and responds to short term peak values rather than average measurements. Adjusting the maximum current from 75A to 125A and setting stall detection to its highest setting has somewhat mitigated the problem, but the average current consumption for each track module is much lower than 75A. The entire treatment was completed without requiring a battery recharge before completion, and the theoretical battery pack capacity is 3120 Wh.

The treatment was performed at a speed of 0.3 m/s. The limiting factor is the computation speed of the imaging analysis algorithms.

Testing the new track module design has revealed a few issues in the transmission where the shafts caused a lateral displacement of the tooth wheels. This problem has already been corrected in a more recent prototype of the track modules.

5. CONCLUSION

A novel cell sprayer designed for large scale tests of weed detection algorithms and spraying strategies is presented in this paper. The cell sprayer consists of a spray implement with camera module and spraying module mounted in a shock absorbing frame carried by the Armadillo robot.

In this work the spray implement was developed, a new track module was applied to the Armadillo robot, and new software components for Transverse Mercator coordinate conversion, pose estimation, waypoint navigation and polygon mapping were developed for the FroboMind software platform.

Data analyzed from test runs performed in conjunction with the trials substantiates that the Armadillo robot with the cell sprayer implement attached is capable of navigating the test field smoothly with a lateral positioning accuracy better than the anticipated 50 mm. The test results presented in this paper shows at over a distance of approximately 145 m the 95% percentile of the absolute distance to the AB line is 0.0250 m and the 95% percentile of the absolute heading error is 1.17 degrees which is within the requirements of the spray implement.

The results from the weed detection algorithms and the resulting weeding performance will be published in a later publication and the cell sprayer is expected to be utilized in further precision spraying experiments. Furthermore the results from this work also contributes to the ongoing development of computer vision technologies for sustainable weeding as well as the Armadillo field robot and the open source FroboMind software platform.

6. ACKNOWLEDGEMENTS

This research is linked to and partially funded from the MODICOVI Proof of Concept project, a collaboration between the Danish Ministry of Science, Innovation and Higher Education, the University of Southern Denmark and Aarhus University, as well as the Danish National

Advanced Technology Foundation project: The Intelligent Sprayer Boom. The authors wish to thank Thomas Giselsson, Jørgen Maagaard Pedersen, Martin Skriver and Uffe Pilegaard Larsen for their contributions.

7. REFERENCES

Gloeckler, F., Joy, R., Simpson, J., Specht, D. (1996). *Handbook for Transformation of Datums, Projections, Grids and Common Coordinate Systems.* U.S. Army Topographic Engineering Center.

Jensen K., Nielsen, S., Bøgild, A., Jørgensen, O.J., Jacobsen, N.J.; Jaeger-Hansen, C., Jørgensen, R.N. (2012) *A low cost modular robotics tool carrier for precision agriculture research.* 11th International Conference on Precision Agriculture, July 2012, Indianapolis, USA., 2012

Jensen, K., Nielsen, S. H., Larsen, M, Bøgild, A., Green, O., Jørgensen, R. N. (2012). *FroboMind, proposing a conceptual architecture for field robots*. CIGR-Ageng. July 8-12 2012. Valencia, Spain.

Kurstjens, D. A. (2007). *Precise tillage systems for enhanced non-chemical weed management.* Soil and Tillage Research 97.2, pp. 293–305. issn: 01671987.

Larsen, T.; Hansen, K.; Andersen, N., Ravn, O. (1999) *Design of Kalman filters for mobile robots; evaluation of the kinematic and odometric approach.* Control Applications Proceedings of the 1999 IEEE International Conference on, 1999, 2, 1021-1026 vol. 2

Midtiby, H. S. (2007). "Real time computer vision technique for robust plant seedling tracking in field environment". PhD thesis. University of Southern Denmark.

Pedersen, S. M., Fountas, S., Have, H., Blackmore, B. S. (2006) *Agricultural robots - system analysis and economic feasibility Precision Agriculture*. Springer, 2006, 7, 295-308

Sørensen C.G. Green O., Nørremark, M., Jørgensen, R.N., Jensen, K., Maagaard, J., Jensen, L.A. (2007). *Hortibot: Feasibility study of a plant nursing robot performing weeding operations – part IV*. 2007 ASABE Annual International Meeting. Paper Number: 077019

Thorling, l., Hansen, B., Langtofte, C., Brüsch, W., Møller, R.B., Mielby, S., Højberg, A.L. (2011) *Ground water monitoring (Grundvandsovervågning) 2011*. Publication from GEUS DK. http://www.geus.dk/publications/grundvandsovervaagning/1989_2010.htm

Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y. (2009) *ROS: an open-source Robot Operating System.* ICRA Workshop on Open Source Software 2009.

Rodríguez, M. & Gómez, J. (2009). *Analysis of Three Different Kalman Filter Implementations for Agricultural Vehicle Positioning.* The Open Agriculture Journal, 2009, 3, 13-19

Slaughter, D. C., Giles, D. K., Downey, D. *Autonomous robotic weed control systems: A review Computers and Electronics In Agriculture*, Elsevier Sci Ltd, 2008, 61, 63-78

Snyder, J. P. (1987) *Map Projections: A Working Manual* Geological Survey (U.S.).

Laursen, M.S., Midtiby, H., Jørgensen, R.N., Krüger, N. (2012). *Validation of Modicovi - Monocot and Dicot Coverage Ration Vision based method for real time estimation of Canopy Coverage Ration between Cereal Crops and Dicotyledon Weeds.* International Conference on Precision Agriculture, Indianapolis, US. July 2012.

Laursen, M.S. (inventor), Midtiby, H. (inventor), Jørgensen, R.N. (inventor), Krüger, N. (inventor). (2012). *Spray boom for selectively spraying a herbicidal composition onto dicots.* Patent number: WO 2012/122988 A1. mar 16, 2011.

Thrun, S., Burgard W., Fox, D. (2006) *Probabilistic Robotics.* MIT Press

Jensen, K., Laursen, M.S., Midtiby, H., Jørgensen, R.N. “Autonomous precision spraying trials using a novel cell spray implement mounted on an Armadillo tool carrier”. International Commission of Agricultural and Biological Engineers, Section V. CIOSTA XXXV Conference “From Effective to Intelligent Agriculture and Forestry”, Billund, Denmark, 3-5 July 2013.