Fault-Tolerant Vision for Vehicle Guidance in Agriculture

Blas, Morten Rufus; Blanke, Mogens; Madsen, Tommy Ertbølle

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

Technical University of Denmark

Morten Rufus Blas

Fault-Tolerant Vision for Vehicle Guidance in Agriculture

PhD thesis, April 2010

DTU Electrical Engineering
Department of Electrical Engineering

Fault-Tolerant Vision for Vehicle Guidance in Agriculture

Morten Rufus Blas

Kongens Lyngby 2010
DTU Elektro-PHD

www.elektro.dtu.dk
Department of Electrical Engineering
Automation and Control
Technical University of Denmark
Ørsteds Plads
Building 348
DK-2800 Kgs. Lyngby
Denmark
Tel: (+45) 45253800
Fax: (+45) 45931634
Email: info@elektro.dtu.dk

ISBN 978-87-92465-22-1

Summary

The emergence of widely available vision technologies is enabling for a wide range of automation tasks in industry and other areas. Agricultural vehicle guidance systems have benefitted from advances in 3D vision based on stereo camera technology. By automatically guiding vehicles along crops and other field structures the operator's stress levels can be reduced. High precision steering in sensitive crops can also be maintained for longer periods of time as the driver is less tired.

Safety and availability must be inherent in such systems in order to get widespread market acceptance. To tolerate dropout of 3D vision, faults in classification, or other defects, redundant information should be utilized. Such information can be used to diagnose faulty behavior and to temporarily continue operation with a reduced set of sensors when faults or artifacts occur.

Additional sensors include GPS receivers and inertial sensors. To fully utilize the possibilities in 3D vision, the system must also be able to learn and adapt to changing environments. By learning features of the environment new diagnostic relations can be generated by creating redundant feed-forward information about crop location. Also, by mapping the field that is seen by the stereo camera, it is possible to support the guidance system by storing salient information about the environment. By tracking the motion of the vehicle, vision output can be fused over time to create more reliable and robust estimates of crop location.

This thesis approaches these challenges by considering systematic design methods using graph-based analysis. It is demonstrated how diagnostic relations can be derived and remedial actions can be done to maintain safety and healthy

functioning of vision systems. The combination of redundant information from 3D vision, mapping, and aiding sensors such as GPS provide means to detect and isolate single faults in the system.

In addition, learning is employed to adapt the system to variational changes in the natural environment. 3D vision is enhanced by learning texture and color information. Intensity gradients on small neighborhoods of pixels are shown to provide a superior approach to modeling texture information than other methods. Stochastic automatas using optimally quantized data is demonstrated as a strong approach for offline learning.

It is considered how 3D vision provides labeling of training data that subsequently can be fed into a learning system. Statistical change detection theory is shown to be a suitable approach to detecting artifacts in the learning process so safe operation can be maintained. The system can be used to perform real-time classification using a fast online approach that is superior to state-of-the-art.

Advances in tracking vehicle motion using 3D vision is demonstrated to allow unprecedented high accuracy maps to be created of the local environment. Features in the environment are extracted and tracked using novel feature detectors relying on approximating the Laplacian operator with a bi-level octagonal kernel. It is shown how these features display high levels of accuracy and stability while being considerable faster than similar feature detectors. Artifacts in 3D vision range measurements are demonstrated to be detectable by using the generated 3D maps and a probabilistic approach to fusing and comparing range measurements.

Resumé

Udviklingen af bredt tilgængelige teknologier indenfor billedanalyse muliggør en lang række af automatiseringsopgaver indenfor industrien og andre områder. Styresystemer til landbrugsmaskiner drager nytte af fremskridt i 3D billedanalyse baseret på stereokameRateknologi. Ved automatisk styring af køretøjer langs afgrøder og andre markstrukturer kan førerens stressniveau reduceres. Højpræcisionsstyring i skrøbelige afgrøder kan udføres over længere tidsperioder eftersom føreren bliver mindre træt.

Sikkerhed og oppetid skal være en integreret del af sådanne systemer for at kunne opnå bred accept på markedet. For at kunne tolerere udfald i 3D syn, klassifikationsfejl eller andre fejl skal redundant information være tilgængelig. Sådan information kan når fejl opstår bruges til diagnostisering af problemet samt midlertidig fortsættelse af kørslen med et reduceret antal sensorer.

Øvrige sensorer inkluderer GPS modtagere og inertisensorer. For fuld udnyttelse af mulighederne indenfor 3D billedanalyse skal systemet være i stand til at tilpasse sig skiftende omgivelser og lære af disse. Ved at lære omgivelsers karakteristika at kende kan redundant information skabes til brug ved diagnosticering. Desuden kan kortlægning af marken med stereokamera muliggøre support til styringssystemet ved indsamling af vigtige informationer. Ved at følge køretøjets bevægelser over tid kan de indsamlede informationer sammenholdes og derved give et mere pålideligt estimat af afgrødernes placering.

I denne afhandling bliver disse udfordringer taget op med anvendelse af grafbaseret analyse til systematisk design. Det bliver demonstreret hvordan diagnostiske relationer kan udledes og afhjælpende handlinger kan udføres for at opretholde sikkerhed og funktion af visionssystemer. Kombinationen af redun-

dant information fra 3D billedanalyse, kortlægning og ekstra sensorer som for eksempel GPS gør det muligt at detektere og isolere individuelle fejl i systemet.

Læring benyttes for at tilpasse systemet til variationer i de naturlige omgivelser. 3D vision forstærkes ved læring af information om tekstur og farve. Det bliver demonstreret at intensitetsgradienter i små områder af pixels er bedre til tekstur-modellering end andre kendte metoder. Stokastisk tilstandsmaskine med kvantiseret data demonstreres som en effektiv tilgang til offline læring.

Det undersøges hvordan 3D vision kan bruges til klassificering af træningsdata som efterfølgende kan indsættes i et læringssystem. Statistisk ændringsdetektionsteori demonstreres som en passende tilgang til detektering af fejl i læringssytemet, således at sikker kørsel kan opretholdes. Systemet kan blive brugt til real-time klassifikation ved brug af en hurtig online tilgang som overgår state-of-the-art.

3D vision-baseret sporing af et køretøjs bevægelser demonstreres at kunne tillade dannelse af højpræcisionskort af lokale omgivelser. Karakteristika i omgivelserne ekstraheres og følges med moderne detektorer, der afhænger af approksimering af Laplaceoperatoren. Det vises hvordan dette kan udvise stor akuratesse og stabilitet og samtidig være betydeligt hurtigere end lignende detektorer. Uventede genstande i 3D vision range målinger vises at være detekterbare ved brug af de skabte 3D kort og en statistisk tilgang til fusion og sammenligning af målinger.

Preface

This thesis was prepared at Department of Electrical Engineering, Automation and Control Group, the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Ph.D. degree in engineering. The project was funded by CLAAS Agrosystems, The Danish Agency for Science, Technology and Innovation, and The Technical University of Denmark.

The thesis deals with different aspects of using stereo vision for controlling agricultural vehicles. The main focus is on mapping, learning and fault-tolerance.

The project was supervised by Professor Mogens Blanke, DTU Elektro, and Tommy Ertbølle Madsen from CLAAS Agrosystems. Part of the research was also conducted at The Stanford Research Institute (SRI) with Dr. Robert C. Bolles and Dr. Kurt Konolige acting as supervisors.

The thesis consists of a summary report and a collection of seven research papers written during the period 2006–2010.

Kgs. Lyngby, April 2010

Morten Rufus Blas

Dissemination of Results

The following papers have been published during the PhD period and are included as part of this thesis.

[A] Morten Rufus Blas, Motilal Agrawal, Aravind Sundaresan and Kurt Konolige. Fast Color/Texture Segmentation For Outdoor Robots. *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 4078-4085, Nice, France, 2008. Published.

[B] Motilal Agrawal, Kurt Konolige and Morten Rufus Blas. CenSurE for Realtime Feature Detection and Matching. *Proc. of the European Conf. on Computer Vision*, pages 102-115, Marseille, France, 2008. Published.

[C] Kurt Konolige, Motilal Agrawal, Morten Rufus Blas, Robert C. Bolles, Brian Gerkey, Joan Sola, Aravind Sundaresan. Mapping, Navigation, and Learning for Off-Road Traversal. *J. of Field Robotics*, pages 88-113, 2009. Published.

[D] Morten Rufus Blas, Mogens Blanke, Radu Bogdan Rusu and Michael Beetz. Fault-Tolerant 3D Mapping with Application to an Orchard Robot. *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 893-898, Barcelona, Spain, 2009. Published.

[E] Morten Rufus Blas and Mogens Blanke. Natural Environment Modeling & Fault-Diagnosis for Automated Agricultural Vehicle. *Proc. 17th IFAC World Congress*, pages 1590-1595, Seoul, Korea, 2008. Published.

[F] Morten Rufus Blas and Mogens Blanke. Automatic Baling Using Stereo Vision and Texture Learning. *J. of Computers and Electronics in Agriculture*, 2010. Submitted.

[G] Fabio Caponetti, Morten Rufus Blas and Mogens Blanke. Stochastic Automata with Optimal Signal Quantisation for Classification of Outdoor Environments. *Control Engineering Practice*, 2009. Submitted.

The following papers have been published during the PhD period but have not been included in the final thesis. They relate to other projects done during the period.

- Tommy Ertbølle Madsen, Kristian Kirk and Morten Rufus Blas. 3D Camera for Forager Automation. *67th Int. Conf. on Agricultural Engineering*, Hannover, Germany, 2009. Published.
- Mads Fogtmann Hansen, Morten Rufus Blas and Rasmus Larsen. Mahalanobis Distance Based Iterative Closest Point. *Proc. of SPIE : Medical Imaging : Image Processing*, page 11, San Diego, USA, 2007. Published.

Acknowledgements

This thesis has been made possible by the amalgamation of the efforts and ideas of a number of bright people who I can only thank for letting me work with them: Mogens Blanke, Kurt Konolige, Motilal Agrawal, Robert C. Bolles, Brian Gerkey, Joan Sola, Aravind Sundaresan, Radu Bogdan Rusu, Fabio Caponetti, and Michael Beetz.

I would like to express my gratitude to SRI for letting me visit for half a year as an international fellow. Working on the LAGR project was very exciting. Likewise, I would like to thank Willow Garage for letting me visit for a short three week period.

My company supervisor, Tommy Madsen deserves considerable thanks along with Jesper Vilander for supporting and believing in the idea of doing this industrial PhD project. I would also like to thank Mogens Blanke again for providing valuable supervision at the university.

Contents

Chapter 1

Introduction

1.1 Background

During the 20th century, food production has seen a number of advances. These advances have allowed food production to keep pace with worldwide population growth. Increased productivity has been achieved through industrialization. Advances in genetics, chemicals, infrastructure, and mechanization have been major contributing factors.

Even with these advances, agricultural work to this day is still thought of as physically hard work and it is difficult to attract skilled labor. The work is often seasonal with long irregular working hours. The working environments are often harsh and dangerous due to weather, and proximity to chemicals and heavy machinery. On top of this, farms are often situated in isolated areas.

At the start of the 21st century information technology is now revolutionizing vehicles by making them more intelligent through automation. The positive effects of automation have been in: reducing drudgery, reducing skill level, and giving freedom for the masses.

Automation still has much potential, and might just be the next major step in making agriculture more productive. Vehicle guidance systems are currently

having a major impact. Historically, guidance systems have been largely mechanic with tactile sensors being actively used for automatic guidance along rows of corn/maize for more than 30 years. During the last 10 years GPS has seen a major breakthrough with more and more vehicles being equipped with this type of system for automatic plowing, seeding, spraying, and harvesting.

This thesis looks at stereo vision sensors as it is enabling for a wide variety of automation tasks. Stereo vision is gaining acceptance for guidance tasks such as automatically steering agricultural vehicles and implements along field structures. Unlike GPS it can actually see the field so can account for local conditions. Compared to tactile sensors it can see large parts of the field while avoiding physical contact with sensitive crops. In the future such systems will be expected to be able to handle more and more situations. This will require being able to extract and reason about additional information from the sensors. Stereo vision has a large potential as considerable information about the environment can be extracted from the images. Fault-tolerance and robustness will be critical to maintain safe operation.

1.2 Problem Formulation

From a fault-tolerant perspective there is little redundancy in typical vision guidance systems. Single faults in software, algorithms or in hardware will cause the system to behave in an ill-defined manner which may jeopardize safety and potentially cause injury or damage to both humans, crop, and machinery. This thesis has investigated what can be done to alleviate some of these problems.

Systematic design methods exist and have been demonstrated for obtaining fault-tolerance in complex sensor systems. Graph-based analysis has in the past proven to be an efficient tool to obtain diagnostic relations to analyze remedial actions for technical systems. They have however never considered the vision dimension. A paradigm in this research is hence that graph-based modeling will be feasible even for vision-based systems and that they can be used to generate diagnostic relation.

Vision guidance systems generally work by tracking the 3D profile of field structures. In order to handle dropout of 3D tracking, faults in classification, or other artifacts of the vision system, it is stipulated that redundant sensor information should be used to diagnose faulty behavior. It should thus be possible to temporarily allow guidance to continue using a reduced set of sensors when faults or artifacts occur.

It is of interest to consider how 3D tracking can be enhanced by extracting supplemental information from the stereo camera. A paradigm is that learning and mapping can be used to provide additional feed-forward information about field structures to create redundancy in the 3D tracking.

Sometimes field structures may become flat or their 3D structure may be ill-defined. In such cases 3D tracking may fail. It is thus of interest to investigate whether learning algorithms can be used to learn 2D features of the environment to support 3D tracking.

Mapping is a step that provides an internal representation of the essentials of an environment. With appropriate mapping information low confidence structures may still be recognizable. This project has thus investigated the possibility of representing semi-structured agricultural environments using state-of-the-art mapping techniques. As mapping requires estimating the position of the vehicle it has also been investigated how this can be extracted from the camera. A method known as visual odometry (VO) has been explored for this task.

With data fusion combining immediate observations from different sources such as mapping and learning there is an imminent risk that single faults in these components may also cause system failure. An objective of the research presented here has hence been to also make the data fusion fault tolerant.

1.3 Stereo Vision Guidance

A typical design of a guidance system using stereo vision is exemplified in Figure 1.1. A stereo camera with onboard processing identifies field structures using a 3D tracking algorithm which is subsequently fed into a controller. The type of field structure to recognize can be configured via the user interface. The field structures may include but are not limited to: rows of plants, ridges, tramlines, and swath.

The controller then uses the information provided by the stereo camera to steer the wheels of the vehicle. This is then used to align the vehicle with the field structure and allows guidance.

The user interacts with the system through a user interface as well as a safety subsystem that checks whether the user is sitting in his seat and/or is using the steering wheel. The user can engage/disengage automatic guidance. In case of signal loss from the stereo camera an alarm is used to notify the user and the wheels lock in their current position.

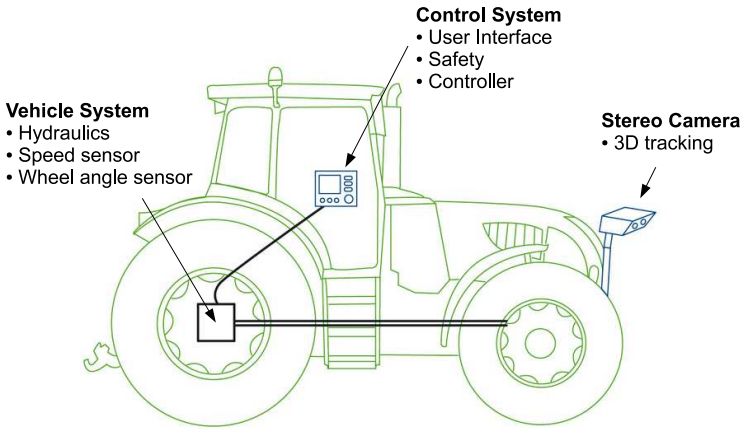


Figure 1.1: Overview of vision guidance system.

1.3.1 Stereo Vision

Stereo vision perceives depth using triangulation. The distance to a point is determined by the triangle between the point and where it appears in each of two images. To do this the two images must be aligned. Given a calibrated stereo camera the images can be aligned by warping them. This is known as rectification. This gives two cameras with parallel optical axes and horizontal epipolar lines. A dense estimation of ranges is then performed at each pixel by matching along the epipolar lines [65]. This is done using a correlation window with typical sizes of around 11x11 pixels. The correlation window matches texture in the two images with each other. The output of the matching process is a disparity image (Figure 1.3(b)). This gives the image difference between the position of objects in the two cameras. The horizontal distance from the image center to the object image is dl for the left image and dr for the right image (Figure 1.2). Then the disparity value d is given by:

$$d = dl - dr \tag{1.1}$$

The transformation between a 3D point $\mathbf{M} = (X, Y, Z)$ observed by the stereo camera and its corresponding coordinate in disparity space $\omega = (\bar{x}, \bar{y}, d)$ in the left image is then given by:

$$\begin{aligned} \bar{x} &= u - u_0 = f\frac{X}{Z} \\ \bar{y} &= v - v_0 = f\frac{Y}{Z} \\ d &= dl - dr = \frac{f \cdot b}{Z} \end{aligned} \tag{1.2}$$

Where (u_0, v_0) is the optical center of the image in pixels. f is the focal length, and b is the baseline between the two cameras.

Some filtering is then typically done after the matching on the disparity image to remove regions of low confidence. These are regions that either could not be matched between the two images or the match was not unique enough.

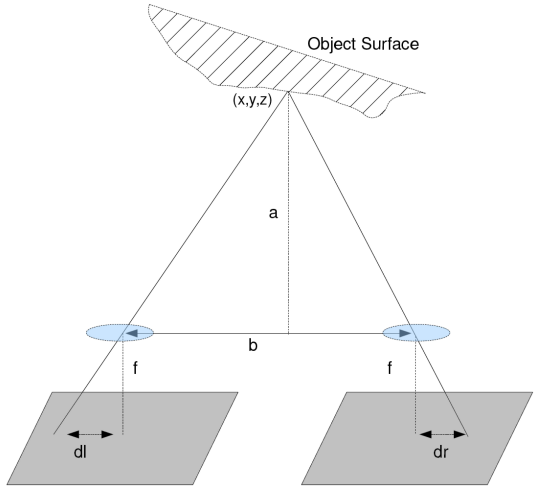


Figure 1.2: A simplified view of stereo geometry. Disparity is the offset of the image location of an object: $d = dl - dr$. a is the range to an object. b is the baseline between images. f is the focal length of the cameras.

1.3.2 3D tracking

Tracking of field structures using 3D is typically done by detecting the height of features in the environment relative to the ground plane [62], [112]. This can be done either using information about how the camera is mounted or by estimating the ground plane from the 3D data (paper C).

3D tracking is most easily performed by labeling pixels in either image based on their calculated height (Figure 1.3(c)). Shape based template matching can then be used to recognize the pose of the field structure in the 2D image using known 3D constraints such as the width and height of the structure (Figure 1.3(d)).

This pose can then be calculated relative to the vehicle ground plane in 3D. A parametrization of this pose into an angular and lateral deviation of the field structure relative to the vehicle can be used as input to the controller. In case of a poor match result then this is also typically signaled to the controller.

1.3.3 Experimental Platforms

The work in this thesis has been conducted on a number of different platforms. Specifically, an outdoor robot and an autonomous tractor have been used to test algorithms on before these were applied to a tractor with a guidance system.

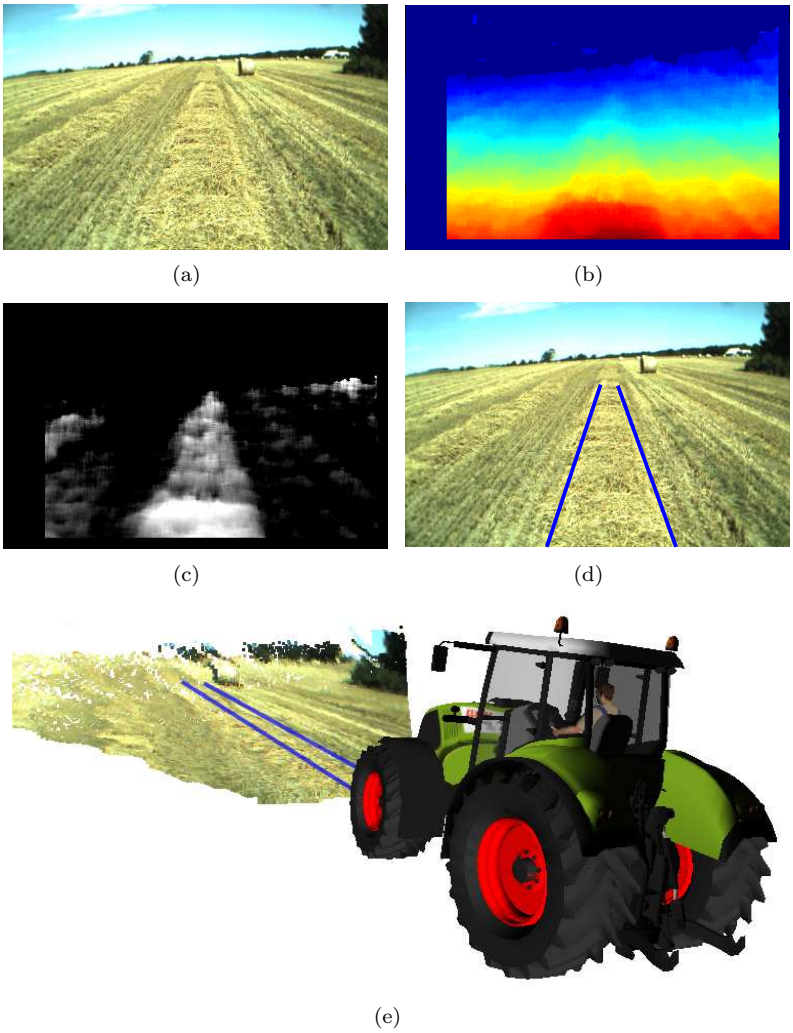
The outdoor robot used was part of the Defense Advanced Research Project Agency (DARPA) Learning Applied to Ground Robots (LAGR) program. The project focused on applying learning to robots navigating in an outdoor environment with the ambitious goal of achieving vision-only autonomous traversal of off-road terrain. This gave the possibility to test state-of-the-art algorithms for mapping and learning on a working fully autonomous platform (Figure 1.4(a)). The work was carried out at the Stanford Research Institute (SRI), USA.

Research was also conducted on an autonomous tractor provided by University of Copenhagen, Faculty of Life Sciences (KU Life) (Figure 1.4(b)). A project entitled "Safe and Reliable" was carried out on the platform in parallel with the thesis work which focused on safe navigation in an orchard environment. This gave a unique opportunity to test and further develop algorithms previously developed for LAGR, in an agricultural environment.

CLAAS Agrosystems provided a tractor with research primarily focused on the agricultural application of baling (Figure 1.4(c)). This vehicle was used during the summer months to test the applicability of the developed algorithms on a commercial guidance system.

1.4 Abbreviations

Abbreviation	Explanation
AI	Artificial Intelligence
CenSurE	*Cen*ter *Sur*round *E*xtrema
CIE	Commission Internationale d'Eclairage
DOF	Degrees Of Freedom
DARPA	Defense Advanced Research Project Agency
ECEF	Earth Centered Earth Fixed coordinate system
EM	Expectation-Maximization
FAST	Features from Accelerated Segment Test
GPS	Global Positioning System
GMM	Gaussian Mixture Model
GHMM	Gaussian Mixture emitting Hidden Markov Models
IMU	Inertial Measurement Unit
JM	Jeffreys-Matusita
LAB	L for Lightness and A,B for the color opponent channels
LAGR	Learning Applied to Ground Robots
LBP	Local Binary Patterns
LM	Levenberg-Marquardt, or Leung-Malik
LRT	Likelihood Ratio Test
MU-SURF	Modified Upright Speeded Up Robust Features
NCC	Normalized Cross Correlation
NED	North East Down coordinate system
RANSAC	RAndom SAmpling Consensus
RGB	Red, Green, and Blue (image color channels)
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SURF	Speeded Up Robust Features
SAD	Sum of Absolute Differences
SBA	Sparse Bundle Adjustment
SVM	Support Vector Machines
U-SURF	Upright Speeded Up Robust Features
VO	Visual Odometry
WGN	White Gaussian Noise



(a) (b)

(c) (d)

(e)

Figure 1.3: (a) Left image from stereo camera. (b) Disparity image with warmer colors indicating shorter range. (c) Pixels in the left image labeled with height above the ground plane. (d) 3D tracking showing borders of detected field structure. (e) 3D points estimated by stereo.

(a)

(b)

(c)

Figure 1.4: (a) The LAGR robot used to test outdoor vision algorithms at SRI. (b) The autonomous tractor provided by KU Life. Its purpose is to conduct research in autonomous spraying and mowing of orchards. (c) The semi-autonomous CLAAS tractor. The stereo camera is configured to follow cut crop in the form of swath. A baler attached to the back is then used to process the swath into bales.

Chapter 2

Contributions

This thesis has been organized as a collection of papers. These have been reformatted and form individual pieces of work at the back of the thesis. The four following chapters summarize the findings in the included papers. This chapter aims at pinpointing the novel elements presented in this thesis. For a review of related work the reader is advised to consult the chapters relating to a particular subject.

Novelty exists at different levels in this thesis. At the application level much of what is presented here is novel since very little research has previously been done in using online learning and mapping in agriculture. In robotics, learning and mapping have been researched for decades. Only little efforts have however been done in applying them to stereo vision (specifically in outdoor environments). With this in mind, it has been investigated how to enhance 3D vision guidance by adding:

- Learning for recognizing color and texture.
- Visual odometry for position estimation.
- Mapping for tracking field structures over time.
- Fault-tolerance for handling faults in vision guidance systems.

A brief summary of the contributions will now be discussed for each of these topics.

2.1 Learning

A compact color and texture descriptor has been developed to describe local color and texture variations in an image. It was initially used to perform online segmentation of natural images. The method has been compared to other state-of-the-art descriptors and has shown to be both faster and more robust at discriminating between textures than other methods. It uses a texton approach. A novel method for learning the texture of field structures aided by 3D vision is demonstrated.

A stochastic automata is also introduced for classification of outdoor environments. A new signal quantisation approach is used to learn optimal thresholds for classifying environments. The automata is compared to other state-of-the-art approaches in terms of training and classification times, as well scalability. It is found to be competitive with the other approaches while allowing easy inclusion of vehicle motion and spatial connectiveness of environments.

Learning is presented in Chapter 3.

Dissemination has been done in papers A, C, F and G.

2.2 Visual Odometry

Visual Odometry (VO) is a method for vision based positioning based on tracking features in an image. This provides additional position information and is enabling for mapping. The developed algorithm has shown to be more precise than existing algorithms while still being computable at frame rates. This precision has come by introducing:

- a novel scale-invariant feature called CenSurE.
- a novel method for matching features called MU-SURF.

Using CenSurE and MU-SURF it is demonstrated that features can be tracked for longer with fewer failed image matches, and better motion estimates. This

was done by comparing different types of state-of-the-art features for outdoor VO. Such an analysis has previously not been done.

The use of 6 degrees of freedom (DOF) VO in agriculture has not previously been demonstrated.

Lastly, an optimization for quick motion hypothesis rejection in the VO algorithm is introduced by enforcing a geometric consistency check.

Visual Odometry is presented in Chapter 4.

Dissemination has been done in papers B,C and F.

2.3 Mapping

It is demonstrated how maps can be created in agriculture using VO and a dynamic tree-based decomposition of space known as octrees. This goes far beyond the state-of-the-art in agriculture in terms of precision, scalability, and size.

Modeling of field structures using splines and clothoids are demonstrated for vision guidance.

Diagnosing artifacts in stereo algorithms is also shown using a novel method relying on comparing redundant range measurements taken from different camera positions.

Mapping is presented in Chapter 5.

Dissemination has been done in papers D, E and F.

2.4 Fault-tolerance

It is demonstrated how systematic methods in fault-tolerant design can readily be applied to a vision system to obtain fault-tolerance. It is shown how a behavioral model can be setup to describe a fault-free system including stereo vision, GPS, and inertial sensors. Such an analysis has not previously been done and can be considered novel by including the stereo vision sensor.

Using statistical change detection algorithms for diagnosing faults in a vision guidance system has previously not been demonstrated in literature.

It is also shown how learnt texture information can be used to provide redundancy when 3D recognition fails. While the 3D recognition is working new texture models are learnt and evaluated. This provides a novel method for self-supervised learning which has not previously been demonstrated in agriculture.

Fault-tolerance is presented in Chapter 6.

Dissemination has been done in papers D, E and F.

Chapter 3

Learning

Learning can be seen as an automatic method for building statistical models of data which would otherwise have been difficult or impossible to do by hand.

Learning is important for automation in agriculture. Experienced drivers have plethora of knowledge on which to draw upon. Such knowledge cannot be manually coded and will require learning from large amounts of training data. Field situations vary from day to day. For example crop appearance change daily and are also different for different parts of a field. These situations cannot be known beforehand so will require in-field learning.

This thesis has looked at learning for recognition and has concentrated on methods to learn texture models of field structures. An underlying assumption has been that this information could not be learnt offline and instead tracking algorithms must be adapted online to the in-field situations at hand. By learning such texture models it is possible to reinforce 3D vision and thereby provide redundancy in the system.

3.1 Related Work

For outdoor texture recognition it has been demonstrated that texture could be used to identify different types of terrain [5]. In this paper a stereo camera was used to extract patches of terrain which were subsequently compared to a learnt texton texture database containing different types of terrain (sand, soil, grass, gravel, asphalt, and wood-chip). In [42] a similar approach was used to extend navigable terrain beyond stereo range by learning nearby textures. In both cases offline learning was used to create filter banks that could be used to discriminate between different textures. These methods are slow whereby they are currently unsuitable for real-time navigation. In [130] texture was modeled using Gabor filters which were used to discriminate between different kinds of weeds in an agricultural environment.

[5] and [42] used color in conjunction with texture and 3D information. In [105] spatial and temporal context is also integrated which should allow better class discrimination (in this case between ground, buildings, and different objects).

Pure color based recognition has had considerable use in agriculture for segmenting plants such as in [100] and [131].

Most of the methods in literature focus on offline learning. Two notable exceptions are [42] and [132] who both use 3D to train a color/texture classifier online and then use it to classify outside of 3D range. One of the reasons for this is that it is difficult to come up with an offline derived model that can handle all expected variation.

3.2 Learning Algorithms

Creating accurate mathematical models of nature has proven difficult for researchers. After decades of research, Artificial Intelligence (AI) research has failed to come up with such models. What AI in turn has given us are statistical approaches for modeling variation in natural systems. These statistical approaches are commonly referred to as learning. Learning algorithms excel at data-mining large amounts of data for establishing patterns which can subsequently be used for classification.

For image-based recognition the first step in learning usually involves manually specifying basic image statistics that may be useful for classification. This drastically reduces the dimensionality of the problem but leaves the question open

about what basic statistics to use. Some basic statistics could be mean, variance, and/or first and second derivatives computed from pixel data over small neighborhoods.

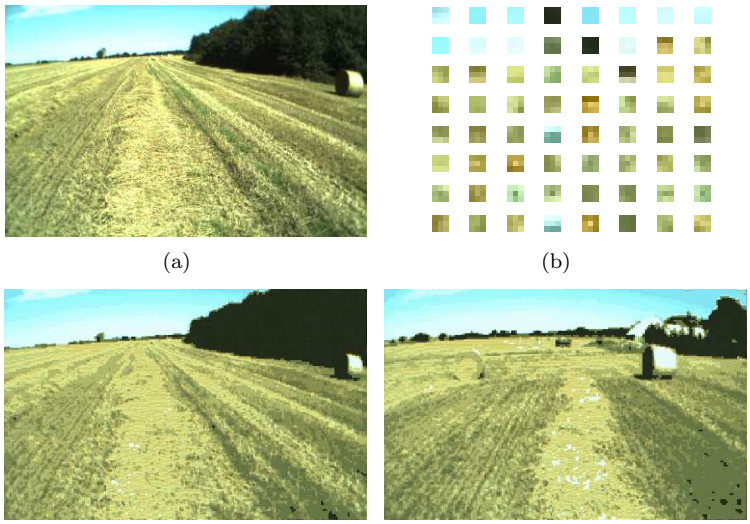
There are many forms of learning but the work here will split algorithms into two general categories: unsupervised and supervised.

A typical use of unsupervised learning is for clustering. K-means is an unsupervised clustering method that partitions observations into K-clusters in which each observation belongs to the cluster with the nearest mean [50]. Another unsupervised clustering method is the use of Gaussian Mixture Model's (GMM) which use an Expectation-Maximization (EM) algorithm to estimate the distribution of data as being composed of a number of Gaussians. The approaches are unsupervised because labels of which observations belong to what clusters is not needed.

State-of-the-art for supervised learning include Support Vector Machines (SVM), Adaboost, and Gaussian mixture emitting Hidden Markov Models (GHMM). SVM uses hyperplanes to separate classes and for subsequent classification [19],[25]. Adaboost is a boosting technique which linearly combines simple weak classifiers on the basis of classification performances on a training set [35]. GHMM's consist of a discrete time and space Markov process that contains hidden parameters and emits observable outputs [106].

The main problem with SVM's and GHMM's is that they are quite slow when working with high dimensional data. Adaboost is fast but does not provide information about the confidence of the classification. This makes the method unsuitable for supervision and diagnosis tasks where low confidence estimates should not trigger false alarms.

In paper G a new supervised learning method is introduced for training a classifier to discriminate between different types of agricultural environments. The method relies on using a probabilistic model on quantized data. The core of the probabilistic model is built around a stochastic automata. A formal analysis of using quantized data has not previously been done. The method is compared to SVM's, GHMM's, and Adaboost. It is shown how data from both vision and laser can be classified over time to discriminate between different orchard environments. The stochastic automata shows similar performances to the other state-of-the-art methods. Compared to general learning algorithms this method has some clear advantages. It allows for straightforward inclusion of how dynamical systems evolve over time (vehicle motion, and connections between environments) which is important for robotics.



(a) (b)

(c) (d)

Figure 3.1: (a) A training image. (b) The 64 textons extracted from the training image. (c) Training image labeled with the textons. (d) Another image labeled using the same textons.

3.3 Learning Texture

Modeling natural textures is a difficult problem because they are very varied. The concept behind the approach presented here is that the texture patterns in a given environment can be learnt online. This should theoretically allow faster and better discrimination between textures in an environment since classification can be optimized for these specific patterns.

3.3.1 Textons

The approach adopted relies on intensity gradients computed over small neighborhoods (E.g. 3×3 pixels). Such gradients have been shown to be quite robust to lighting changes and are also used by other descriptors such as SIFT [78]. In paper A it is proposed to combine intensity gradients with a texton approach that also allows inclusion of color information. It is demonstrated to be superior to other state-of-the-art methods: Local Binary Patterns (LBP) [81], RGB Textons [5], and the Leung-Malik (LM) Filter Bank [74].

The image is first transformed to the CIE*LAB colorspace using an efficient lookup table to do the RGB to LAB conversion. Colors in LAB are more perceptually linear than in the RGB space, thereby resulting in better clusters. This gives the brightness information L and the color channels a, b. The texture information is taken as the surrounding pixel intensities minus the center intensity. Each pixel location p_i in the image can then be represented using the descriptor:

$$p_i = \begin{bmatrix} W_1 * L_c \\ W_2 * a_c \\ W_2 * b_c \\ W_3 * (L_1 - L_c) \\ \vdots \\ W_3 * (L_8 - L_c) \end{bmatrix} \tag{3.1}$$

Here (L_c, a_c, b_c) is the color of the center pixel, and $L_1, L_2, ..., L_8$ are the intensities of the surrounding pixels in a 3×3 neighborhood. The set of weights $\{W_1 = 0.5, W_2 = 1, W_3 = 0.5\}$ is used to balance how much to rely on color, texture, and brightness for the clustering. These were set as to weigh chrominance higher than luminance. Also, since texture takes up many of the descriptor rows it must be down-weighted so the color still has an impact on the clustering. The assumption here is that in a local neighborhood the color does not vary much, so including the color channels for all 3×3 pixels does not provide much additional information.

Textons are a set of mean textures (represented by a vector of the intensity gradients and color) extracted by a K-means algorithm on a large set of these descriptors (acquired from one or more training images). The K-means algorithm then finds the set of textons μ_j that partitions the descriptors into k sets $\mathbf{S} = S_0, S_1, ..., S_k$ by trying to minimize:

$$\mathbf{S}^* = \text{argmin}_{\mathbf{S}} \sum_{j=1}^{k} \sum_{p_i \in S_j} \|p_i - \mu_j\|^2 \tag{3.2}$$

where p_i are the descriptors in the training image(s) and S_j is the set of descriptors belonging to cluster j out of k clusters.

An illustration of extracted textons is given in Figure 3.1. Each pixel neighborhood in an image can then be labeled as belonging to a texton by comparing its

descriptor to the set of textons and finding the nearest one (Euclidean distance). For the experiments shown in Figure 3.1 a total of $k = 64$ textons were used.

Let $p(x, y)$ be the texton descriptor at row x and column y in an image. Then the texton labeled image $L(x, y)$ assigns a label to a pixel depending on which set its texton descriptor belongs to:

$$L(x, y) = j, \quad p(x, y) \in S_j \tag{3.3}$$

The idea is that the textons that best describe an image area create a unique signature that can subsequently be used for recognition. The labeling of image pixels reduces the input dimensionality of the image (E.g. 24bit, 3 color channels ranging from 0-255) to a lower dimensional space consisting of a single channel with values $1 - k$. This first step causes some generalization of the texture patterns. K-means was chosen because it was very fast.

3.3.2 Texton Histograms

Histograms of the occurrence of textons in different regions of the image is used as the basis for comparing similarity between textures. By constructing histograms geometrical constraints can be relaxed so that textons do not have to follow some specific pattern but just need to occur equally as much for two textures to be the same. Histograms can be made quite robust to noise by using enough samples.

3.3.2.1 Histogram Computation

In this work histograms of texton occurrence have been computed over $(2n+1)\times(2n+1)$ pixel neighborhoods ($n = 16$) which corresponds roughly to $1\% - 2\%$ percent of the image size. Since this is usually done in a dense manner across an image it was important that this could be done efficiently. The method makes use of integral images to efficiently compute histograms in constant time, independent of window size [140, 76].

Integral images $I_j(x, y)$ are constructed for each texton ($j = 1, ..., k$).

$$I_j(x, y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} \begin{cases} 1, & L(x', y') = j \\ 0, & L(x', y') \neq j \end{cases} \tag{3.4}$$

The integral image is computed recursively, requiring only one scan over the image. Once the integral image is computed, it takes only two additions and two subtractions to calculate the occurrence of a texton over any upright, rectangular area, independent of its size. Let $o_j(x, y)$ be the texton occurrence at pixel x, y for texton j then:

$$o_j(x,y) = I_j(x-n-1, y-n-1) - I_j(x-n-1, y+n) \\ -I_j(x+n, y-n-1) + I_j(x+n, y+n) \tag{3.5}$$

Negative indexes are treated by padding the image with zeros. The histogram of texton occurences $\mathbf{h}$ at pixel x, y is then:

$$\mathbf{h}(j) = o_j(x,y), \quad j = 1, ..., k \tag{3.6}$$

3.3.2.2 Histogram Comparison

To compare the similarity between two textures involves comparing how similar their histograms are. This has been explored in papers A and F. A normalized histogram represents a probability distribution so similarity measures typically used to compare probability distributions can be used: Kullback Leibler divergence [69], chi-square divergence [38], earth-movers distance [114], and Battacharrya distance [54]. These methods are however non-trivial to use for high dimensional distributions. Simple distance measures (such as SAD and Euclidean distance) seemed to offer the best ratio of complexity versus performance over the other methods. In the work presented here the SAD distance was used.

Given two histograms $\mathbf{h}_1$ and $\mathbf{h}_2$ the distance between them is (defined by the $\oplus$ operator):

$$\mathbf{h}_1 \oplus \mathbf{h}_2 = \sum_{j=1}^{k} |\mathbf{h}_1(j) - \mathbf{h}_2(j)| \tag{3.7}$$

3.4 Learning Field Structures

Learning a texton model for a specific environment is performed unsupervised. To recognize field structures some information must be supplied so a classifier can know what to look for. A self-supervised method was proposed in paper A for recognizing outdoor structures. It relies on clustering to find areas of the image with a similar texture. Recognition of field structures is then done by identifying a set of areas that have a geometric shape similar to the structure. This method relies on tight geometric constraints for reducing false positives. Some field structures vary considerably in shape, which reduces the usefulness of this method.

Instead in paper F it is proposed to relax some of these geometric constraints and instead use 3D vision to label data that can be used for training a texture classifier. This follows the trail of thought given in [42] and [132] for augmenting 3D information with image-based perception.

The algorithm runs through the following steps:

1. Learn textons from training image.
2. Construct texton histograms for each pixel location.
3. Label histograms using 3D vision.
4. Learn a texture model based on the labeled histograms.

Given a texton histogram $\mathbf{h}$ extracted at a given pixel. The classification task then becomes in determining which of two hypotheses the histogram belongs to: $\mathcal{H}_s$ for "field structure" and $\mathcal{H}_n$ for "not field structure".

The likelihood ratio is then a method to decide between the two hypotheses [56]:

$$L(\mathbf{h}) = \frac{p(\mathbf{h}; \mathcal{H}_s)}{p(\mathbf{h}; \mathcal{H}_n)} \tag{3.8}$$

To estimate the likelihood of either hypothesis the probability density function (PDF) must be modeled. The histogram distributions for textons follow complicated distributions. In Figure 3.3 it is clear that different parts of the ground may have very different texton histograms even within the same image.

In [132] it was demonstrated that GMM's could be used to model the distribution of colors of a dirt road. Each Gaussian could be made to recognize different sets of colors and allow modeling of multi-modal color distributions by mixing the Gaussians. RGB color data is 3 dimensional which is much smaller than a texton histogram with 64 dimensions. Modeling texture even with just one Gaussian was found too slow on current hardware and a number of Gaussians seem needed to represent the distributions well.

A compromise was achieved by modeling the texture histograms as a number of K-means clusters. A major penalty in doing this is that the density of the distributions is not modeled. Likelihood is then just approximated by the distance from a histogram to either of the cluster centers. Although this is a crude approximation it does allow for very fast learning and classification.

The output of the 3D vision tracking algorithm allows the histograms at each pixel in an image to be labeled as belonging to either $\mathcal{H}_s$ or $\mathcal{H}_n$. Then these histograms are clustered separately for each hypothesis using K-means with $m = 3$ clusters.

Then $\mathcal{H}_s$ is $H_s = \{\mathbf{h}_{s,1},...,\mathbf{h}_{s,m}\}$ and $H_n = \{\mathbf{h}_{n,1},...,\mathbf{h}_{n,m}\}$ for $\mathcal{H}_n$. A graphical illustration of this training process is given in Figure 3.2.

The classifier is then formulated as a distance ratio to the nearest cluster under each hypothesis:

$$d = \frac{\min(\mathbf{h} \oplus \mathbf{h}_{n,1},..., \mathbf{h} \oplus \mathbf{h}_{n,m})}{\min(\mathbf{h} \oplus \mathbf{h}_{s,1},..., \mathbf{h} \oplus \mathbf{h}_{s,m})} \tag{3.9}$$

The distance is computed for each pixel in the image. A geometrical analysis is then used to detect if a field structure is present in the image. An example of classifying an image is given in Figure 3.3.

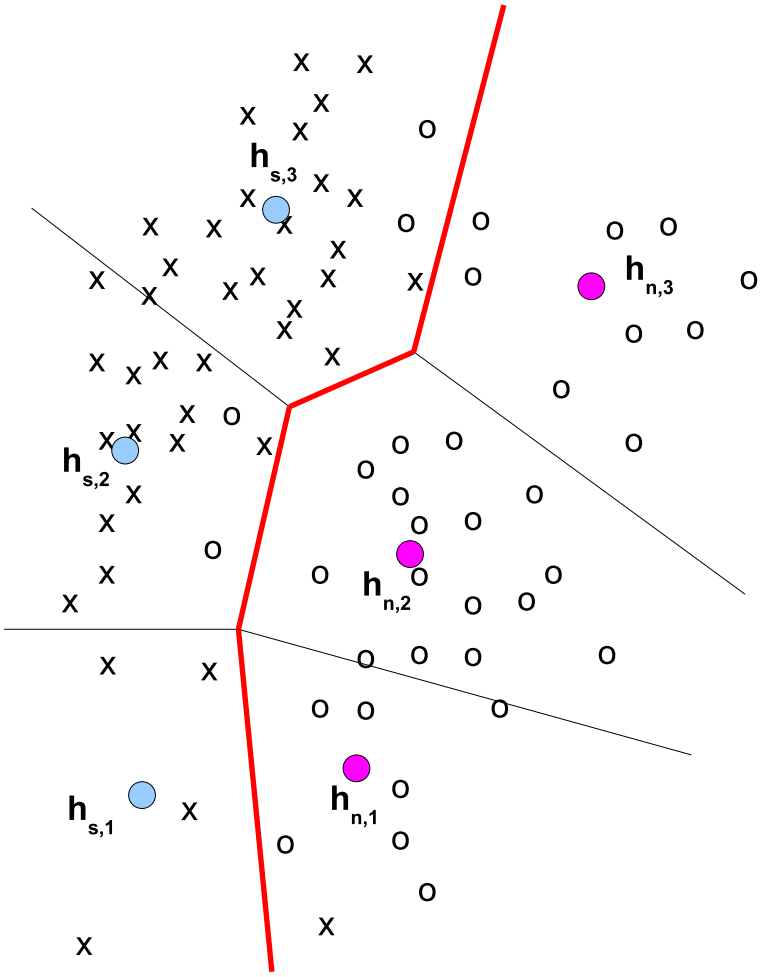
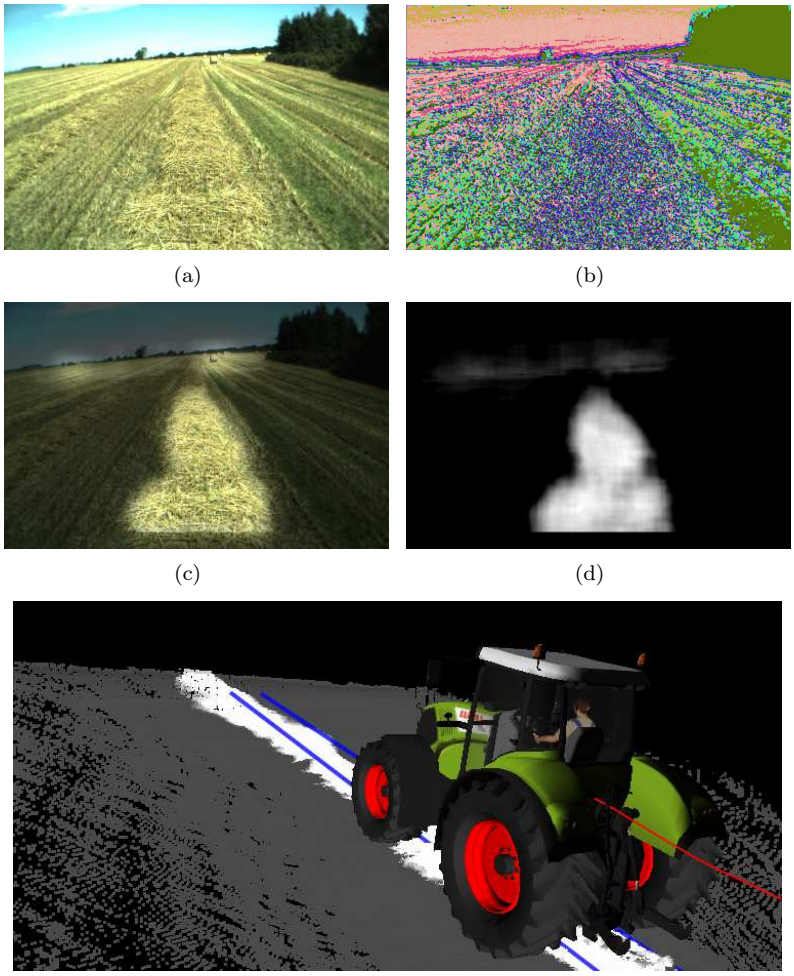


Figure 3.2: Let $\mathbf{x}$ be training samples for H_s and $\mathbf{o}$ for H_n (as given by 3D vision). Then this can be represented by a set of histogram clusters with $H_s = \{\mathbf{h}_{s,1},...,\mathbf{h}_{s,m}\}$ and $H_n = \{\mathbf{h}_{n,1},...,\mathbf{h}_{n,m}\}$ respectively. This is here illustrated for the two dimensional case with $k = 2$ textons and $m = 3$ histogram centers. Using a Voronoi tessellation a border (illustrated in red) shows points that are equidistant to the two hypotheses. In higher dimensions this border forms a series of hyperplanes.



(a) (b) (c) (d) (e)

Figure 3.3: (a) An image from the stereo camera. (b) Texton classification. Each pixel color represents a different texton. (c) The stereo camera image with a transparency mask based on the swath classification. (d) Swath classification based on texture with intensity representing the strength of classification. (e) The texture classification integrated into the 3D mapping.

Chapter 4

Visual Odometry

Obtaining reliable pose estimates for agricultural vehicles can be difficult. However such information is essential for mapping.

Current systems for positioning in agriculture typically use GPS with more expensive systems integrating inertial sensors for estimating the pose of a vehicle. Such systems are however highly dependent upon having a good GPS signal. GPS signals may dropout due to satellite occlusion or exhibit artifacts due to atmospheric conditions or multi-path errors where signals are bounced off nearby objects.

A cheap redundant source of position information in agricultural vehicles is wheel odometry where encoders measure the wheel revolutions to estimate the vehicle pose. Such systems are generally robust but suffer if the vehicle slips or tire pressures change.

A method for estimating the vehicle pose using stereo vision is commonly referred to as Visual Odometry (VO). This chapter summarizes the research conducted in conjunction with developing high accuracy VO for outdoor use.

4.1 Related Work

VO systems use structure-from-motion methods to estimate the relative position of two or more camera frames, based on matching features between those frames. There have been a number of recent approaches to VO [97, 82, 53], including motion estimation on the Mars vehicles [86]. The system presented here is most similar to the recent work of Mouragnon et al. [93] and Sunderhauf et al. [128], which exploit bundle adjustment techniques to obtain increased precision.

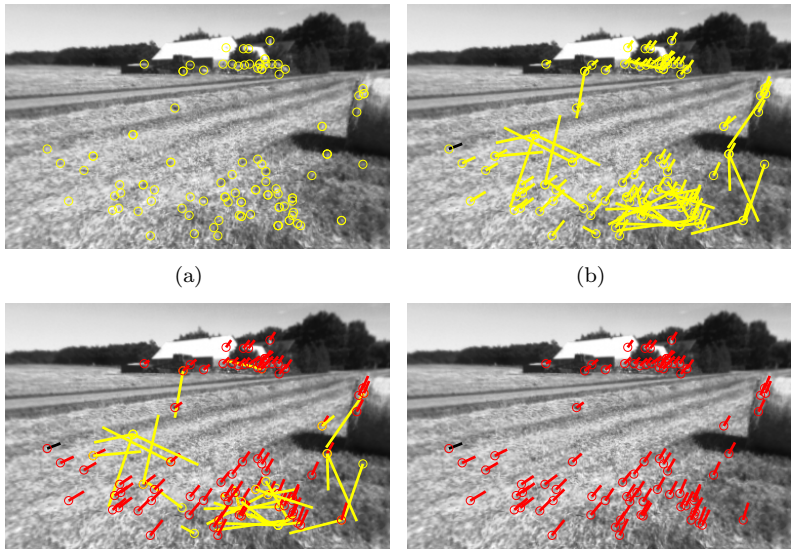
VO accumulates error over distance whereby GPS is needed to correct the error as seen in [1] and for an agricultural application in [4].

4.2 Overview

The used VO system works by matching sparse features between frames (Figure 4.1). The algorithm follows a series of steps as previously proposed in literature [67, 1, 2]:

1. Distinctive features are extracted from each new frame in the left image. Standard stereo methods are used to find the corresponding point in the right image.
2. Left-image features are matched to the features extracted in the previous frame.
3. From these uncertain matches, a consensus motion estimate is recovered using a RANdom SAmpling Consensus (RANSAC) method [32]. Several hundred relative motion hypotheses are generated by randomly selecting three matched non-collinear features, and then scored using pixel re-projection errors.
4. If the motion estimate is small and the percentage of inliers is large then the image is discarded to avoid integrating errors caused by composing such small motions.
5. The motion estimate is refined further using non-linear minimization on the re-projection errors.

The research conducted in the various steps of the algorithm will now be discussed.



(a) (b)

(c) (d)

Figure 4.1: (a) Sparse features are detected in an image. (b) Features are then matched to a previous image by comparing image patches in local neighborhoods. (c) RANSAC is used to identify false matches using a consensus motion estimate. (d) The motion estimate is then refined using the correct matches.

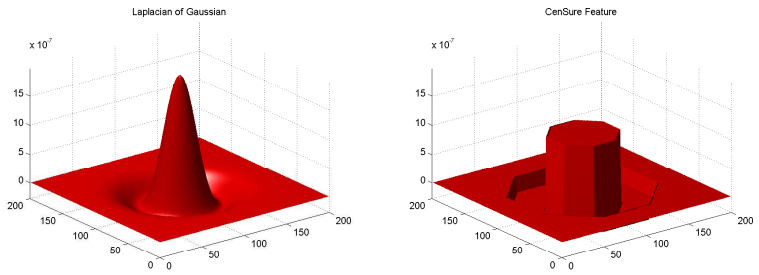
4.3 Feature Detection

When a new pair of images are acquired by the stereo camera the first step is to detect features in these images that can be used for tracking the camera movement.

In paper B experiments were done in comparing different state-of-the-art feature detectors for outdoor VO. Two important criteria have been established for comparing feature performance:

- Stability: the persistence of features across viewpoint change
- Accuracy: the consistent localization of a feature across viewpoint change

The performance of commonly used features (FAST [110], SIFT [78], SURF [46], and Harris [45, 120]) have been compared to a new feature type called CenSurE



(a) (b)

Figure 4.2: The Laplacian of Gaussian can be efficiently approximated using a bi-level octagonal kernel. (a) Laplacian of Gaussian. (b) CenSurE feature.

(*Cen*ter *Sur*round *E*xtrema). CenSurE uses scale-space extremas of a center-surround response as features. This is done by approximating the Laplacian of Gaussian operator with a bi-level octagonal kernel that can efficiently be computed at all scales (Figure 4.2).

Scale invariant features in general work by using down-sampled images to compute features at larger scales. An important finding of the research is that this directly affects the accuracy (localization) of features at larger scales. The CenSurE feature provides a solution for this by finding features at all scales in the original image.

The CenSurE feature is also shown to be much faster to compute than other scale-space approaches. For visual odometry, CenSurE features result in longer track lengths, fewer frames where images fail to match, and better motion estimates compared to the other methods.

4.3.1 CenSurE Computation

The octagon shape used to approximate the bi-level Laplacian of Gaussian operator is actually one in a series of approximations of using a circle as shown in Figure 4.3. The box filter (Figure 4.3(d)) has previously been shown to be computed efficiently using integral images [140, 76]. This has been extended to cover more complicated shapes. By using an octagon the feature becomes more invariant towards rotations in the image plane compared to the box filter.

An integral image $I(x, y)$ is an intermediate representation for the image and contains the sum of gray scale pixel values of image N with height y and width

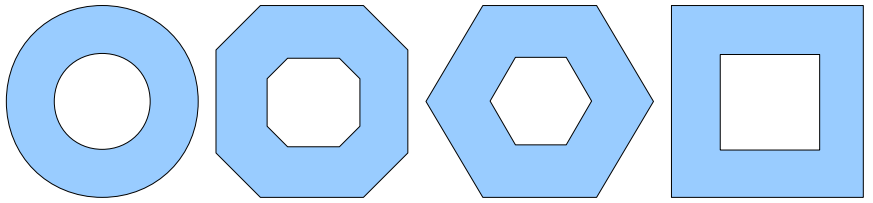


Figure 4.3: Progression of Center-Surround bi-level filters. Successive filters (octagon, hexagon, box) have less symmetry.

x:

$$I(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} N(x',y') \tag{4.1}$$

The integral image is computed recursively, requiring only one scan over the image. Once the integral image is computed, it it takes only two additions and two subtractions to calculate the sum of the intensities over any upright, rectangular area, independent of its size (Eq. 3.5).

Modified versions of integral images can be exploited to compute the other polygonal filters. The idea here is that any trapezoidal area can be computed in constant time using a combination of two different *slanted* integral images, where the sum at a pixel represents an angled area sum. The degree of slant is controlled by a parameter α:

$$I_\alpha(x,y) = \sum_{y'=0}^{y} \sum_{x'=0}^{x+\alpha(y-y')} N(x',y') \tag{4.2}$$

When $\alpha = 0$, this is just the standard rectangular integral image (Eq. 4.1). For $\alpha < 0$, the summed area slants to the left; for $\alpha > 0$, it slants to the right. Slanted integral images can be computed in the same time as rectangular ones, using incremental techniques.

Adding two areas together with the same slant determines one end of a trapezoid with parallel horizontal sides; the other end is done similarly, using a different slant. Each trapezoid requires three additions, just as in the rectangular case. Finally, the polygonal filters can be decomposed into 3 trapezoids for an octagon.

4.3.2 Non-maximal Suppression

The response of the CenSurE feature is calculated at seven different scales as detailed in paper B for every pixel in the left image. Different weights are likewise assigned to the sums of the inner and outer regions of the filter depending on the feature scale. A non-maximal suppression is then done by comparing responses in small 3×3 pixel neighborhoods and across scale. The response for a given scale is suppressed (response set equal to zero) if there exists a response within this neighborhood for one of the seven scales that has a higher value. This is because partial responses will be exhibited near maxima responses.

A strong response indicates a feature that is likely repeatable and thus more likely to be tracked. Therefore a threshold is used to remove weak responses. The strong responses that are left are then used as features.

4.4 Feature Matching

Feature matching involves matching features between two images. This is subsequently used for estimating the camera pose between the images. Matching occurs by comparing the texture of features in a local neighborhood and finding the best match. Similar to feature detectors the goal is to have the matching process be invariant towards different influences: affine transformations, lighting, and localization. To maximize the number of correct matches the matching scheme must generate a unique signature for each feature. Two simple approaches to matching is to use Sum of Absolute Differences (SAD) or Normalized Cross Correlation (NCC). These approaches are simple in that they work with the image pixels directly. More advanced approaches such as the DAISY descriptor [134], SURF [46], and SIFT [78] construct a vector based descriptor using image gradients computed from the image patches which are then subsequently used to match features.

SAD and NCC are relatively sensitive to in-plane rotations (roll), larger changes in perspective, and inaccuracies in feature localization. The problems related to roll and perspective changes become more significant as the region size increases. For scale-invariant features it is desirable to be able to match features across scale changes.

However, the other methods in literature are too slow for real-time implementation. In paper B a modification to U-SURF [46] called MU-SURF is proposed (Modified Upright SURF). MU-SURF is demonstrated to be faster and also bet-

ter than U-SURF for matching features by accounting for boundary conditions in the descriptor. This has allowed it to be used for real-time applications.

The SURF descriptor builds on from the SIFT descriptor by encoding local gradient information. It uses integral images to compute Haar wavelet responses, which are then summed in different ways in 4×4 subregions of the region to create a descriptor vector of length 64.

4.4.1 MU-SURF

Given a CenSurE feature calculated at scale s the feature descriptor used for matching is calculated by first extracting an image patch centered around the feature of $24s$ by $24s$ pixels. The Haar wavelet responses in the horizontal (d_x) and vertical (d_y) directions are then computed for each 24×24 point in the region with filter size $2s$ by first creating a summed image, where each pixel is the sum of a region of size s (Figure 4.4(a)).

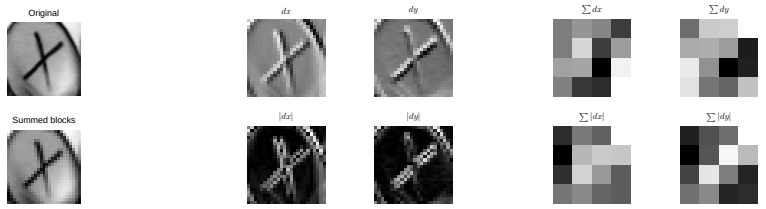
The Haar wavelet output results in four fixed-size d_x,d_y, $|d_x|$ and $|d_y|$ images that have the dimensions 24×24 pixels irrespective of the scale (Figure 4.4(b)).

Each d_x,d_y, $|d_x|$ and $|d_y|$ image is then split into 4×4 square overlapping subregions of size 9×9 pixels with an overlap of 2 pixels with each of the neighbors. For each subregion the values are then weighted with a Gaussian ($\sigma_1 = 2.5$) centered on the subregion center and summed into the descriptor vector for each subregion: $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. This is an extension compared to the standard SURF descriptor. This makes the matching performance more robust when pixels shift from one subregion to another due to affine transformations on the feature texture. This extension is known in the literature from the SIFT descriptor.

Each subregion vector is then weighted using another Gaussian ($\sigma_2 = 1.5$) defined on a mask of size 4×4 and centered on the feature point (Figure 4.4(c)).

Each subregion value is then stored in the form of a normalized vector.

Features can then be matched by finding the nearest vector using Euclidean distance. A threshold on the distance is used to remove features that are matched with insufficient confidence.



(a) (b) (c)

Figure 4.4: Modified U-SURF descriptor (MU-SURF). (a) Scaling is removed by creating a summed image where each pixel is the sum of a region of size s. (b) The Haar wavelet output results in four fixed-size d_x,d_y, $|d_x|$ and $|d_y|$ images.(c) Each image is then split into 4×4 square overlapping subregions. Two Gaussian weights are then applied, one with center around each subregion, and one with center in the image. The output is finally summed for each subregion to create an 8×8 dimensional descriptor.

4.5 Motion Estimation

Using a RANSAC method a consensus motion estimate is recovered using the matched features [32]. RANSAC allows such a motion estimate to be robustly recovered even if a large percentage of the features have been wrongly matched.

The RANSAC algorithm works by iterating over the following steps (Algorithm 4.1):

1. Select 3 matched non-collinear features at random.
2. Check geometrical consistency of matches.
3. Generate motion hypothesis for the features.
4. Score hypothesis using re-projection errors.

The hypothesis with the largest score forms a consensus estimate of the most valid camera motion by checking how many feature matches agree with the motion.

The geometrical consistency check is novel and has previously not been published.

4.5.1 Hypothesis Generation

Let M and M' be a 3D point before and after a rigid transformation. Then this transformation in 3D space can be defined by a rotation $\mathbf{R}$ and translation $\mathbf{t}$:

$$\begin{bmatrix} M' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M \\ 1 \end{bmatrix} \tag{4.3}$$

Three features can then be used to estimate $\mathbf{R}$ and $\mathbf{t}$ by calculating the least squares solution. This can be done efficiently using Singular Value Decomposition ($svd()$) as shown by [137]. With $n = 3$ matched non-collinear features the mean of the 3D points are given by:

$$\begin{aligned} \mu_M &= \frac{1}{n}\sum_{i=1}^{n} M_i \\ \mu_{M'} &= \frac{1}{n}\sum_{i=1}^{n} M'_i \end{aligned} \tag{4.4}$$

The covariance matrix of the 3D points is then:

$$\sum\nolimits_{M'M} = \frac{1}{n}\sum_{i=1}^{n}\left(M'_i - \mu_{M'}\right)\left(M_i - \mu_M\right)^T \tag{4.5}$$

Let $UDV^T = svd(\sum_{M'M})$ then:

$$\begin{aligned} \mathbf{R} &= USV^T \\ \mathbf{t} &= \mu_{M'} - \mathbf{R}\mu_M \end{aligned} \tag{4.6}$$

Where S is given by:

$$S = \begin{cases} I, & \text{rank}(\sum_{M'M}) = n-1 \wedge \det(U)\det(V) = 1 \\ \text{diag}(1,1,...,1,-1), & \text{rank}(\sum_{M'M}) = n-1 \wedge \det(U)\det(V) = -1 \\ I, & \text{rank}(\sum_{M'M}) > n-1 \wedge \det(\sum_{M'M}) \geq 0 \\ \text{diag}(1,1,...,1,-1), & \text{rank}(\sum_{M'M}) > n-1 \wedge \det(\sum_{M'M}) < 0 \end{cases} \tag{4.7}$$

4.5.2 Geometric Consistency Check

A large part of the time in RANSAC is spent generating motion hypotheses. To speed it up one must realize that for a change in pose to be valid there must not be a scale change (only rotation and translation). This corresponds to checking that the distance between two points in the hypothesis generation does not change. Given three 3D points: $\{M_0, M_1, M_2\}$ that have been matched correspondingly to three points in another image: $\{M'_0, M'_1, M'_2\}$ then the Euclidean distance between the features is checked for scale change. If the change in scale is more than 10% of the distance the motion hypothesis is rejected.

Formally, this can be expressed by a function G that evaluates a logical expression:

$$G(M_0, M'_0, M_1, M'_1, M_2, M'_2) = D_i < 1.1 \wedge D_i > 0.9, \quad i = \{0,1,2\} \tag{4.8}$$

Where D_i is given by:

$$D_0 = \frac{\|M_0 - M_1\|}{\|M'_0 - M'_1\|}, D_1 = \frac{\|M_0 - M_2\|}{\|M'_0 - M'_2\|}, D_2 = \frac{\|M_1 - M_2\|}{\|M'_1 - M'_2\|} \tag{4.9}$$

This check has previously not been discussed in the literature.

4.5.3 Hypothesis Evaluation

Voting is used in the RANSAC algorithm to decide which hypothesis is the best. For each of several hundred hypotheses the number of inlier features is counted

by re-projecting features from one image into the other. The best hypothesis is the one with the most inliers. This is done in disparity space as this space has isotropic noise independent of feature distance and location in the 2D image.

The 3D transformation in disparity space can be described by:

$$\begin{bmatrix} \omega \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ d \\ 1 \end{bmatrix} = \Gamma \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \Gamma \begin{bmatrix} \mathbf{M} \\ 1 \end{bmatrix} \tag{4.10}$$

where the transformation Γ is:

$$\Gamma = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & f \cdot b \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.11}$$

Let w and w' be the corresponding transformation in disparity space. Then by substitution [27]:

$$\begin{aligned} \Gamma^{-1} \begin{bmatrix} w' \\ 1 \end{bmatrix} &\simeq \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \Gamma^{-1} \begin{bmatrix} w \\ 1 \end{bmatrix} \\ \begin{bmatrix} w' \\ 1 \end{bmatrix} &\simeq \Gamma \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \Gamma^{-1} \begin{bmatrix} w \\ 1 \end{bmatrix} \\ \begin{bmatrix} w' \\ 1 \end{bmatrix} &\simeq \mathbf{H}(\mathbf{R}, \mathbf{t}) \begin{bmatrix} w \\ 1 \end{bmatrix} \end{aligned} \tag{4.12}$$

The symbol $\simeq$ denotes the equality up to a scale factor.

A feature is then counted as an inlier if:

$$\|w' - w''\| < t \tag{4.13}$$

where $t = 2$ is a threshold in pixels defining the maximum allowable error in the re-projection. w'' is the re-projected point of w under a motion hypothesis setup using $\mathbf{H}(\mathbf{R}, \mathbf{t})$.

Algorithm 4.1 RANSAC Motion Estimation

$k := 0$
$score_{\max} := 0$
while $(k < k_{max})$ **do**
  $k := k + 1;$
  $score := 0$
  pick at random 3 matched features: (M_i, M_i'), $i = 0, 1, 2$
  if $G(M_0, M_0', M_1, M_1', M_2, M_2')$ **then**
    Solve $\begin{bmatrix} M_0' & M_1' & M_2' \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M_0 & M_1 & M_2 \\ 1 & 1 & 1 \end{bmatrix}$
    for $j = 0$ to $N - 1$ **do**
      $\begin{bmatrix} w_j'' \\ 1 \end{bmatrix} \simeq \mathbf{H}(\mathbf{R}, \mathbf{t}) \begin{bmatrix} w_j \\ 1 \end{bmatrix}$
      $score := score + (\left\| w_j' - w_j'' \right\| < t)$
    end for
  end if
  if $score > score_{\max}$ **then**
    $score_{\max} = score$
    $\mathbf{R}_{best} = \mathbf{R}$
    $\mathbf{t}_{best} = \mathbf{t}$
  end if
end while

4.6 Motion Refinement

The RANSAC motion estimate can be refined further by minimizing the re-projection error. This is a non-linear least squares problem due to the use of Euler angles and can be solved using non-linear minimization algorithms.

A popular algorithm for motion refinement is the Levenberg-Marquardt (LM) algorithm [80]. LM provides a numerical solution to minimizing a function over a space of input parameters. It works by interpolating between a Gauss-Newton algorithm and the method of gradient descent.

Given a vector function $\mathbf{f}$ that is wanted minimized then the general non-linear least squares problem can be formulated as:

$$\mathbf{x}^* = \text{argmin}_x \left\{ F\left(\mathbf{x}\right) \right\} \tag{4.14}$$

where:

$$F(x) = \frac{1}{2} \sum_{i=1}^{m} \left(f_i\left(\mathbf{x}\right) \right)^2 = \frac{1}{2} \left\| \mathbf{f}\left(\mathbf{x}\right) \right\|^2 = \frac{1}{2} \mathbf{f}\left(\mathbf{x}\right)^\top \mathbf{f}\left(\mathbf{x}\right) \tag{4.15}$$

To minimize the re-projection error between two frames then $\mathbf{f}$ is the vector function that calculates the re-projection error for each feature:

$$\mathbf{f}\left(\mathbf{x}\right) = \begin{bmatrix} f_0\left(\mathbf{x}\right) \\ f_1\left(\mathbf{x}\right) \\ \vdots \\ f_{N-1}\left(\mathbf{x}\right) \end{bmatrix} \tag{4.16}$$

where:

$$f_i\left(\mathbf{x}\right) = w_i' - w_i'' \tag{4.17}$$

N is the number of inliers after RANSAC and w_i'' is w_i re-projected after undergoing a transformation defined by $\mathbf{x}$.

$\mathbf{x}$ is $\mathbf{R}$ and $\mathbf{t}$ parameterized by Euler angles {roll φ pitch θ, yaw ψ}, and three translational components $\{t_x,t_y,t_z\}$ respectively:

$$\mathbf{x} = [\varphi, \theta, \psi, t_x, t_y, t_z] \tag{4.18}$$

Then the LM algorithm refines $\mathbf{x}$. As initial estimates for $\mathbf{x}$ the output of the RANSAC algorithm is used.

4.6.1 Sparse Bundle Adjustment

Given that many features can be tracked over multiple images it makes sense to extend the motion refinement to handle this (Figure 4.5). A method in literature to do this for VO is to use a technique called Sparse Bundle Adjustment (SBA) [29, 135]. By tracking features over multiple images fewer errors in pose are integrated. SBA takes this a step further by also estimating the position of features. This allows errors in the 3D coordinates of features to be corrected and should also produce a more accurate motion estimate. Given n 3D features observed in m images then this gives $3n + 6m$ free parameters that must be estimated. As features generally are only observed over four or five images a sliding window SBA is used. The number of features tracked is generally a few hundred which means the number of free parameters to be estimated are also in the order of hundreds.

The parameters to be estimated are $\mathbf{x} = \mathbf{x}_0, .., \mathbf{x}_m, M_0, .., M_n$ for the camera positions $\mathbf{x}_j$ and 3D coordinates of the features M_i respectively. $\mathbf{f}(x)$ stays the same except that instead of re-projecting w_i the 3D coordinate of the feature M_i is transformed into disparity space and used to calculate the error for each image in which it occurs.

The equation $(\mathbf{A}+\mu\mathbf{I})\mathbf{h_{lm}} = -\mathbf{g}$ is then solved by exploiting the sparse structure of $\mathbf{J}(x)$ as done in [129]. This allows the problem to be solved in real-time. $\mathbf{J}(x)$ is sparse because the re-projection error for a given image only depends upon the features present in that image. Also the re-projection error for a feature does not depend on the 3D coordinates of other features.

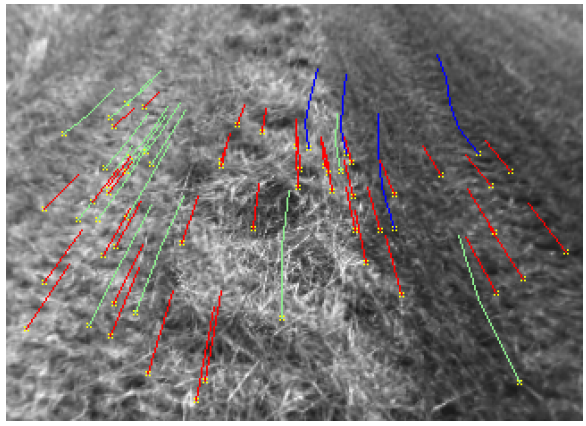


Figure 4.5: In Sparse Bundle Adjustment camera poses and 3D coordinates of features are estimated using non-linear minimization. This image shows some of the extracted features and their tracks. Red are features tracked over 1 frame. Green are features tracked over 2 frames. Blue are features tracked over 3 or more frames.

Chapter 5

Mapping

Mapping provides an internal representation of the essentials of an environment. A map can be created by storing processed sensor data in a common reference frame. Reliable positioning is critical in maintaining the transformation from sensor data to this reference frame.

There have generally been very few attempts at using mapping in agriculture in conjunction with vision. Some work has been done in the area of satellite imagery but such maps have low spatial resolution and are unsuitable for navigation. Mapping using vision has a huge potential as it enables very high spatial resolution maps to be created. Such maps can allow novel methods of steering agricultural vehicles.

A use of mapping is to compensate for the limited field of view of vision sensors. Using a memory of past observations it is possible to construct a more complete view of the local environment by fusing multiple observations. This can facilitate recognition and allow more intelligent control decisions to be made. Mapping provides a method to do sensor fusion between data acquired from different sensors as well as over time. This creates redundancy in the recognition systems by allowing observations stored in a map to be compared to new observations.

5.1 Related Work

In [61] it was demonstrated that a few meters of crop could be mapped by stitching together stereo images but the method was found too slow for real-time application. In [24] a wheel odometry system was used to stitch multiple laser measurements together which allowed the heading of a swath to be calculated from a local map.

In robotics, mapping has been very heavily researched for indoor environments with much of it focusing on Simultaneous Localization And Mapping (SLAM) [41, 73]. Much of the recent research on outdoor navigation has been driven by DARPA projects on mobile vehicles [12]. The sensor of choice is a laser rangefinder, augmented with monocular or stereo vision. In much of this work, high-accuracy GPS is used to register sensor scans; exceptions are [40, 91]. In contrast, the work here forgoes laser rangefinders, and explicitly use image-based registration to build accurate maps. Other approaches to mapping with vision are [108, 124], although they are not oriented towards realtime implementations.

Current technology permits city-scale reconstruction of 3D triangular meshes of environments [104] using vision and GPS. These systems still have limited handling of dynamic environments and are very dependent on good GPS signals. Outdoor SLAM methods such as [64] can potentially make GPS redundant by allowing VO systems to correct their own drift by re-localizing themselves in a map. Some issues make it difficult to use SLAM in agriculture. It relies on matching old features to new images when the vehicle revisits an area. Current research in wide-baseline matching [133] is limited which means it is hard to match features across large distances or angles. Also, agricultural environments change rapidly so old features are difficult to match.

5.2 Frames of Reference

A short overview of the reference frames used for mapping is now presented. Since work has been done in combining maps produced by both GPS and vision the next sections demonstrate how observations from these two can be brought into a common reference frame.

5.2.1 Global Frame

Typically GPS data uses the world geodetic system 1984 (WGS84) to determine the location of a point near the surface of the Earth and is expressed in terms of latitude (Φ) and longitude (λ) in degrees and a height h above the Earths surface.

5.2.2 Navigation Frame

The navigation frame used is the North, East, Down (NED) coordinate system (Figure 5.1). It is a local tangent plane to the Earths surface with local reference point (origin) specified at (Φ_0, λ_0) [30].

In order to convert global coordinates to the navigation frame the following conversion is used to first get the coordinates in Earth Centered Earth Fixed coordinates (ECEF):

$$X_{ECEF} = \left(\frac{a}{\chi} + h\right)\cos\Phi\cos\lambda \tag{5.1}$$

$$Y_{ECEF} = \left(\frac{a}{\chi} + h\right)\cos\Phi\sin\lambda \tag{5.2}$$

$$Z_{ECEF} = \left(\frac{a(1-e^2)}{\chi} + h\right)\sin\Phi \tag{5.3}$$

The height above the surface of the ellipsoid is given by h. The constants χ, a (semi-major axis) and e^2 (square of the first numerical eccentricity of the ellipsoid) are:

$$\chi = \sqrt{1 - e^2\sin^2\Phi} \tag{5.4}$$

$$a = 6378137.0m \tag{5.5}$$

$$e^2 = 6.69437999014x10 - 3 \tag{5.6}$$

$$\tag{5.7}$$

This models the Earth as an ellipsoid.

The ECEF coordinates are then converted to the NED coordinates using the

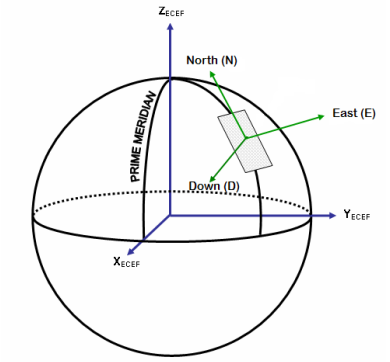


Figure 5.1: ECEF versus NED coordinate system.

following matrix:

$$\mathbf{p}_{gps}^{n} = \begin{bmatrix} -\sin\Phi_0\cos\lambda_0 & -\sin\Phi_0\sin\lambda_0 & \cos\Phi_0 \\ -\sin\lambda_0 & \cos\lambda_0 & 0 \\ -\cos\Phi_0\cos\lambda_0 & -\cos\Phi_0\sin\lambda_0 & -\sin\Phi_0 \end{bmatrix} \begin{bmatrix} X_{ECEF} \\ Y_{ECEF} \\ Z_{ECEF} \end{bmatrix} \tag{5.8}$$

5.2.3 Body Frame

The origin of the body frame is centered around the middle of the tractors rear axle on the ground plane with the x-axis along the length of the tractor and the y-axis along the width of the tractor. The z-axis increases downwards. This is illustrated in Figure 5.2(a).

Conversion of a 3D point between body frame and navigation frame can be made as follows:

$$\begin{bmatrix} X^n \\ Y^n \\ Z^n \end{bmatrix} = \mathbf{R}_n^b(\Theta) \begin{bmatrix} X^b \\ Y^b \\ Z^b \end{bmatrix} + \mathbf{p}^n \tag{5.9}$$

$\mathbf{p}^n$ is the coordinate vector of the middle of the rear axle in the navigation frame. $\mathbf{R}_n^b(\Theta)$ is the rotation matrix from body to navigation frame which is a function of the attitude of the vehicle in the navigation frame parameterized by the Euler angles $\Theta = \{\varphi, \theta, \psi\}$.

$\mathbf{p}^n$ can be calculated from the GPS receiver as:

$$\mathbf{p}^n = \mathbf{p}^n_{gps} - \mathbf{R}^b_n(\Theta)\mathbf{p}^b_{gps} \tag{5.10}$$

Where $\mathbf{p}^b_{gps}$ is the location of the GPS receiver in the body frame.

5.2.4 Camera Frame

The camera frame is defined with origin in the sensor, the x-axis right, y-axis down, and the z-axis along the direction the sensor is pointing. Furthermore the vision sensor is at an angle relative to the ground plane along the x-axis in the body frame. This is illustrated in Figure 5.2(b).

Conversion of a 3D point between camera frame and body frame can be made as follows:

$$\begin{bmatrix} X^b \\ Y^b \\ Z^b \end{bmatrix} = \mathbf{R}^c_b \begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} + \mathbf{p}^b_c \tag{5.11}$$

$\mathbf{R}^c_b$ is the rotation matrix from camera to body frame. $\mathbf{p}^b_c$ is the location of the vision sensor in the body frame. Similar transformations can be applied to other sensors such as IMU[1] (Inertial Measurement Unit) and wheel odometry.

5.3 Mapping Using Grid Maps

In the LAGR project 2D grid maps were used to store map information. 2D grid maps work by storing information in a quantized grid (usually from a top-down view). Grids are usually represented as images with each pixel representing a part of the environment. They are by far the most common way of representing maps in robotics literature. One of the reasons for this is that it is easy to add information to the map as it becomes a simple task of labeling pixels. For example all pixels with a value of 1 could be obstacles. Projecting 3D points observed from a stereo camera into a grid map reduces to a simple coordinate

[1] IMU's consist of 3 accelerometers and 3 gyroscopes which measure accelerations and angular velocities, respectively.

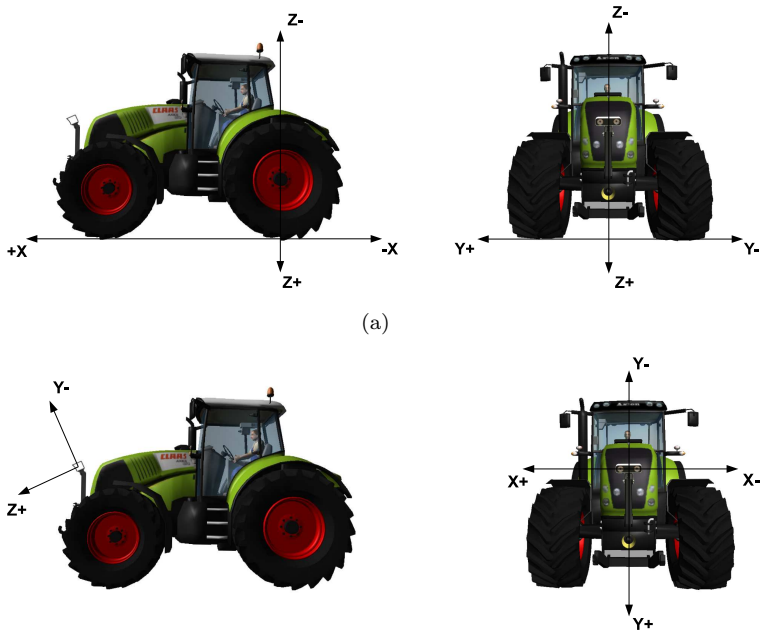


Figure 5.2: (a) Body frame. (b) Camera frame.

transformation. Stereo data is however sparse for points far from a camera which can lead to holes between samples in the grid map. This can make interpretation more difficult.

In paper A it is demonstrated how a robot can follow a path represented in a grid map (see Figure 5.3). The system maps obstacles, navigable terrain, paths, and GPS waypoints. It relies on an efficient global planner based on gradient techniques [63] which allows the robot to plan a route along the center of a path if it can see one. Without a path it switches to GPS waypoint navigation. The planner makes intelligent decisions so it can leave a path if an obstacle is encountered. This method could directly be applied to let an autonomous agricultural vehicle follow a rough GPS trajectory and switch to following field structures when it sees them.

For high accuracy agricultural applications the size of pixels need to be very small. This gives problems in terms of sparse data and high memory consumption. Centimeter accuracy maps of entire fields are likely intractable to work with. If the goal is to follow a field structure represented in the map then some

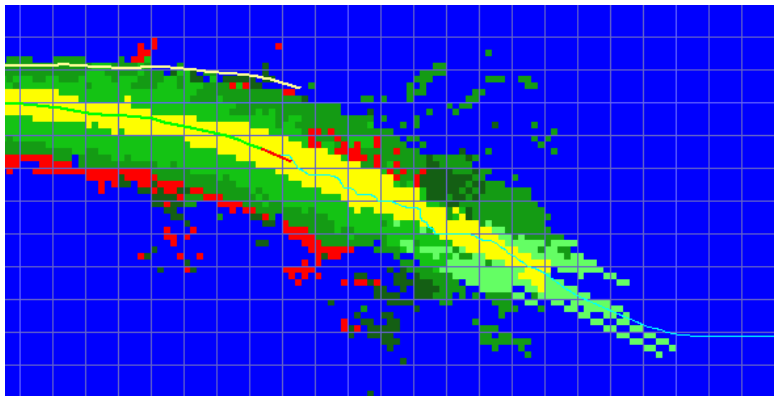


Figure 5.3: Path following using 2D grid maps. Blue is unknown terrain. Obstacles are shown in red. Ground plane is shown in various shades of green (with brighter colors indicating lower cost). The yellow region is the detected path. The robot position is marked with a red line. The cyan line indicates the planned trajectory. The green line indicates where the robot has driven. The super-imposed grid squares have a length of 1 m.

kind of interpretation must be done to transform the data to control parameters. Planning algorithms are well researched and allow optimal planning of the trajectory but require a certain amount of computation.

To give some perspectives on 2D grid maps in agriculture, some experiments have been made in using 2D grid maps to create high precision maps of crops. By creating local maps of crops it is easier to discern the rows of crops when there are high levels of weed penetration. Such maps can also be useful for precision farming and control (See Figure 5.4).

5.4 Mapping Using Clothoids

Mapping for guidance has been approached by considering a vector representation of field structures. Mapping using vectorized data has a number of advantages. It is usually compact as the input data is parameterized into 2D/3D polygons, lines, and points. Vectorized data can typically be stored in high spatial resolution which is good for high accuracy scenarios.

It was initially investigated how splines could be used for representing field structures in paper E. Later it was decided to switch to a clothoid representation

in paper F.

Clothoids are often used for modeling vehicle trajectories in robotics. Given that a:

- Vehicle's steering assembly can be represented by a bicycle model.
- Vehicle's steering rate is constant.
- Vehicle is traveling at a constant velocity.

Then the trajectory exhibited by the vehicle should follow that of a clothoid [121]. Kinematic models of Ackermann steered vehicles are typically modeled using a "bicycle model" where the front pair of wheels and the rear pair of wheels are modeled as each pair being a single wheel [59]. Such models produce a linear change in turning radius for a constant rate input.

Field structures in agriculture are generally made by Ackermann steered vehicles. Most field operations are done at relatively constant velocities and assuming the steering rates to be locally quite constant then the field structures should also follow a clothoid shape.

The Taylor series expansion of a clothoid can be represented as [123]:

$$y(x) = y_0 + \tan(\phi)x + C_0\frac{x^2}{2} + C_1\frac{x^3}{6} \tag{5.12}$$

Where y_0 is the lateral offset between the vehicle and the field structure center, ϕ is the angle of the structure relative to the vehicle. C_0 is the curvature of the structure. C_1 is the rate of curvature. This parametrization also has the advantage that it it is easy to incorporate into a vehicle controller.

The approach used in paper F involves first detecting the location of the field structure in the left image and then fitting a point to the center of mass of the detected structure in each image row. The points are then projected to 3D and transformed to the body frame. The points are then stored as a list. When the vehicle moves the list is updated to keep them in the body frame. The clothoid is then fitted to the list of points that lie along the current trajectory within ± 10 meters of the vehicle. This is done using least-squares.

For storing maps for later use it makes sense to transform them to the navigation frame which was done in paper E for storing an a priori map of field structures.

This necessitates GPS information. An example of fitting a clothoid is shown in Figure 5.5.

5.5 Mapping 3D Point Clouds

It has been investigated how accurate 3D maps can be constructed from stereo data (See Figure 5.6). The point based measurements are stored in the navigation frame to form a "cloud". By keeping data in full 3D more complete interpretation can be done of 3D objects than what is possible with grid maps. This makes it a superior approach to recognizing objects in 3D.

A typical stereo camera operating at 10Hz delivers around 1 million range measurements a second. Storing this information for even small trajectories is clearly impractical. Also, indexing information stored in the 3D point clouds can become difficult. Two solutions are discussed in paper D to deal with these problems. The first is to match measurements in overlapping images so that redundant information is fused. This is further discussed in section 5.5.1. The second solution is to store the information in a dynamic octree which was previously demonstrated in [116]. Octrees provide a hierarchical decomposition of 3D space (Figure 5.7(c)). This is done by recursively partitioning 3D space into eight octants. Searches in 3D space can then efficiently be done by traversing the octree.

5.5.1 Artifact Detection in Stereo Images

Stereo processing is the first step in 3D recognition where the 3D information is reconstructed from two stereo images. It is thus also a critical step as artifacts in the reconstruction process will make 3D recognition more difficult. Flying debris and occlusions in dense vegetation are some factors that can make reconstruction difficult in agricultural environments. Such situations will cause the stereo algorithms to match up wrong features in the image pairs. This in turn produces artifacts in the resulting disparity images which are subsequently used for 3D recognition.

In paper D a probabilistic method is introduced for detecting artifacts in the stereo range measurements. It approaches the problem by looking at the raw 3D range measurements. The method assumes that the image scene is static. This is acceptable for a large range of tasks, for example most guidance algorithms such as swath guidance focus on recognizing static structures.

The concept behind the method is to compare the range measurements in a series of stereo images. This provides redundant information which can be used for fault diagnosis. Using VO the range measurements in pairs of images are registered relative to each other in the navigation frame. This is done by storing range measurements as a 3D point cloud in a dynamic octree.

This allows residuals to be constructed by comparing range measurements of the same objects. The actual matching of range measurements is done using a ray-tracing approach to estimate which range measurements belong to the same object. An efficient approach to ray-tracing is identified by considering this in disparity space and using the octree for fast indexing. Measurements far away from the camera have a higher covariance associated to them. This is modeled by considering the uncertainty in the disparity images (using a Gaussian). Stereo artifacts can then be removed by identifying conflicting range measurements. Measurements that agree are subsequently fused by updating their mean and covariances.

The results is a more accurate 3D map of the local environment where most of the artifacts have been filtered away (Figure 5.8).

5.5.2 Measurement Model

The uncertainty in the measured ranges is affected by a number of things such as image noise, matching inaccuracies, and low-pass effects from using a correlation window in the stereo algorithm. Additionally, the actual range accuracy is governed by camera calibration errors, lens distortion and camera alignment errors.

Using the same assumption as in [119] then the image error in the stereo matching algorithm is governed by:

$$\left(\sigma_v^2, \sigma_u^2, \sigma_d^2\right) = (0.5, 0.5, 1.0) \tag{5.13}$$

These are the assumed variances for a point in the disparity image of being in a specific row (σ_v^2) and column (σ_u^2), as well as having a specific disparity (σ_d^2).

The diagonals of the covariance estimate for the 3D projection of a point in the

image is then:

$$\begin{aligned}\sigma_x^2 &= \frac{b^2\sigma_u^2}{d^2} + \frac{b^2\left(u-u_0\right)\sigma_d^2}{d^4} \\ \sigma_y^2 &= \frac{b^2\sigma_v^2}{d^2} + \frac{b^2\left(v_0-v\right)\sigma_d^2}{d^4} \\ \sigma_z^2 &= \frac{f^2b^2\sigma_d^2}{d^4}\end{aligned} \tag{5.14}$$

The covariance $\mathbf{Q}^c$ in camera 3D coordinates is then:

$$\mathbf{Q}^c = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix}. \tag{5.15}$$

This is rotated from the camera to the navigation frame as:

$$\mathbf{Q}^n = \left(\mathbf{R}_n^b\mathbf{R}_b^c\right)^\top \mathbf{Q}^c \left(\mathbf{R}_n^b\mathbf{R}_b^c\right) \tag{5.16}$$

where $\mathbf{Q}^n$ is thus the covariance in navigation frame.

The estimated location of the point in 3D is transformed to the navigation frame by:

$$\bar{\mathbf{x}}_g = \mathbf{R}_n^b\left(\mathbf{R}_b^c\bar{\mathbf{x}}^c + \mathbf{p}_c^b\right) + \mathbf{p}^n \tag{5.17}$$

5.5.3 Artifact detection

Let each measurement $(\mathbf{x}_i)$ be associated with a covariance $\mathbf{Q_i}$ according to 5.15. The divergence (distance between stochastic distributions) between measurements $[\mathbf{x}_i, \mathbf{Q_i}]$ and $[\mathbf{x}_j, \mathbf{Q_j}]$ then needs to be expressed.

A general measure of distance between distributions f_i and f_j [87], is the Jeffreys-Matusita (JM) divergence defined as

$$J_{ij} = \left(\int_\Omega \left(\sqrt{f_i(r)} - \sqrt{f_j(r)}\right)^2 dr\right)^{\frac{1}{2}} \tag{5.18}$$

The JM distance has the salient feature to be easily applicable on arbitrary distributions. The JM distance $J_{ij} = 0$ when the distributions $f_i(r)$ and $f_j(r)$ are

equal and overlapping. The JM distance takes the value $J_{ij} = \sqrt{2}$ when the two distributions are totally separated. Bhattacharyya introduced the parameter

$$\rho_{ij} = \int_{\Omega} \sqrt{f_i(r)} \sqrt{f_j(r)} dr, \tag{5.19}$$

and the negative logarithm, α_{ij}, of this quantity,

$$\alpha_{ij} = -\ln \rho_{ij}, \tag{5.20}$$

to obtain

$$J_{ij}^2 = 2(1 - \rho_{ij}) = 2(1 - \exp(-\alpha_{ij})) \tag{5.21}$$

[54] showed that when the two distributions are normal multivariate of degree n: $f_i(r) = N(\mathbf{x}_i, \mathbf{Q}_i)$ and $f_j(r) = N(\mathbf{x}_j, \mathbf{Q}_j)$ then

$$\alpha_{ij} = \frac{1}{8}(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{Q}_{ij}^{-1} (\mathbf{x}_i - \mathbf{x}_j) \tag{5.22}$$

$$+ \frac{1}{2} \ln \frac{\det(\mathbf{Q}_{ij})}{\sqrt{\det \mathbf{Q}_i \det \mathbf{Q}_j}}, \tag{5.23}$$

$$\text{where } \mathbf{Q}_{ij} = \frac{\mathbf{Q}_i + \mathbf{Q}_j}{2}.$$

Further, the probability of misclassification P_e is bounded:

$$\frac{1}{16}(2 - J_{ij}^2)^2 \leq P_e \leq 1 - \frac{1}{2}(1 + \frac{1}{2} J_{ij}^2), \tag{5.24}$$

which is equivalent to

$$(0.5 \exp(-\alpha_{ij}))^2 \leq P_e \leq 0.5 \exp(-\alpha_{ij}). \tag{5.25}$$

In this context α_{ij} is calculated from Eq. 5.23 and the upper bound in 5.25 is used to estimate misclassification. When a new point is outside this bound, the point is considered an artifact of the 3D stereo processing and discarded.

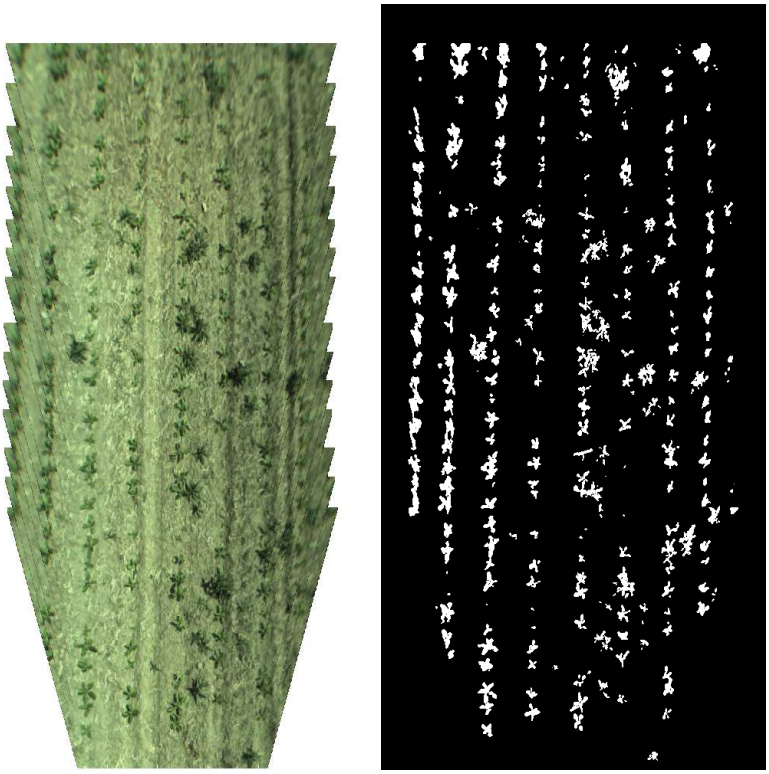
5.5.4 Startup and algorithm procedure

A mapping can be initialized by a prior map or it can be initialized by a first stereo image of the map. Hence there exist octrees occupied by sets $[\mathbf{x}_i, \mathbf{Q}_i]$ and subsequent stereo images provide $[\mathbf{x}_j, \mathbf{Q}_j]$, $j = i+1, i+2, ..i+N$. If the first two points are outside the accepted divergence, a third is processed, until at least two points agree within the chosen value of α_{ij}. If the point lies within the interval then it is assumed that both measurements pertain to the same

object and the map is updated by merging the two distance estimates into a new estimate given by:

$$\begin{aligned} \mathbf{Q}_m^{-1} &= \mathbf{Q}_i^{-1} + \mathbf{Q}_j^{-1}, \\ \bar{\mathbf{x}}_m &= \mathbf{Q}_m \left(\mathbf{Q}_i^{-1} \bar{\mathbf{x}}_i + \mathbf{Q}_j^{-1} \bar{\mathbf{x}}_j \right). \end{aligned} \tag{5.26}$$

Subsequent measures are compared with $[\mathbf{x}_m, \mathbf{Q}_m]$, which replaces $[\mathbf{x}_i, \mathbf{Q}_i]$ in the calculations. The two measurement i and j are deleted from the octree and replaced by the merged estimate m.

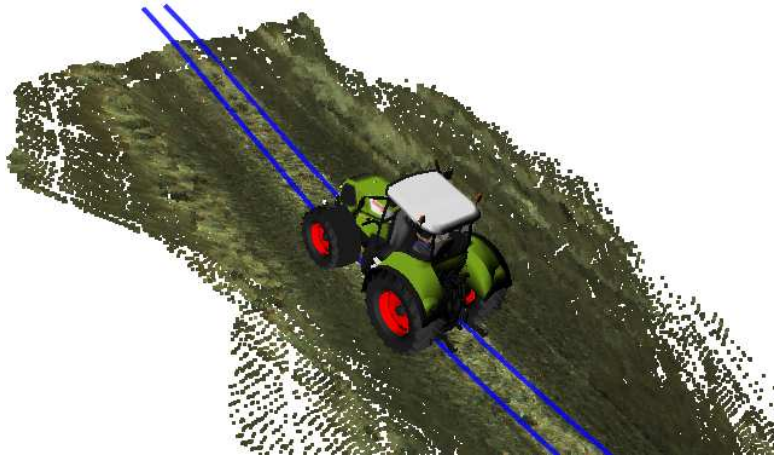


(a) (b)

(c)

Figure 5.4: (a) 2D grid map of tiled color images. (b) 2D grid map of color segmentation of crop using an approach similar to [100]. (c) 3D point cloud of corresponding map.

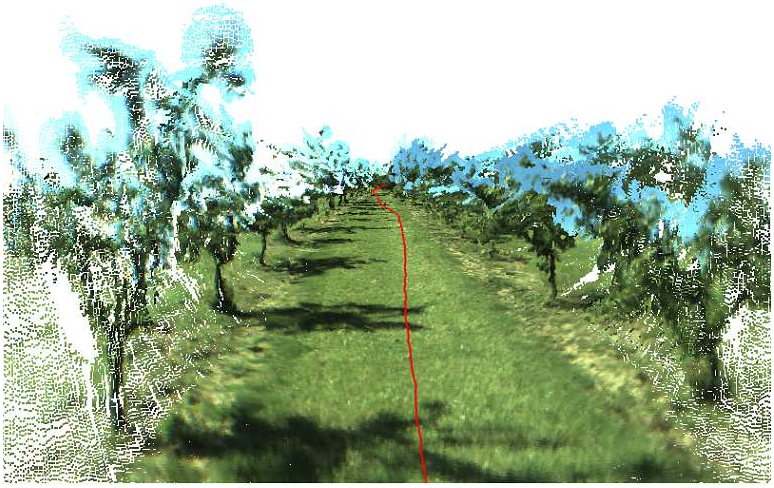
(a)



(b)

Figure 5.5: Estimating Curvature by fitting a clothoid to a list of points extracted by the recognition algorithms. The blue lines represent the bounds of the field structure.

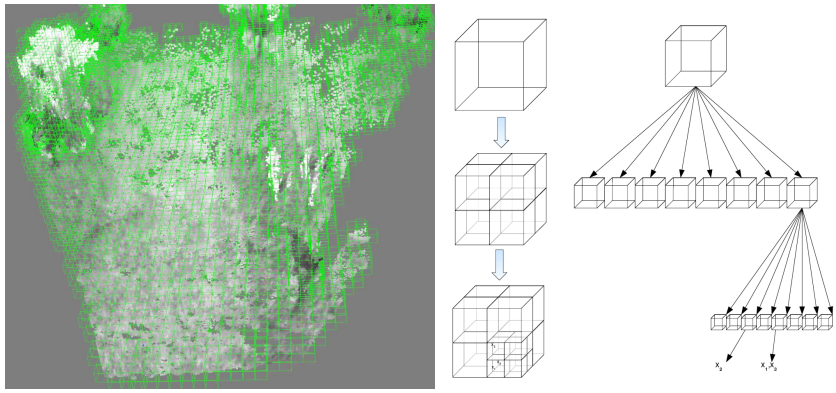
(a)



(b)

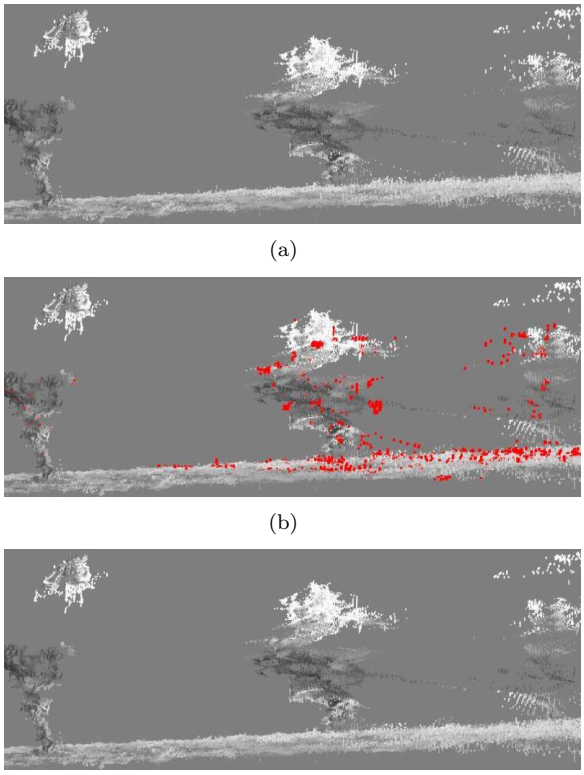
Figure 5.6: (a) left image from the stereo camera with bounding box for the stereo.. (b) Mapping using 3D point clouds. The red line shows where the camera has been.

(a)



(b) (c)

Figure 5.7: (a) A left image from the stereo camera. (b) Octree decomposition for the corresponding 3D data. Green cubes illustrate the bounds of the bottom-most nodes in the octree hierarchy. (c) In octree decomposition each cube of space can recursively be split into 8 sub-cubes called nodes. At the bottom of the hierarchy (node leaves) the 3D points themselves are stored. Here illustrated with points $\mathbf{x}_1, \mathbf{x}_3$ at the end of one leaf, and $\mathbf{x}_2$ at the end of another.



(a)

(b)

(c)

Figure 5.8: (a) New measurements from stereo camera (sideways view with camera position to the left). (b) The identified artifacts highlighted in red. (c) Filtered measurements.

Chapter 6

Fault-Tolerance

Fault-tolerance can enhance the safety and availability of a system by making it less susceptible to faults. In [15] it was proposed that faults can be modeled as deviations from a nominal system behavior. Given that a fault can be detected then remedial action to compensate the fault can be done by reconfiguring the system architecture.

Fault-tolerance is achieved by exploiting system redundancy. Redundancy provides both a method to diagnose faults and also a method to reconfigure. Redundancy exists both as physical redundancy (in the form of redundant hardware) but also as analytical redundancy. In analytical redundancy an explicit mathematical model is used to describe relations between signals. This creates redundancy by allowing certain system states to be estimated through the use of different subsets of signals. Automatic treatment of such models to identify redundant components can be done using structural analysis techniques [15].

GPS and IMU are becoming common sensor modalities on agricultural vehicles. Clearly, such sensors can be used along with vision and wheel odometry for increased hardware redundancy. Another source of redundancy comes from the many vision algorithms that have been described in this thesis. This chapter explores how fault-tolerance can be achieved by exploiting redundancy in sensors and algorithms.

6.1 Related Work

Literature exists to treat fault detection systematically [14], [15]. In [13] these methods were used to detect faults in positioning on a ship but the methods have never been applied to systems incorporating vision sensors. Systematic methods for applying fault-tolerant design to learning and mapping have received minor attention. The concept behind the methods investigated here is to use a behavior-based model. The purpose of the model is to arrive at a set of constraints that can be used for analysis of system structure and subsequent generation of residuals for fault diagnosis. This idea was brought into the field of fault diagnosis by [126] and later expanded, see [125] and [15]. The advantage of this approach over classical methods, [136], include the ability to use a formulation of behaviors at a high level of abstraction.

There does not exist much literature on detecting faults in vision systems in agriculture. In [113] a fuzzy logic approach with selective fusion was used to choose between vision and GPS guidance based on how much they agree and their signal qualities. The approach presented here is more systematic and exploits additional redundancy so that fault-detection becomes less dependent upon the measured signal qualities which may themselves be faulty.

6.2 Behavioral Model

Investigating the fault-tolerant properties of a system with many sensors and algorithms requires a systematic approach. Behavioral models of a system can be used to formulate models of a system at high levels of abstraction. This can be used to decrease complexity for an otherwise complicated problem.

A behavioral model uses mathematical constraints (equations) to describe the relationships between system variables over time. According to fault-tolerant theory it is only necessary to model the nominal situation without faults. If a constraint is no longer satisfied then this is a fault in the system.

In paper E it was discussed how such a model can be created for a vision guidance system. Such an analysis has previously not been done for a vision system. The following sections expand upon this research by considering also texture learning, wheel odometry, and IMU.

6.2.1 Position Constraints

For positioning sensors such as GPS, IMU, and VO the approach used in [13] was adapted for the vision guidance system:

$$
\begin{aligned}
c_1 &: \dot{\mathbf{p}}^b = \mathbf{R}_b^n(\Theta)\frac{d}{dt}\mathbf{p}^n \\
c_2 &: \ddot{\mathbf{p}}^b = \mathbf{R}_b^n(\Theta)\frac{d^2}{dt^2}\mathbf{p}^n \\
c_3 &: \omega^b = J(\Theta)\frac{d}{dt}\Theta \\
m_1 &: \mathbf{p}_{gps}^n = \mathbf{p}^n + \mathbf{R}_b^n(\Theta)\mathbf{p}_{gps}^b \\
m_2 &: \mathbf{v}_{vo}^b = \dot{\mathbf{p}}^b \\
m_3 &: \mathbf{v}_{wo}^b = \dot{\mathbf{p}}^b \\
m_4 &: \mathbf{a}_{imu}^b = \ddot{\mathbf{p}}^b \\
m_5 &: \omega_{vo}^b = \omega^b \\
m_6 &: \omega_{imu}^b = \omega^b \\
m_7 &: \omega_{wo}^b = \omega^b
\end{aligned}
\tag{6.1}
$$

The position of the vehicle in navigation frame is given by $\mathbf{p}^n$. This position is measured by the GPS, $\mathbf{p}_{gps}^n$. VO measures the vehicle velocity vector in body coordinates $\mathbf{v}_{vo}^b$ which is also measured by the wheel odometry $\mathbf{v}_{wo}^b$. The rotation from navigation to body frame is given by $\mathbf{R}_b^n$. Likewise, the IMU accelerometers measure the vehicle accelerations $\mathbf{a}_{imu}^b$. The angular velocity of the vehicle in body frame (ω^b) is measured by VO (ω_{vo}^b), wheel odometry (ω_{wo}^b) and the IMU gyroscopes (ω_{imu}^b). ω^b is related to the Euler rate vector $\frac{d}{dt}\Theta$ using a function $J(\Theta)$.

Note that IMU and VO provide 6 DOF estimates where the wheel odometry is 3 DOF as it assumes motion on a plane. This means when residuals are created using constraints incorporating wheel odometry, the IMU and VO information must also be reduced to 3 DOF.

Θ is not directly measured by any sensor and can be estimated using GPS combined with IMU and/or VO, or using a multi-antenna GPS. In the analysis done in this thesis it has been approximated using the course over ground given by the GPS (Θ_{gps}):

$$m_8 : \Theta_{gps} = \Theta \tag{6.2}$$

6.2.2 Map Constraints

It has been investigated how map information from a number of different sources can be used to provide redundancy for 3D vision.

First, for many farming applications it is plausible to have some kind of a priori knowledge of the field. This could be provided by recording a previous traversal of the field where the location of relevant field structures are then stored. To use such a map it must be defined in the navigation frame so that the field structure can be localized relative to the vehicle using GPS.

Another source of information is from the learning system described in Chapter 3 where texture information is used to supplement 3D vision.

Let the position of a field structure in navigation frame be formulated by variable s^n, and s^b in body frame. Then the measurements from the various components can be seen as a measurement of this variable:

$$\begin{aligned} c_4 &: s^b = \mathbf{S}_b^n(\Theta, \mathbf{p}^n, s^n) \\ c_5 &: \mathcal{M}_b(\omega^b, \dot{\mathbf{p}}^b) = s^b \\ c_6 &: \mathcal{M}_n(\Theta, \mathbf{p}^n) = s^n \\ m_9 &: s_l^b = s^b \\ m_{10} &: s_c^b = s^b \end{aligned} \tag{6.3}$$

$\mathbf{S}_b^n$ defines a transformation of the field structure from navigation to body frame. It depends on the vehicle attitude Θ and position $\mathbf{p}^n$ in navigation frame. $\mathcal{M}_b$ is an incrementally built map from the mapping system in the body frame. Given ω^b and $\dot{\mathbf{p}}^b$ the location of the field structure from previous measurements can be transformed to the current vehicle position. This transformation is updated at each time step. $\mathcal{M}_n$ is the a priori map in navigation frame. Given Θ and $\mathbf{p}^n$ the position of the field structure in navigation frame can be looked up in the map. The measured position of the field structure in body frame from the 3D vision is given by s_c^b and s_l^b for the texture learning.

6.3 Structural Analysis

Analytical derivation of the parity relations can be done using structural analysis [15]. This can be done automatically using the software SaTool [16]. A graphical representation of the result of the SaTool analysis is given in Figure 6.3. From the parity relations, residuals can be formulated which in turn can be used for detecting and isolating faults. The analytical result for examining the described system gives rise to the following residuals:

$$
\begin{aligned}
r_1 &: \omega^b_{imu} - \omega^b_{vo} = 0 \\
r_2 &: \omega^b_{wo} - \omega^b_{vo} = 0 \\
r_3 &: \mathbf{v}^b_{wo} - \mathbf{v}^b_{vo} = 0 \\
r_4 &: s^b_l - s^b_c = 0 \\
r_5 &: s^b_c - \mathbf{S}^n_b(\Theta_{gps}, \mathbf{p}^n_{gps}, \mathcal{M}_n(\Theta_{gps}, \mathbf{p}^n_{gps})) = 0 \\
r_6 &: \mathbf{a}^b_{imu} - \mathbf{R}^n_b(\Theta_{gps}) \frac{d^2}{dt^2} \mathbf{p}^n_{gps} = 0 \\
r_7 &: \mathbf{v}^b_{vo} - \mathbf{R}^n_b(\Theta_{gps}) \frac{d}{dt} \mathbf{p}^n_{gps} = 0 \\
r_8 &: s^b_c - \mathcal{M}_b(\omega^b_{vo}, \mathbf{v}^b_{vo}) = 0 \\
r_7 &: \omega^b_{vo} = J(\Theta_{gps}) \frac{d}{dt} \Theta_{gps} = 0
\end{aligned}
$$

Based on these residuals all single faults should be structurally isolable as shown in Table 6.3 and outputted from the SaTool report generator. Note how constraints c_1,c_2,c_3,c_4 can't fail since they do not rely on measured properties.

Constraint	c_1	c_2	c_3	c_4	c_5	c_6	m_1	m_2	m_3	m_4	m_5	m_6	m_7
Status	-	-	-	-	i	i	i	i	i	i	i	i	i

Constraint	m_8	m_9	m_{10}
Status	i	i	i

Table 6.1: Summary of the complete d/i properties of the investigated system. (- : can't fail, 0 : undetectable, i : isolable, d : detectable)

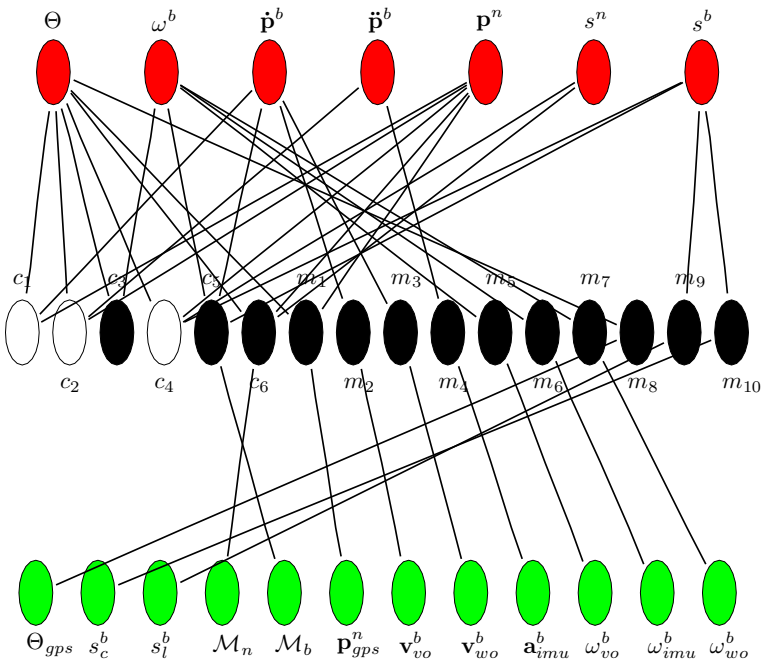


Figure 6.1: Bipartite graph from SaTool. Legend: Red - Unknown variable, Green - Known variables, Black - Isolable, Purple - Detectable, Blue - Undetectable and White - Can not fail.

6.3.1 Discussion of Structure Analysis

The results demonstrate that GPS and IMU provides useful redundant information to the vision system. Specifically, the ability to incorporate a priori map information using GPS is extremely useful if the vision component fails. On the other hand it is interesting to note that the different vision algorithms can also be used for self-supervision. For example residual r_8 provides a means to detect a fault in VO by using the fact that the position of field structures extracted by 3D vision should change in accordance with the VO motion. Also, residual r_4 can be used to detect inconsistencies between 3D vision and the output of tracking based on texture. In effect these redundancies are brought about by having vision algorithms that are tracking independent observations.

In paper E it was demonstrated how a fault in the GPS could be diagnosed using the vision system. An observation was that detecting a GPS fault is only weakly detectable in residual r_7 since VO can only monitor the derivative of the GPS position for faults. However, using mapping information a GPS fault

can become strongly detectable through residual r_5 since the information in the map should in the no-fault case coincide with the measured position of the field structure from vision.

In paper F it was investigated how faults could be detected in the learning algorithms by validating the tracking performance from texture learning with the output from 3D tracking. This allowed novel self-supervision of the learning process.

6.4 Design of Detectors

Given $\mathcal{H}_0$ is the non-faulty hypothesis and $\mathcal{H}_1$ is the hypothesis in the faulty scenario, then a likelihood ratio test (LRT) can be used to detect a fault [56]:

$$L(x) = \frac{p(x;\mathcal{H}_1)}{p(x;\mathcal{H}_0)} > \gamma \tag{6.4}$$

where the probability densities from the observed distributions should be used, $p(x;\mathcal{H}_1)$ for the case of a fault, $p(x;\mathcal{H}_0)$ for the normal case, respectively. x is a signal sample. γ then becomes a decision threshold to decide between the two hypotheses.

In paper E the distributions of the residuals were analyzed for the case with and without a GPS fault. A similar analysis was also done in paper F for the case with a fault in texture learning. The distributions were found close enough to Gaussian in both cases to warrant modeling faults as a DC level in White Gaussian Noise (WGN):

$$\begin{aligned} \mathcal{H}_0 &: x[n] = w[n] \qquad n = 0, 1, ..., N-1 \\ \mathcal{H}_1 &: x[n] = A + w[n] \quad n = 0, 1, ..., N-1 \end{aligned} \tag{6.5}$$

$x[n]$ is the signal under both hypotheses and $w[n]$ is WGN with variance σ^2. A represents the magnitude of the fault and N is the sample window.

A Neyman-Pearson detector can then be formulated to detect a fault [56]:

$$\frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}}\exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}\left(x\left[n\right]-A\right)^2\right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}}\exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}\left(x^2\left[n\right]\right)^2\right]} > \gamma \tag{6.6}$$

Which yields the test function:

$$\frac{1}{N}\sum_{n=0}^{N-1} x\left[n\right] > \frac{\sigma^2}{NA}\ln\gamma + \frac{A}{2} > \gamma' \tag{6.7}$$

Taking the mean of the signal over N samples corresponds to decreasing the variance of the resulting signal to $\frac{\sigma^2}{N}$. Thus from [56] it can be calculated what the probability of a false alarm P_{FA} and the probability of detection P_D then are:

$$P_{FA} = Q(\frac{\gamma'}{\sqrt{\sigma^2/N}}) \tag{6.8}$$

$$P_D = Q(\frac{\gamma' - A}{\sqrt{\sigma^2/N}}) \tag{6.9}$$

Where Q is the right-tail probability. A suitable value of N and γ' then need to be configured in order to the detection performance P_D while assuring a low rate of false alarms P_{FA}. As faults can be both positive and negative in magnitude, the test is setup for both positive and negative values of A.

6.4.1 Discussion of Detectors

The Neyman-Pearson detector was used in papers E and F where it was successfully demonstrated that it could be used to detect faults. In paper F it was shown how field structures could be parameterized for easy integration into the detector. It has been shown that suitable values for P_{FA} and P_D can be found that allow fast detection while at the same time assuring low false alarm rates for the cases analyzed.

For different field structures the residual variance may be different in the nominal case. It has only been analyzed for swath. A solution may require online estimation of this parameter which could potentially be incorporated into the mapping system. Also, the faults exhibited by for example GPS may potentially leave different traces in the output residuals based on the type of fault. This must be investigated further to see if further extensions are needed in formulating the residual generators. Lastly, it could also be investigated whether a vector-based approach could be used for combining one or more residuals. This could potentially allow better detection properties.

Using statistical change detection theory is clearly a valid approach for detecting faults in a vision guidance system. The computational requirements are also very low compared to the vision algorithms so there is little overhead in including such detectors in future guidance systems.

Chapter 7

Conclusion

Systematic design methods for obtaining fault-tolerance in a vision system have been demonstrated. Graph-based modeling can be used to generate diagnostic relation. It is shown how single faults in sensors and/or algorithms can be detected and diagnosed using statistical change detection theory.

Dropouts of 3D vision and faults in classification can be handled using redundant hardware as well as using mapping and learning methods. It is shown how GPS enhanced with a priori map information complements 3D vision. Positioning can be made fault-tolerant by combining visual odometry, GPS, inertial sensors, and/or wheel odometry. Mapping is shown to provide valuable information about past observations which can be used to detect artifacts in the vision system as well as enhancing recognition.

Research in mapping has been taken beyond state-of-the-art by enhancing the accuracy of existing visual odometry systems and by using diagnostic residuals to detect faults in the mapped information. Visual odometry has been enhanced by investigating new and improved methods for feature detection and tracking.

Learning is shown to provide valuable information about field structures by learning texture information from 3D vision tracking results. It is demonstrated how texture and 3D vision can be fused in a mapping system to facilitate guidance. It was found necessary to also make the learning process fault-tolerant

by constructing diagnostic residuals to evaluate the performance of the learned models.

Care has been taken to consider the computational burden of algorithms so that the systems could be tested online. This has allowed experimental validation to be done which confirms the validity of the results.

The advances in outdoor vision-based positioning presented in this thesis has allowed more accurate and reliable maps to be made of local environments. This has the potential for a broad range of agricultural applications including cultivating, spraying, and harvesting. As farm vehicles start incorporating more sensory information, mapping will likely become an important component of such systems.

Putting the project into perspective there are a number of topics for future research. Within the vision domain it could be attractive to be able to extend VO to also allow SLAM. Also, coming up with a system for doing long-term learning would be attractive so textures do not need to be learnt each time. Recognizing objects in the far field outside of stereo range could also be an interesting topic. From a control perspective the current system can allow novel steering methods by using VO for example to detect and model wheel slippage. Likewise, variable velocity control based on estimating properties of the environment using texture and 3D could help farmers optimize field operations.

A push towards complete automation in agriculture will require a holistic approach. This will require integrating perception, planning, and control. It will require using many sensors and not just vision. This thesis has taken some first steps in showing how a fault-tolerant system can be designed at a guidance level. With dynamic route planning and telematics also being integrated the fault-tolerant concepts will also have to be extended to cover these areas.

Paper A

Fast Color/Texture Segmentation For Outdoor Robots

Morten Rufus Blas, Motilal Agrawal, Aravind Sundaresan and Kurt Konolige. Fast Color/Texture Segmentation For Outdoor Robots. *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 4078-4085, Nice, France, 2008. Published.[1]

[1]This material is based upon work supported by the United States Air Force under Contract No. FA8650-04-C-7136. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

Abstract:
We present a fast integrated approach for online segmentation of images for outdoor robots. A compact color and texture descriptor has been developed to describe local color and texture variations in an image. This descriptor is then used in a two-stage fast clustering framework using K-means to perform online segmentation of natural images. We present results of applying our descriptor for segmenting a synthetic image and compare it against other state-of-the-art descriptors. We also apply our segmentation algorithm to the task of detecting natural paths in outdoor images. The whole system has been demonstrated to work online alongside localization, 3D obstacle detection, and planning.

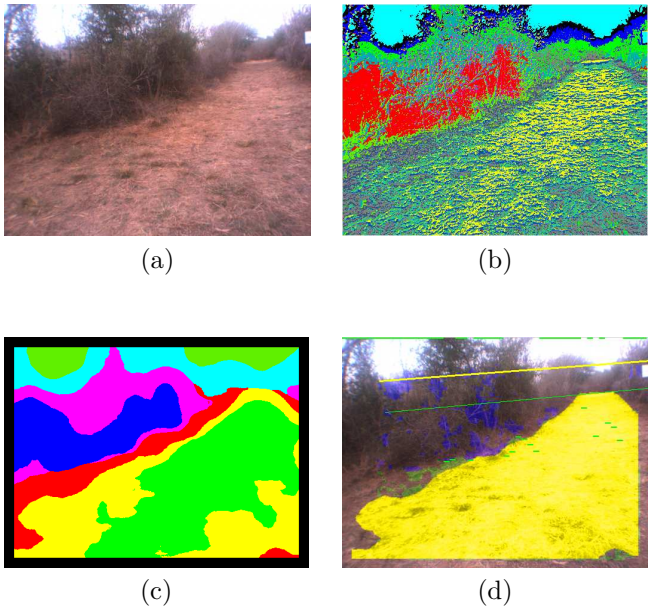
A.1 Introduction

Autonomous navigation for outdoor, unstructured environments is an important research problem in robotics with numerous applications. The ability to recognize navigable terrain and avoid obstacles is a critical component for autonomous navigation. Current state-of-the-art systems employ a range sensor such as stereo cameras or LADAR to reason about the geometry of the world and identify geometrical obstacles. However, geometrical reasoning alone is unlikely to result in intelligent behavior of the robot. For example, it is hard to distinguish between tall grass and a short wall based on geometry alone. In addition, learning is a very important component of an intelligent system. If the robot has traversed over tall grass earlier, it can learn that as traversable terrain and mark it as such if it sees it again. It is clear that appearance-based terrain recognition plays an important role in such intelligent behaviors and segmentation is the first step toward recognition.

Appearance-based segmentation is a classical problem in computer vision. For robotics, the specific challenge is to be able to do reliable segmentation of outdoor scenes in an efficient manner so that it can be used online. In our experience, color alone is not a reliable feature. For example, in Figure A.1(a) it is hard to distinguish between the bushes and the darker grass on the ground based on color. However, the texture of the grass and the bushes is very different.

In this paper, we present an online segmentation algorithm that combines color with texture information to group similar regions. Our algorithm has several novel features.

- Compact texture/color descriptors. It is important to have compact rep-



(a) (b)

(c) (d)

Figure A.1: Various steps of our segmentation algorithm on a typical outdoor image. (a) The image from one of the stereo cameras. (b) Each pixel assigned to a texton. (c) Each histogram of textons gets assigned to a histogram profile. (d) A path is recognized (in yellow).

resentations of the information necessary to distinguish textures. Here we carefully choose a small local neighborhood vector that incorporates the important aspects of texture and color.

- A two-stage unsupervised online learning process. For each image, we cluster neighborhood vectors to find a small set of basis vectors (textons [74]) that characterize scene textures (Figure A.1(b)). Then, we cluster histograms of textons over larger areas to find more coherent regions with the same mixture of textons (Figure A.1(c)).

Note that the problem we are interested in here is online, unsupervised segmentation, not classification based on a library. One application is finding paths in off-road terrain, where the path appearance may be unlike anything seen previously (Figure A.1(d)). We have successfully demonstrated online path detection in a complete outdoor navigational system that uses stereo-vision as its primary sensor.

A.2 Algorithm Overview and Related Work

A.2.1 Texture Representation

Approaches to texture representation include co-occurrence probabilities [44], Markov modeling [58, 11, 83], multichannel filtering [51, 138, 20, 84], Local Binary Patterns (LBP) [81], and texton-based approaches [84, 5, 74]. The more recent approaches use either a filter bank or a small neighborhood as a feature descriptor for each pixel – for example, LBP's are formed by subtracting the intensity of the center in a small local neighborhood and then binarizing the intensity variation in the neighborhood.

In a seminal paper, Leung and Malik [74] showed that many textures could be represented and re-created using a small number of basis vectors extracted from the local descriptors; they called the basis vectors *textons*. While Leung and Malik used a filter bank, later Varma and Zisserman [139] showed that small local texture neighborhoods may be better than using large filter banks. In addition, a small local neighborhood vector can be much faster to compute than multichannel filtering such as Gabor filters over large neighborhoods.

Many schemes exist for combining local texture with color information [84, 5, 75]. The sheer number of variations makes it hard to decide what is a good representation of both color and texture for segmentation. In this paper, we describe

a segmentation algorithm that uses a compact descriptor for representing color and texture. Our descriptor fits into the class of local texture neighborhoods and in that sense is similar to LBP's. For each local neighborhood (a 3x3 or 5x5 region centered at a pixel), the descriptor is composed of the color information of the center and the relative change in intensity in the neighborhood. This is computed by subtracting out the intensity of the center from the intensities in the neighborhood. Unlike LBP, we do not binarize the center subtracted intensity values, thereby retaining the actual gradient values. In contrast with other local neighborhood descriptors, ours is more compact since we do not store the color variation in the neighborhood. For a typical 3x3 window size, for example, our descriptor is an 11-dimensional vector, whereas storing the raw RGB values will result in a 27-dimensional vector. A compact descriptor becomes crucial for the clustering step to be fast and real time.

A.2.2 Segmentation

The raw descriptors must be grouped to segment the image; a number of clustering algorithms exist for this task. The K-means algorithm and its many variations is a standard way of doing this [50, 28]. Graph-cut-based approaches [84] generally result in better and sharper boundaries but are computationally more demanding. Self-Organizing Maps [85] yield clusters such that neighborhood relations between the clusters are preserved, allowing one to better visualize the input space. Another approach are level-sets [75] which handles boundaries by first finding homogeneous areas in the image and then propagates these areas to unlabelled parts of the image.

In our algorithm we use two-stage, unsupervised clustering to find smooth similar regions based on the descriptors. The choice of clustering framework was largely dictated by the need for it to be fast and efficient. In our method, we use the K-means algorithm to perform clustering – K-means has the best trade-off between good results and speed.

For the first clustering step, our descriptors are computed at each pixel and are then clustered using K-means to find the basis vectors or textons. Each pixel then gets assigned to the closest texton. This is shown in Figure A.1(b). As can be seen, a segmentation based on simple pixel classification is very noisy. To capture statistics of larger areas, we compute a histogram of these textons over a window, and cluster the histograms again using K-means to find similar regions in the image. Histograms of textons are computed efficiently using integral images [140]. Regions that are close together are then merged to give the final segmentation. Figure A.1(c) shows the final segmentation results.

Not only is our descriptor compact and faster to compute but it captures all the local texture variations, resulting in better segmentations. We present experimental results of comparing these texture descriptors to segment a synthetic image. While there has been previous work [139] on comparing the different types of texture descriptors for the task of texture classification, to our knowledge no comparisons have been done earlier for the task of texture segmentation.

A.2.3 Path Finding

Finally, for an application, we use the segmentation algorithm to recognize natural paths in outdoor images. The image is segmented, and then we look for regions that share the geometric attributes of a path. Because there are only a small number of regions, various combinations of regions that could possibly be a path can all be checked. This path detection algorithm runs online on the robot in real time and we present results of our path detection algorithm on several types of outdoor paths. Other work has previously been done in road detection where we mention the work by Fernandez and Price [31] who used region growing in HSI color values to find the road borders. Others include Dahlkamp et al. [26], who used self-supervised learning on color images to extend roads found by LADAR. In the area of stereo-vision Soquet et al.[122] used a stereo-based color segmentation algorithm to determine road segments. Texture has also been used as seen in Zhang and Nagel [144], who explore anisotropic texture features of roads for segmentation.

While all the individual components of our segmentation algorithm are known, we have judiciously selected each step of the processing pipeline so that we are able to run our algorithm online on the robot in real time. It is the integration of these fast techniques, coupled with our compact texture descriptor and a two-stage online learning process, that characterizes our work.

The rest of the paper is organized as follows. Section A.3 describes our segmentation algorithm in detail, and the results of our segmentation algorithm for a synthetic image are presented and compared with other texture descriptors in Section A.4. Our path recognition algorithm is discussed in Section A.5 and results of this algorithm are discussed in Section A.6. Section A.7 concludes the paper and discusses ongoing and future work.

A.3 Segmentation Algorithm

The first step of our segmentation algorithm is to learn a set of textons (basis descriptor vectors) for the image. A local descriptor (3x3 or 5x5 window) is computed at each pixel location, and the ensemble of descriptors is clustered to find a small set of textons. Each pixel location then gets assigned to one of these textons by comparing its descriptor using Euclidean distance.

A.3.1 Textons

Our descriptor is composed of color and texture information for a 3x3 or 5x5 pixel neighborhood. The image is first transformed to the CIE*LAB colorspace using an efficient lookup table to do the RGB to LAB conversion. Colors in LAB are more perceptually linear than in the RGB space, thereby resulting in better clusters. This gives the brightness information L and the color channels a, b. The texture information is taken as the surrounding pixel intensities minus the center intensity. Each pixel location p_i in the image can then be represented using the descriptor:

$$p_i = \begin{bmatrix} W_1 * L_c \\ W_2 * a_c \\ W_2 * b_c \\ W_3 * (L_1 - L_c) \\ \vdots \\ W_3 * (L_8 - L_c) \end{bmatrix} \tag{A.1}$$

Here (L_c, a_c, b_c) is the color of the center pixel, and $L_1, L_2, ..., L_8$ are the intensities of the surrounding pixels. The set of weights $\{W_1 = 0.5, W_2 = 1, W_3 = 0.5\}$ is used to balance how much to rely on color, texture, and brightness for the clustering. These were set as to weigh chrominance higher than luminance. Also, since texture takes up many of the descriptor rows it must be downweighted so the color still has an impact on the clustering. The assumption here is that in a local neighborhood the color does not vary much, so including the color channels for all 3x3 pixels does not provide additional information. The L_c, a_c, b_c components could also be computed as an average of the 3x3 neighborhood but this was not done to speed up computation. See top of Figure A.2 for an overview. The 5x5 version of the descriptor is similar but uses a larger neighborhood size resulting in a 27-dimensional descriptor.

A.3.2 Clustering to Textons

The K-means algorithm seeks to minimize

$$J = \sum_{i=1}^{n} \min_{j} |p_i - c_j|^2 , \tag{A.2}$$

where p_i is each descriptor in the image and c_j are the textons; basically, it finds a set of basis descriptors such that the Euclidean distance between them and all descriptors is minimized. $j = 1, .., k$ is the number of textons we desire to learn. For our outdoor robotic sequences $k = 16$ gave a good trade-off between accuracy and efficiency. In the K-means iterations, reclassification attempts are not made for points that lie less than half the mean absolute distance away from their currently classified center (similar to [55]). This considerably speeds up the implementation without a significant loss in precision.

A.3.3 Histogram Clustering

Once the 16 textons for a given image have been established, each pixel is classified as belonging to one of these using Euclidean distance. A simple threshold identifies outliers. Integral images [140] are then constructed for each of the 16 textons. An entry in the integral image at location x is simply the sum of the count of each of the 16 textons in the rectangle formed by the point x and the origin. With the integral image calculated, it takes only four additions to calculate the total number of each texton over any upright, rectangular area, independent of its size. This is then used to extract a histogram profile for a window neighborhood across the image. Experimentally, a 32x32 window gives the best results for our image size. This is similar to what was observed in [75] where the window was chosen to 1-2% of the total image size.

K-means is then run on the histograms to extract a set of histogram profiles, using Euclidean distance as the norm. Boundary conditions between areas of different texture are not explicitly treated and thus may receive their own cluster representing "mixed terrain". The choice of histogram clusters ($k = 8$) was set so as to slightly over-segment the image. Texton outliers are not included in the histogram clustering. See bottom of Figure A.2 for an overview.

Finally, the Earth Movers Distance [114] was used to merge similar clusters if the threshold was below 100. EMD is a good distance measure for histograms, but too computationally expensive to be used directly in the K-means clustering. The EMD ground distance matrix for the histograms was set to the Euclidean

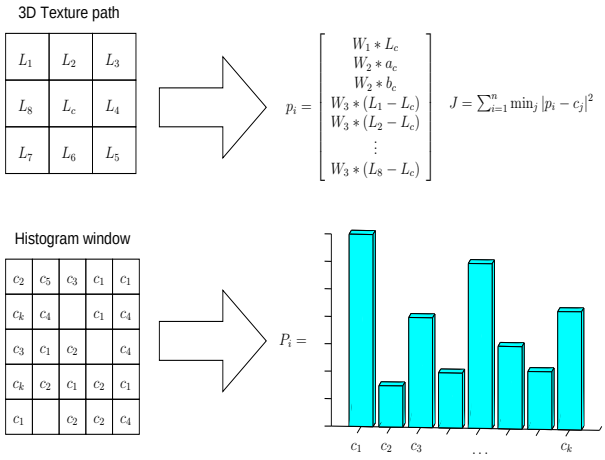


Figure A.2: The segmentation algorithm works in two steps. First textons are learned from the image. Then histograms are constructed from textons and clustered. The missing values in the histogram window represent outliers.

distance between each basis texton. Given two textons $c_{t,i}$ and $c_{t,j}$ the Euclidean distance between them can be written as $d_{t,ij}$:

$$d_{t,ij} = \|c_{t,i} - c_{t,j}\|^2 \tag{A.3}$$

Given that we have 16 basis textons this gives a 16x16 distance matrix $\mathbf{D}$ for comparing two histograms (generalized for an m-by-n matrix):

$$\mathbf{D} = \begin{bmatrix} 0 & d_{t,01} & d_{t,02} & \cdots & d_{t,0n} \\ d_{t,10} & 0 & d_{t,12} & \cdots & d_{t,1n} \\ d_{t,20} & d_{t,21} & 0 & \cdots & d_{t,2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{t,m0} & d_{t,m1} & d_{t,m2} & \cdots & 0 \end{bmatrix} \tag{A.4}$$

The EMD then attempts to solve the transportation problem of

$$\text{WORK}(P, Q, \mathbf{F}) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{t,ij} f_{ij} \tag{A.5}$$

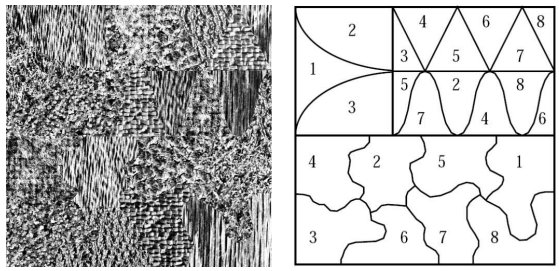


Figure A.3: Synthetic texture mosaic used (provided by USC via its website). The left image is the texture mosaic. The right image shows which texture regions belong to which texture.

subject to a number of constraints on f_{ij} as described in [114]. P and Q are the two compared histograms and $\mathbf{F}$ is the flow that minimizes the above cost. If the flow is very small the clusters are similar and are merged based on a threshold.

Last, the image is reclassified using the computed histogram profiles. Histograms that are not close to the computed histogram profiles are thresholded as outliers.

A.4 Segmentation Results

It is important that the textons contain the information necessary to accurately discriminate between different textures. We compare our texture descriptor to various other state-of-the-art descriptors by applying them to the task of segmenting a synthetic image into different textures.

The University of Southern California (USC) hosts the Brodatz texture database and also provides texture mosaics that are a number of Brodatz textures stitched together in a jigsaw-type pattern. *texmos3* was selected as the texture mosaic for benchmarking our texton descriptors. Figure A.3 shows this mosaic along with the ground truth segmentation. This mosaic has eight textures and does not contain color, which tests the descriptors' ability to discriminate textures. Four basic descriptors are tested: a 48-dimensional descriptor composed of the responses from the Leung-Malik filter bank (LM,32); a 75-dimensional descriptor of 5x5 raw RGB (RGB,5x5,32) values as used in [5] (which in effect is 25-dimensional on grayscale images); the LBP in a 3x3 neighborhood (LBP,3x3,32); and two versions of our descriptor – the 3x3 neighborhood (11-dimensional with

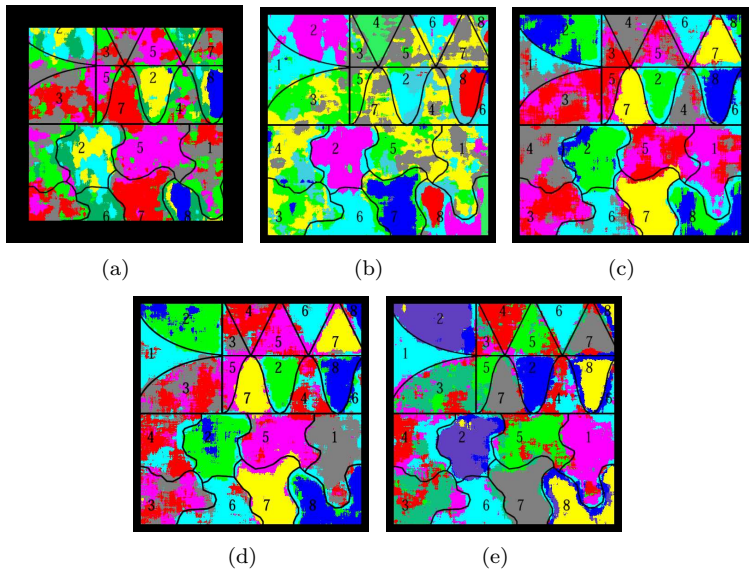


Figure A.4: Results for the synthetic texture segmentation. Each color represents a different histogram cluster. An overlay shows which regions should have homogeneous colors. (a) LM Filter, 32, (b) RGB 5x5, 32, (c) LBP 3x3,32, (d) SRI 3x3,32, (e) SRI 5x5,64.

the L,a,b color components set to zero), (SRI,3x3,32) and a 5x5 neighborhood (SRI,5x5,64) with the descriptor components still being the intensities minus the center intensity. For the test, the LM filter bank is the only one where the descriptors are not learned on the image itself. For all other descriptors, 32 textons are learned from the image itself. Our 5x5 version used 64 textons illustrating our best possible result. The lack of color information meant that more textons were needed to discriminate the textures. The second stage of clustering is then applied to give the segmentation results. It is important to note that the underlying segmentation algorithm is the same (as described in section A.3) for each of these descriptors.

Each descriptor is then scored using two scores – the detection rate and the confusion rate. The detection rate gives a measure of how much of a given texture it managed to classify correctly. The confusion rate gives a measure of how many correct versus false detections to expect. A good segmentation will have a high detection rate and a low confusion rate. For the detection rate we look only at texture regions that are entirely inside our 32x32 histogram window (so only one texture is present inside the window). This is done by eroding the borders of each texture region with a flat 16 pixel radius circle structuring

element. This gives us a maximum on the number of possible correct inliers $D_{\max,t}$ for a given texture. The cluster that takes up the most area of a given texture is chosen as the cluster that belongs to that texture. For a specific texture t, the number of correct detections is called $D_{c,t}$, and the number of false detections is $D_{f,t}$. The two scores for a specific texture are then calculated as

$$\text{Confusion Rate} = 100 \times \frac{D_{f,t}}{D_{c,t} + D_{f,t}} \tag{A.6}$$

$$\text{Detection Rate} = 100 \times \frac{D_{c,t}}{D_{\max,t}} \tag{A.7}$$

The total confusion and detection rates are simply the sums over all the eight textures.

$$\text{Total Confusion Rate} = 100 \times \frac{\sum_{t=1}^{8} D_{f,t}}{\sum_{t=1}^{8} (D_{c,t} + D_{f,t})} \tag{A.8}$$

$$\text{Total Detection Rate} = 100 \times \frac{\sum_{t=1}^{8} D_{c,t}}{\sum_{t=1}^{8} D_{\max,t}} \tag{A.9}$$

The actual segmentations obtained for each descriptor can be seen in Figure A.4. Figures A.5 and A.6 show the two scores for each of the eight textures present in the mosiac. The total confusion and detection rates are shown in Table I. The LM filter bank performs the worst, as it has higher confusion and lower detection rates than all the other descriptors. This fits with the observations in [139]. The raw intensity value descriptor also performs poorly. LBP has problems discriminating between textures 2 and 8 but is otherwise clearly better than the raw intensities and LM filter bank. Our descriptors do a much better job at discriminating between textures 2 and 8, which indicates that the intensity gradients are necessary to do this and that it is not enough to rely just on the gradient direction. All the methods find it hard to discriminate between textures 3 and 4 except the LBP, which aids it greatly in the total scores. The results for our descriptor are on average better than the other methods on this dataset. Interestingly, for our descriptors the 3x3 version actually gets a better

%	LM,32	RGB	LBP	SRI,3x3	SRI,5x5
Total Conf.	50	56	46	38	34
Total Det.	40	53	68	79	68

Table A.1: Total Rates

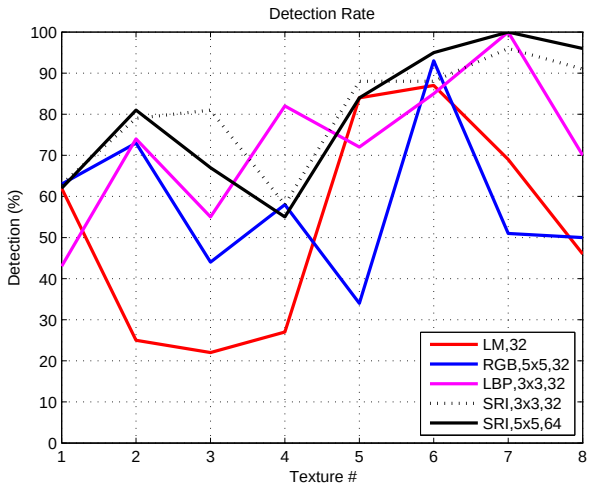


Figure A.5: Detection rate of the individual textures for the five tested feature descriptors on the artificial dataset.

total detection rate than the 5x5 version at the cost of a higher total confusion score.

The results presented here are typical for other mosaics in the synthetic dataset and have been omitted beacuse of space constraints.

A.5 Application: Path Recognition

This work has been carried out in conjunction with a larger research project entitled Learning Applied to Ground Robotics (LAGR). The project deals with outdoor navigation in unstructured environments using stereo vision. The goal is to navigate a robot autonomously to a GPS waypoint through unknown terrain at a speed of roughly $1m/s$. Many of the environments tested include small paths in the form of dirt/asphalt roads as well as more natural paths such

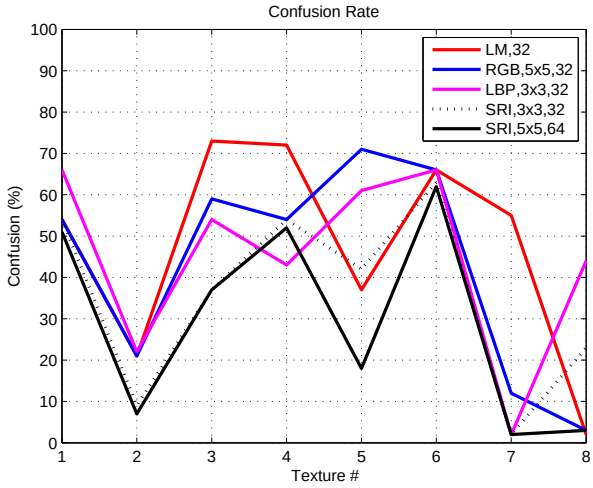


Figure A.6: Confusion rate of the individual textures for the five tested feature descriptors on the artificial dataset.

as beaten-down tracks through vegetation. Many of the paths do not have a clear signature in the 3D output of the stereo-vision sensor. We have used our segmentation algorithm to recognize these paths, using geometrical constraints from the stereo sensors (flatness, width) to find segmented regions that could be paths.

Figure A.7 illustrates a sample image (a), the texture-based segmentation (b), the disparity map computed from stereo (c), and the ground plane inliers (d). The ground plane is computed from stereo information. The objective is to determine if any of the segments in the image (b) constitute a path. We project the segmented image onto a 2D grid on the ground plane (Figure A.8 (a)) to obtain the segmentation map (Figure A.8 (b)), which is a bird's-eye view of the textures placed on the ground plane. In order to determine if a set of segments constitutes a path, we first obtain the corresponding path map (Figure A.8 (c)) and compute path statistics on the path map as described in Section A.5.2. The statistics such as the width profile help us determine if the selected segments actually constitute a path. In Section A.5.1, we describe how different segments or textures are combined to detect paths.

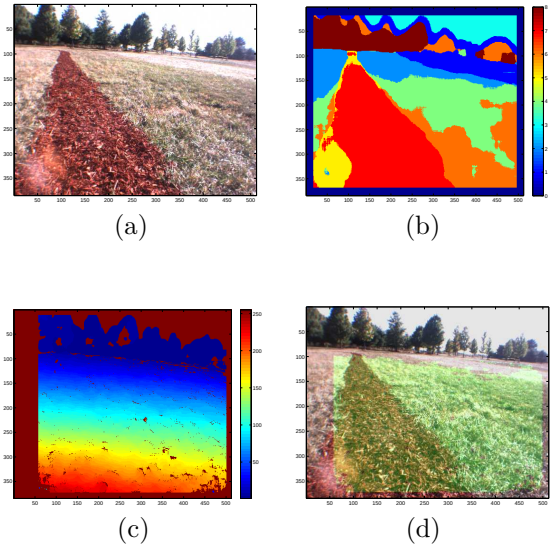


Figure A.7: Images illustrating the information used in path detection. (a) The image from one of the stereo cameras. (b) Assigned texture labels. (c) Disparity values of the pixels; red is closer, blue is farther away. (d) The inliers in the ground plane in green overlay, computed from (c).

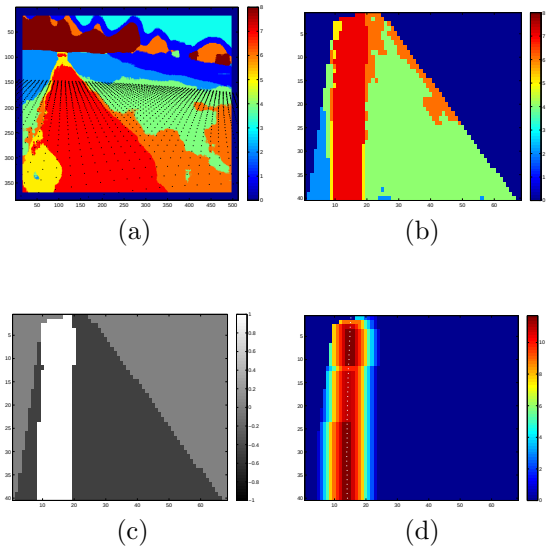


Figure A.8: Computing the path map and path statistics. (a) Ground plane uniform grid projected onto the image. (b) The texture values at the grid points (segmentation map), showing an overhead geometrical view of the textures on the ground plane. Note the main path texture (red) is now clearly a path. (c) A path map obtained by combining the red and yellow textures. The hypothesized path is in white, textures that are "not path" are in black, and unknown areas are in gray. (d) The width of the hypothesized path at each pixel. The center of the path is marked by a white dot.

A.5.1 Path Detection Using a Segmented Image

We sample the segmented image on a 2D grid on the ground plane to obtain the "segmentation map". The grid points are illustrated in Figure A.8(a) and the corresponding $N_i \times N_j$ "segmentation map", $T_{i,j}$, is illustrated in Figure A.8(b). We note that the path can be composed of a single segment or a combination of segments. For a given combination of segments, $\mathcal{S}$, the width profile can be computed directly from the segmentation map as

$$w_i^S = \sum_{j=1}^{N_j} \sum_{k \in \mathcal{S}} \delta(T_{i,j} - k). \tag{A.10}$$

The mean and deviation (A.15)-(A.16) serve as a simple means to identify segments or combinations of segments that could constitute a path. A set of segments is detected as a path if its mean width, deviation, and length lie within certain predetermined thresholds. In the LAGR experiments, for example, we considered paths whose width was in the range 0.5 m to 2 m, with deviation less than 0.15 m and length greater than 4 m and the thresholds were set accordingly. The simple width profile can be computed quickly and is also linear in the number of segments, i.e.,

$$w_i^S(\mathcal{S}_1 + \mathcal{S}_2) = w_i^S(\mathcal{S}_1) + w_i^S(\mathcal{S}_2). \tag{A.11}$$

We see from (A.11) that it is easy to compute the width profile of a path comprising multiple segments using the width profile of the component segments. The width profile computed in this manner can be used to identify single and compound segments that constitute a path for different combinations of segments. Once we obtain a list of candidate segments we can check for both row-wise and column-wise spatial coherence (next subsection). For a compound segment path consisting of the set of segments, $\mathcal{S}$, the path map is assigned as

$$p_{i,j} = \begin{cases} 0 & T_{i,j} = 0 \\ 1, & T_{i,j} \in \mathcal{S} \\ -1, & \text{otherwise.} \end{cases} \tag{A.12}$$

Figure A.8(c) illustrates the path map obtained by combining segments labeled red and yellow. Figure A.8(d) illustrates the width at each pixel ($W_{i,j}$) as well as the path center that was computed by fitting a quadratic curve.

A.5.2 Computing the Path Profile

We describe how we compute the width profile and other statistics of a $N_i \times N_j$ 2D path map such as the one in Figure A.8 (c). The path map is a 2D grid on the ground plane whose grid points are labeled as "path", "not path", or "unknown" with values as follows.

$$\text{Pixel } p_{i,j} \text{ is labeled as } \begin{cases} \text{path}, & \text{if } p_{i,j} = 1 \\ \text{not path}, & \text{if } p_{i,j} = -1 \\ \text{unknown}, & \text{if } p_{i,j} = 0 \end{cases} \tag{A.13}$$

The basic idea is to determine the existence of a consistent path by computing its width profile, i.e, its width in each row. We first determine the simple width profile (w^S), and then consider the spatial coherency in each row (w^R), and finally across the columns (w^C). We compute the simple width profile, w_i^S, which is the width of the path in the i^{th} row, as the number of pixels that are labeled as path in each row.

$$w_i^S = \sum_{j=1}^{N_j} \delta(p_{i,j} - 1) \tag{A.14}$$

The mean width and the deviation can be computed for a width profile as

$$\mu = \frac{1}{N_j} \sum_i w_i \tag{A.15}$$

$$d = \frac{1}{N_j} \sum_i |w_i - \mu| \tag{A.16}$$

While w^S is a simple means of testing if the width is consistent, it fails to take into account if the path pixels are spatially adjacent. We therefore compute a "running average" of the path width centered at each pixel using a window of length $2L + 1$, which is computed as

$$W_{i,j} = {}^{+}\sum_{k=j-L}^{j} p_{i,k} + {}^{-}\sum_{k=j+L}^{j+1} p_{i,k} \tag{A.17}$$

where

$${}^{+}\sum_{k=0}^{j} x_k = \max\left(0, {}^{+}\sum_{k=0}^{j-1} x_k + x_j\right), \text{ and} \tag{A.18}$$

$${}^{-}\sum_{k=L}^{j} x_k = \max\left(0, {}^{-}\sum_{k=L}^{j+1} x_k + x_j\right). \tag{A.19}$$

The two terms in (A.17) describe the widths on the left and right of the pixel (i, j). These can be computed recursively (A.18-A.19) and are constrained to be nonnegative. The new width profile, w_i^R, of row i is computed as

$$w_i^R = \max_j W_{i,j} \tag{A.20}$$

and takes into account spatial coherency in each row. Thus, if a row has a certain number of path pixels, its width is highest when all of them are adjacent.

We next check the spatial consistency of the path across rows. We note that $W_{i,j}$ in each row obtains the maximum value at the center of the path and we obtain the column corresponding to the maxima of $W_{i,j}$ for each row i : $j_i^{\max} = \arg_j \max W_{i,j}$. We then fit a quadratic curve to the set of points $(i, j_i^{\max})$ using RANSAC to obtain the path center in each row. The column-wise spatially coherent width profile, w_i^C is computed as the path width at the fitted path center

$$w_i^C = W_{i,j_i}, \tag{A.21}$$

where j_i is the fitted path center at row i.

A.6 Path Recognition Results

As part of the LAGR program, an independent testing group ran monthly blind tests of the perception and control software. The nine competing teams in the LAGR program were compared to a baseline system; each team was scored based on the time taken by its robot to reach the goal. Figure A.7(a) is one of the tests. Here, a path can be identified as a dirt section among the grass.

In real time tests, we run the segmentation algorithm at slightly slower than a 1 Hz rate. The perception system passes information about paths to the planner, along with other information about obstacles and freespace. Figure A.9 shows the trail amongst the bushes detected as a path. This information is then passed onto the planner. Figure A.10 shows the planner operating on the information returned by the perception algorithms. The path segmentation contributes the yellow center section, which is preferred by the planner. Using the path helps the robot to stay away from the bushes surrounding the path (where the robot wheels might get stuck). Also, since path costs are lower, the robot avoided squeezing through the open space between the bushes. This behavior is similar to that of a person, who would prefer the easy path rather than more dense terrain among the bushes.

Because of the strong geometric tests, no false positives of recognized paths have been experienced in any of the LAGR tests.

	LM,32	RGB	LBP	SRI,3x3	SRI,5x5
Time (s)	N/A	5.14	2.59	1.11	3.01

Table A.2: Segmentation Times

Figure A.9: Example of path classification. The costmap in Fig. A.10 was created by driving along this path. The yellow color indicates the detected path. Green is ground plane and blue are obstacles. The two horizontal lines are estimates for the location of the horizon.

Timings for our segmentation algorithm on a 512 by 384 color image are shown in Table A.2. The computational platform is a 2 GHz Pentium-M machine. For our K-means algorithm, the maximum number of iterations was fixed at 100. In practice, K-means converges much before that and is stopped when it has converged. It takes about 1 s to perform the two-stage online learning process, and about 150 ms to classify a 512 by 384 image using our 3x3 descriptor. The LM filters have not been implemented to work with color and so are excluded from the timings.

A.7 Conclusions

We have presented a segmentation algorithm suitable for robotic applications in outdoor environments. Our segmentation algorithm uses a compact feature descriptor in a two-stage K-means-based clustering algorithm. We have shown that our descriptor does a better job at texture segmentation than other commonly used texture descriptors. We have also applied our segmentation algorithm for recognizing natural paths in outdoor environments in real time. The approach has been demonstrated online for following natural paths on an outdoor robot. Although false positives have not been experienced in the LAGR tests, they could potentially occur if the segments happen to resemble a path geometrically. Another failure mode could occur if the path is segmented into too many

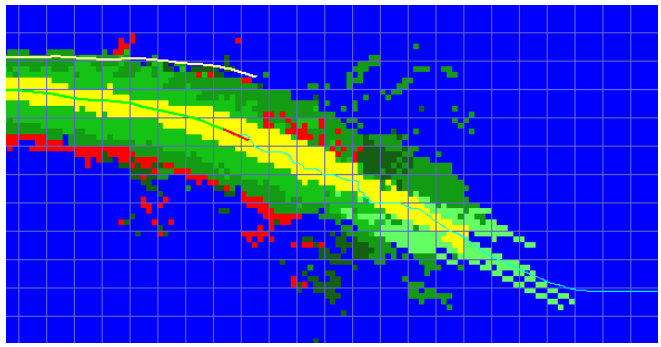


Figure A.10: Example of a costmap built by the LAGR robot and subsequently used by the planner. Blue is unknown terrain. Obstacles are shown in red. Ground plane is shown in various shades of green (with brighter colors indicating lower cost). The yellow region is the detected path. The robot position is marked with a red line. The cyan line indicates the planned trajectory. The green line indicates where the robot has driven. The super-imposed grid squares have a length of 1 m.

regions. We also need to look into the maximum angle and distance relative to the robot at which the path can be detected.

The segmentation algorithm forms a basis for performing appearance-based terrain classification. We are currently looking into building a database of commonly seen terrain types such as tall grass, sandy soil, gravel, and mulch; textons can be learned offline for each class, and then online each segment can be classified into one of these terrain types. Such terrain classification can also be performed entirely online, wherein the robot learns from its own experience. For each terrain type that the robot has been on, the robot can learn its color, texture, and navigability characteristics. Subsequently, it can predict the navigability characteristics of an unknown terrain by recognizing its color and texture. Indeed, such behaviors can make the robot appear 'intelligent'.

Paper B

CenSurE for Realtime Feature Detection and Matching

Motilal Agrawal, Kurt Konolige and Morten Rufus Blas. CenSurE for Realtime Feature Detection and Matching. *Proc. of the European Conf. on Computer Vision*, pages 102-115, Marseille, France, 2008. Published.[1]

[1]This material is based upon work supported by the United States Air Force under Contract No. FA8650-04-C-7136. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

Abstract:
We explore the suitability of different feature detectors for the task of image registration, and in particular for visual odometry, using two criteria: stability (persistence across viewpoint change) and accuracy (consistent localization across viewpoint change). In addition to the now-standard SIFT, SURF, FAST, and Harris detectors, we introduce a suite of scale-invariant center-surround detectors (CenSurE) that outperform the other detectors, yet have better computational characteristics than other scale-space detectors, and are capable of real-time implementation.

B.1 Introduction

Image matching is the task of establishing correspondences between two images of the same scene. This is an important problem in Computer Vision with applications in object recognition, image indexing, structure from motion and visual localization – to name a few. Many of these applications have real-time constraints and would benefit immensely from being able to match images in real time.

While the problem of image matching has been studied extensively for various applications, our interest in it has been to be able to reliably match two images in real time for camera motion estimation, especially in difficult off-road environments where there is large image motion between frames [67, 1]. Vehicle dynamics and outdoor scenery can make the problem of matching images very challenging. The choice of a feature detector can have a large impact in the performance of such systems.

We have identified two criteria that affect performance.

- Stability: the persistence of features across viewpoint change
- Accuracy: the consistent localization of a feature across viewpoint change

Stability is obviously useful in tracking features across frames. Accuracy of feature localization is crucial for visual odometry tasks, but keypoint operators such as SIFT typically subsample the image at higher scales, losing pixel-level precision.

Broadly speaking, we can divide feature classes into two types. *Corner detectors* such as Harris (based on the eigenvalues of the second moment matrix [45, 120]) and FAST [110] (analysis of circular arcs [109]) find image points that are well localized, because the corners are relatively invariant to change of view. Both these detectors can be implemented very efficiently and have been used in structure-from-motion systems [1, 93, 96] because of their accuracy. However, they are not invariant to scale and therefore not very stable across scale changes, which happen constantly with a moving camera. The Harris-Laplace and the Hessian-Laplace features [89] combine scale-space techniques with the Harris approach. They use a scale-adapted Harris measure [77] or the determinant of the Hessian to select the features and the Laplacian to select the scale. Supposedly, visual odometry can benefit from scale-space features, since they can be tracked for longer periods of time, and should lead to improved motion estimates from incremental bundle adjustment of multiple frames.

While we expect scale-space features to be more stable than simple corner features, are they as accurate? The answer, at least for visual odometry, is "no". The reason is that, as typically implemented in an image pyramid, scale-space features are not well localized at higher levels in the pyramid. Obviously, features at high levels have less accuracy relative to the original image. The culprit in loss of accuracy is the image pyramid. If the larger features were computed at each pixel, instead of reducing the size of the image, accuracy could be maintained. However, computing features at all scales is computationally expensive, which is why SIFT features [78], one of the first scale-space proposals, uses the pyramid – each level incurs only 1/4 the cost of the previous one. SIFT attempts to recover some of the lost accuracy through subpixel interpolation.

Our proposal is to maintain accuracy by computing features at all scales *at every pixel* in the original image. The extrema of the Laplacian across scale have been shown to be very stable [90], so we consider this operator, or more generally, extrema of a center-surround response (CenSurE, or *Cen*ter *Sur*round *E*xtrema). We explore a class of simple center-surround filters that can be computed in time independent of their size, and show that, even when finding extrema across all scales, they are suitable for real-time tasks such as visual odometry. CenSurE filters outperform the best scale-space or corner features at this task in terms of track length and accuracy, while being much faster to compute; and they are also competitive in standard tests of repeatability for large-viewpoint changes.

While the main focus of this paper is on a novel feature detector, visual odometry (and other motion estimation tasks) can benefit from matching using a descriptor that is robust to viewpoint changes. In this paper, we develop a fast variant of the upright SURF descriptor, and show that it can be used in real-time tasks.

B.1.1 Related Work

The two scale-space detectors that are closest to our work, in technique and practicality, are SIFT [78] and SURF [46]. The main differences between approaches is summarized in the table below.

	CenSurE	**SIFT**	**SURF**
Spatial resolution at scale	full	subsampled	subsampled
Scale-space operator Approximation	Laplace (Center-surround)	Laplace (DOG)	Hessian (DOB)
Edge filter	Harris	Hessian	Hessian
Rotational invariance	approximate	yes	no

The key difference is the full spatial resolution achieved by CenSurE at every scale. Neither SIFT nor SURF computes responses at all pixels for larger scales, and consquently do not detect extrema across all scales. Instead, they consider each scale octave independently. Within an octave, they subsamples the responses, and find extrema only at the subsampled pixels. At each successive octave, the subsampling is increased, so that almost all computation is spent on the first octave. Consequently, the accuracy of features at larger scales is sacrificed, in the same way that it is for pyramid systems. While it would be possible for SIFT and SURF to forego subsampling, it would then be inefficient, with compute times growing much larger.

CenSurE also benefits from using an approximation to the Laplacian, which has been shown to be better for scale selection [90]. The center-surround approximation is fast to compute, while being insensitive to rotation (unlike the DOB Hessian approximation). Also, CenSurE uses a Harris edge filter, which gives better edge rejection than the Hessian.

Several simple center-surround filters exist in the literature. The bi-level Laplacian of Gaussian (BLoG) approximates the LoG filter using two levels. [102] describes circular BLoG filters and optimizes for the inner and outer radius to best approximate the LoG filter. The drawback is that the cost of BLoG depends on the size of the filter. Closer to our approach is that of Grabner et al. [37], who describe a difference-of-boxes (DOB) filter that approximates the SIFT detector, and is readily computed at all scales with integral images [140, 76]. Contrary to the results presented in [37], we demonstrate that our DOB filters outperform SIFT in repeatability. This can be attributed to careful selection of filter sizes and using the second moment matrix instead of the Hessian to filter out responses along a line. In addition, the DOB filter is not invariant to rotation, and in this paper we propose filters that have better properties.

The rest of the paper is organized as follows. We describe our CenSurE features in detail in Section B.2. We then discuss our modified upright SURF (MU-SURF) in Section B.3. We compare the performance of CenSurE against several other feature detectors. Results of this comparison for image matching are presented in Section B.4.1 followed by results for visual odometry in Section B.4.2. Finally, Section B.5 concludes this paper.

B.2 Center Surround Extrema (CenSurE) Features

Our approach to determining accurate large-scale features demands that we compute all features at all scales, and select the extrema across scale and location. Obviously, this strategy demands very fast computation, and we use simplified bi-level kernels as center-surround filters. The main concern is finding kernels that are rotationally invariant, yet easy to compute.

B.2.1 Finding Extrema

In developing affine-invariant features, Mikolajczyk and Schmid [88] report on two detectors that seem better than others in repeatability – the Harris-Laplace and Hessian-Laplace. Mikoljczyk and Schmid note that the Harris and Hessian detectors (essentially corner detectors) are good at selecting a location within a scale, but are not robust across scale. Instead, they show that the maximum of Laplacian operator across scales gives a robust characteristic scale - hence the hybrid operator, which they define as follows: first a peak in the Harris or Hessian operator is used to select a location, and then the Laplacian selects the scale at that location.

This strategy requires computing the Hessian/Harris measure at all locations and all scales, and additionally calculating the Laplacian at all scales where there are peaks in the corner detector. In our view, the Laplacian is easier to compute and to approximate than the Hessian, as was discovered by Lowe for SIFT features. So in our approach, we compute a simplified center-surround filter at all locations and all scales, and find the extrema in a local neighborhood. In a final step, these extrema are filtered by computing the Harris measure and eliminating those with a weak corner response.

B.2.2 Bi-level Filters

While Lowe approximated the Laplacian with the difference of Gaussians, we seek even simpler approximations, using center-surround filters that are bi-level, that is, they multiply the image value by either 1 or -1. Figure B.1 shows a progression of bi-level filters with varying degrees of symmetry. The circular filter is the most faithful to the Laplacian, but hardest to compute. The other filters can be computed rapidly with integral images (Section B.2.7), with decreasing cost from octagon to hexagon to box filter. We investigate the two endpoints: octagons for good performance, and boxes for good computation.

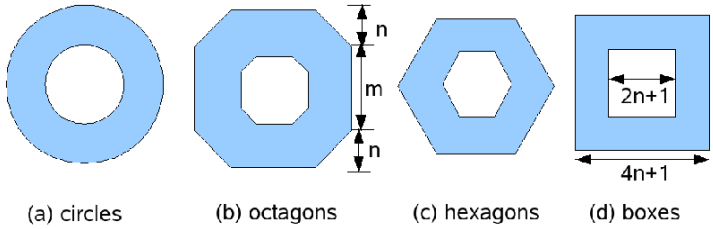


Figure B.1: Progression of Center-Surround bi-level filters. (a) circular symmetric BLoG (Bilevel LoG) filter. Successive filters (octagon, hexagon, box) have less symmetry.

B.2.3 CenSurE Using Difference of Boxes

We replace the two circles in the circular BLoG with squares to form our CenSurE-DOB. This results in a basic center-surround Haar wavelet. Figure B.1(d) shows our generic center-surround wavelet of block size n. The inner box is of size $(2n+1) \times (2n+1)$ and the outer box is of size $(4n+1) \times (4n+1)$. Convolution is done by multiplication and summing. If I_n is the inner weight and O_n is the weight in the outer box, then in order for the DC response of this filter to be zero, we must have

$$O_n(4n+1)^2 = I_n(2n+1)^2 \tag{B.1}$$

We must also normalize for the difference in area of each wavelet across scale.

$$I_n\,(2n+1)^2 = I_{n+1}\,(2\,(n+1)+1)^2 \tag{B.2}$$

We use a set of seven scales for the center-surround Haar wavelet, with block size $n = [1, 2, 3, 4, 5, 6, 7]$. Since the block sizes 1 and 7 are the boundary, the

scale	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
inner (m,n)	$(3,0)$	$(3,1)$	$(3,2)$	$(5,2)$	$(5,3)$	$(5,4)$	$(5,5)$
outer (m,n)	$(5,2)$	$(5,3)$	$(7,3)$	$(9,4)$	$(9,7)$	$(13,7)$	$(15,10)$

Table B.1: CenSurE-OCT: inner and outer octagon sizes for various scales

lowest scale at which a feature is detected corresponds to a block size of 2. This roughly corresponds to a LoG with a sigma of 1.885. These five scales cover $2\frac{1}{2}$ octaves, although the scales are linear. It is easy to add more filters with block sizes 8,9, and so on.

B.2.4 CenSurE Using Octagons

Difference of Boxes are obviously not rotationally invariant kernels. In particular, DOBs will perform poorly for 45 degrees in-plane rotation. Octagons, on the other hand are closer to circles and approximate LoG better than DOB.

In using octagons, the basic ideas of performing convolutions by inner and outer weighted additions remain the same. As in DOB, one has to find weights I_n and O_n such that the DC response is zero and all filters are normalized according to the area of the octagons.

An octagon can be represented by the height of the vertical side (m) and height of the slanted side (n) (Figure B.1(b)). Table B.1 shows the different octagon sizes corresponding to the seven scales. These octagons scale linearly and were experimentally chosen to correspond to the seven DOBs described in the previous section.

B.2.5 Non-Maximal Suppression

We compute the seven filter responses at each pixel in the image. We then perform a non-maximal suppression over the scale space. Briefly, a response is suppressed if there is a response greater (maxima case) or a response less than (minima case) its neighbors in a local neighborhood over the location and scales. Pixels that are either maxima or minima in this neighborhood are the feature point locations. We use a 3x3x3 neighborhood for our non-maximal suppression.

The magnitude of the filter response gives an indication of the strength of the feature. The greater the strength, the more likely it is to be repeatable. Weak

responses are likely to be unstable. Therefore, we can apply a threshold to filter out the weak responses.

Since all our responses are computed on the original image without subsampling, all our feature locations are localized well and we do not need to perform subpixel interpolation.

B.2.6 Line Suppression

Features that lie along an edge or line are poorly localized along it and therefore are not very stable. Such poorly defined peaks will have large principal curvatures along the line but a small one in the perpendicular direction and therefore can be filtered out using the ratio of principal curvatures. We use the second moment matrix of the response function at the particular scale to filter out these responses.

$$H = \begin{bmatrix} \sum L_x^2 & \sum L_x L_y \\ \sum L_x L_y & \sum L_y^2 \end{bmatrix} \tag{B.3}$$

L_x and L_y are the derivatives of the response function L along x and y. The summation is over a window that is linearly dependent on the scale of the particular feature point: the higher the scale, the larger the window size. Note that this is the scale-adapted Harris measure [88, 77] and is different from the Hessian matrix used by SIFT [78, 37] to filter out line responses. Once the Harris measure is computed, its trace and determinant can be used to compute the ratio of principal curvatures. We use a threshold of 10 for this ratio and a 9×9 window at the smallest scale of block size 2.

The Harris measure is more expensive to compute than the Hessian matrix used by SIFT. However, this measure needs to be computed for only a small number of feature points that are scale-space maxima and whose response is above a threshold and hence does not present a computational bottleneck. In our experience it does a better job than Hessian at suppressing line responses.

B.2.7 Filter Computation

The key to CenSurE is to be able to compute the bi-level filters efficiently at all sizes. The box filter can be done using integral images [140, 76]. An integral image I is an intermediate representation for the image and contains the sum

of gray scale pixel values of image N with height y and width x, i.e.,

$$I(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} N(x',y') \tag{B.4}$$

The integral image is computed recursively, requiring only one scan over the image. Once the integral image is computed, it it takes only four additions to calculate the sum of the intensities over any upright, rectangular area, independent of its size.

Modified versions of integral images can be exploited to compute the other polygonal filters. The idea here is that any trapezoidal area can be computed in constant time using a combination of two different *slanted* integral images, where the sum at a pixel represents an angled area sum. The degree of slant is controlled by a parameter α:

$$I_\alpha(x,y) = \sum_{y'=0}^{y} \sum_{x'=0}^{x+\alpha(y-y')} N(x',y'). \tag{B.5}$$

When $\alpha = 0$, this is just the standard rectangular integral image. For $\alpha < 0$, the summed area slants to the left; for $\alpha > 0$, it slants to the right (Figure B.2, left). Slanted integral images can be computed in the same time as rectangular ones, using incremental techniques.

Adding two areas together with the same slant determines one end of a trapezoid with parallel horizontal sides (Figure B.2, right); the other end is done similarly, using a different slant. Each trapezoid requires three additions, just as in the rectangular case. Finally, the polygonal filters can be decomposed into 1 (box), 2 (hexagon), and 3 (octagon) trapezoids, which is the relative cost of computing these filters.

B.3 Modified Upright SURF (MU-SURF) Descriptor

Previously, we have demonstrated accurate visual odometry using ZNCC for feature matching [67] (using a 11×11 region). However, it is relatively sensitive to in-plane rotations (roll), larger changes in perspective, and inaccuracies in keypoint localization. The problems related to rolls and perspective changes become more significant as the region size increases. We have therefore decided to switch to an upright SURF type descriptor [46].

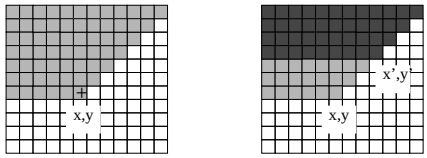


Figure B.2: Using slanted integral images to construct trapezoidal areas. Left is a slanted integral image, where the pixel x, y is the sum of the shaded areas; α is 1. Right is a half-trapezoid, from subtracting two slanted integral image pixels.

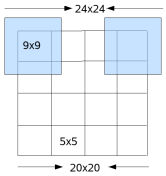


Figure B.3: Regions and subregions for MU-SURF descriptor. Each subregion (in blue) is 9x9 with an overlap of 2 pixels at each boundary. All sizes are relative to the scale of the feature s.

The SURF descriptor builds on from the SIFT descriptor by encoding local gradient information. It uses integral images to compute Haar wavelet responses, which are then summed in different ways in 4×4 subregions of the region to create a descriptor vector of length 64.

As pointed out by David Lowe [78], "it is important to avoid all boundary effects in which the descriptor abruptly changes as a sample shifts smoothly from being within one histogram to another or from one orientation to another". The SURF descriptor [46] weighs the Haar wavelet responses using a Gaussian centered at the interest point. This single weighting scheme gave poor results and we were unable to recreate the SURF descriptor results without accounting for these boundary effects.

To account for these boundary conditions, each boundary in our descriptor has a padding of $2s$, thereby increasing our region size from $20s$ to $24s$, s being the scale of the feature. The Haar wavelet responses in the horizontal (d_x) and vertical (d_y) directions are computed for each 24×24 point in the region with filter size $2s$ by first creating a summed image, where each pixel is the sum of a region of size s. The Haar wavelet output results in four fixed-size d_x,d_y, $|d_x|$,$|d_y|$ images that have the dimensions 24×24 pixels irrespective of the scale.

Each d_x,d_y, $|d_x|$,$|d_y|$ image is then split into 4×4 square overlapping subregions of size 9×9 pixels with an overlap of 2 pixels with each of the neighbors. Figure fig:descriptor shows these regions and subregions. For each subregion the values are then weighted with a precomputed Gaussian ($\sigma_1 = 2.5$) centered on the subregion center and summed into the usual SURF descriptor vector for each subregion: $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Each subregion vector is then weighted using another Gaussian ($\sigma_2 = 1.5$) defined on a mask of size

4×4 and centered on the feature point. Like the original SURF descriptor, this vector is then normalized.

The overlap allows each subregion to work on a larger area so samples that get shifted around are more likely to still leave a signature in the correct subregion vectors. Likewise, the subregion Gaussian weighting means that samples near borders that get shifted out of a subregion have less impact on the subregion descriptor vector.

From an implementation point of view the dynamic range of the vector was small enough that the end results could be scaled into C++ shorts. This allows for very fast matching using compiler vectorization.

CenSurE features themselves are signed based on their being dark or bright blobs. This is similar to SURF and can also be used to speed up the matching by only matching bright features to bright features and so forth.

We have compared the performance of MU-SURF with U-SURF for matching and found them to be similar. As will be pointed out in Section B.4.3, our implementation of MU-SURF is significantly faster than U-SURF. It is unclear to us as to why MU-SURF is so much faster. We are currently looking into this.

B.4 Experimental Results

We compare CenSurE-DOB and CenSurE-OCT to Harris, FAST, SIFT, and SURF feature detectors for both image matching and visual odometry. Results for image matching are presented in Section B.4.1 and VO in Section B.4.2.

B.4.1 Image Matching

For image matching, we have used the framework of [90] to evaluate repeatability scores for each detector on the graffiti and boat sequences.[2] We have used the default parameters for each of these detectors. In addition, since each of these detectors has a single value that represents the strength of the feature, we have chosen a strength threshold such that each of these detectors results in the same number of features in the common overlapping regions. Figure B.4 (a) & (b) shows a plot of the detector repeatability and number of correspondences for each detector using 800 features and an overlap threshold of 40% for the graffiti

[2]available from `http://www.robots.ox.ac.uk/~vgg/research/affine/`

sequence. For Harris and FAST, the scale of all detected points was assumed to be the same and set at 2.0.

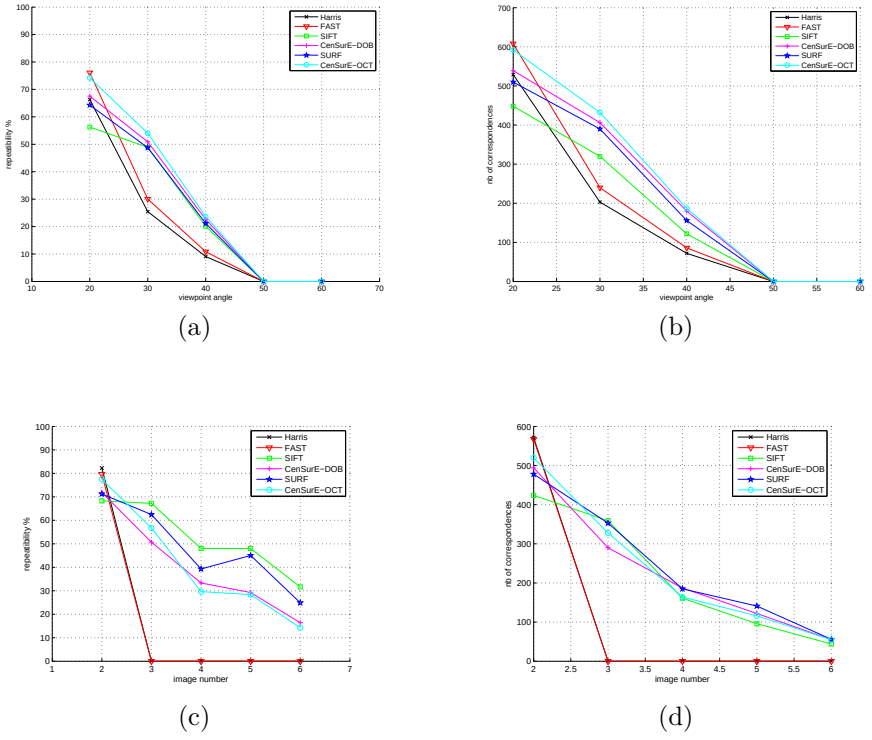


Figure B.4: Repeatability and number of correspondences for different detectors for the graffiti and boat sequences. The number of features is the same for each detector. (a) & (b) graffiti sequence. (c) & (d) boat sequence.

Both versions of CenSurE are better than SIFT or SURF, although for large viewpoint changes, the differences become only marginal. As can be expected, CenSurE-OCT does better than CenSurE-DOB.

The *boat* sequence is more challenging because of large changes in rotation and zoom. Figure B.4 (c) & (d) shows the detector performance for this sequence for 800 features. On this challenging sequence, CenSurE performs slightly worse than either SIFT or SURF, especially for the larger zooms. This can be attributed to CenSurE's non-logarithmic scale sampling. Furthermore, CenSurE filters cover only $2\frac{1}{2}$ octaves and therefore has less degree of scale-invariance for large scale changes.

To evaluate the matching performance, we used our MU-SURF descriptor for each of those detectors and matched each detected point in one image to the one with the lowest error using Euclidean distance. A correspondence was deemed as matched if the true match was within a search radius r of its estimated correspondence. Note that this is a different criterion than considering overlap error and we have chosen this because this same criterion is used in visual odometry to perform image registration. Figure B.5 shows the percentage of correct matches as a function of search radius when the number of features is fixed to 800.

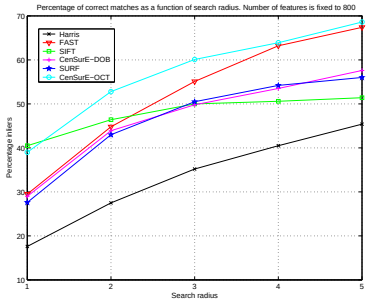


Figure B.5: Percentage of correct matches as a function of search radius

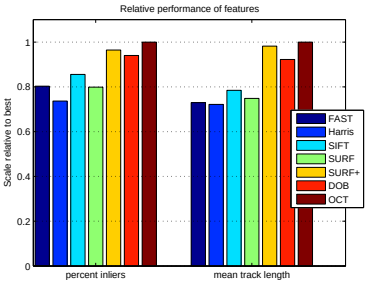


Figure B.6: Basic performance of operators in the VO dataset.

B.4.2 Visual Odometry

We evaluate the performance of CenSurE for performing visual odometry in challenging off-road environments. Because there can be large image motion between frames, including in-plane rotations, the tracking task is difficult: essentially, features must be re-detected at each frame. As usual, we compare our method against Harris, FAST, SIFT, and SURF features. Note that this is a test of the *detectors*; the same MU-SURF descriptor was used for each feature.

The Visual Odometry (VO) system derives from recent research by the authors and others on high-precision VO [67, 1] using a pair of stereo cameras. For each new frame, we perform the following process.

1. Distinctive features are extracted from each new frame in the left image. Standard stereo methods are used to find the corresponding point in the right image.
2. Left-image features are matched to the features extracted in the previous frame using our descriptor. We use a large area, usually around 1/5 of the

image, to search for matching features.

3. From these uncertain matches, we recover a consensus pose estimate using a RANSAC method [32]. Several thousand relative pose hypotheses are generated by randomly selecting three matched non-collinear features, and then scored using pixel reprojection errors.

4. If the motion estimate is small and the percentage of inliers is large enough, we discard the frame, since composing such small motions increases error. A kept frame is called a *key frame*. The larger the distance between key frames, the better the estimate will be.

5. The pose estimate is refined further in a sparse bundle adjustment (SBA) framework [29, 135].

The dataset for this experiment consists of 19K frames taken over the course of a 3 km autonomous, rough-terrain run. The images have resolution 512x384, and were taken at a 10 Hz rate; the mean motion between frames was about 0.1m. The dataset also contains RTK GPS readings synchronized with the frames, so ground truth to within about 10 cm is available for gauging accuracy.

We ran each of the operators under the same conditions and parameters for visual odometry, and compared the results. Since the performance of an operator is strongly dependent on the number of features found, we set a threshold of 400 features per image, and considered the highest-ranking 400 features for each operator. We also tried hard to choose the best parameters for each operator. For example, for SURF we used doubled images and a subsampling factor of 1, since this gave the best performance (labeled "SURF+" in the figures).

The first set of statistics shows the raw performance of the detector on two of the most important performance measures for VO: the average percentage of inliers to the motion estimate, and the mean track length for a feature (Figure B.6). In general, the scale-space operators performed much better than the simple corner detectors. CenSurE-OCT did the best, beating out SURF by a small margin. CenSurE-DOB is also a good performer, but suffers from lack of radial symmetry. Surprisingly, SIFT did not do very well, barely beating Harris corners.

Note that the performance of the scale-space operators is sensitive to the sampling density. For standard SURF settings (no doubled image, subsampling of 2) the performance is worse than the corner operators. Only when sampling densely for 2 octaves, by using doubled images and setting subsampling to 1, does performance approach that of CenSurE-OCT. Of course, this mode is much more expensive to compute for SURF (see Section B.4.3).

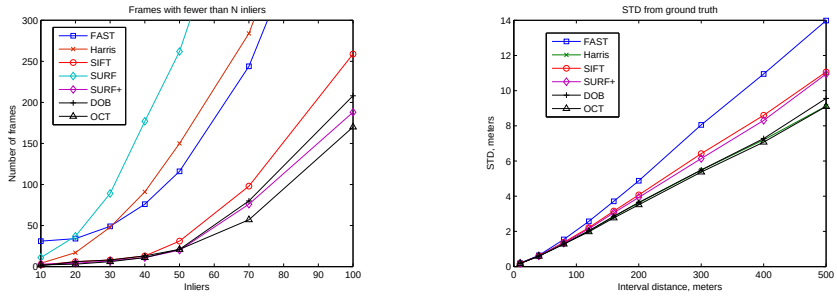


Figure B.7: Accuracy statistics. Left: number of frames with inliers less than a certain amount, out of 19K frames. For example, FAST and Harris both have around 50 frames with fewer than 30 inliers. Right: standard deviation from ground truth, over trajectories of varying length.

The question to ask is: do these performance results translate into actual gains in accuracy of the VO trajectory? We look at two measures of accuracy, the number of frames with low inlier counts, and the deviation of the VO trajectory from ground truth (Figure B.7). The graph at the left of the figure can be used to show how many frames are not matched, given a threshold for inliers. For example, we typically use 30 inliers as a cutoff: any frames with fewer matches are considered to have bad motion estimates. With this cutoff, SIFT, SURF+, OCT, and DOB all have less than 10 missed frames, while Harris and FAST have around 50. To show the influence of low-resolution localization, standard SURF does very poorly here, as we expect from the previous performance graph.

Finally, we looked at the deviation of the VO estimates from ground truth, for different trajectory lengths. At every 10 key frames along the VO trajectory, we compared a trajectory of length N against the corresponding ground truth, to give a dense sampling (about 1000 for each trajectory length). The standard deviation is a measure of the goodness of the VO trajectory. Here, OCT, DOB and Harris were all about equivalent, and gave the best estimates. Although Harris does not do well in getting large numbers of inliers for difficult motions, it is very well localized, and so gives good motion estimates. SIFT and SURF+ give equivalent results, and are penalized by their localization error.

Overall, CenSurE-OCT gives the best results in terms of accurate motion estimates, and misses very few frames. Harris does very well in accuracy of motion, but misses a large number of frames. SURF+ is a reasonable performer in terms of missed frames, but is not as accurate as the CenSurE or Harris features.

detector

SURF+	SURF-1	SIFT	SURF	OCT	DOB	Harris
3408	292	304	75	23	17	10

descriptor

U-SURF	MU-SURF
308	16

Table B.2: Time in milliseconds for different feature detectors and descriptors

B.4.3 Timing Results

Timing results for our CenSurE and MU-SURF implementations on an Intel Pentium-M 2 GHz machine for a 512×384 image are presented in Table 2. For comparison, SURF timings based on the original author's implementations[3] (on the same computational platform and on the same images) are also included.

SURF has default parameters (no doubled image, subsampling of 2), whereas SURF-1 has subsampling set to 1, and SURF+ is SURF-1 with a doubled image. For the descriptor, both U-SURF and MU-SURF are given the same features (about 1000 in number).

For VO the best performance is with SURF+. In this case, CenSurE-OCT yields more than a hundred-fold improvement in timing. Our MU-SURF is also more than twenty times faster than U-SURF. It is clear that feature detection using CenSurE features and matching using MU-SURF descriptors can be easily accomplished in real time.

B.5 Conclusion

We have presented two variants of center-surround feature detectors (CenSurE) that outperform other state-of-the-art feature detectors for image registration in general and visual odometry in particular. CenSurE features are computed at the extrema of the center-surround filters over multiple scales, using the original image resolution for each scale. They are an approximation to the scale-space Laplacian of Gaussian and can be computed in real time using integral images. Not only are CenSurE features efficient, but they are distinctive, stable and repeatable in changes of viewpoint. For visual odometry, CenSurE features result in longer track lengths, fewer frames where images fail to match, and better motion estimates.

[3]available from `http://www.vision.ee.ethz.ch/~surf/download.html`

We have also presented a modified version of the upright SURF descriptor (MU-SURF). Although the basic idea is same as the original SURF descriptor, we have modified it so as to handle the boundaries better, and it is also faster. It has been our experience that MU-SURF is well suited for visual odometry and performs much better than normalized cross-correlation without much computational overhead.

CenSurE is in constant use on our outdoor robots for localization; our goal is to ultimately be able to do visual SLAM in real time. Toward this end, we are exploiting CenSurE features to recognize landmarks and previously visited places in order to perform loop closure.

Paper C

Mapping, Navigation, and Learning for Off-Road Traversal

Kurt Konolige, Motilal Agrawal, Morten Rufus Blas, Robert C. Bolles, Brian Gerkey, Joan Sola, Aravind Sundaresan. Mapping, Navigation, and Learning for Off-Road Traversal. *J. of Field Robotics*, pages 88-113, 2009. Published.[1]

[1]This material is based upon work supported by the United States Air Force under Contract No. FA8650-04-C-7136. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

Abstract:
The challenge in the DARPA Learning Applied to Ground Robots (LAGR) project is to autonomously navigate a small robot using stereo vision as the main sensor. During this project, we demonstrated a complete autonomous system for off-road navigation in unstructured environments, using stereo vision as the main sensor. The system is very robust – we can typically give it a goal position several hundred meters away, and expect it to get there. In this paper we describe the main components that comprise the system, including stereo processing, obstacle and freespace interpretation, long-range perception, online terrain traversability learning, visual odometry, map registration, planning, and control. At the end of three years, the system we developed outperformed all 9 other teams in final blind tests over previously-unseen terrain.

C.1 Introduction

The DARPA LAGR project began in Spring 2005 with the ambitious goal of achieving vision-only autonomous traversal of off-road terrain. Further, the participating teams were to be tested "blind" – sending in code to be run on a robot at a remote, unseen site. The hope was that by using learning algorithms developed by the teams, significant progress could be made in robust navigation in difficult off-road environments, where tall grass, shadows, deadfall, and other obstacles predominate. The ultimate goal was to achieve better than 2x performance over a Baseline system already developed at the National Robotics Engineering Center (NREC) in Pittsburgh. All participant teams used the same robotic hardware provided by NREC (Figure C.1(a)); testing was performed by an independent team on a monthly basis, at sites in Florida, New Hampshire, Maryland, and Texas.

Although work in outdoor navigation has preferentially used laser rangefinders [91, 12, 40], LAGR uses stereo vision as the main sensor. One characteristic of the vision hardware is that depth perception is good only at fairly short range – its precision deteriorates rapidly after 7m or so. Even where good stereo information is available, it is often impossible to judge traversability on the basis of 3D form. For example, tall grass that is compressible can be traversed, but small bushes cannot, and they might have similar 3D signatures. The robots would often slip on sand or leaves, and be unable to climb even small grades if they were slippery. These conditions could not be determined even at close range with stereo vision.

Another area that the testing team was keen on developing was the ability of the robots to make decisions at a distance. Many of the tests had extensive cul-de-sacs, dead ends, or paths that initially led towards the goal but then turned away. Here, the robot could not rely on local information to find a good way out. The expectation was that the teams would cope with such situations using long-range vision sensing, that is, be able to tell from the appearance of the terrain whether it was traversable or not.

Throughout the project life, we evaluated the potential of learning methods and appearance-based recognition. The emphasis was always on general methods that would work well in all situations, not just artificial ones designed to test a particular ability, like bright orange fencing that could easily be recognized by its distinctive color. In the end, the most useful and novel technique we developed was an online method for path-finding based on color and texture. While we also developed algorithms for classifying obstacles at a distance, they did not work reliably enough to be included in a final system.

In addition to appearance-based learning, we had to build improved algorithms for many different aspects of vision-based offroad navigation. The paragraphs below summarize the methods that distinguished the SRI system, and which contributed to its overall performance.

Online Color and Texture Segmentation

It became clear from the early stages of the project that color-only methods for recognizing vegetation or terrain were not sufficient. We concentrated on developing fast combined color/texture methods that could be used online to learn segmentations of the image. These methods advance state-of-the-art in appearance-based segmentation, and are the key to our online path-finding method. They reliably finds paths such as the one in Figure C.1(b), even when the particular appearance of the path is new.

(a) LAGR Robot (b) View from the robot cameras

Figure C.1: (a) LAGR robot with two stereo sensors. (b) Typical outdoor scene as a montage from the left cameras of the two stereo devices.

Precisely Registered Maps

If the robot's reconstruction of the global environment is faulty, it cannot make good plans to get to its goal. After noticing navigation failures from the very noisy registration provided by GPS, we decided to give high priority to precise registration of local map information into a global map. Here, we developed realtime visual odometry (VO) methods that are more precise than existing ones, while still being computable at frame rates. To our knowledge, this is the first use of VO as the main registration method in an autonomous navigation system. VO enabled us to learn precise maps during a run, and so escape efficiently from cul-de-sacs. In the last stage of the project, we also discovered that the precision of VO made it possible to reuse maps from a previous run, thereby avoiding problem areas completely. This *run-to-run learning*, or map re-use, was unique among the teams, and on average halved the time it took to complete a course.

Efficient Planner and Controller

The LAGR robot was provided with a "baseline" system that used implementations of D* [127] for global planning and Dynamic Window Approach (DWA) [34] for local control. These proved inadequate for realtime control – for example, the planner could take several seconds to compute a path. We developed an efficient global planner based on previous gradient techniques [63], as well as a novel local controller that takes into account robot dynamics, and searches a large space of robot motions. These technqiues enabled the robot to compute optimal global paths at frame rates, and to average 85% of its top speed over most courses.

It is hard to over-emphasize the contribution of consistent map construction and map re-use. Without well-registered maps, the robot would often spend large amounts of time getting cornered as badly-remembered obstacles filled in open spaces, or it would re-enter dead-end areas that had shifted in the map. Because the testing team emphasized cul-de-sacs and garden path scenarios, it was critical to have accurate representations of areas that were no longer within the short stereo range. As it turned out, an accurate map was a more robust way to deal with these scenarios than unreliable long-range sensing. Further, once the map was constructed, map re-use led to very efficient runs: if you memorize the route, there's no need to repeat your mistakes. While the idea is simple, the execution was difficult, requiring very precise localization based on visual odometry.

At the end of the project, the teams were tested in a series of courses (Tests 25–27) with a variety of challenges (see Section C.6.2). We choose these last tests because our system was complete, having just added the map re-use feature.

Over these tests, we averaged about 4x the score of Baseline, the best of any team. In each of these tests, our score beat or tied the best other team; and in the aggregate, we scored 60% higher than the best other team. These results validate the applicability of our techniques to autonomous navigation.

In this paper we show how we built a system for autonomous off-road navigation that embodies the methods described above, and in particular performs online path learning and run-to-run map learning to increase its performance. In the following sections, we first discuss local map creation from visual input, with a separate section on learning color models for paths and traversable regions. Then we examine visual odometry and registration in detail, and show how consistent global maps are created and reused. The next section discusses the global planner and local controller. Finally, we present performance results for the last series of tests at the end of the project.

C.1.1 Related work

There has been an explosion of work in mapping and localization (SLAM), most of it concentrating on indoor environments [41, 73]. Much of the recent research on outdoor navigation has been driven by DARPA projects on mobile vehicles [12]. The sensor of choice is a laser rangefinder, augmented with monocular or stereo vision. In much of this work, high-accuracy GPS is used to register sensor scans; exceptions are [40, 91]. In contrast, we forego laser rangefinders, and explicitly use image-based registration to build accurate maps. Other approaches to mapping with vision are [108, 124], although they are not oriented towards realtime implementations. Obstacle detection using stereo has also received some attention [108].

Visual odometry systems use structure-from-motion methods to estimate the relative position of two or more camera frames, based on matching features between those frames. There have been a number of recent approaches to visual odometry [97, 82, 53], including motion estimation on the Mars vehicles [86]. Other teams in LAGR also developed visual odometry systems to aid in navigation [47]. Our system [67, 1, 2] is most similar to the recent work of Mouragnon et al. [93] and Sunderhauf et al. [128], which exploit bundle adjustment techniques to obtain increased precision. One difference is the introduction of a new, more stable keypoint detector, and the integration of an IMU to maintain global pose consistency. Our system is also distinguished by realtime implementation and high accuracy using a small baseline in realistic terrain. It has been in regular use in demonstrations for over two years as the primary mode of localization and map registration. In addition, the system has been tested over trajectories of up to 9km with a ground truth RTK-GPS dataset, and has

achieved accuracies of under 1% error (see Section C.3.4).

Our segmentation algorithm uses a compact descriptor to represent color and texture. In a seminal paper, Leung and Malik [74] showed that many textures could be represented and re-created using a small number of basis vectors extracted from the local descriptors; they called the basis vectors *textons*. While Leung and Malik used a filter bank, later Varma and Zisserman [139] showed that small local texture neighborhoods may be better than using large filter banks. In addition, a small local neighborhood vector can be much faster to compute than multichannel filtering such as Gabor filters over large neighborhoods.

Our planning approach is an enhanced reimplementation of the gradient technique [63], which computes a global navigation function over the cost map. A similar approach is used in wavefront planning [71], although wavefront planners usually minimize Manhattan or diagonal distance, whereas we minimize Euclidean distance. Level sets [60] offer an equivalent method for computing paths that minimize Euclidean distance. The underlying computation for all such planners is a variation on dynamic programming [10]. For reasons of efficiency, our planner treats the robot as a holonomic cylinder with no kinodynamic constraints. These constraints could be incorporated into the planner by use of sampling-based algorithms such as rapidly-exploring random trees (RRTs) [72].

We enforce kinodynamic constraints in our local controller. Control algorithms such as DWA [34] compute local controls by first determining a target trajectory in position or velocity space (usually a circular arc or other simple curve), then inverting the robot's dynamics to find the desired velocity commands that will produce that trajectory. We instead explore the control space directly, and simulate and evaluate the resulting trajectories, in a manner reminiscent of the controller used in the RANGER system [57], with the key differences being the definition of the state space and the trajectory evaluation function. The Stanley controller [132] also rolls out and evaluates possible trajectories, but divides them into two categories ("nudges" and "swerves"), based on their expected lateral acceleration. Howard et al. [48] present a more general approach to constraining the search for controls by first sampling directly in the vehicle's state space.

C.2 Local map construction

The object of the local map algorithms is to determine, from the visual information, which areas are freespace and which are obstacles for the robot: the *local*

map. Note that this is not simply a matter of geometric analysis – for example, a log and a row of grass may have similar geometric shapes, but the robot can traverse the grass but not the log.

Figure C.2(a) is an outline of visual processing, from image to local map. There are four basic trajectories. From the stereo disparity, we compute a nominal ground plane, which yields free space near the robot. We also analyze height differences from the ground to find obstacles. Via the technique of sight lines we can infer freespace to more distant points. Finally, from color, texture and path analysis, coupled with the ground plane, we determine paths and traversability of the terrain.

All of the processing of local cost maps, with the exception of the color/texture learning, takes place very efficiently. We can run the full algorithm, including stereo computation and obstacle detection, in under 70 ms (15 Hz), enabling very quick response to new features in the environment.

C.2.1 Stereo analysis and ground plane extraction

We use a fast stereo algorithm [65] to compute a disparity image at 512x384 resolution in less than 40 ms[2] (Figure C.3(a)). In typical outdoor scenes, it is possible to achieve very dense stereo results, The high resolution gives very detailed 3D information for finding the ground plane and obstacles. Each disparity image point $[u, v, d]^\top$ corresponds to a 3D point in the robot's frame $([x, y, z, w]^\top = R[u, v, d, 1]^\top$ in homogenous coordinates, where R is the reprojection matrix [68]. The matrix multiplication is done for each disparity point as part of stereo processing.

Output from the stereo process is used in a number of ways – the diagram in Figure C.2(b) summarizes them. Most of our analysis is biased towards finding freespace, especially in areas that are further from the robot. This strategy stems from the high cost of seeing false obstacles, closing off promising paths for the robot.

The most important geometric analysis is finding the ground plane. Although it is possible to detect obstacles using local variation in height [43], using a ground plane simplifies processing and yields more stable results. To extract a ground plane, we use a RANSAC technique [32], choosing sets of 3 noncollinear points. Hypothesized planes are ranked by the number of points that are close to the plane. Figure C.3 shows an example, with a green overlay indicating the inliers. Points that lie too high above the ground plane, but lower than the robot's height, are labeled as obstacles. This method is extremely simple, but has proven to work well in practice, even when the ground has modest dips and rises; one reason is that it only looks out to 6m around the robot. A more sophisticated analysis would break the ground plane into several segments or model more complex shapes. To compute the ground plane efficiently, we subsample the image to 10K points, and apply the RANSAC algorithm above. Hypothesizing a plane and finding inliers are simple matrix operations, and typical running time is 5ms.

To find obstacles, a typical algorithm would cluster the 3D points into grid cells on the ground plane. Then, by analyzing the points in each cell, it would be declared ground, obstacle, or unknown. The problem is that there is a geometric mismatch between the 3D cells and the image point density: as the points projected on further cells become sparse, it is difficult to determine obstacle boundaries. Instead, we find obstacles using algorithms in the disparity plane. The ground plane is projected back onto the disparity points, which are shown in green in Figure C.3(b). The disparity image is divided into thin columns, and each column is traversed from the bottom of the image (red line in the

[2] All processing times referenced in this paper are on a 2 GHz Intel CPU.

image). When the ground plane ends and enough disparity points are found, there must be an obstacle at that endpoint in the ground plane. In practice this technique is much faster and more reliable than to ground plane projection, typically consuming only 5ms.

C.2.2 Sight lines

Although we cannot precisely locate obstacles past 6-8m, we can determine if there is freespace, using the following observation. Consider the interpreted image of Figure C.3(c). There is a path that goes around the bushes and extends out a good distance. The ground plane extends over most of this area, and then ends in a distant line of trees. The trees are too far to place with any precision, but we can say that *there is no obstacle along the line of sight to the trees*. Given a conservative estimate for the distance of the trees, we can add freespace up to this estimate; typically we would add freespace to at most 25m. The computation of sight lines is most efficiently accomplished in disparity space, by finding columns of ground plane pixels that lead up to a distant obstacle (red line in Figure C.3(b)). Note that the example sight line follows the obvious path out of the bushes.

C.2.3 Learning color and texture models

Our learning algorithm uses an online unsupervised segmentation algorithm that uses color and texture to group and cluster similar regions. This segmented image is then used to learn a color and texture model for *path-like* regions in outdoor images. Our segmentation algorithm is based on textons [74] and is accomplished in two stages. The main design issues have been speed (for real-time segmentation) and robustness (to minimize false-positives).

In the first stage, we cluster color and texture vectors over small local neighborhoods to find a small set of basis vectors (also known as textons) that characterize different scene textures. For reasons of speed, this vector should be as compact as possible without losing appearance characteristics of the region. [7] use the three color components over a 5×5 region centered on each pixel. This results in a large 75 dimensional feature vector for each pixel. For speed, we use a 3×3 neighborhood and use the pixel intensity gradients between the surrounding pixels relative to the center pixel to represent texture compactly. We also augment this eight dimensional feature vector with the three dimensional color vector of the center pixel in the CIELAB color space. CIELAB has the property that colors are perceptually uniform. The resulting 11 dimensional

color/texture feature vector is very compact and we have found that it still retains the crucial appearance properties to discriminate and segment regions. A detailed comparison with other representations can be found in [17].

In the second stage, we cluster histograms of these textons over larger 32×32 regions (which is dependent on the scale of the image) to find more coherent regions with the same mixture of textons using k-means as our clustering algorithm. These histograms can be constructed efficiently (irrespective of window size) using integral images [140]. The algorithm is generally set to over-segment the image slightly as in our case over-segmentation can be dealt with by a subsequent geometrical analysis of the image. Under-segmentation is harder to deal with as considerable information is lost. The number of clusters was set to 8 in the second stage. There are other ways of doing the second stage which provides better results by specifically dealing with boundary conditions but they are much slower and are thus currently ill-suited for our real-time needs (See graph-cut [84] and level-sets [75]).

C.2.3.1 Segmentation results

The University of Southern California (USC) hosts the Brodatz texture database and also provides texture mosaics that are a number of Brodatz textures stitched together in a jigsaw-type pattern. *texmos3* was selected as the texture mosaic for benchmarking our texton descriptors. Figure C.4 shows this mosaic along with the ground truth segmentation. This mosaic has eight textures and does not contain color, which tests the descriptors' ability to discriminate textures. Four basic descriptors are tested: a 48-dimensional descriptor composed of the responses from the Leung-Malik [74] filter bank (LM,32); a 75-dimensional descriptor of 5×5 raw RGB (RGB,5×5,32) values as used in [7] (which in effect is 25-dimensional on gray-scale images); the local binary pattern [81] (LBP) in a 3×3 neighborhood (LBP,3×3,32); and two versions of our descriptor – the 3×3 neighborhood (11-dimensional with the L,a,b color components set to zero), (SRI,3×3,32) and a 5×5 neighborhood (SRI,5×5,64) with the descriptor components still being the intensities minus the center intensity. For the test, the LM filter bank is the only one where the descriptors are not learned on the image itself. For all other descriptors, 32 textons are learned from the image itself. Our 5×5 version used 64 textons illustrating our best possible result. The lack of color information meant that more textons were needed to discriminate the textures. The second stage of clustering is then applied to give the segmentation results. It is important to note that the underlying segmentation algorithm is the same for each of these descriptors.

The actual segmentations obtained for each descriptor can be seen in Figure

%	LM,32	RGB	LBP	SRI,3×3	SRI,5×5
Total confusion rate	50	56	46	38	34
Total detection rate	40	53	68	79	68

Table C.1: Total confusion and detection rates for different types of descriptors.

C.5. Each descriptor is then scored using two scores – the detection rate and the confusion rate. The detection rate gives a measure of how much of a given texture it managed to classify correctly. The confusion rate gives a measure of how many correct versus false detections to expect. A good segmentation will have a high detection rate and a low confusion rate.

The total confusion and detection rates are shown in Table C.1. The LM filter bank performs the worst, as it has higher confusion and lower detection rates than all the other descriptors. The raw intensity value descriptor also performs poorly. LBP has problems discriminating between textures 2 and 8 but is otherwise clearly better than the raw intensities and LM filter bank. Our descriptors do a much better job at discriminating between textures 2 and 8, which indicates that the intensity gradients are necessary to do this and that it is not enough to rely just on the gradient direction. All the methods find it hard to discriminate between textures 3 and 4 except the LBP, which aids it greatly in the total scores. The results for our descriptor are on average better than the other methods on this dataset. Interestingly, for our descriptors the 3×3 version actually gets a better total detection rate than the 5×5 version at the cost of a higher total confusion score.

C.2.3.2 Learning paths

We use our segmentation algorithm to learn and subsequently recognize both natural and man-made *paths* in outdoor images. Paths are characterized by their color, texture and geometrical properties. Training samples for a path can come from tele-operation or from a priori knowledge that the robot is starting on a path. The robot can also search for paths by trying to identify image clusters that have the geometry of a path. We deal with over-segmentation of the path (wherein a path is split into multiple segments due to possibly differing textures) by grouping multiple segments based on their overall geometry. We compute geometrical properties of the path that could be composed of a single or multiple segments. The properties include width, length and spatial continuity of the path in order to verify if it geometrically resembles a path. These geometrical properties are computed in 3D using the ground plane information available from the stereo-cameras and are hence not affected by the perspec-

tive projection. We assume a fixed path width (but allow a certain deviation from this assumption). Once a path is identified, the robot learns the texton histograms of the component segments as a model for the path. This model can be used to identify even paths that are partially outside the field of view by allowing the path to end prematurely at the image boundary.

For classification, each pixel is first labeled using shortest Euclidean distance on the color/texture vector at this pixel to the clustered textons. Likewise histograms of textons at each pixel are classified using Euclidean distance to the clustered histograms. The texton histograms from our training provide positive examples (the histograms that belong to a path) as well as negative examples (the histograms that don't belong to a path). This helps prevent false-positives. A final geometrical analysis of the labeled histograms makes sure that potential path regions have the right geometry.

The learning process runs at 1Hz for training on a single image and is typically performed at the beginning of a run (although it could be performed at regular intervals to update the path model). Classification based on the learned model runs at around 5Hz. Figure C.6 shows the various steps of our algorithm on one of our test runs. The path between bushes is identified in yellow in Figure C.6(d). Details on this algorithm can be found in [17].

C.2.4 Results of local map construction

The combined visual processing results in local maps that represent traversability with a high degree of fidelity. Figure C.7 shows the results of an autonomous run of about 130m, over a span of 150 seconds. We used offline learning of mulch paths on a test site, then used the learned models on the autonomous run. The first part of the run was along a mulch path under heavy tree cover, with mixed sunlight and deep shadows. Cells categorized as path are shown in yellow; black is freespace. Obstacles are indicated by purple (for absolute certainty), and white-to-gray for decreasing certainty. We did not use sight lines for this run.

The path did not lead directly to the goal, and there were many opportunities for the robot to head cross-country. About two-thirds of the way through the run, no more paths were available, and the robot went through heavy grass and brush to the goal. The robot's pose, as estimated from filtered visual odometry (see Section C.3.2), is in green; the filtered GPS path is in yellow. Because of the tree cover, GPS suffered from high variance at times.

A benefit of using visual odometry is that wheel slips and stalls are easily detected, with no false positives (Section C.5.4). For example, at the end of the

run, the robot was caught on a tree branch, spinning its wheels. The filtered GPS, using wheel odometry, moved far off the global pose, while the filtered visual odometry pose stayed put.

C.3 Constructing consistent global maps

In this section we provide solutions to two problems: representing and fusing the information provided by visual analysis, and registering local maps into a consistent global map.

C.3.1 Map representation

For indoor work, a standard map representation is a 2D *occupancy grid* [92], which gives the probability of each cell in the map being occupied by an obstacle. Alternatives for outdoor environments include 2.5D elevation maps and full 3D voxel maps [49]. These representations can be used to determine allowable kinematic and dynamic paths for an outdoor robot in rough terrain. We choose to keep the simpler 2D occupancy grid, foregoing any complex calculation of the robot's interaction with the terrain. Instead, we abstract the geometrical characteristics of terrain into a set of categories, and fuse information from these categories to create a *cost* of movement.

We use a grid of 20cm×20cm cells to represent the global map. Each cell has a probability of belonging to each of the four categories derived from visual analysis (Section C.2): obstacle, ground plane freespace, sight line freespace, and path freespace. Note that these categories are not mutually exclusive, since, for example, a cell under an overhanging branch could have both path and obstacle properties. We are interested in converting these probabilities into a cost of traversing the cell. If the probabilities were mutually exclusive, we would simply form the cost function as a weighted sum. With non-exclusive categories, we chose a simple prioritization schedule to determine the cost. Obstacles have the highest priority, followed by ground plane, sight lines, and paths. Each category has its own threshold for significance: for example, if the probability of an obstacle is low enough, it will be ignored in favor of one of the other categories. The combination of priorities and thresholds yields a very flexible method for determining costs. Figure C.7 shows a color-coded version of computed costs.

C.3.2 Registration and visual odometry

The LAGR robot is equipped with a GPS that is accurate to within 3 to 10 meters in good situations. GPS information is filtered by the IMU and wheel encoders to produce a more stable position estimate. However, because GPS drifts and jumps over time, it is impossible to differentiate GPS errors from other errors such as wheel slippage, and the result is that local maps cannot be reconstructed accurately. Consider the situation of Figure C.8. Here the robot goes through two loops of 10m diameter. There is a long linear feature (a low wall) that is seen as an obstacle at the beginning and end of the loops. Using the filtered GPS pose, the position of the wall shifts almost 2m during the run, and obstacles cover the robot's previous tracks.

Our solution to the registration problem is to use *visual odometry* (VO) to ensure local consistency in map registration. Over larger regions, filtering VO with GPS information provides the necessary corrections to keep errors from growing without bounds. We describe these techniques in the next two sections.

The LAGR robot presents a challenging situation for visual odometry: wide FOV and short baseline make distance errors large, and a small offset from the ground plane makes it difficult to track points over longer distances. We have developed a robust visual odometry solution that functions well under these conditions. We briefly describe it here; for more details consult [1, 67].

For each new frame, we perform the following process.

1. Distinctive features are extracted from each new frame in the left image. Standard stereo methods are used to find the corresponding point in the right image.

2. Left-image features are matched to the features extracted in the previous frame using our descriptor. We use a large area, usually around 1/5 of the image, to search for matching features.

3. From these uncertain matches, we recover a consensus pose estimate using a RANSAC method [32]. Several thousand relative pose hypotheses are generated by randomly selecting three matched non-collinear features, and then scored using pixel reprojection errors.

4. If the motion estimate is small and the percentage of inliers is large enough, we discard the frame, since composing such small motions increases error. A kept frame is called a *key frame*. The larger the distance between key frames, the better the estimate will be.

5. The pose estimate is refined further in a sparse bundle adjustment (SBA) framework [29, 135]. SBA is a nonlinear batch optimization over camera poses and tracked features. An incremental form of SBA can reduce the error in VO by a large factor at very litle computational overhead. A feature that is long lived, that is, can be tracked over more frames, will give better results.

Precise VO depends on features that can be tracked over longer sequences. Hence, the choice of a feature detector can have a large impact in the performance of such a VO system. Harris corner features are widely used for VO. We have found that although Harris corners give good results and are very efficient to compute, they fail in a lot of situations in outdoor environments. In addition, these features are not very stable resulting in very short track lengths. Other widely used feature detectors such as SIFT [78] and SURF [46] work well but are not suitable for a real time system. We have developed a novel feature (named CenSurE) [3] that has improved stability and is inexpensive to compute. While the basic idea of CenSurE features is similar to that of SIFT, the implementation is extremely efficient, comparable to Harris. Just as SIFT approximates the Laplacian of Gaussian with difference of gaussians, CenSurE features approximates the LOG with bi-level center surround filters. The extreme simplicity of these filters makes them extremely fast to compute, but without sacrificing performance. Figure C.10 shows a progression of bi-level filters with varying degrees of symmetry. The circular filter is the most faithful to the Laplacian, but hardest to compute. The other filters can be computed rapidly with integral images with decreasing cost from octagon to hexagon to box filter. Further Details of our CenSurE feature detector are described in [3]. Figure C.11 shows the CenSurE features tracked over several frames.

The IMU and the wheel encoders are used to fill in the relative poses when visual odometry fails. This happens due to sudden lighting changes, fast turns of the robot or lack of good features in the scene (e.g. blank wall).

C.3.3 Global consistency

Bundle adjusted incremental motions between consecutive frames are chained together to obtain the absolute pose at each frame. Obviously, this is bound to result in accumulation of errors and drifting. We use GPS and the IMU to correct the pose of the vehicle. We perform two types of filtering.

1. Gravity Normal – the IMU's accelerometers measure the gravity normal in vehicle frame together with vehicle accelerations. Vehicle accelerations

have zero mean in the long run and can therefore be considered as white perturbations of the gravity measurements. We apply regular EKF corrections to the tilt and roll angles. By assigning very large noise values to the perturbing accelerations (we used 10g standard deviation), the effect is imperceptible in the short term but sufficient to cancel the long term angular drifts, otherwise unbounded.

2. GPS Yaw – the IMU yaw data is very bad, and cannot be used for filtering (for example, over the 150 m run, it can be off by 60 degrees). Instead, we used the yaw estimate available from the LAGR GPS. These yaw estimates are comparable to a good-quality IMU. Over a very long run, the GPS yaw does not have an unbounded error, as would an IMU, since it is globally corrected.

To maintain globally consistent maps, we have turned off any position filtering based on GPS. We completely ignore position estimates from the GPS in calculating our pose. In addition, to limit the effect of velocity noise from GPS on the heading estimate, GPS yaw is used only when the GPS receiver has at least a 3D position fix and the vehicle is travelling 0.5 m/s or faster. Our filter is a simple linear filter that nudges the tilt/roll (for gravity normal) and yaw (for GPS yaw) towards global consistency, while maintaining local consistency.

The quality of the registration from filtered VO, shown in Figure C.9, can be compared to the filtered GPS of Figure C.8. The low wall, which moved almost 2m over the short loops when using GPS, is much more consistent when VO is employed. And in cases where GPS is blocked or degraded, such as under heavy tree cover in Figure C.7, VO still produces maps that are locally consistent. It also allows us to determine wheel slips and stalls with almost no false positives – note the end of the run in Figure C.7, where the robot was hung up and the wheels were slipping, and wheel odometry produced a large error.

C.3.4 Results of visual odometry

In Test 17, the testing team surveyed a course using an accurate RTK GPS receiver. The 'Canopy Course' was under tree cover, but the RTK GPS and the LAGR robot GPS functioned well. Sixteen waypoints were surveyed, all of which were within 10 cm error according to the RTK readout (one waypoint was deemed inaccurate and not included). The total length of the course was about 150 meters. Subsequently, the LAGR robot was joysticked over the course, stopping at the surveyed points. The robot was run forward over the course, and then turned around and sent backwards to the original starting position.

The course itself was flat, with many small bushes, cacti, downed tree branches, and other small obstacles. Notable for VO was the sun angle, which was low and somewhat direct into the cameras on several portions of the course. Figure C.12 shows two images acquired by the robot. The left image shows a good scene in the shadow of the trees, and the right image shows a poor image where the sun washes out a large percentage of the scene. (The lines in the images are horizon lines taken from VO and from ground plane analysis). The uneven image quality makes it a good test of the ability of VO under realistic conditions.

Since the initial heading of the robot is unknown, we used an alignment strategy that assumes there is an initial alignment error, and corrects it by rotating the forward VO path rigidly to align the endpoint as best as possible. This strategy minimizes VO errors on the forward path, and may underestimate them. However, for the return path, the errors will be caused only by VO, and can be taken as a more accurate estimate of the error.

For this test, our CenSurE features were not ready and we were able to match frames along the whole route using Harris corners. Figure C.13 (a) shows the RMS error between VO (with different filters) and the RTK waypoints, on the return path. As noted above, the forward VO path of the robot has been aligned with the RTK path. As can be seen, the best results are obtained using bundle-adjusted VO with gravity normal and GPS yaw filtering. In this case, the errors between waypoints is very small, amounting to $< 1\%$ of distance traveled. Without filtering, the results are worse (Figure C.13(b)), amounting to about 3% of distance traveled. At some points in the middle of the return trip, the VO angle starts to drift, and at the end of the backward trip there is about a 10m gap. Note that this effect is almost entirely caused by the error in the yaw angle, which is corrected by GPS yaw. It is also worth mentioning that the use of CenSurE features substantially improves the performance of VO although we do not have results of using CenSurE on this dataset.

We present results of VO with CenSurE features on two other large outdoor datasets. These datasets were collected using a larger tank-like vehicle called Crusher (also developed by NREC, Pittsburg under DARPA's UPI program). They have frame-registered ground truth from RTK GPS, which is accurate to several cm in XY and 10 cm in Z. For these datasets, the camera FOV is 35 deg, the baseline is 50 cm, and the frame rate is 10 Hz (512x384), so there is often large image motion. We took datasets from Little Bit (9 km trajectory, 47K frames) in Pennsylvania, and Ft Carson (4 km, 20K frames) in Colorado, to get variety in imagery. The Ft Carson dataset is more difficult for matching, with larger motions and less textured images. In the experiments, we use only CenSurE features, which failed the fewest times (0.17% for Little Bit, 4.0% for Ft Carson).

Table C.2: Trajectory error statistics, in meters and percent of trajectory

		RMS error in XYZ	Max error in XYZ
Little Bit (9 Km)	VO No SBA	97.41 (1.0%)	295.77 (3.2%)
	VO SBA	45.74 (0.49%)	137.76 (1.5%)
	VO No SBA + IMU	7.83 (0.08%)	13.89 (0.15%)
	VO SBA + IMU	4.09 (0.04%)	7.06 (0.08%)
Ft Carson (4 Km)	VO No SBA	263.70 (6.9%)	526.34 (13.8%)
	VO SBA	101.43 (2.7%)	176.99 (4.6%)
	VO No SBA + IMU	19.38 (0.50%)	28.72 (0.75%)
	VO SBA + IMU	13.90 (0.36%)	20.48 (0.54%)

The VO angular errors contribute nonlinearly to trajectory error. On the two datasets, we compared RMS and max XYZ trajectory errors. In the case of matching failure, we substituted IMU data for the angles, and set the distance to the previous value. In Table C.2, the effects of bundle adjustment and IMU filtering are compared.

In both datasets, IMU filtering plays the largest role in bringing down error rates. This isn't surprising, since angular drift leads to large errors over distance. Even with a noisy IMU, global gravity normal will keep Z errors low. The extent of XY errors depends on how much the IMU yaw angle drifts over the trajectory - in our case, a navigation-grade IMU has 1 deg/hr of drift. Noisier IMU yaw data would lead to higher XY errors.

The secondary effect is from SBA. With or without IMU filtering, SBA can lower error rates by half or more, especially in the Ft. Carson dataset, where the matching is less certain.

C.3.5 Map Reuse

VO and IMU/GPS filtering enable us to construct consistent maps on a single run. These maps are useful for getting out of traps and cul-de-sacs in the environment, which occurred quite frequently. In fact, the testing team was interested in long-range sensing capabilities, and would use natural or constructed traps as a way of rewarding robots that could detect them from a distance. Unfortunately, the vision sensors on the robots were not very capable at a distance (see Section C.6 and Figure C.18(a)). So, our strategy was to use map information learned in the first run to compute an optimal path for the second and subsequent runs. This type of learning, *run-to-run learning*, turned out to be the most powerful form of learning for the tests, and the key to performing better than any other LAGR team.

Our first successful test of map learning and reuse was in Test 25 at the end of the project (Figure C.14 and Figure C.18(a)). The direct line to the goal was through a small copse of trees, where there were barriers of deadfall and tall grass. In the first run, the robot wandered through this area, eventually finding a way out to the goal. In the second run, the robot started with the map constructed on the first run, and headed around the problem area. Note that the robot actually started into the cul-de-sac, then decided to go around. The planner had a finite horizon of about 40m, and only recognized the blockage at that point. In subsequent tests we extended the horizon of the planner to the goal.

Our map-reuse technique is simple: at the start of a run, match the robot's view to the start of the previous run, using the same method as for matching frames in VO. If a good match is found, the map from the previous run is brought in and adjusted to the robot's current position. From this point the robot's position on the old map is "open loop," that is, there is no re-registration or localization of the robot within the map. Since VO performance is generally within 1% over 100m, this strategy was overwhelmingly successful during the tests. Still, a true visual SLAM algorithm would work better in more difficult conditions, and we have made significant progress here, closing loops over 5 km datasets [64]; but unfortunately this research was done too late to incorporate into the LAGR system.

C.4 Planning

The LAGR robot was provided with a "baseline" system that used implementations of D* [127] for global planning and Dynamic Window Approach (DWA) [34] for local control. Using this system, we (as well as other teams) had frequent crashes and undesirable motion. The main causes were the slowness of the planner and the failure of the controller to sufficiently account for the robot's dynamics. The D* planner is optimized for very large-scale environments. It uses dynamic programming to compute the minimum-cost potential to the goal at each cell; it needs significant resources to maintain the indices necessary to unravel the minimum-cost computations incrementally. In our environments (100m×200m, 20 cm^2 cells) it would take many seconds to compute a plan, even when only a small portion of the map was filled. For large-scale maps this may be acceptable, but we need much faster response to tactical maneuvers over smaller scales (e.g., cul-de-sacs).

Instead, we re-implemented a gradient planner [63, 103] that computes optimal paths from the goal to the robot, given a cost map. The gradient planner is a

wavefront planner that computes the cost of getting to a goal or goals at every cell in the workspace. It works by using a local neighborhood to update the cost of a cell. If the cell's cost is higher than the cost of a neighbor cell plus the local transit cost, then it is updated with the new cost. The overall algorithm starts by initializing the goal with a zero cost, and everything else with a very large cost. All goal cells are put onto an "open" list. The algorithm runs by popping a cell of the open list, and updating each of the cell's neighbors. Any neighbor that has a lowered cost is put back onto the open list. The algorithm finishes when the open list is empty.

There are many variations on this algorithm that lead to different performance efficiences. Our algorithm has several unique modifications.

- Unlike other implementations, it uses a true Euclidean metric, rather than a Manhattan or diagonal metric, in performing the update step [60]. The update can be performed on the four nearest neighbors of a cell. Generally speaking, the two lowest-cost neighbors can be used to determine the direction of propagation of the cost potential, and the cell updated with an appropriate distance based on this direction.

- The algorithm computes the configuration space for a circular robot, and includes safety distances to obstacles. This is one of the interesting parts of the gradient method. Since there is already a method for computing the distance transform from a set of points, the configuration space can be computed efficiently. The obstacle points are entered as goal points, and the update algorithm is run over each of these points, generating a new open list. Each open list is processed fully, leading to a sequence of open lists. At the end of n cycles, the distance to obstacles has been determined up to $n * c$, where c is the cell size. Usually this is done to a distance of 3 or 4 times the robot radius, enough to establish a safety cushion to the obstacle. Finally, a cost is associated with the distance: an infinite cost within the robot radius to an obstacle, and a decreasing cost moving away from this.

- The queue handling is extremely efficient, using threshold-based queues, rather than a best-first update, which has high overhead for sorting. Instead, we use a 2-priority-queue method. A threshold shuttles new cells to one queue or the other, depending on whether their cost is greater or less than the threshold. The low-cost queue is always processed first. When no more cells remain in it, the threshold is increased, the second queue becomes the low-cost queue, and a new high-cost queue is initialized. This queue strategy is the key to the good performance of the algorithm: each update step happens very rapidly. Although the complexity of the algorithm is the order of the area to be covered, and there is no "best first"

search from the goal to the robot position, still the extreme rapidity of each step makes it possible to cover reasonable areas (e.g., 80m×80m) in several tens of milliseconds.

- Rapid switching of global paths is avoided by including hysteresis - lowering the cost along the path. There is a tradeoff between sticking to the current path, and exploring some new path if current readings indicate it might be better. We lower the cost enough so that it takes a significant amount of new information to turn the path aside.

Typically we run the global planner within a subregion of the whole map, since the robot is continuously moving towards the goal and encountering new areas. On longer runs, up to 200m, we use an 80m x 80m area; the global planner runs in about 30 ms in this region. Unless there is a large cul-de-sac, longer than 80m, this area is sufficient to maneuver the robot tactically around obstacles. For more global planning, which occurs when starting a run with a previously-made map, we run the planner over the whole area, which can take up to 100 ms for a large 100m×200m map.

The global planner is optimistic in assuming the robot to be circular, with a diameter equal to the width of the robot. Also, it does not take into account the nonholonomic nature of the robot's motion. Instead, we rely on a local controller to produce feasible driving motions (Section C.5).

C.4.1 Line goals

One of the problems encountered in directing the robot towards a point goal is that the plans tend to constantly urge the robot towards the center of the map. This is not necessarily an efficient strategy, because, for example, the robot will prefer to run near vegetation on the side of a path that does not point directly towards the goal. Instead, when the robot is far from the goal, we posit a relaxed *virtual goal line* that allows the robot to pursue more indirect paths to the goal (Figure C.15). In a line goal, any point on the line is considered to be a goal, and the robot navigates to the nearest (lowest-cost path) position on the line. For example, in Figure C.15 the robot is shown with the direction of travel straight ahead to its line goal, while with the goal point at the end it would want to move diagonally.

The line goal is easily implemented in the gradient planner, by simply adding all points on the line as goal points. The navigation function then computes the lowest-cost path to any point on the line. The line goal is always place about 60m ahead of the robot, and its extent grows in the middle of the run,

and contracts as it gets nearer to the goal. In experiments, the robot is able to navigate more than 50m off the center line to the goal, and consequently find easily traversed paths that would have been difficult to find if it had headed directly to the goal (Figure C.7).

C.5 Control

Given the global cost information produced by the gradient planner, we must decide what local controls to apply to the robot to drive it toward the goal.

C.5.1 Trajectory generation

We take an approach that is opposite to techniques such as DWA. Instead of searching the space of feasible *trajectories*, we search the space of feasible *controls*. As is the case with most differentially-driven platforms, the LAGR robot is commanded by a pair $(\dot{x}, \dot{\theta})$ of desired translational and rotational velocities.[3] Thus we have a 2D space of possible commands to consider.

This space is bounded in each dimension by velocity limits that reflect the vehicle's capabilities. Because we are seeking *good*, as opposed to *optimal*, control, we sample, rather than exhaustively search, this rectangular region of allowed velocities. We take a regular sampling ($\sim$25 in each dimension, $\sim$625 total), and for each sample simulate the effect of applying those controls to the robot over a short time horizon ($\sim$2s). The simulation predicts the robot's trajectory as a sequence of 5-dimensional $(x, y, \theta, \dot{x}, \dot{\theta})$ states with a discrete-time approximation of the vehicle's dynamics.

Of significant importance in this simulation are the vehicle's acceleration limits. While the LAGR robot can achieve a speed of 1.3 m/s, its low-level motor controller (which we cannot modify) follows a trapezoidal velocity profile that limits the translational acceleration to approximately 0.5 m/s^2 (we determined this value empirically). Thus more than 2 seconds may elapse between commanding and achieving a desired velocity. We found that the ability to accurately predict the LAGR robot's future state depends vitally on appropriate integration of these acceleration limits. We expect this to be the case for any vehicle with a similarly large ratio of maximum velocity to maximum acceleration.

[3] We could instead work in terms of left and right wheel velocities; the two velocity spaces are equivalent, being related by a simple geometric transformation.

The generated trajectories, projected into the (x, y) plane, are smooth, continuous 2-dimensional curves that, depending on the acceleration limits, may not be easily parameterizable. For the LAGR robot, the trajectories are generally not circular arcs (Figure C.16).

C.5.2 Trajectory evaluation

Each simulated trajectory t is evaluated by the following weighted cost:

$$C(t) = \alpha \text{Obs} + \beta \text{Gdist} + \gamma \text{Pdist} + \delta \frac{1}{\dot{x}^2} \tag{C.1}$$

where *Obs* is the sum of grid cell costs through which the trajectory passes (taking account of the robot's actual footprint in the grid); *Gdist* and *Pdist* are the estimated shortest distances from the endpoint of the trajectory to the goal and the optimal path, respectively; and $\dot{x}$ is the translational component of the velocity command that produces the trajectory. We choose the trajectory for which the cost in Equation C.1 is minimized, which leads our controller to prefer trajectories that: (a) remain far from obstacles, (b) go toward the goal, (c) remain near the optimal path, and (d) drive fast. Trajectories that bring any part of the robot into collision with a lethal obstacle are discarded as illegal.

Note that we can compute $C(t)$ with minimal overhead: *Obs* is a simple summation over grid cell costs, *Gdist* and *Pdist* were already computed by the planner for all map cells, and $\dot{x}$ is a known constant for each trajectory.

C.5.3 Supervisory control

We could generate, evaluate, and compare all potential trajectories. However, given the kinematic design (driven wheels in front, passive casters behind) and sensor configuration (forward-facing cameras and forward-mounted bumper) of the LAGR robot, we found it useful to add supervisory logic to direct the order in which candidate velocities are simulated and evaluated.

All forward velocities ($\dot{x} > 0$) are tried first; if any legal forward trajectory is found, the best one is selected. If there are no legal forward velocities, then the controller tries in-place rotations ($\dot{x} = 0$), and then backward velocities ($\dot{x} < 0$). This preference ordering encourages the robot to make forward progress whenever possible, and discourages driving backward (during which the robot is essentially blind). If no legal trajectory is found, the default behavior of the robot is to move slowly backward.

C.5.4 Slip handling

Because the robot may have to traverse rough, steep terrain, it is necessary to detect and react to conditions in which the wheels slip or become stuck. We employ two mechanisms to handle these situations. In both cases, we are comparing the motion reported by the wheels to the motion estimated by visual odometry (VO), which is sufficiently accurate to be treated as ground truth (Section C.3.2).

First, the controller continuously compensates for the slip in each wheel by reducing its maximum speed. Our approach is similar to automotive traction control. For each wheel, we monitor the slip ratio s, defined as [6]:

$$s = \frac{\omega r - v}{\omega r} \in [0, 1] \tag{C.2}$$

where ω is the measured angular velocity of the wheel, r is the wheel radius, and v is the actual linear velocity of the wheel. We obtain ω directly from the wheel encoders. To compute v, we difference sequential VO poses to produce translational and rotational velocities for the vehicle, then use the vehicle geometry to distribute these velocities between the two wheels. When the slip ratio s for a wheel exceeds a minimum threshold ($\sim$0.25), we compensate by proportionally reducing the maximum allowable speed for that wheel, which produces better traction on most terrain. Importantly, the controller takes account of the current speed limits, ensuring that predicted trajectories will be achievable under these limits. The slip ratios and speed limits are recomputed at the frequency of VO pose estimation ($\sim$15Hz).

While continuous slip compensation improves performance, there are situations in which the robot can become truly stuck, and require explicit escape mechanisms. The robot usually becomes stuck because of extremely slippery soil (e.g., sand), or ground clutter (e.g., fallen branches). We detect these conditions by looking for significant, time-extended disparities among the velocities that are: commanded by the controller, reported by wheel odometry, and estimated by VO (we maintain a running window of each velocity). If a slip or stall is detected, or if the front bumper is triggered, the robot enters a stochastic finite state machine of preprogrammed escape maneuvers (e.g., drive forward, turn in place, drive backward). These maneuvers are executed blindly, on the assumption that the vision system failed to identify the terrain as dangerous and so is unlikely to yield good advice on how to escape it.

One indication of how well slip detection performed was on Test 18, about 2 years into the program. This was a difficult test around small sand dunes, and the robots would spin easily on the sand on small inclines. The slip detection code

and escape maneuvers, coupled with visual odometry for localization, allowed us to finish this challenging course, the only team to do so.

C.6 Performance

For the LAGR program, the government testing group (LGT) ran monthly blind demos of the perception and control software developed by the teams, and compared their performance to a baseline system. Teams were encouraged but not required to send code for each test; they could also send the same code on successive tests. In general the tests would change each month, to expose the teams to different environmental conditions.

There were two major checkpoints for all the teams, the first at the end of Phase I of the program after 1.5 years (Tests 12 and 13), and the second at the end of the program (3 years, Test 27). The goal was to beat the baseline system at the end of Phase I, and to do better by a factor of 2 for the second. In this section we show results from all of these tests, and additionally the penultimate tests 25 and 26. At this point, our system was essentially complete, and Tests 25 and 26 presented interesting terrain challenges, while Test 27 was somewhat artificial and designed to isolate specific learning capabilities.

On each test, the robot was given 4 runs, and the best 3 were taken to give a combined score. The highest achievable score is 1.0, calculated by measuring the shortest possible path to the goal, and measuring the time it took a skilled operator to manually drive the robot. There were also penalties for not getting to the goal within a cutoff time.

C.6.1 Phase I Tests

The tests at the end of Phase I were designed to test the overall ability of the system to handle typical outdoor terrain (make a consistent map, recover from slips, etc.), while focusing on perceptual skills and learning. For these tests, we had completed a basic VO system, the global planner, and the controller. All of the main stereo interpretation algorithms were also in place, including sightlines. However, our appearance-based learning was only a simple supervised color classifier, and there was no map reuse, because the VO system was not precise enough.

In Test 12 (Figure C.17(a)), there was a dark mulch path that formed the easiest

way to the goal, and teams were invited to submit automatic supervised learning code that would be trained on a similar path from log files. The minimum times for a skilled operator were 69 seconds along the path (103m), and 77 seconds through the maze (distance not measured).

Several teams managed to follow the path after training, and generally had good times. We did not; the dark color of the path confused our color-based learner, and led us to develop the combined color/texture model described in Section C.2.3. We completed the maze course 3 times, with times of 106, 110, and 112 seconds. Our best time equaled that of the best team following the mulch path, and our average score of 0.73 was the highest of any team, 3 times the baseline score (which was also through the maze). Although the LGT expected the maze to be a significant problem for the teams (as it was for the baseline), our combination of consistent map-making, fast global planning, and rollout controller combined to make the robot zip through the maze with no hesitation.

In Test 13, the objective was to stay along an old dirt and asphalt road for much of the course, until there was an opening in brush to the left of the road towards the goal (see Figure C.17(b)). The easiest way (most open route) to the goal was along the third route from the left, which required the robot to stray far from the direct route to goal, following the open road. The second route was actually the shortest, requiring a skilled operator just 90 seconds, while the third route took 95 seconds.

In all three scoring runs, we followed the third route, utilyzing sight lines (Section C.2.2) to find open space along the road, and then the open space along the third route. The times for three runs were 132, 132, and 148 seconds. All of our times were faster than the best time of any other team, and our overall score, 0.86, was just under 3x the score of the baseline. We expect that the better online color/texture path learning of Section C.2.3 would have found the second route, and a shorter time, but we did not develop this technique until the end of the project.

C.6.2 End-of-project Tests

The end-of-project tests were through different types of terrain, and with different degrees of difficulty. For these test, our full system was operational, including online color/texture learning of paths, and map reuse. Here is a summary of the courses (Figure C.18).

Test 25 83m straight-line distance to the goal, through a copse of trees with a cul-

de-sac and tall grass (Figure C.18(a)). Ideal behavior was to go around the copse, following a mulch path as in Test 12.

Test 26a (93m) Narrow paths through tall bushes, with several false turns that might lead more directly to the goal. Desired behavior was to avoid the false turns.

Test 26b (106m) A challenging course with man-made and natural obstacles, including a cul-de-sac of parked cars; stacked pipes; hay bales; and rock piles (Figure C.18(b)). The course to the goal was indirect and involved narrow passageways, and finding it was a challenge.

Test 27a (34m) A simple course on a grassy field with jersey barriers stretched directly across the route (Figure C.18(c)). Ideal behavior would be to avoid the barrier without getting close.

Test 27b (34m) Similar to 27a, but using low hay bales for obstacles, with two gaps in the barrier containing tall grass. The object was to identify the tall grass and push through it directly to the goal.

The first four tests were designed to reward behavior that could avoid routes that were temptingly direct, but ultimately dead-ends. There were two methods of doing this – long-range perception (>10m), and map memorization and reuse. For Test 26a, the narrow routes through the bushes were easily detected by our online learning algorithms, and the path planner moved the robot quickly along the center of the path. On the first run, the robot turned twice to look briefly at side paths that could have been more direct, but then turned back to the main route. Figure C.19 shows the scores for this run. The Baseline score is 0.23, and SRI's score is 0.83, which is better by a factor of 3.6. In this test, since the long-range perception of paths worked well, the first run was very good (2.9x Baseline), and subsequent map reuse only contributed a modest amount, by not turning to examine the dead-end paths. In fact, our score could have been higher, but the fourth run failed because of a map registration error in the middle of the run, closing off the narrow path.

In the other three tests (25, 26b, 27a), map reuse is the primary enabler of good performance – it improved by almost a factor of 2 from the first run. For example, in Test 25, after wandering through the copse and encountering the cul-de-sac and tall grass obstacles, the robot made its way to the goal. On the second run, the robot avoided the copse entirely, choosing a path around it as less costly.

Test 27b was a learning-by-example test. The robots were shown samples of the hay bales and tall grass. Operators would drive the robots into the hay bales and over the grass, to give the robot an idea of the traversability of each. Our

online learning algorithms correctly picked out the grass as driveable, based on primarily on its texture, since the color was similar to the hay bales. We also learned that hay bales were obstacles; however, we had set the suppression of obstacles by driveable objects a little too high, and the robot bumped the hay bales next to the grass area. After a few bumps, it drove through the grass and onto the goal. In subsequent runs, of course, map reuse allowed an optimal plan directly through the grass.

C.6.3 Analysis

There is no doubt that our system achieves both robustness and good performance, on a wide variety of outdoor, unstructured terrain. Map building relies on VO to provide good localization, efficient realtime stereo and robust ground-plane analysis for obstacle detection, and sight lines to identify distant regions that are likely to be navigable. Online path learning helps in the very common situation of tracks through vegetation, or man-made dirt and asphalt roads. Together these techniques allow us to construct well-registered, precise maps that serve well during the first run to get the robot reliably to the goal. Even more importantly, on subsequent runs, the path planner is able to construct an optimal path to the goal from the start of the run.

Moving quickly is very important to achieving good performance, especially since many small obstacles such as branches could be traversed at speed, but might hang up the robot if it was moving slower. As described in Section C.5, the path planner and local controller combined to give the robot a very agile feeling. Our average speed was over 1.1 m/s, even while exploring unknown terrain (top speed of the robot is 1.3 m/s).

The government team was very interested in creating scenarios to test the long-range perception of the robot. Unfortunately, the robot's vision sensors had very little resolution at distance. Depth information from stereo was very uncertain after about 7m. Even using monocular information, there were very few pixels available for long-range sensing. In Figure C.18(a), a high-resolution camera with a longer focal length clearly shows routes around the barrier. But with a similar distance to the tree on the right image, looking through the robot cameras, there is very little to show that the copse of trees could be avoided to the left – perhaps there are a few more vertical pixels of brown-colored grass on that side. But this information is insufficient to reliably navigate from the robot's perspective, and teams that tried to do this would as often pick a bad way as a good one.

What we could reliably learn is the map structure from the first run. With this

in hand, subsequent runs could be much more efficient. We had this technique working reliably only in the last tests (25–27), and it was difficult for the government team to react and set up tests that would allow long-range perception to do as well as map learning and reuse. It was also difficult for other teams to adopt our technique, because it required very good map registration, and a badly-registered map is worse than no map at all. In Test 26a, the narrow paths (≈2m wide) meant that even small registration errors could cause a prior map to close off the current path, which happened to us in the fourth run. Note that the map reuse was run open-loop: after registering with an initial image at the beginning of the run, we relied on VO to keep the robot localized.

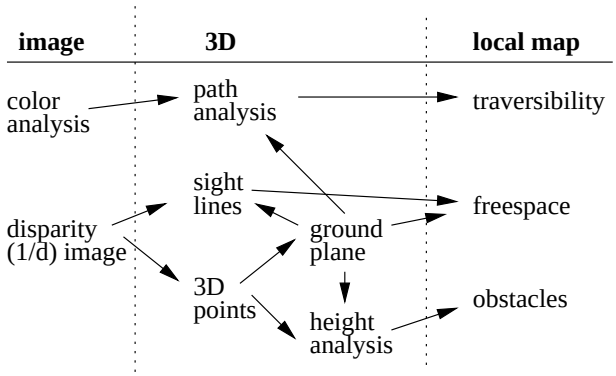
We compared our results with the published results of the other teams, both the average and the best for each test (Figure C.19). In all these tests, we had the best score (or tied for the best). Typically we out-performed the average team by a factor of two. In the most difficult test, 26b, even our first-run score was almost as good as the best overall team score; map reuse enabled us to do even better. The controller, planner, and visual odometry system were used in the best-in-class NIST system, and in fact NIST was our closest competition in two of the tests, including the difficult Test 26b. We also surpassed the target of 2x baseline performance, achieving almost a 4x improvement, better than the 3x improvement of the Phase I tests. There is no doubt that map reuse was the primary enabler for this performance.

While many teams concentrated on finding obstacles at a distance using color-based learning, we decided that the risk of using this technique in complicated environments was not worth the results. In some simple (and contrived) scenarios, such as Test 27a (Figure C.18(c)), it could indeed help, but if it were used all the time, it would be as likely to lead to bad choices as good (dark green areas could be shadows instead of trees), and cause poor overall behavior. Our online path-learning, by contrast, used both geometric and color/texture cues to reliably find good paths, with almost no false positives.

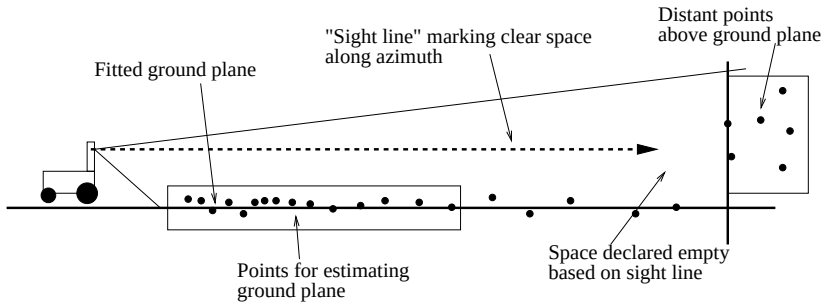
C.7 Conclusion

We have demonstrated a complete autonomous system for off-road navigation in unstructured environments, using stereo vision as the main sensor. The system is very robust – we can typically give it a goal position several hundred meters away, and expect it to get there. It is also one of the first to demonstrate the practical use of visual odometry as the primary method of registration, with extremely good results. The precision of VO is such that maps can be reused on subsequent runs, doubling the performance of the system.

To be sure, there are hazards that are not dealt with by the methods discussed in this paper: water and ditches are two robot-killers. We also were restricted to running open-loop in reusing maps; we would like to use visual landmarks to re-register the position of the robot in the map, but this work was not ready at the time of the last tests.

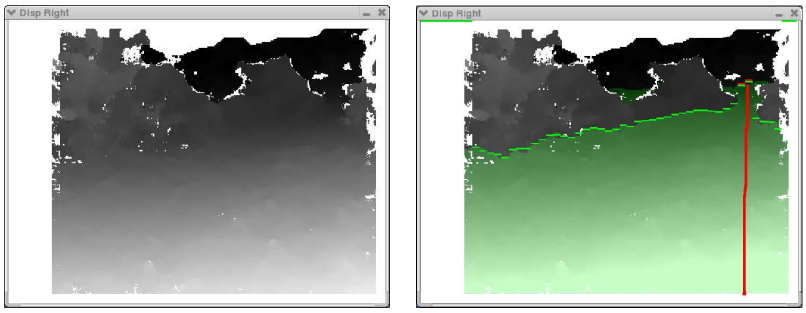


(a) Processing Diagram

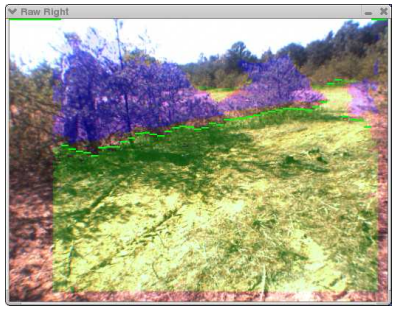


(b) Stereo point interpretation

Figure C.2: Visual processing. In (a), the paths from visual input depict the processing flow in constucting the local map. The interpretation of stereo data points is in (b): nearby points (out to 6m) contribute to the ground plane and obstacle detection; further points can be analyzed to yield probably freespace ("sight lines") and extended ground planes.



(a) Disparity Image (b) Disparity Image Ground Plane



(c) Interpreted Image

Figure C.3: (a) Disparity image from the left stereo pair of the robot in Figure C.1. Closer pixels are lighter. (b) Extracted ground plane, in green overlay. Limit of ground plane is shown by green bar; sight line has a red bar. (c) Ground plane overlayed on original image, in green. Obstacles are indicated in purple.

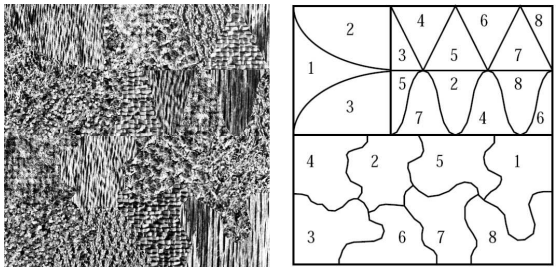


Figure C.4: Synthetic texture mosaic used (provided by USC via its website). The left image is the texture mosaic. The right image shows which texture regions belong to which texture.

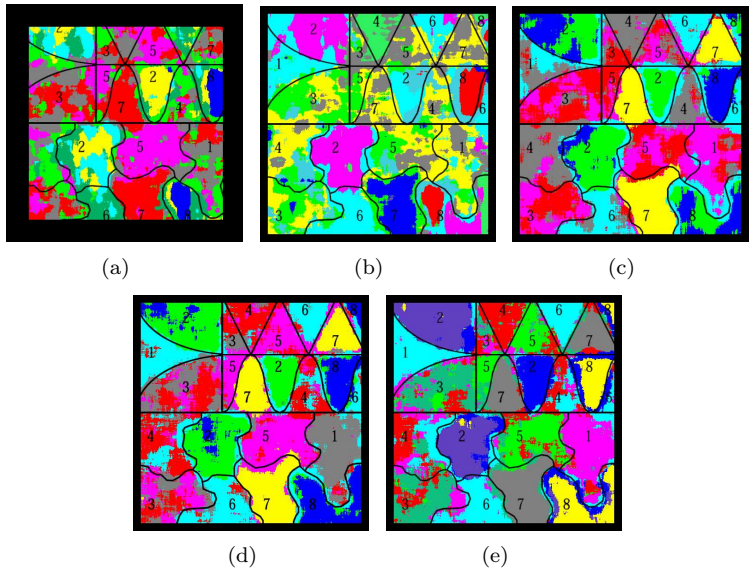
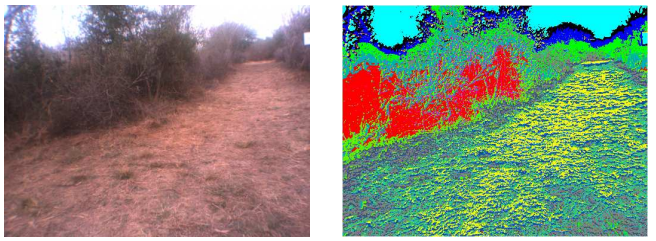
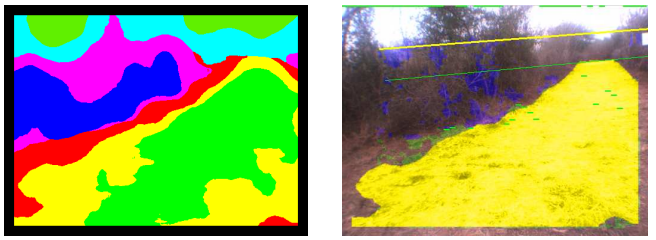


Figure C.5: Results for the synthetic texture segmentation. Each color represents a different histogram cluster. An overlay shows which regions should have homogeneous colors. (a) LM Filter, 32, (b) RGB 5x5, 32, (c) LBP 3x3,32, (d) SRI 3x3,32, (e) SRI 5x5,64.



(a) (b)



(c) (d)

Figure C.6: Various steps of our segmentation algorithm on a typical outdoor image. (a) The image from one of the stereo cameras. (b) Each pixel assigned to a texton. (c) Each histogram of textons gets assigned to a histogram profile. (d) A path is recognized (in yellow).

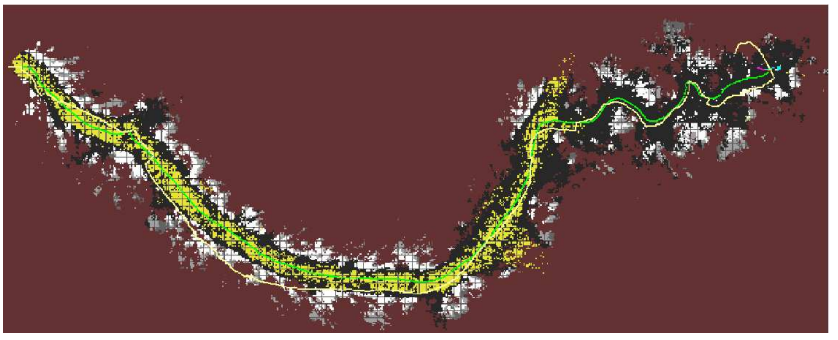


Figure C.7: Reconstruction on a 130m autonomous run. Yellow is recognized path, black is freespace, and white and gray are obstacles.

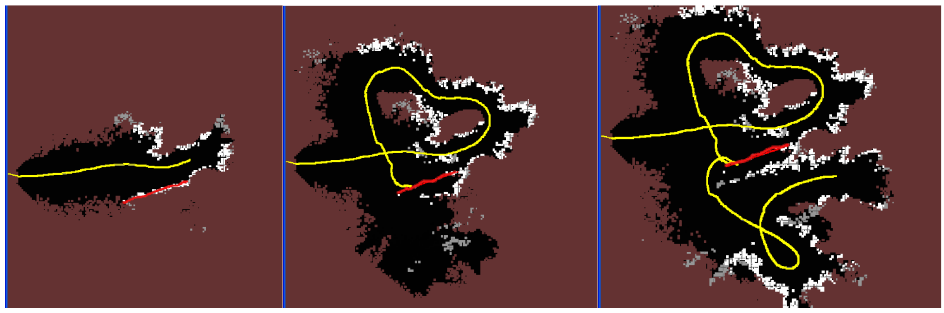


Figure C.8: Three stages during a run using GPS filtered pose. Obstacle points are shown in white, freespace in black, and the yellow line is the robot's path. The linear feature is marked by hand in red in all three maps, in its initial pose. Map extent is 35m on a side.

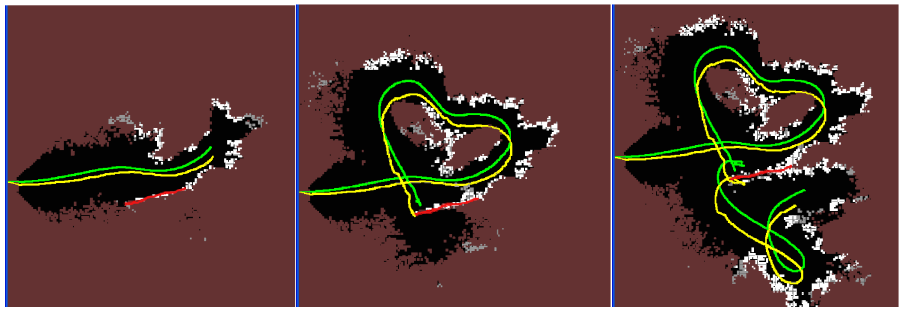


Figure C.9: VO in the same sequence as Figure C.8. GPS filtered path in yellow, VO filtered path is in green.

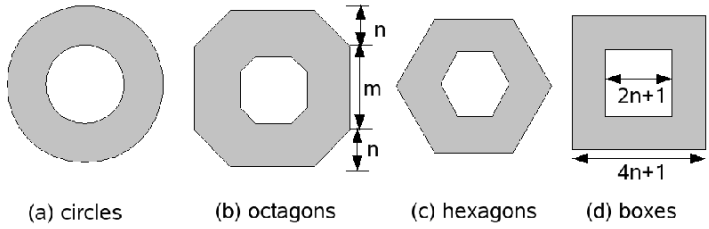
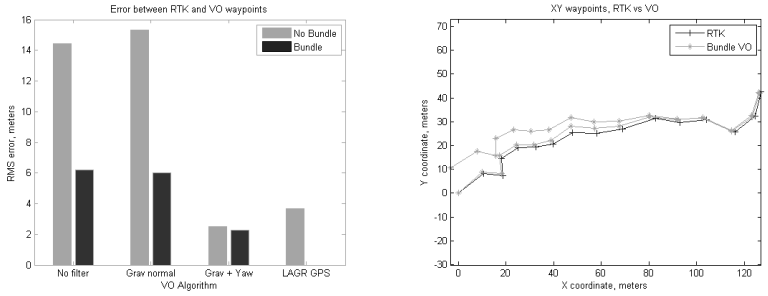


Figure C.10: Progression of Center-Surround bi-level filters. (a) circular symmetric BLOG (Bilevel LOG) filter. Successive filters (octagon, hexagon, box) have less symmetry.

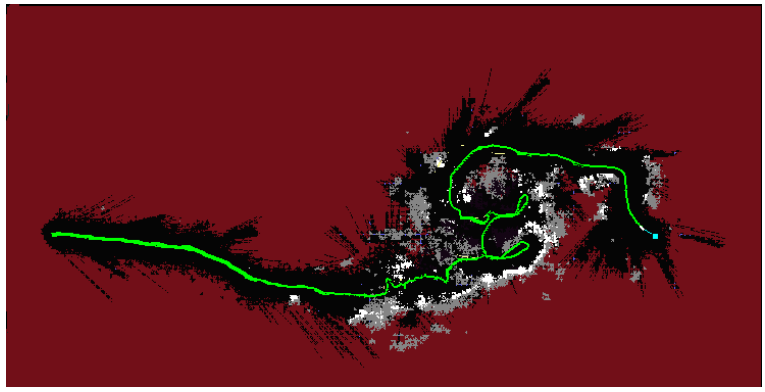
Figure C.11: CenSurE features tracked over several frames.

Figure C.12: Images from the Canopy dataset.

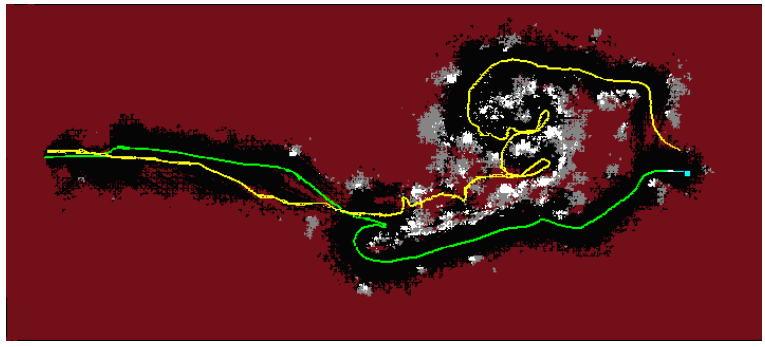


(a) (b)

Figure C.13: Results of VO on the Canopy dataset. (a) RMS error between VO (with different filters) and the RTK waypoints, on the return path. (b) Trajectory of bundle adjusted VO (without any filtering) compared to RTK groundtruth.



(a) Test 25 Initial Run



(b) Test 25 Second Run

Figure C.14: Map reuse during Test 25. The global map in (a) shows the first run: black is freespace (including long sightlines), white and gray are obstacles. The robot path estimated from VO is the yellow line. Starting position of the robot is the left side of the screen; goal is on the right at about 80m. Note the many extended concave obstacles and cul-de-sacs. Image (b) shows the robot's trajectory for the second run in green, bypassing the cul-de-sac obstacles and heading around to the right. The original run is superimposed.

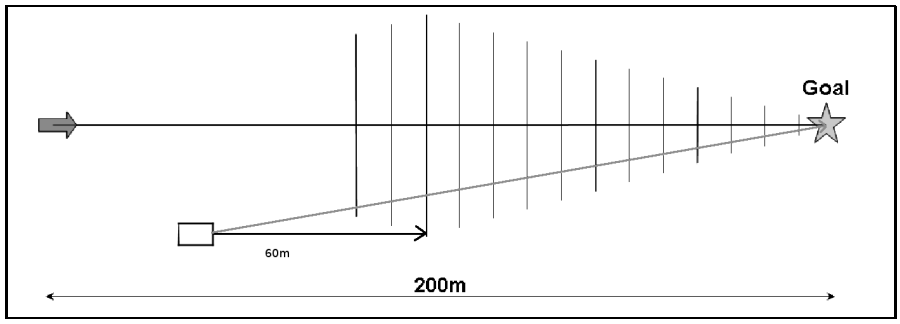
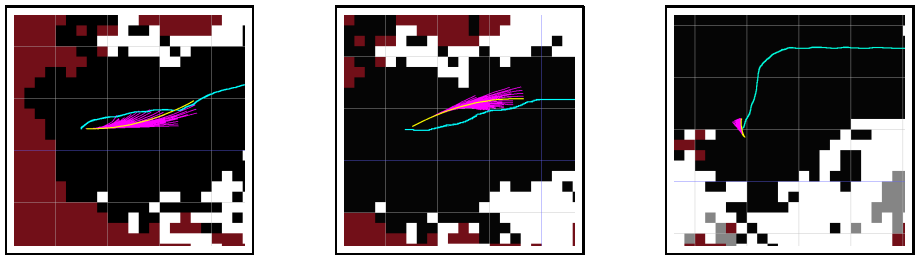
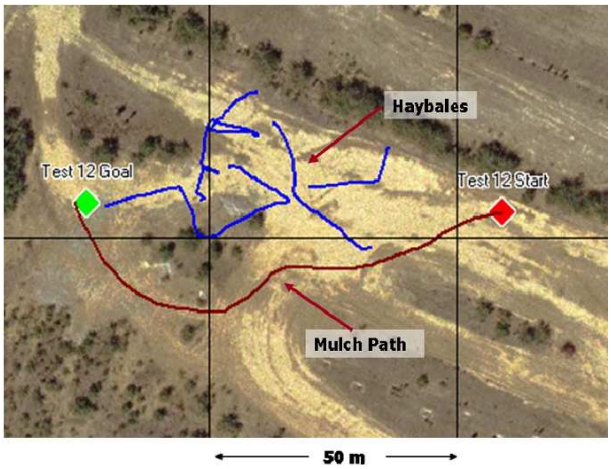


Figure C.15: Line goals for a robot in a 200m environment. The line goal is placed 60m ahead of the robot, and its extent varies with the distance to the goal.

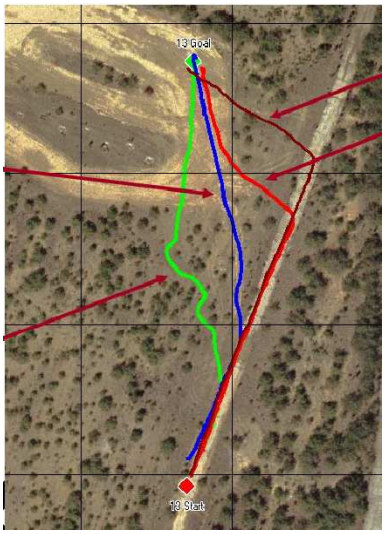


(a) (b) (c)

Figure C.16: The controller generates trajectories by sampling feasible velocities and simulating their application over a short time horizon. Generated trajectories are purple, the chosen trajectory is yellow, the desired global path is cyan, and obstacles are white. As shown in (a) and (b), the trajectories are smooth but not easily parameterizable as they depend on the vehicle's current velocity and its acceleration limits. When forward motion is not possible, backward trajectories are considered (c) - robot is facing down towards obstacles.



(a) Test 12



(b) Test 13

Figure C.17: Aerial views of the Phase I tests. In (a), the haybale maze is drawn in blue, while the mulch path leading to the goal is in red. The tree copse leading directly to the goal was not traversable. In (b), four possible routes to the goal are drawn, with the third from the left being the easiest (154m).

(a) Test 25

(b) Test 26b

(c) Test 27a

Figure C.18: Views of three final tests. In (a), a robot's-eye view of the beginning of Test 25. The copse in the distance could be avoided on the right or left. The yellow line is the robot's horizon from a noisy INS, while the green line is the VO-stabilized horizon. In (b), a pipe corridor from Test 26b – note the blocked left corridor. In (c), Test 27a shows the jersey barrier, with the goal immediately behind.

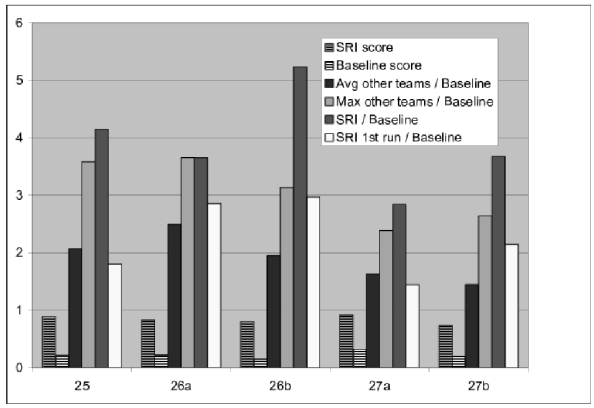


Figure C.19: Summary of results from the last 3 LAGR tests. Raw scores are given for the Baseline software and the SRI system, where 1 is a perfect score (as fast as the robot can go). The other scores are presented as a factor over Baseline; the target performance for the project was 2x Baseline.

Paper D

Fault-Tolerant 3D Mapping with Application to an Orchard Robot

Morten Rufus Blas, Mogens Blanke, Radu Bogdan Rusu and Michael Beetz. Fault-Tolerant 3D Mapping with Application to an Orchard Robot. *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 893-898, Barcelona, Spain, 2009. Published.[1]

[1]This work was supported by The Danish Food Industry Agency under contract 3412-06-01729, and the CoTeSys (Cognition for Technical Systems) excellence cluster at the Technische Universität München.

Abstract:
In this paper we present a geometric reasoning method for dealing with noise as well as faults present in 3D depth maps. These maps are acquired using stereo-vision sensors, but our framework makes no assumption about the origin of the underlying data. The method is based on observations made on the environment from different camera poses (viewpoints), where the occupied space as well as uncertainties in the range measurement are modelled using dynamic octree structures. This scheme allows us to detect and diagnose faulty range measurements in an efficient manner. We present results on the acquisition of comprehensive 3D maps for an agricultural robot operating in an orchard.

D.1 Introduction

Mobile robotic research is typically organized into perception, planning, and control. Our work concentrates on the first problem, namely perception, where the acquisition and interpretation of 3D maps representing the surrounding environment are critical for the robot's operating safety and reliability. As a sensing modality, we make use of stereo-vision techniques, but the methods presented in this paper are crafted with generality in mind, thus the data can be acquired using any general range sensor. Stereo-vision is cheap, and passive, as it relies on inferring distance by matching 2D features between pairs of images taken by two synchronized cameras.

The penalty of stereo-vision however is that range accuracy quickly degrades with distance from the cameras, and that areas of the image where it is hard to match features can result in wrong or no range measurements. The outliers can be filtered based on the matching confidence but there is no assurance that all such faulty measurements can be removed. The output of stereo-vision is a depth map commonly referred to as a disparity image where each pixel contains depth information that can be projected out in 3D. Since there is overlap in the image stream a number of depth maps can be created for the same objects taken from different ranges and viewpoints. This creates a certain data redundancy in the system which can be used to detect inconsistencies in the depth maps such as faulty measurements. To do this it is a necessity that the depth maps can be considered in the same coordinate system. For moving stereo platforms this involves precise relative positioning of the images. In this paper we present a method for using this data redundancy along with an accurate positioning system to fuse multiple depth maps in order to reduce noise and filter out faulty measurements. We use a robust image derived positioning system called Visual

Odometry (VO) [67].

Figure D.1: An overview of the evaluated dataset with GPS overlayed on Google Maps. The robot was driven around in the area that it is likely to operate in. The driven path from the GPS is drawn in color to illustrate signal quality. Blue is GPS DOP less than 2, purple is 2 to 3, and red is DOP higher than 3.

The amount of research in 3D mapping is overwhelming, and due to space constraints we are unable to cover all similar initiatives. Instead, we will just reference the ones most appropriate for our work. Traditionally, highly accurate maps are created using low-noise laser range finders, as for example shown in [98]. Stereo sensors have considerable less accuracy but can be tweaked to still obtain reasonably good results for some applications as presented in [116]. A critical point in any mapping system is good positioning information. For stereo-vision a solution to this problem is given by the use of VO techniques (see [67]), which offer ways of obtaining accurate 6-degrees of freedom poses. There are many alternatives where GPS is traditionally widely used in agriculture. We demonstrate a fault-tolerant positioning system integrating GPS and stereo-vision for applications in agriculture in [18].

The system presented in this paper uses a stereo camera pair as its primary sensor input data stream, and is comprised of a series of geometric reasoning techniques, which are at the core of data acquisition, filtering, and 3D mapping (see fig. D.2). Due to the poor accuracy of stereo sensors in terms of distance measurements, these techniques must be robust enough to account for the faulty measurements. Filtering bad data at an early level by fusing multiple depth maps acquired from different camera poses constitutes a primary goal of our approach.

In detail, the depth map fusion is produced by aligning the data in the same coordinate system and then merging several views together while trying to eliminate noise. Similar initiatives are presented in [141], where an algorithm is used to consider only surfaces in cells that were supported by a consensus from multiple depth maps. In [117] each range measurement votes for a cell to contain a surface or free space between the camera and the surface. In [104] all data is merged into a single reference view, and conflicts in visibility are resolved by looking for occlusions as well as support for different hypotheses.

The 3D mapping system presented here goes beyond what has previously been demonstrated in agricultural environments in terms of robustness and accuracy - specifically when using stereo-vision. The proposed method of dealing with faulty range measurements in the context of mapping is also novel.

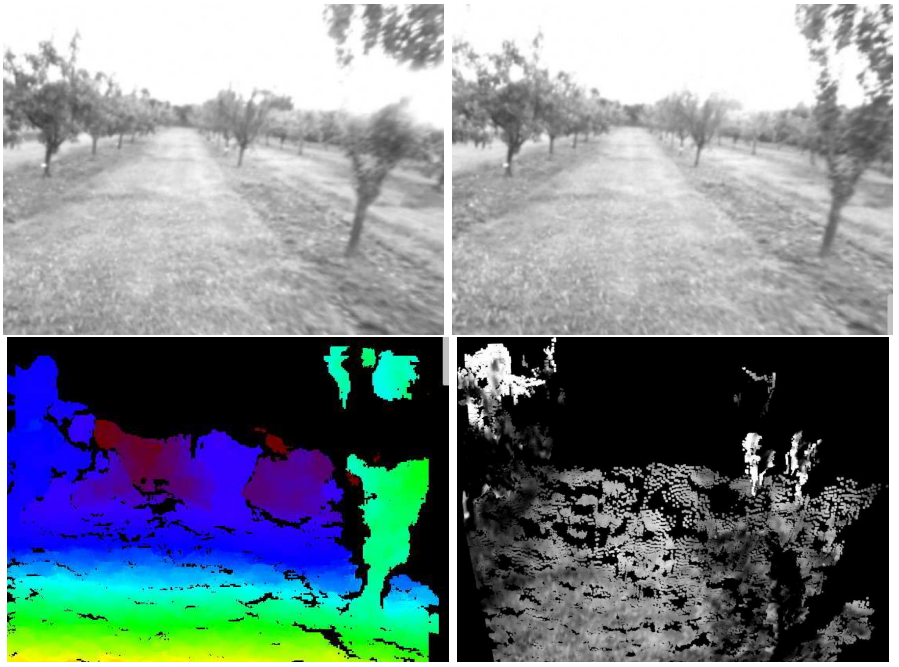


Figure D.2: Top row: left and right camera images acquired with the stereo pair; bottom row: disparity image (depth map) and the associated Point Cloud Data (PCD) frame.

D.2 Stereo Processing

Stereo vision perceives depth using triangulation. The distance to a point is determined by the triangle between the point and where it appears in each of two images. To do this the two images must be aligned. the process of aligning the images is part of a calibration step. Given a calibrated stereo camera the images are then aligned by warping them. This is known as rectification. This gives two cameras with parallel optical axes and horizontal epipolar lines. A dense estimation of ranges is then performed at each pixel by matching along the epipolar lines. This is done using a correlation window with typical sizes of around 10x10 pixels. The correlation window matches texture in the two images with each other. The output of the matching process is a disparity image. This gives the image difference between the position of objects in the two cameras. The horizontal distance from the image center to the object image is dl for the left image and dr for the right image (See Fig. D.3). Then the disparity value d is given by:

$$d = dl - dr \tag{D.1}$$

A pixel in the disparity image can then be projected to a 3D coordinate using triangulation.

For a stereo-camera a point in the depth map is defined by (u, v, d) which is the column, row, and disparity value in the disparity image. This can be projected to and from a 3D coordinate defined by (x, y, z) in the camera coordinate system using the following formula:

$$\bar{x}_l = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{(u-u_0)b}{d} \\ \frac{(v-v_0)b}{d} \\ \frac{fb}{d} \end{bmatrix} \tag{D.2}$$

where u_0 and v_0 are the column and row coordinate of the optical center of the image in pixels. b is the camera baseline and f is the focal length for the rectified image.

D.2.1 Artifacts

The stereo processing algorithm relies on objects being matched correctly in the two images. Incorrect matches give a wrong disparity value in the disparity

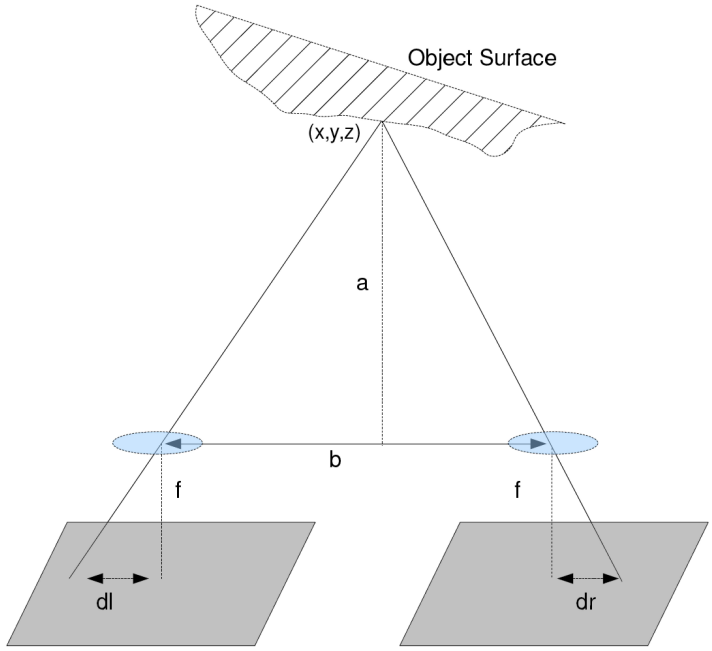


Figure D.3: A simplified view of stereo geometry. Disparity is the offset of the image location of an object: $d = dl - dr$. a is the range to an object. b is the baseline between images. f is the focal length of the cameras.

image. This in turn yields an incorrect range measurements for a given pixel in the disparity image. These incorrect matches represent artifacts (faults) in the stereo imaging. A filtering step is usually performed after the stereo algorithm to remove artifacts. We use 3 well-known methods. A confidence filter removes matches in areas of low texture (where it is hard to match). A uniqueness filter removes matches with high ambiguity. A speckle filter removes small regions using a salt-and-pepper filter. Generally these filters do well but they will not catch all incorrect matches. The motivation is to exploit the redundancy that using multiple images gives in order to detect these artifacts. This requires us to be able to transform multiple images to the same coordinate system. This is the output of VO.

D.3 Visual Odometry

The VO system derives from recent research on high-precision positioning (see [67]) using a stereo camera. By tracking image features between images the change in pose of the camera can be estimated. This is done by computing changes in position between image frames. In this type of system GPS or other aiding positioning sensors are typically needed to maintain global consistency as VO will slowly drift as noise in the positioning estimate will also be integrated. GPS in orchards generally have poor performance as seen in our dataset Fig. D.1.

For each new frame, we perform the following process in the VO algorithm:

1. Distinctive features are extracted from each new frame in the left image. Standard stereo methods are used to find the corresponding point in the right image.
2. Left-image features are matched to the features extracted in the previous frame.
3. From these uncertain matches, a consensus pose estimate is recovered using a RANSAC method [32]. Several thousand relative pose hypotheses are generated by randomly selecting three matched non-collinear features, and then scored using pixel reprojection errors.
4. The pose estimate is refined further in a sparse bundle adjustment (SBA) framework, as presented in [29, 135]. SBA is a nonlinear batch optimization over camera poses and tracked features.

The output of the VO system is the transformation matrix (rotation $\mathbf{R}$ and translation $\mathbf{T}$) between the camera poses.

D.4 3D Model

D.4.1 Visual Odometry and PCD registration

Each pair of images from the stereo camera delivers a set of point cloud data (PCD), that is, every point in the disparity image is projected in local 3D coordinates. The local coordinate system has the current camera pose as the origin. Our 3D map building is performed by registering multiple PCDs in a global coordinate frame. VO allows us to register individual PCD frames relative to each. Let (R_t,t_t) be the transformation from camera coordinates to global coordinates for the PCD in frame t. Then we denote $\mathcal{P}_t$ the *registered* PCD in the global coordinate frame.

D.4.2 Spatial decomposition of PCD using Octrees

To account for spatial visibility across the entire map, we construct a dynamic fixed-width octree structure. This is a hierarchical data structure that is based on the recursive decomposition of a 3D region. An octree node is a cube in 3D space. Each node has eight children, and the actual data is stored at the node leaves.

As soon as the first PCD frame is acquired, the octree is initialized and its bounds set to the bounds of the PCD. For each subsequent PCD frame ($\mathcal{P}_t$), the octree structure is grown and adapted to fit the new data in. In general the use of octrees allow for compact representation and fast indexing. This in turn gives good scalability properties. An illustration of an octree is given in Fig. D.4.

D.4.3 Modeling Measurement Uncertainties

The uncertainty in the measured ranges is affected by a number of things such as image noise, matching inaccuracies, and low-pass effects from using a correlation window in the stereo algorithm. Additionally, the actual range accuracy is governed by camera calibration errors, lens distortion and camera alignment errors.

We apply a similar assumption as in [119] that the image error in the stereo

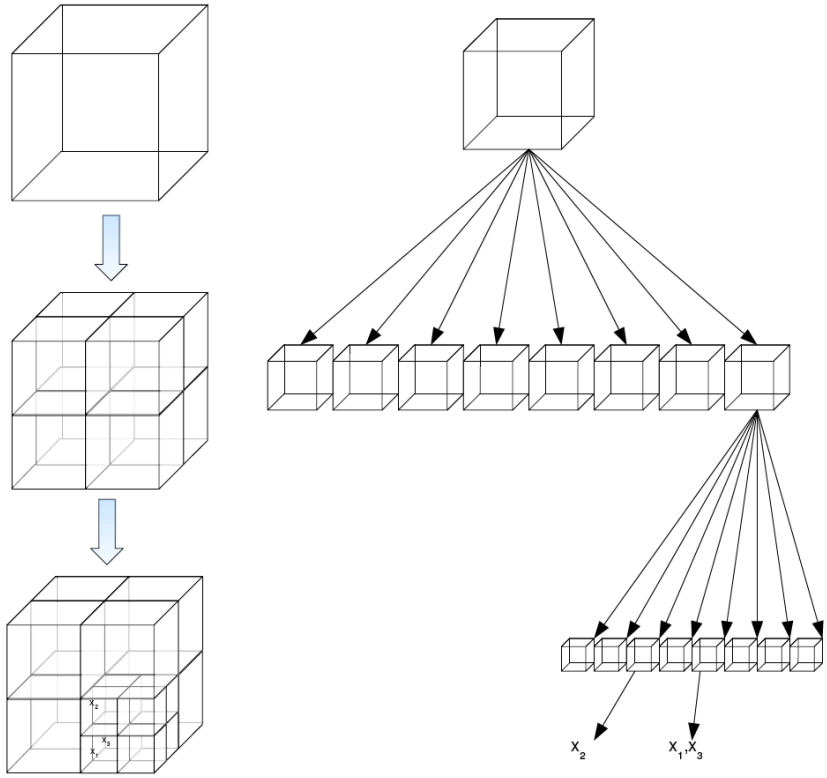


Figure D.4: Octree Decomposition. Each cube of space can recursively be split into 8 sub-cubes called nodes. At the bottom of the hierarchy (node leaves) the 3D points themselves are stored. Here illustrated with points $\mathbf{x}_1, \mathbf{x}_3$ at the end of one leaf, and $\mathbf{x}_2$ at the end of another.

matching algorithm is governed by:

$$\left(\sigma_v^2, \sigma_u^2, \sigma_d^2\right) = (0.5, 0.5, 1.0) \tag{D.3}$$

These are the assumed variances for a point in the disparity image of being in a specific row (σ_v^2) and column (σ_u^2), as well as having a specific disparity (σ_d^2).

The diagonals of the covariance estimate for the 3D projection of a point in the image is then:

$$\begin{aligned} \sigma_x^2 &= \frac{b^2\sigma_u^2}{d^2} + \frac{b^2\left(u-u_0\right)\sigma_d^2}{d^4} \\ \sigma_y^2 &= \frac{b^2\sigma_v^2}{d^2} + \frac{b^2\left(v_0-v\right)\sigma_d^2}{d^4} \\ \sigma_z^2 &= \frac{f^2b^2\sigma_d^2}{d^4} \end{aligned} \tag{D.4}$$

The covariance $\mathbf{Q}_l$ in camera 3D coordinates is then:

$$\mathbf{Q}_l = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix}. \tag{D.5}$$

This is rotated from the local to the global coordinates as:

$$\mathbf{Q}_g = \mathbf{R}_{l2g}^T \mathbf{Q}_l \mathbf{R}_{l2g}, \tag{D.6}$$

where $\mathbf{Q}_g$ is the covariance in global frame and $\mathbf{R}_{l2g}$ is the rotation matrix from local to global coordinate frames.

The estimated location of the point in 3D is transformed to the global frame by:

$$\bar{\mathbf{x}}_g = \mathbf{R}_{l2g}\bar{\mathbf{x}}_l + \mathbf{T}_{l2g}, \tag{D.7}$$

where $\mathbf{T}_{l2g}$ is the translational from local to global.

For each point in the octree we store not only the coordinate of the point but also now its covariance Q_g. As we won't be looking up points based on their covariance it does not add further dimensionality to the octree structure. It is simply metadata that gets attached to a point.

D.5 Point Filtering

When a new disparity image is acquired, all existing points in the octree which lie inside the field of view of the camera and inside stereo-camera range are transformed into the new local frame and projected back into the disparity image. The octree structure allows us to find these relevant points in a fast and efficient manner. Classification techniques are used to detect and isolate points (distance measures), which represent artifacts in the stereo imaging.

D.5.1 Artifact detection

Let each measurement $(\mathbf{x}_i)$ be associated with a covariance $\mathbf{Q_i}$ according to D.5. We then wish to express the divergence (distance between stochastic distributions) between measurements $[\mathbf{x}_i, \mathbf{Q_i}]$ and $[\mathbf{x}_j, \mathbf{Q_j}]$.

A general measure of distance between distributions f_i and f_j [87], is the Jeffreys-Matusita (JM) divergence defined as

$$J_{ij} = \left(\int_\Omega \left(\sqrt{f_i(r)} - \sqrt{f_j(r)} \right)^2 dr \right)^{\frac{1}{2}} \tag{D.8}$$

The JM distance has the salient feature to be easily applicable on arbitrary distributions. The JM distance $J_{ij} = 0$ when the distributions $f_i(r)$ and $f_j(r)$ are equal and overlapping. The JM distance takes the value $J_{ij} = \sqrt{2}$ when the two distributions are totally separated. Bhattacharyya introduced the parameter

$$\rho_{ij} = \int_\Omega \sqrt{f_i(r)} \sqrt{f_j(r)} dr, \tag{D.9}$$

and the negative logarithm, α_{ij}, of this quantity,

$$\alpha_{ij} = -\ln \rho_{ij}, \tag{D.10}$$

to obtain

$$J_{ij}^2 = 2(1 - \rho_{ij}) = 2(1 - \exp(-\alpha_{ij})) \tag{D.11}$$

[54] showed that when the two distributions are normal multivariate of degree n: $f_i(r) = N(\mathbf{x}_i, \mathbf{Q}_i)$ and $f_j(r) = N(\mathbf{x}_j, \mathbf{Q}_j)$ then

$$\alpha_{ij} = \frac{1}{8} (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{Q}_{ij}^{-1} (\mathbf{x}_i - \mathbf{x}_j) \tag{D.12}$$

$$+ \frac{1}{2} \ln \frac{\det(\mathbf{Q}_{ij})}{\sqrt{\det \mathbf{Q}_i \det \mathbf{Q}_j}}, \tag{D.13}$$

$$\text{where } \mathbf{Q}_{ij} = \frac{\mathbf{Q}_i + \mathbf{Q}_j}{2}.$$

Further, the probability of misclassification P_e is bounded:

$$\frac{1}{16}(2 - J_{ij}^2)^2 \leq P_e \leq 1 - \frac{1}{2}(1 + \frac{1}{2}J_{ij}^2), \tag{D.14}$$

which is equivalent to

$$\left(0.5\exp(-\alpha_{ij})\right)^2 \leq P_e \leq 0.5\exp(-\alpha_{ij}). \tag{D.15}$$

In this context we calculate α_{ij} from Eq. D.13 and use the upper bound in D.15 to estimate misclassification. When a new point is outside this bound, we consider the point an artifact of the 3D stereo processing and discard the point.

D.5.2 Startup and algorithm procedure

A mapping can be initialized by a prior map or it can be initialized by a first stereo image of the map. Hence there exist octrees occupied by sets $[\mathbf{x}_i, \mathbf{Q}_i]$ and subsequent stereo images provide $[\mathbf{x}_j, \mathbf{Q}_j]$, $j = i+1, i+2, ..i+N$. If the first two points are outside the accepted divergence, a third is processed, until at least two points agree within the chosen value of α_{ij}. If the point lies within the interval then it is assumed that both measurements pertain to the same object and the map is updated by merging the two distance estimates into a new estimate given by:

$$\begin{aligned} \mathbf{Q}_m^{-1} &= \mathbf{Q}_i^{-1} + \mathbf{Q}_j^{-1}, \\ \bar{\mathbf{x}}_m &= \mathbf{Q}_m \left(\mathbf{Q}_i^{-1}\bar{\mathbf{x}}_i + \mathbf{Q}_j^{-1}\bar{\mathbf{x}}_j\right). \end{aligned} \tag{D.16}$$

Subsequent measures are compared with $[\mathbf{x}_m, \mathbf{Q}_m]$, which replaces $[\mathbf{x}_i, \mathbf{Q}_i]$ in the calculations. The two measurement i and j are deleted from the octree and replaced by the merged estimate m.

D.5.3 Algorithm simplification from Octree quantization

To gain computational efficiency, the procedure can be simplified when the minor axis of the covariance ellipse have dimensions lower than those of the octree. Then points need only be considered that lie on the projection between camera and object. This simplification is supported by the fact that main source of artifacts in the disparity image is a wrong disparity value, which gives a wrong range estimate along the ray. This is done efficiently by projecting the points in the map back to the disparity image. All points which land in the same pixel lie

along the ray.[2] To handle occlusions, i.e. a point in the map lies behind a new measurement and α_{ij} is outside the upper bound, the point is not discarded but measurements are taken to belong to different objects.

D.6 Results

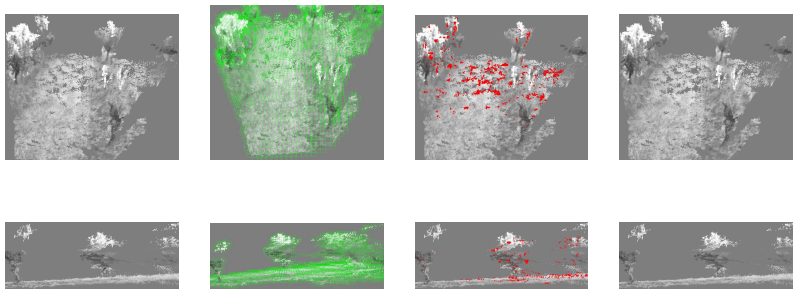
An example of the results for filtering is shown in Fig. D.5. The sequence contains some ground and a few trees. In Fig. D.5(c) some faulty measurements are removed near the ground plane. Specifically in the bottom image of Fig. D.5(c) many of these points are groups of data, which were not present in the real environment. The Figure has been done using only 2 disparity images. This is mostly for illustration purposes as the 3D map gets messy if all faulty measurements are shown for larger amounts of disparity images. Of special note is that most of the points being removed are at long range. This is consistent with the fact that range measurements are less accurate at long range.

The method has been evaluated on a manually recorded dataset of an orchard. Due to space limits, only parts of the results are given here. Fig. D.6. shows the 3D map for the trajectory in headland. This is the part at the southern end of the field in Fig. D.1. The total distance covered in this part of the trajectory is roughly 114 meters. As seen there is little noise in the final map. Due to lack of texture there are some holes in the map in certain parts of the image. We currently do not consider dynamically moving obstacles which means that the person moving at the end of the field appears multiple times in the final map.

D.7 Conclusions

This paper presented a method for fault detection in depth maps generated from stereo data through multiple depth map fusion and geometric reasoning. We presented the theory behind our approach, and gave a demonstration of its efficiency on data collected by an agricultural robot in an orchard environment. The proposed method can account for measurement noise at different ranges while trying to detect faults in stereo data. The implementation showed that computational effort need be considered and the fault-tolerant 3D imaging could need a dedicated processor to run in real time. Extensions could well be to

[2]This concept is similar to that employed in computer graphics called z-buffering to decide which elements of a rendering scene are visible, and which are hidden.

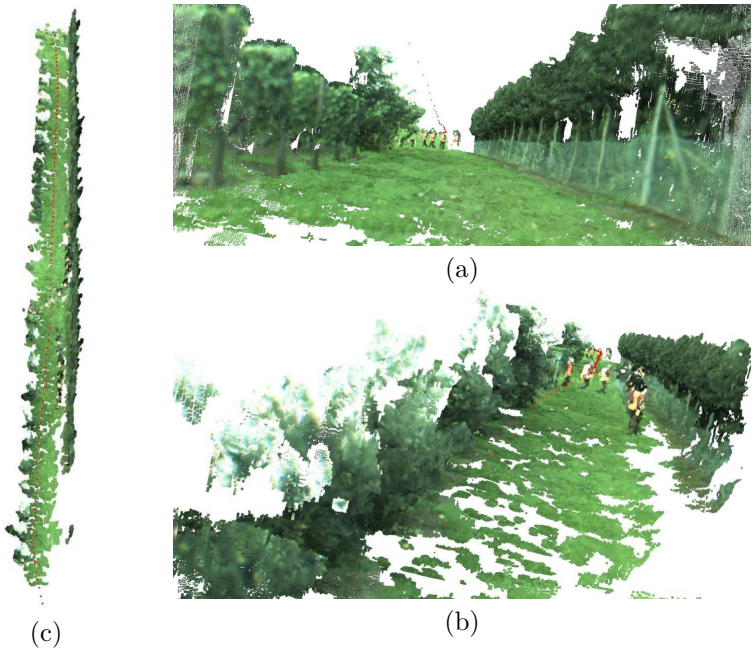


(a) (b) (c) (d)

Figure D.5: Results for the filtering for faulty range measurements. (a) Raw data (b) Raw data overlayed with octree cubes (c) Red are faulty measurements (d) Filtered data.

combine semantic labeling techniques for the objects in the map [115], as well as extend the method to dynamic obstacles.

Acknowledgments The authors would like to thank Dr. M. Agrawal at SRI International for numerous advices and for providing the VO library. We would also like to thank Dr. J.C. Andersen at DTU for collecting the used dataset.



(a)

(b)

(c)

Figure D.6: Three views of headland in the orchard from the 3D map reconstruction. Blue dots represent the trajectory of the camera poses. The fence can be clearly seen to the right. The orchard rows are on the left side. A moving person taken from different views can be seen in the distance. (a) view from below camera. (b) view from above camera. (c) Top-down view.

Paper E

Natural Environment Modeling & Fault-Diagnosis for Automated Agricultural Vehicle

Morten Rufus Blas and Mogens Blanke. Natural Environment Modeling & Fault-Diagnosis for Automated Agricultural Vehicle. *Proc. 17th IFAC World Congress*, pages 1590-1595, Seoul, Korea, 2008. Published.

Abstract:
This paper presents results for an automatic navigation system for agricultural vehicles. The system uses stereo-vision, inertial sensors and GPS. Special emphasis has been placed on modeling the natural environment in conjunction with a fault-tolerant navigation system. The results are exemplified by an agricultural vehicle following cut grass (swath). It is demonstrated how faults in the system can be detected and diagnosed using state of the art techniques from fault-tolerant literature. Results in performing fault-diagnosis and fault accomodation are presented using real data.

E.1 Introduction

Agricultural machinery is increasingly getting automated. For example, agricultural vehicles have seen a revolution in automation by adoption of GPS for automatic steering. A number of technical limiting factors of these GPS systems however do exist. Some of these are faults inherent to GPS receivers, such as those induced by satellite occlusions and multipath errors. Others include that the GPS systems require a detailed navigation plan as they themselves cannot "see" features in the field that want to be followed. It is thus of interest to find a solution to these limitations.

Earlier research results can be split into 3 broad categories: work in fault-tolerance, stereo-vision, and position estimation for navigation in agriculture. Stereo-vision is an active topic in agriculture. It has been shown that stereo-vision can be used for navigation by finding the relative position of a vehicle to a variety of agricultural structures: [112], [62]. Relative position estimation has been fused in [111]. Visual odometry has been fused in [4]. Sensor faults in these articles are typically treated in simple fashion by gating on, for example, the innovation in a kalman filter. Literature however exists to treat fault detection systematically, [14], [15], by using systematic fault-tolerant design tools. They have been demonstrated to work in practice on for example ships: [13]. How the systematic methods as presented in [15] can be applied to navigation in agriculture and especially with vision-based navigation sensors has not previously been demonstrated.

This paper will deal with a specific field operation that involves the agricultural vehicle to follow cut grass (swath) in order to pick it up with a baler (see fig. E.1). The system to be analyzed is equipped with stereo-vision, a single antenna GPS and an IMU, in one configuration. GPS positions of the vehicle

Figure E.1: This is an example of a swath that the navigation system should follow. The vision system locks on and tracks the swath most central in the image.

that formed the swath are known. The combination of stereo-vision and GPS allows the system both to "see" the swath but also navigate based on the given map. This creates system redundancy that is essential for achieving fault-tolerance. A visual odometry algorithm on the stereo-camera allows for the relative positionment of the vehicle without GPS or IMU. The GPS receiver used was a high-end EGNOS receiver and the IMU was a tactical grade (low accuracy) MEMS based unit.

The two main ideas presented here is first a behavior model for representing the natural environment, namely the swath. Secondly, it is shown how parts of this model (the swath location) can be used in conjunction with sensor inputs to create a fault tolerant sensor fusion system. The fault diagnosis is illustrated using real data. Combining the sensor information optimally for state estimation beyond estimating faults is not delved into.

E.2 Swath Model

A model of the swath requires extracting the salient features of the environment required for the field operation and storing them in the model representation. The salient features are the location of the swath and the distribution of the

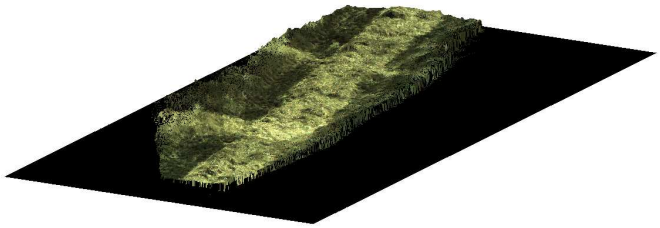


Figure E.2: Using position information and 3D data from the stereo-camera the entire field (if necessary) can be reconstructed in 3D. This is a 15 m section of a swath.

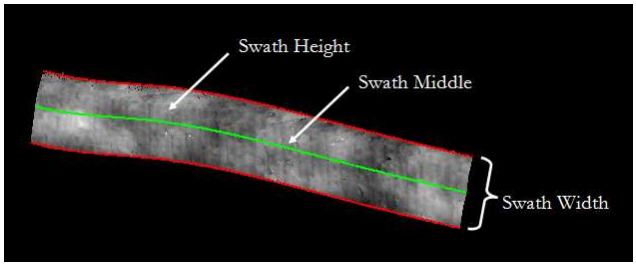


Figure E.3: The middle of the swath can be represented by a cubic spline with a number of knots. This is accompanied by the width of the swath at each location along the spline as well as the height of the material within the bounds of the swath. The height of the swath is shown as different intensities. The illustration is from a top-down view.

swath material across the swath. The location of the swath is modeled as a cubic spline with the number of knots being fixed for a certain length of swath. The spline is positioned along the middle of the swath. The distribution of the swath material is first modeled by defining the swaths width at any given point along the spline. Secondly, points inside the 2D volume enclosed by the swath width are assigned a value pertaining to the height of the swath at this point relative to the ground. The swath height is represented by a grid map of resolution 2 cm for each grid point. The model is illustrated in fig. E.3. The concept behind the swath location, width, and height is now explained in more detail.

E.2.1 Swath Location

The swath location is defined as being in a 2D coordinate system on the ground plane. A function f represents the lines down the middle of the swaths. Given coordinate pairs (x, y) then f is:

$$y = f(x) \tag{E.1}$$

The model of the swath location is then $s(x)$ with $s \in \mathcal{S}_3(\boldsymbol{k}_{0:n})$, where $\boldsymbol{k}_{0:n}$ are the spline knots and the spline coefficients and $\mathcal{S}_3$ is the cubic spline domain. Then the model is equal to the swath location plus the approximation error ϵ_a of fitting a spline to f:

$$s(x) = f(x) + \epsilon_a \tag{E.2}$$

Based upon the concept of having a controller that allows the vehicle to follow the swath location, the x-track error ε_x (signed shortest distance from the control point to the spline) can be found as a function of the tractor position and the spline. Defining the spline s_b in body coordinates a function $\mathcal{E}$ can be set to find the x-track error:

$$\varepsilon_x = \mathcal{E}(s_b) \tag{E.3}$$

E.2.2 Swath Width

For each point on s the swath width is defined by the two points on either side of s that are orthogonal to s at these points and which lie a distance d_w away from s at each point. The function g defines d_w as a function of x and the model s. The tangent of s at a specific point is given by differentiation of s. This allows for the swath width to change along the length of the swath.

$$d_w = g(x, s) \tag{E.4}$$

E.2.3 Swath Height

The swath height is the mean swath height inside each grid square in the grid map bounded by the swath sides as defined by the swath width and location. Given the grid map coordinates (x, y) the swath height z_m at this point is defined by the function h:

$$z_m = h(x, y) \tag{E.5}$$

E.2.4 Swath Sets

The above model is only for a single swath. Each swath in the field is then described by a set of the above functions with $\mathcal{M}$ being the set of such swaths and n being the number of swaths:

$$\mathcal{M} = \{s_i, g_i, h_i\}, i = 1...n$$

E.3 Behavior Models

The following sections outline how the behavior-based model of the swath can combined with models of GPS, stereo-vision sensor and odometry data. The purpose is to arrive at a set of constraints that can be used for analysis of system structure and subsequent generation of residuals for fault diagnosis. This idea was brought into the field of fault diagnosis by [126] and later expanded, see [125] and [15]. The advantage of this approach over classical methods, [136], include the ability to use a formulation of behaviors at high level of abstraction.

E.3.1 GPS

The GPS positions of the vehicle that formed the swath is fitted with the spline (see fig. E.4). The distribution of the swath cannot be measured by the GPS receiver. Knowledge of the machine settings used to construct the swath are assumed known giving an approximate height h_{est} and width $d_{w,est}$ of the swath. These settings can be used as estimates for the swath model fitting. Given the measured GPS positions $\hat{\mathbf{p}}_2^n$ of the vehicle and the attitudes of the vehicle $\hat{\mathbf{\Theta}}_2$

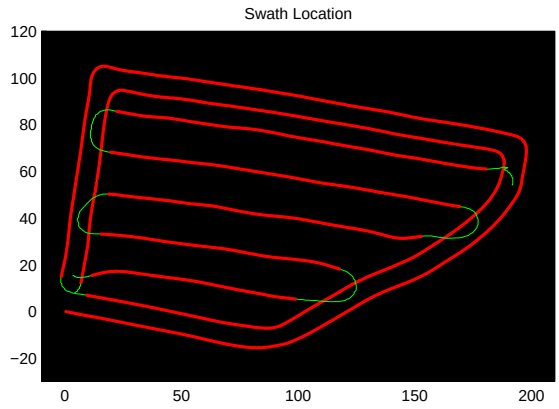


Figure E.4: The location of the swath as recorded by the GPS mounted on the vehicle that formed the swath. It is assumed the start and end points of the GPS trajectory have been recorded. The swath location is shown in red and has been fitted with a spline. Green indicates the trajectory followed by the vehicle where it was not forming swath - raw GPS readings. All units are in meters in the NED coordinate system.

the position of swath formation behind the vehicle can be calculated. The path formed by these positions is then fitted with the spline model using the function k to provide an estimate of the swath location. The information in the GPS map is then, in an abstract formulation:

$$s_g = k(\hat{\mathbf{p}}^n, \hat{\mathbf{\Theta}}_2) \tag{E.6}$$

E.3.2 Stereo-Camera

A stereo algorithm is used to find the correspondence between features in the left and right image sensor (i_l, i_r). The position of the features relative to the stereo-camera can then be inferred in 3D. Modern vision algorithms then exist to register 3D models with the 3D point cloud provided by the stereo-camera: [36]. An algorithm has been constructed that allows such registration between the swath model and the 3D points. The stereo-algorithm and registration will be denoted by the function a_{reg}. Thus, given the two images a measurement of the swath location s_c, width g_c, and height h_c can be computed for the part

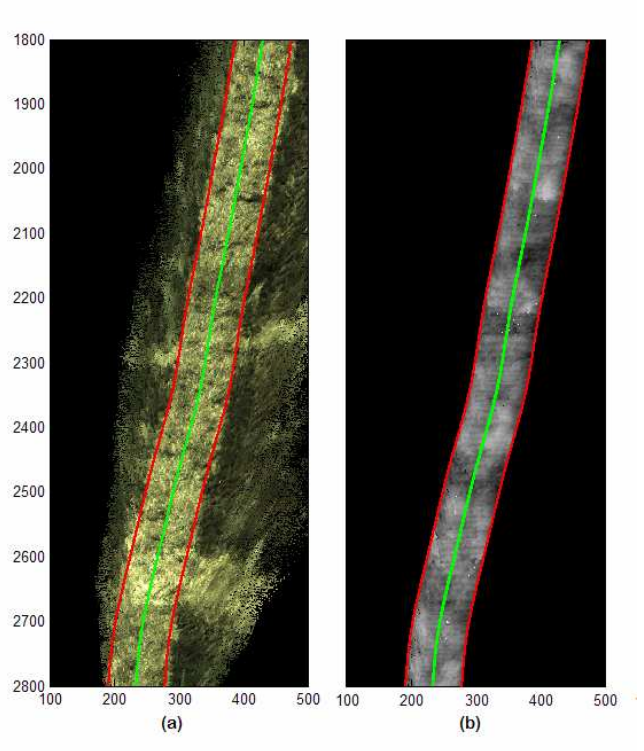


Figure E.5: (a) RGB topdown-view of the swath with swath location and swath width superimposed on the swath. (b) The swath model showing both swath location, width and height by image feature extraction.

of the swath in the image.

$$\begin{bmatrix} s_c \\ g_c \\ h_c \end{bmatrix} = a_{reg}(i_l, i_r) \tag{E.7}$$

These measurements are stored in a map representation for an individual swath. s_m is the spline formed by combining previous measurements.

E.4 Structural Model

The structural model describes the behavior of variables in the normal, fault-free system using a behavior based approach. A violation of a behavior indicates a fault in the system. Given the current setup a fault could for example be in a sensor, an algorithm, and/or an assumption about the environment. The structural approach treats faults unambiguously.

The constraints are composed of a number of measurement (m), differential (d), and system constraints (c). The variables in the constraints are likewise composed of two groups: the subset of known variables $\mathcal{K}$ and the subset of unknown variables $\mathcal{X}$. The constraints are listed in Eq. E.9.

$$\begin{aligned} K &= \{\mathbf{v}^b, \mathbf{a}^b, \mathbf{p}_1^n, s_g, s_c, s_m, \mathbf{R}_b^n(\Theta, \lambda)\} \\ X &= \{\mathbf{p}^n, \dot{\mathbf{p}}^n, s^b, s, \varepsilon_x\} \end{aligned} \tag{E.8}$$

$$\begin{aligned} c_1 &: s^b = \mathbf{R}_b^n(\Theta, \lambda)s + \mathbf{p}^n \\ c_2 &: \varepsilon_x = \mathcal{E}(s^b) \\ d_1 &: \dot{\mathbf{p}}^n = \frac{d}{dt}\mathbf{p}^n \\ m_1 &: \mathbf{v}^b = \mathbf{R}_b^n(\Theta, \lambda)\dot{\mathbf{p}}^n \\ m_2 &: \mathbf{a}^b = \frac{d}{dt}\mathbf{R}_b^n(\Theta, \lambda)\dot{\mathbf{p}}^n \\ m_3 &: \mathbf{p}_1^n = \mathbf{p}^n \\ m_4 &: s_g = s \\ m_5 &: s_c = s^b \\ m_6 &: s_m = s^b \end{aligned} \tag{E.9}$$

Following the notation in [33]: $\mathbf{v}^b$ is the tractor's velocity vector over ground seen in body coordinates as measured by visual odometry; $\mathbf{a}^b$ is the acceleration vector given by the IMU; $\mathbf{p}^n$ the position in (North, East) coordinates with $\mathbf{p}_1^n$ being the position measurement from the GPS; $\mathbf{R}_b^n$ is the rotation matrix from body to navigation frame, which is a function of Θ, the attitude vector (Euler angles roll, pitch and yaw) and of λ, the latitude. s^b is the spline body coordinates. In this analysis, the $\mathbf{R}_b^n$ matrix is assumed to be known.

Table E.1: Incidence Matrix.

	known						unknown		
\	s_g	s_c	s_m	$\mathbf{v}^b$	$\mathbf{a}^b$	p_1	$\mathbf{p}$	$\dot{\mathbf{p}}$	ε_x
m_1				1				1	
m_2					1			1	
m_3						1	1		
m_4	1						1		1
m_5			1						1
m_6		1							1

Table E.2: Dependability Matrix.

	constraints						
\	d_1	m_1	m_2	m_3	m_4	m_5	m_6
p_1	1	1		1			
p_2		1	1				
p_3				1	1	1	
p_4						1	1

E.5 Structural Analysis

As described in [13] a structural analysis is then performed on the structural model. The constraint d_1 is a differential constraint and as such cannot fail. To simplify the structural analysis the constraints $\{c_1, c_2\}$ are pulled into the measurement constraints $\{m_4, m_5, m_6\}$ and faults in them are treated as subsystem faults in the respective measurement constraints instead. The incidence matrix can be seen in table E.1. A number of parity relations are then found as shown in Table E.2.

Based on the dependability matrix in Table E.2, the parity relations are derived in analytical form and used as the basis for the residuals (Eq. E.10). As all the columns of Table E.2 are different, it follows that all faults should be structurally detectable and isolable as the faults will have a unique signature in these residuals.

$$
\begin{aligned}
r_1 &: \mathbf{v}^b - \mathbf{R}_b^n(\Theta,\lambda)\frac{d}{dt}\mathbf{p}_1^n = 0 \\
r_2 &: \frac{d}{dt}\mathbf{v}^b - \mathbf{a}^b = 0 \\
r_3 &: \mathcal{E}(s_c) - \mathcal{E}(\mathbf{R}_b^n(\Theta,\lambda)s_g + \mathbf{p}_1^n) = 0 \\
r_4 &: \mathcal{E}(s_c) - \mathcal{E}(s_m) = 0
\end{aligned}
\tag{E.10}
$$

E.6 Field Tests

The properties of residuals were investigated based on recorded data. The data stems from the test field run illustrated in fig. E.4. The position of the swath was first logged by following the middle of the swath manually - emulating the vehicle forming the swath. This was repeated for a second pass emulating the vehicle that should pick up the swath. This provides some form of limited ground truth. The position error of the driver is bounded between the runs as he constantly steers relative to the swath. Experience with driving with balers puts the error associated with not driving exactly over the center of the swath to under $\pm 0.2\,\mathrm{m}$ as this is required to pick up the swath successfully. In the data examined the GPS has a false offset in the second pass relative to the first pass of approximately $0.6\,\mathrm{m}$ for the first approx. $70\,\mathrm{s}$ before it corrects its position estimate to bring it to about $0.15\,\mathrm{m}$ of the swath location. This second offset is acceptable for normal operation. Field tests enabled calculation of residuals r_1, r_3 and r_4 as an instrumentation issue prevented data reception from the IMU. The field test is hence representing a case of one permanent failure and an additional fault occurring.

E.6.1 Detailed Design of Residual Generators

The parity relations Eq. E.10 are now further scrutinized as a basis for change detector and hypothesis evaluation design. A common assumption for readily available change detection algorithms is that of a Gaussian amplitude distribution. A required property of residuals for average run length calculations is whiteness. Fig. E.7 shows the histograms of residuals. The figure also shows Gaussian distributions with mean and variance as observed. Residual 1 suffers from a deadband in the calculation of velocity from the stereo images, hence, the distribution is not Gaussian. The cause of the deadband must be investigated further. Residual 3 appears to follow a shifted Rayleigh distribution.

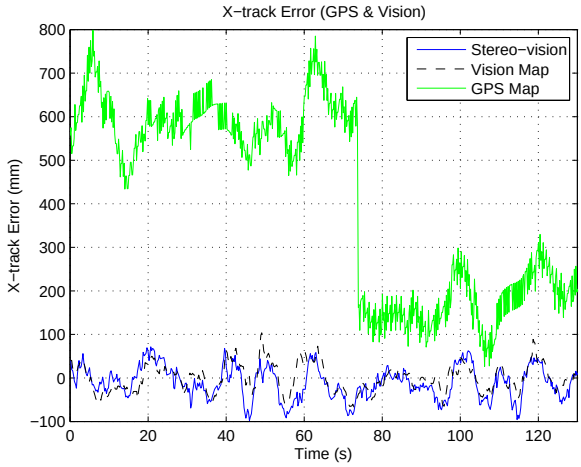


Figure E.6: The vehicle was driven manually over a swath. The driver centered the vehicle over the middle of the swath and drove for 2 min while maintaining this centered position. The x-track errors from the subsystems were recorded.

Statistical change detection will ideally be based on a log-likelihood ratio test

$$s_i = \ln(p_{\theta_1}(r_i) - \ln(p_{\theta_0}(r_i)) \tag{E.11}$$

where the probability densities from the observed distributions should be used, p_{θ_1} for the case of a fault, p_{θ_0} for the normal case, respectively. With the shifted Rayleigh shape distribution of r_3 in the no-GPS-fault case, a change detection of CUSUM or GLR type is straight forward to compute, [9]. The Rayleigh distribution gives, however, some computational burden over the tests when Gaussian distributions are assumed. The testing was therefore conducted using a Gaussian assumption. In the above analysis, it is also noted that a GPS fault is strongly detectable in residual r_3 while only weakly detectable in r_1 due to the position differentiation in the parity relation of r_1. A low-pass filter is hence applied on r_1.

With respect to testing whiteness of the residuals, Fig. E.8 shows the auto-correlation functions for the three available residuals. Residual r_3 is seen to comprise some correlation due to filtering within the algorithm that determines the spline approximation to the swath. Whiteness is particularly important to reach design conclusions about average run length.

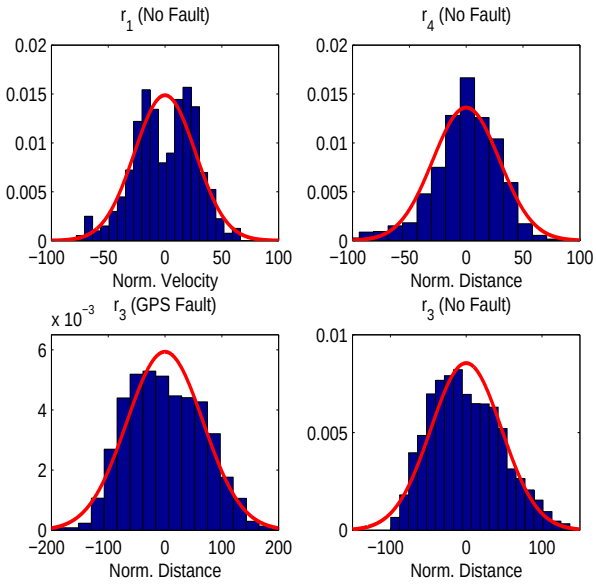


Figure E.7: Normalized histograms for residuals r_1, r_3, r_4 in the faultless scenario along with r_3 in the case of the GPS fault. Fitted gaussian distributions are shown on top of the histograms.

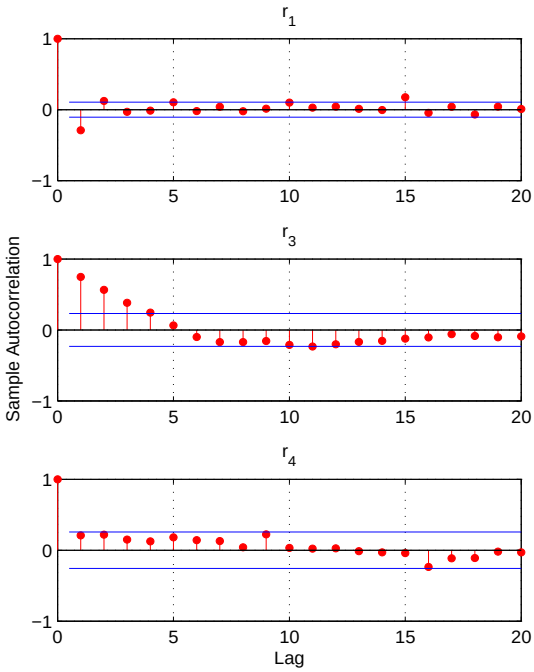


Figure E.8: Autocorrelation function for residuals r_1, r_3, r_4 in the faultless case. The blue lines indicate the bounds of the 95% confidence intervals.

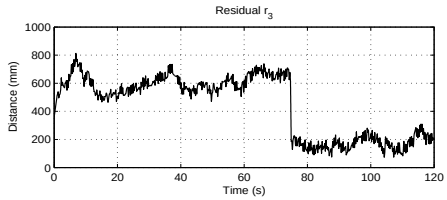


Figure E.9: Residual r_3.

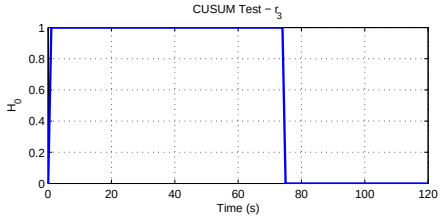


Figure E.10: CUSUM test for residual r_3. $\mathcal{H}_0 = 0$ indicates no fault.

E.6.2 Change Detection and Hypothesis Evaluation

Change detection on the residual vector could be made using a vector-based approach, where a known signature $\rho(\tau) = [\rho_1(\tau), \rho_3(\tau), \rho_4(\tau)]^T$ is sought for in $\mathbf{r}(t) = [r_1(t), r_3(t), r_4(t)]^T$. The computational burden is, however, larger than by applying a simple CUSUM test for change in mean on r_3 and r_4 and make a threshold test on r_1. With adequate logics used for hypothesis testing, this allows for isolation of faults. The result for r_3 is shown in fig. E.10. The CUSUM test is given a mean value to test for so that an erroneous x-track error of up to 0.2 m is permissible. From the system is started it takes 1 s to isolate a fault. The CUSUM test for residual r_4 stays 0 for the data indicating that s_c and s_m agree, and that any variation between them is due to noise.

Isolation of the fault to the GPS subsystem (GPS sensor and GPS Map) can, as minimum be achieved by considering the residual vector $[r_3, r_4]^T$. Stronger isolation is achieved by considering the full residual vector $[r_1, r_2, r_3, r_4]^T$ where the output of the GPS subsystem is also compared to the visual odometry which in turn is compared to the IMU output. The GPS jump is clearly seen as a spike in residual r_1 as shown in Fig. E.11.

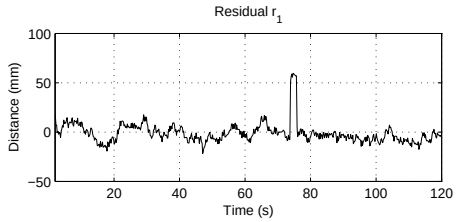


Figure E.11: Low-pass filtered residual r_1.

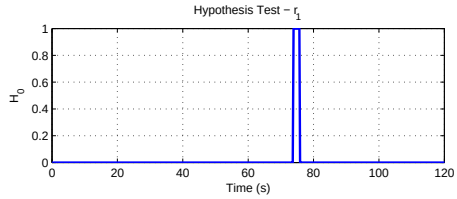


Figure E.12: Hypothesis test by thresholding for residual r_1.

E.7 Fault Handling

There are a number of approaches to fault accomodation. In the present case, a GPS fault can be isolated, and the magnitude of the fault can be estimated. It is then a straight forward task to reconfigure the navigation controller either to avoid using the faulty sensor or to attempt to compensate for the fault by compensating the sensor readings. Given that the residuals have detected the fault in the GPS x-track signal it is indeed possible to handle the fault automatically. The approach adopted here was to do this is by estimating the GPS fault magnitude using a Kalman filter and the vision x-track signal. The estimated mean difference is used as the correction estimate. Threshold detection on $r_1$is used to prevent invalid signals from being passed to the controller. The response of the thus corrected position estimate is shown in Fig. E.13.

It can be seen that discrepencies in the GPS and vision x-track signals quickly converge (the two signals in the example are roughly within $\pm 10\,\mathrm{cm}$ of each other). This approach only works for correcting faults which are stationary errors to the input signal. It is stipulated that most other GPS faults will be caught by the visual odometry or IMU.

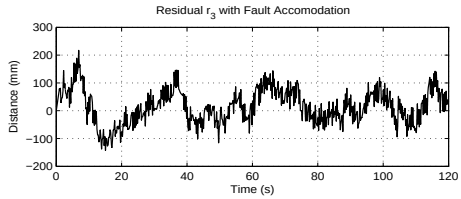


Figure E.13: By estimating the magnitude of the fault in the GPS sensor it was possible to do fault accomodation.

E.8 Conclusion

It has been demonstrated both using principles from fault diagnosis and from fault-tolerant sensor fusion theory, as well as experimentally, that an agricultural vehicle equipped with GPS, IMU and stereo-vision can be made fault-tolerant to sensor faults. The results presented here were done offline and work will be pursued to demonstrate them online. Real-time implementations are available for the image processing algorithms. The computational requirements for the fault-tolerant sensor fusion framework is insignificant compared to the image processing. The system should thus have a very good chance of working online.

Paper F

Automatic Baling Using Stereo Vision and Texture Learning

Morten Rufus Blas and Mogens Blanke. Automatic Baling Using Stereo Vision and Texture Learning. *J. of Computers and Electronics in Agriculture*, 2010. Submitted.

Abstract:
This paper presents advances in automated baling using stereo-vision. A robust classification scheme is developed for learning and classifying based on texture and shape. Using a state-of-the-art texton approach a fast classifier is suggested that can handle non-linearities and artifacts in data. Shape information is employed to make the classifier robust to large variations in lighting conditions and greatly reduce the likelihood that artifacts in signals from the stereo vision system lead to gross errors in estimated object positions. The classifier is tested on data from a stereovision guidance system on a tractor. The system is shown to be able to classify cut plant material (called swath) by learning it's appearance. A 3D classifier is successfully used to train the texture classifier. It is demonstrated from field tests how fault-tolerant fusion of steering reference data are obtained for an automated baling vehicle.

F.1 Introduction

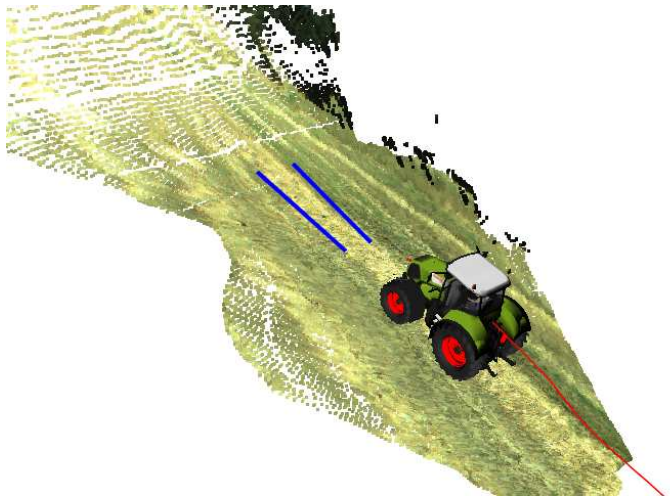


Figure F.1: The system in context. The red line shows the motion of the camera, the blue lines are the swath boundaries inside the camera's field of view.

Autonomous navigation for outdoor, unstructured environments is an important research problem in robotics with numerous applications. Being able to

recognise specific structures in the environment will in many cases enhance the ability of a robot to make the right navigation decisions. Typical outdoor systems rely on range sensors such as stereo cameras or laser range finders for reasoning about the geometry of the world and identifying geometrical shapes.

This paper combines a 3D classifier and a texture classifier for a general problem related to agricultural robotics. In agriculture typical tasks are to follow structures in the field to for example plow, seed, spray or harvest. This paper focuses on the specific harvesting task of baling, which involves following rows of cut straw or grass (swath) in order to pick it up and process it into bales. This is a labour intensive and repetitive task which is of interest to automate. The difficulties pertaining to automating this task are similar to the difficulties in automating a large range of agricultural tasks. The ability to track this structure using 3D shape information from a stereo camera and/or GPS information has previously been demonstrated [18]. In order to add redundancy to the system this paper presents a classifier that uses online learning to learn texture information about the swath and the surroundings. This is then coupled with shape information to extract the swath position. A mapping system keeps track of measured swath positions. The map is then used to guide the vehicle along the swath by steering the tractors front wheels while a driver controls the throttle and brakes.

The main contributions of this paper are, a classifier framework with online learning of texture in an agricultural environment, a method of supervising and fusing 3D and texture in a map, improved results using state-of-the-art stereo camera positioning and the application to automatic baling with round-baler feedback.

F.2 Related Work

In agricultural robotics, most current efforts at following field structures without GPS use 3D such as in [52],[112],[24]. Generally speaking, texture methods have had limited success due to large variations in conditions. Colour methods are generally restricted to for example identifying green plants [8]. Existing camera-based methods generally do not use mapping and instead rely only on data from single images for steering. The mapping system implemented in this paper relies on visual odometry (VO) and GPS for positioning [66]. A simpler and much slower mapping approach using VO has previously been described in [61]. The advantage of VO over for example GPS is that it is generally more robust and accurate over shorts distances. In robotics, recent uses of classifying based on color/texture include [5] where it was used to classify terrain for a

Mars rover. In [105] a combination of visual and geometric features are used for outdoor classification. Texture using a bag-of-words type approach is used with the words trained off-line. This has the disadvantage that a very large set of filter-banks must be used in order to be able to handle all variations. The approach described in [17] forms the basis for the texture representation used in this paper. It has the advantage that the texture representation can be learnt online which allows the classifier to be optimised for individual scenes which reduces the need for a large set of filters.

There has generally been little efforts at improving redundancy in vision based guidance systems. The system presented here improves redundancy by combining 3D measurements, texture, and mapping. This is different than in [24] where a 2D laser scanner is also used to follow a 3D swath profile and mapping is required to extract an angle of the swath relative to the vehicle. Such systems require a clear 3D profile of the structure and do not work in areas where the structure is flat. In [24] the mapping comes from wheel odometry which is often inaccurate and subject to wheel slips.

F.3 System Overview

The automatic baling system is composed of two parts: a vision system and a control system (See Fig.F.2 and Fig.F.14). In the vision system, a left and right image are acquired from a stereo camera. A stereo algorithm is then used to extract 3D information from the images. This is then fed into a 3D tracking algorithm that tracks the swath field structure. A learning algorithm is then used to teach a texture tracking algorithm to track the swath. Supervision is then used to make the learning process fault-tolerant. Mapping then provides fusion of tracking information. To enable mapping a fault-tolerant positioning fuses and supervises visual odometry and GPS information. The output from the mapping system is fed to a track controller that in turn steers the wheels of the tractor.

F.3.1 Hardware

In a proof of concept configuration and for the present paper, image processing was done on a laptop which interfaces to the vehicle controller ECU through the tractor's CAN bus. The driver interfaces to the control system through a terminal to change settings and engages/disengages the automatic steering system through a switch. The baler has integrated pressure sensors which are

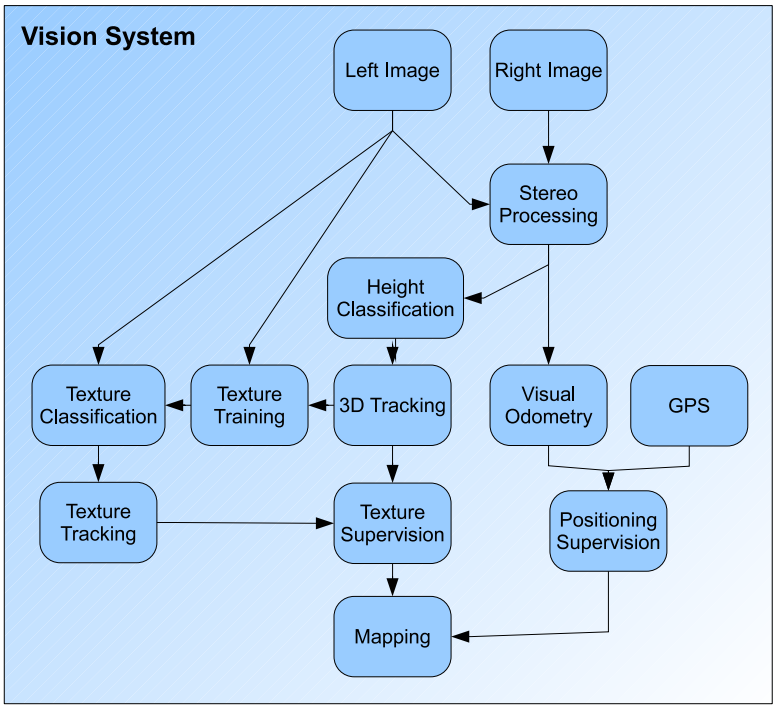


Figure F.2: Overview of the vision system. Images from a stereo camera are used to track 3D field structures. Online learning of texture enables texture tracking of the structure. Supervision of the learning process provides fault-tolerance. Visual odometry and GPS are fused and supervised to allow fault-tolerant mapping of the field structure.

used to measure the bale diameter. This information is used in the controller to assure an even filling of the bale chamber. A wheel angle sensor provides feedback about the angle of the front wheels of the tractor. Hydraulics allow actuation of the front wheels. A stereo camera is mounted in front of the tractor and an RTK GPS on the roof provides position information.

F.4 3D Classification

Stereo-vision perceives depth using triangulation. The distance to a point is determined by the triangle between the point and where it appears in each of two images. To do this the two images must be aligned. The process of aligning

the images is part of a calibration step. Given a calibrated stereo camera the images are then aligned by warping them. This is known as rectification. This gives two cameras with parallel optical axes and horizontal epipolar lines. A dense estimation of ranges is then performed at each pixel by matching along the epipolar lines. This is done using a correlation window with typical sizes of around 10x10 pixels. The correlation window matches texture in the two images with each other. The output of the matching process is a disparity image (See Fig.F.5). This gives the image difference between the position of objects in the two cameras. The horizontal distance from the image centre to the object image is dl for the left image and dr for the right image. Then the disparity value d is given by:

$$d = dl - dr \tag{F.1}$$

A pixel in the disparity image can then be projected to a 3D coordinate using triangulation.

For a stereo camera a point in the depth map is defined by (u, v, d) which is the column, row, and disparity value in the disparity image. This can be projected to and from a 3D coordinate defined by (x, y, z) in the camera coordinate system by:

$$\bar{x}_l = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{(u-u_0)b}{d} \\ \frac{(v_0-v)b}{d} \\ \frac{fb}{d}, \end{bmatrix} \tag{F.2}$$

where u_0 and v_0 are the column and row coordinate of the optical centre of the image in pixels. b is the camera baseline and f is the focal length for the rectified image.

F.4.1 Height Classifier

In order to identify the swath in 3D the ground plane is first estimated (See [66]). The distance from each pixel in the disparity image to the plane is then calculated (Using the 3D coordinates and the point-to-plane distance). Pixels below the plane are set to zero-distance. The height values are then normalised based on a median value of a subset of the largest height values. The output is a 2D image scaled from 0-1 with a higher value indicating a higher likelihood of pertaining to the swath (under the assumption that only the swath is higher than the ground plane). An example result is shown in Fig.F.2.

F.5 Texture Classification

In a seminal paper, Leung and Malik [74] showed that many textures could be represented and re-created using a small number of basis vectors extracted from the local descriptors; they called the basis vectors *textons*. While Leung and Malik used a filter bank, Varma and Zisserman [139] showed that small local texture neighbourhoods may be better than using large filter banks. In addition, a small local neighbourhood vector can be much faster to compute than multichannel filtering such as Gabor filters over large neighbourhoods.

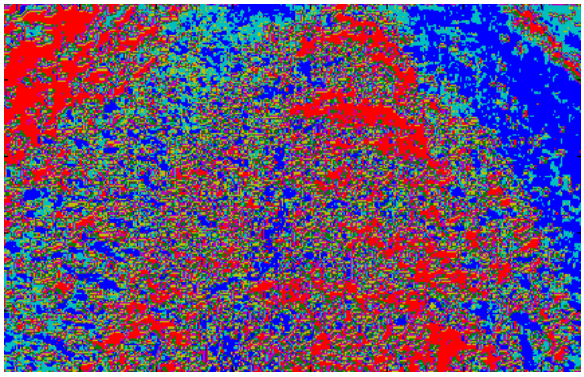
F.5.1 Texton classifier

Given a colour image as input pixel neighbourhoods in the image are grouped into belonging to 1 of 23 texton types (texton image). This number was chosen as a compromise between quality and speed. These textons are learnt from the training image (See [17]). This is done by first extracting a descriptor in the form of a vector from each pixel location in the image. For each location p_i the vector is:

$$p_i = \begin{bmatrix} W_1 * L_c \\ W_2 * a_c \\ W_2 * b_c \\ W_3 * (L_1 - L_c) \\ \vdots \\ W_3 * (L_8 - L_c) \end{bmatrix} \tag{F.3}$$

Where L_c, a_c, b_c is the colour of the pixel at this location in CIE*LAB colour-space. $(L_1 - L_c), ..., (L_8 - L_c)$ are the intensity differences between the pixel at this location and the 8 surrounding pixels in a 3×3 neighbourhood. The vector elements are then weighted using $\{W_1 = 0.5, W_2 = 1, W_3 = 0.5\}$. A K-means algorithm is then run on all these descriptors to extract 23 cluster centres which we refer to as textons. Each pixel location in the image is then classified as belonging to a texton by finding the nearest texton in Euclidean space. An example of such a classification is shown in Fig.F.3.

(a)



(b)

Figure F.3: The input RGB image is transformed to an image where each pixel is labeled by color as belonging to one of 23 textons. (a) The original image with a swath in the middle. (b) The labeled texton classification.

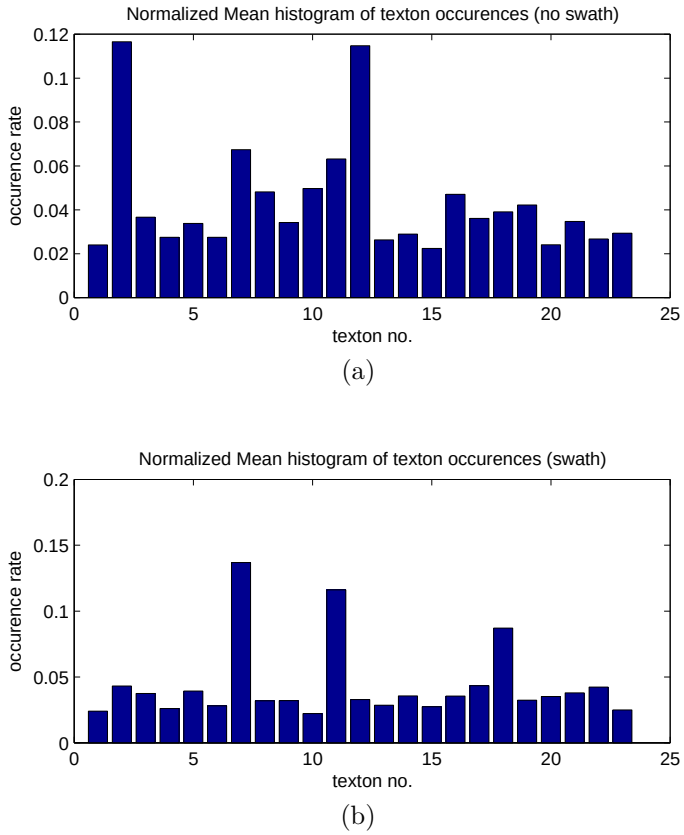


Figure F.4: Normalised Texton Histograms. (a) Texton occurrences for "no swath". (b) Texton occurrences for "swath".

F.5.2 Texture Training

The texton image is used as the basic input to train the texture classifier. Given a training mask representing the location of the swath in an image (representing the location of the "swath" as well as the surroundings "not swath"), and a texton image the average histogram of texton occurrences around 32×32 image patches for the "swath" and "not swath" case can be constructed (With 32×32 pixels corresponding to $1 - 2\%$ of the total image size). These are illustrated in Fig.F.4. The histograms are clearly different in the two cases, for example texton #2 occurs with high probability for the "swath" case and little for the "not swath" case.

As is apparent from the swath images in Fig.F.7 there are multiple objects with unique textures present in both the swath and in the area surrounding the swath. For example, stubble from harvested plants may be present in part of the image or tire tracks from other machines may form separate textures. The solution presented here is to identify and model each texture independently by representing each of them with a mean histogram. Identification the different textures is done by taking the list of histograms for the "swath" and "not swath" cases and clustering them (similarly to what was done for the textons) independently of each other using K-means [50]. For the results presented here, the number of clusters was 3 for each case. The reason for using K-means was simply for speed. Other probabilistic learning approaches could be used, including Support Vector Machines [19] and Gaussian Mixture Models.

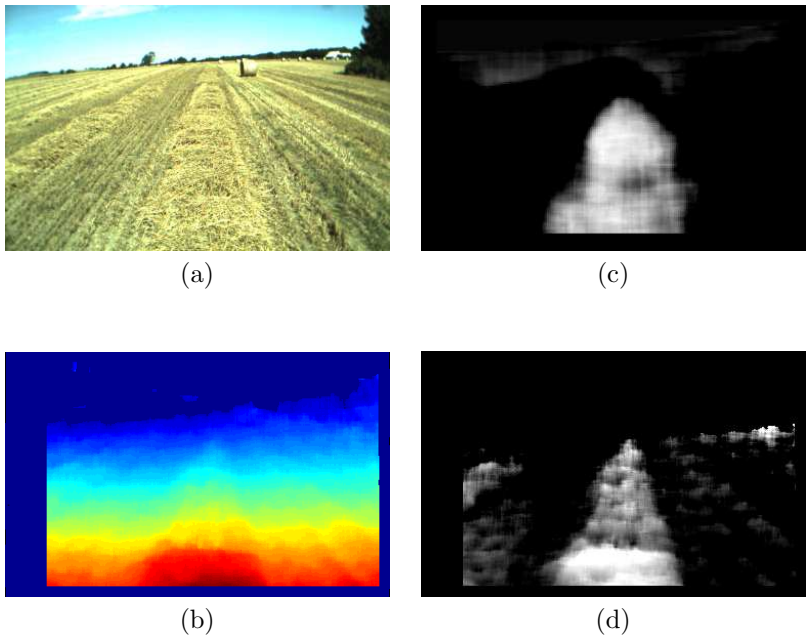
F.5.3 Texture Classifier

Classification is done on a texton image by identifying a likelihood that a histogram centred around a pixel is either "swath" or "not swath". This is represented by the two hypotheses: $\mathcal{H}_s$ for "swath" and $\mathcal{H}_n$ for "not swath". A suitable distance measure must be used to compare the histograms. Having considered several distance measures: Euclidean distance, histogram intersection kernels, Kullback-Leibler divergence, chi-square divergence, earth-movers distance, and Battacharrya distance, the sum-of-absolute (SAD) distances was chosen and the ratio of the distance from an observed distribution to either of the hypotheses were used for classification. The operator $\oplus$ is used here to denote the SAD distance. A main reason for using this distance measure was that it can be implemented to run very fast since it does not involve square roots or exponential functions:

$$a \oplus b = \sum_{i=1}^{k} |a_i - b_i| \tag{F.4}$$

All histograms in this context have the same number of elements and there is no need to normalise when calculating distances. Given a histogram $\mathbf{h}$ and a set of mean histograms for $\mathcal{H}_s$ denoted by $H_s = \{\mathbf{h}_{s,1},...,\mathbf{h}_{s,m}\}$ and $H_n = \{\mathbf{h}_{n,1},...,\mathbf{h}_{n,m}\}$ for $\mathcal{H}_n$ then the distance ratio is formulated as:

$$\text{distance ratio} = \frac{\min(\{\mathbf{h} \oplus \mathbf{h}_{n,1},...,\mathbf{h} \oplus \mathbf{h}_{n,m}\})}{\min(\{\mathbf{h} \oplus \mathbf{h}_{s,1},...,\mathbf{h} \oplus \mathbf{h}_{s,m}\})} \tag{F.5}$$



(a) (c)

(b) (d)

Figure F.5: Illustration of the texture and height classification. (a) original left image. (b) disparity image (warmer colors indicate shorter range). (c) Texture classified image. (d) Height classified image.

The distance is computed for each pixel in the input texton image. Similarly to the height classification, the image is normalised from $0 - 1$. Results are shown in Fig.F.5.

F.5.4 Swath Detection

The swath in the image is parameterised by a width, position and orientation. A mask can then be constructed for all feasible swath parameterisations (See Fig. F.6 for an example of a mask). An exhaustive search is then performed within a quantised set of the parameter space to maximise a match score. In Fig. F.6 the dark region has a value of -1, the white region has a value of 1. This mask is multiplied pixel-wise with the classified images for both the height as well as the texture, and then summed to produce a scalar value indicating goodness of a match.

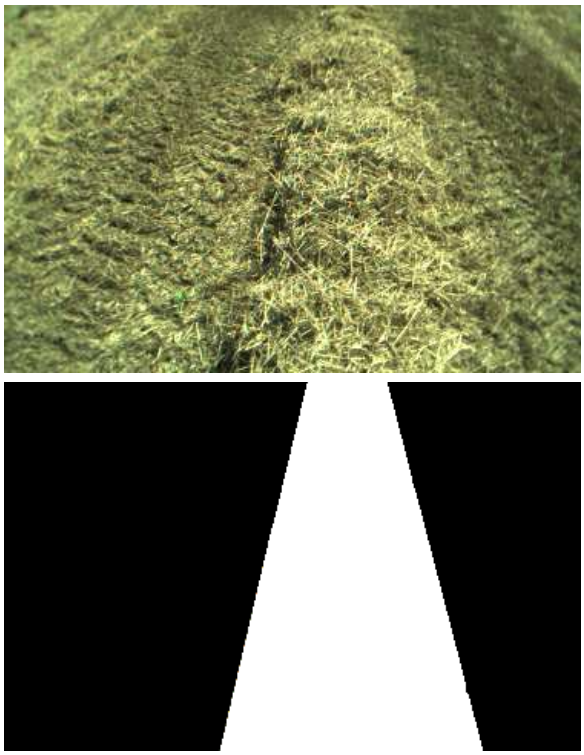


Figure F.6: (a) The original image with a swath in the middle. (b) Mask illustrating width, position, and orientation of swath extracted from stereo-algorithm.

F.5.5 Updating Texture Model

The texture training produces a model of the texture that the classifier uses. In the current implementation, training is performed at a rate of about 1 Hz. Training is only done if the 3D classifier match score is above a threshold. The newly trained texture model is then validated by testing the match score from using the old model against the new one. If better, the old model is discarded. It has been considered whether it could pay off to use an incremental texture model where the old model is not completely discarded. This approach had difficulties with local minima. In effect, as new models are constantly being tested, the learning system does have an incremental attribute in that often the method converges to finding a single training image that best represents the field variations. Nevertheless, it still has the ability of quickly switching model if a new type of environment is encountered.

F.5.6 Texture Results

In order to evaluate the performance of the classifier, a simple scoring scheme was designed. The classification rate was defined as the number of correct pixel classifications D_c normalised by the maximum number of correct pixel classifications, $D_{c,\max}$. High classification rate is the better. The false alarm rate was defined as the number of false pixel classifications (false alarms) D_f normalised by the maximum number of possible false pixel classifications $D_{f,\max}$. Low false alarm rates are preferred. A hand labeled image was used as ground truth.

$$\text{Classification rate} = \frac{D_c}{D_{c,\max}} \tag{F.6}$$

$$\text{False alarm rate} = \frac{D_f}{D_{f,\max}} \tag{F.7}$$

A set of 20 images were selected at random from 5 different data-sets of swaths (See F.7). A hand labeled set of ground truth images were made for all 100 images. For each of the sequences, the classifier was learnt from one image and then applied to the others. For this image the position of the swath was calculated using the 3D classifier that extracts the width and position of the swath based on the 3D profile in the stereo images.

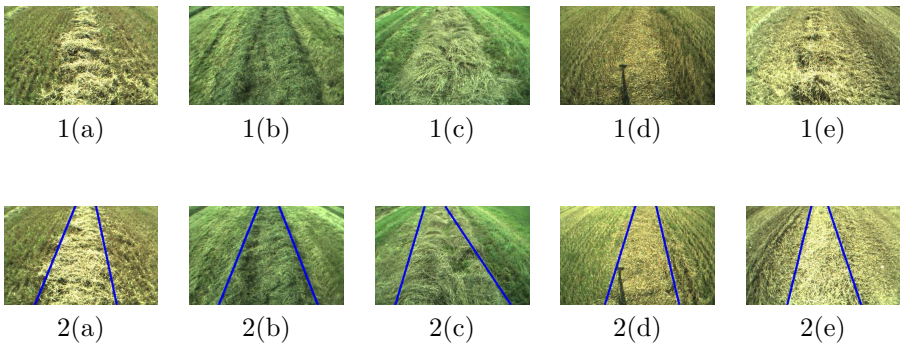


Figure F.7: Images 1(a-e) illustrate the training images used. Image #10 in each dataset is classified in 2(a-e) with the blue lines representing the swath bounds.

The results are very good with an average of around 90% detection rate relative to the ground truth. The false alarm rate is around 4% and is thus also good. Some images in data-set #3 have problems getting the heading of the swath correct as in Fig.F.7 where 2(c) has a slightly wrong heading. This error was attributed to limited resolution this far out meaning that there was not enough texture information to classify reliably on this specific swath. There was a spike on image #17 in data-set #4. This was due to an ambiguity in the image where the classifier chose a solution slightly to the left of the hand labeled image to compensate for a large lump of swath lying off centre relative to the rest of the swath. This is shown in Fig.F.9.

A false positive could potentially be triggered by a patch of grass or ground that has a shape that reasonably looks like a swath. Such objects were not present in the available dataset.

F.5.6.1 Texture Versus 3D

A comparison between the measured lateral and angular deviation of the swath position relative to the vehicle for the 3D and texture algorithms is shown in Fig. F.10. Clearly, the two follow each other closely. It is interesting to note that the two algorithms complement each other in that sudden drops in the match score seldom occur in both at the same time. The match score was on average higher for the texture method than for the 3D method in this sequence. This may be different from swath to swath depending upon how high it is and

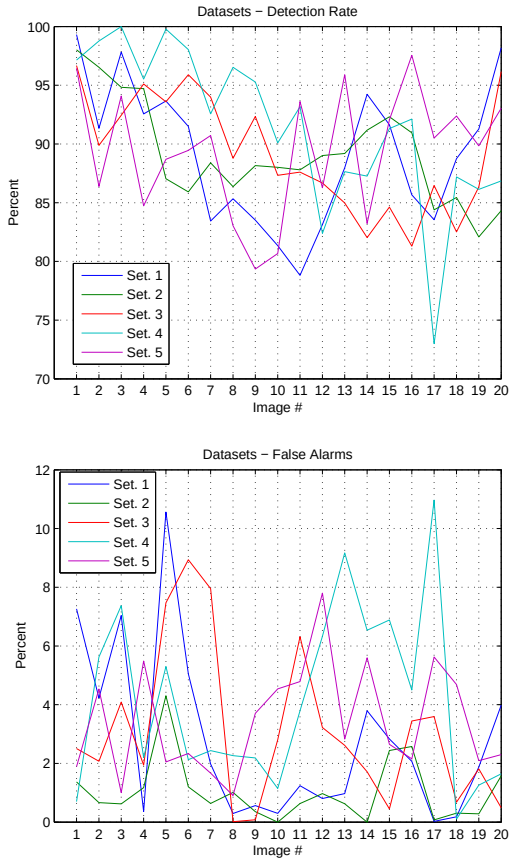


Figure F.8: Classification and false alarm rate for the tested sequences.

Figure F.9: This is the worst match in the used datasets according to the scoring scheme where the algorithm follows a lump of swath slightly to the left of the rest of the swath. Image #17 in dataset #4.

how slanted the edges are. Towards the end of the sequence (around frame 1500) the 3D method started to have problems detecting a height difference on a very flat section of swath, but the texture method continued driving to the end.

F.6 Mapping

The mapping system relies on recent research for high-precision positioning using VO fused with GPS (see [66]). By tracking image features between images, the change in pose of the camera can be estimated. This is done by computing changes in position between image frames. GPS provides global correction so drift in the VO subsystem can be avoided. This fused position estimate is used to maintain a map of the position of the swath.

The swath map is modeled as a Taylor series expansion of a clothoid [123]:

$$y(x) = y_0 + \tan(\phi)x + C_0\frac{x^2}{2} \tag{F.8}$$

Where y_0 is the lateral offset between the vehicle and the swath centre, ϕ is the angle of the swath relative to the vehicle. C_0 is the curvature of the swath.

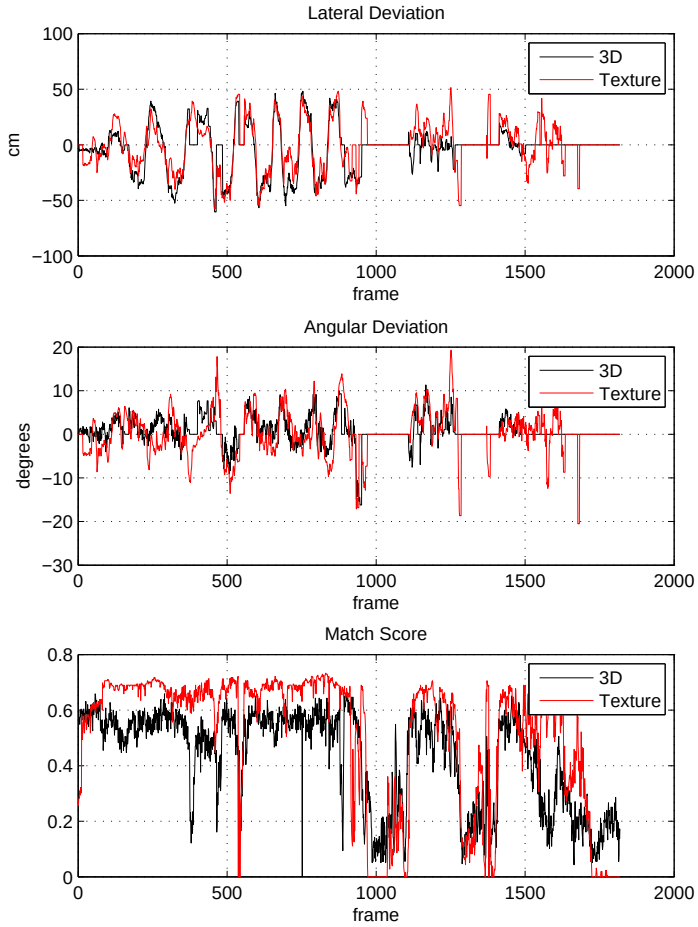


Figure F.10: Results for 1800 frames of video. The measured lateral and angular deviation of the swath relative to the vehicle plotted for both 3D and texture. The match score is shown for the two algorithms. There was no swath inside the field of view between frames 950-1100, 1270-1380, 1600-1800.

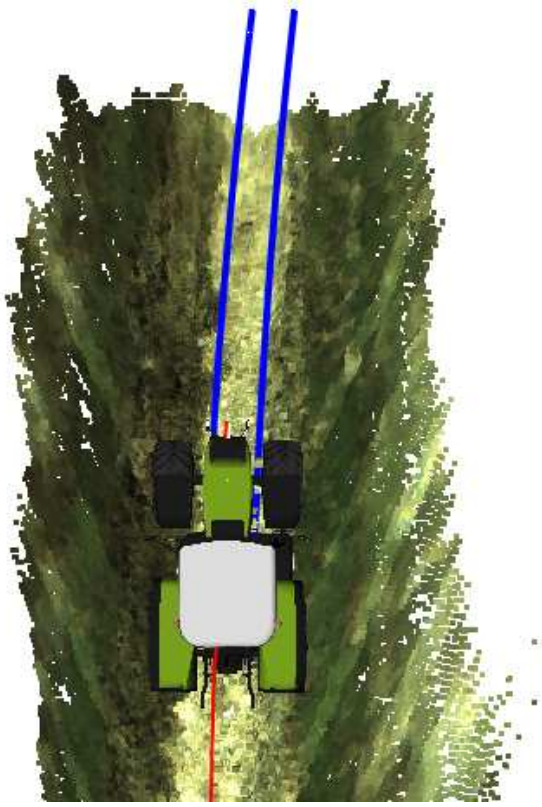


Figure F.11: The swath is modeled as a clothoid, illustrated here with blue lines and extrapolated beyond the field of view.

Estimating the curvature from a single image can be difficult. This is due to the natural variance of the swath position and width. To estimate the curvature requires a larger "field of view" such as that provided by this mapping system (Fig. F.11).

For each tracked image the centre point of the swath for each line in the image is extracted. These centre points are stored as a long list. When a new image is tracked, old centre points from the list which are inside the field of view are deleted and the new points added. The clothoid parameters are then estimated using least-squares on the point list. The map only extends from the camera field of view back to the pickup on the baler. Since the baler picks the swath up it does not make sense to keep track of a larger map.

F.7 Supervision and fault-tolerance

Two supervision systems are run concurrently in the software. The first operates solely on position sensor information and is based on the work described in [18] and is thus not repeated here (*Positioning Supervision* in Fig.F.2). The second is concerned with supervising the trained texture classifier (*Texture Supervision* in Fig.F.2).

To make sure the trained texture classifier is not faulty, a supervision system has been setup to validate it. This is done by evaluating the performance of the classifier relative to the 3D classifier over a number of frames. Change detection theory is then applied to determine if the texture classifier has similar tracking output. Two residuals are formed: one for the lateral and one for the angular deviations. These are calculated as the differences between the measured outputs of the 3D and texture based classifiers.

Statistical change detection will ideally be based on a log likelihood ratio test:

$$s_i = \ln \frac{p_{\theta_1}(r_i)}{p_{\theta_0}(r_i)})) \tag{F.9}$$

where the probability densities from the observed distributions should be used, p_{θ_1} for the case of a fault, p_{θ_0} for the normal case, respectively. A residual sample is denoted r_i.

The implemented change detection algorithm uses the standard CUSUM algorithm, see e.g. [15], to detect a change in mean between samples:

$$S(k) = \sum_{i=1}^{k} \ln \frac{p_{\theta_1}(r_i)}{p_{\theta_0}(r_i)} \tag{F.10}$$

Histograms of the residuals were analysed for cases with and without faults. These are shown in Fig.F.12. By inspection, faults appears to be given by a shift in the mean of the residuals r_i. For the CUSUM detector, only a change in mean need be considered and the residuals are assumed Gaussian. A change in mean of 20 cm and 5 ° are used to reject a trained texture classifier. The variance is estimated to be $\sigma^2 = 88.0\,\mathrm{cm}^2$ and $\sigma^2 = 16.5\,^{\circ 2}$ respectively. A sample window of $N = 20$ (roughly 2 seconds) was used to calculate the cumulative sum with tests for both a positive and negative change in mean. A fault flag was triggered if this value exceeds $\gamma = 5$ for the lateral fault, and $\gamma = 2$ for the angular fault.

For a change in mean of $A = 20\,\text{cm}$ over the sample window it corresponds to detecting a change in mean between $\mathcal{N}(0, \frac{\sigma^2}{N})$ and $\mathcal{N}(A, \frac{\sigma^2}{N})$. The probability of a false alarm P_{FA} and the probability of detection P_D is then given by [56]:

$$P_{FA} = Q(\frac{\gamma}{\sqrt{\sigma^2/N}}) = 0.0086 \tag{F.11}$$

$$P_D = Q(\frac{\gamma - A}{\sqrt{\sigma^2/N}}) = 1.0000 \tag{F.12}$$

Where Q is the right-tail probability. For the angular deviation, similar results can be obtained: $P_{FA} = 0.0138$, $P_D = 0.9995$. This gives a good detection rate while also assuring a reasonably low false alarm rate.

The residuals checks are run concurrently with the rest of the system and can be seen in Fig.F.13 for the run in Fig.F.16. The maximum value is shown at each time step for both positive and negative changes in mean (denoted by $g(k)$). In this run two faults are detected. Only the second fault is detected by both residuals.

Individual match scores for the two classifiers are first thresholded based on their match score. If they are both accepted, they are passed to the supervision system. If no faults are detected, the results are fused by taking a weighted mean of the lateral and angular deviations (based on the match scores). If a fault is detected, the texture classifier information is not added to the map. Mapping provides intelligent filtering of the classifier outputs so jumps are not experienced when switching between the information used. The approach outlined gave good fault-tolerance against artifacts in the image processing.

F.8 Control

A tracking control system to collect the swath is shown in the block diagram in Fig. F.14. The control system remains active as long as the match score is above a predefined threshold and there is map information ahead of the vehicle such that a reference track can be computed. The control system has a variable offset that is calculated to ensure the bale chamber is filled evenly. If the bale chamber is unevenly filled then the bale becomes cone shaped. Pressure sensors inside the baler provide a measure of how evenly it is filled. The bale must have

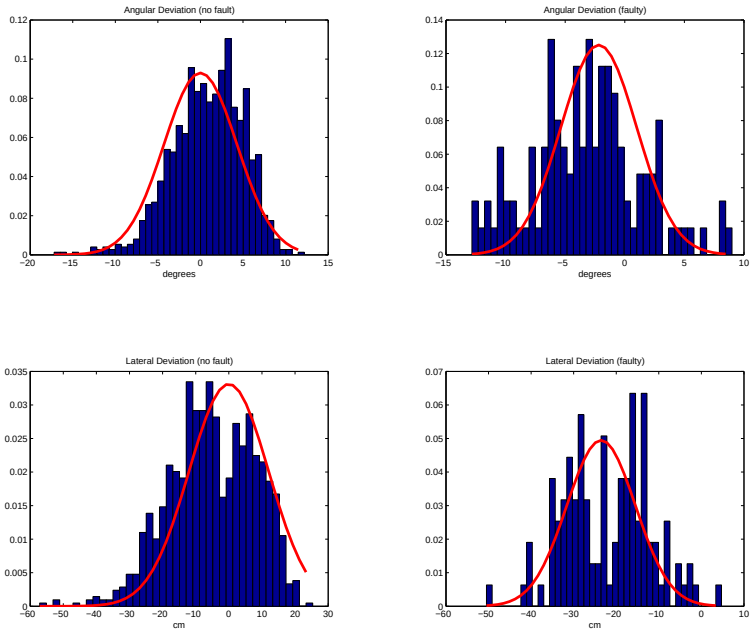


Figure F.12: Normalised histograms for residuals with and without faults. The histograms are shown with fitted Gaussian probability density functions.

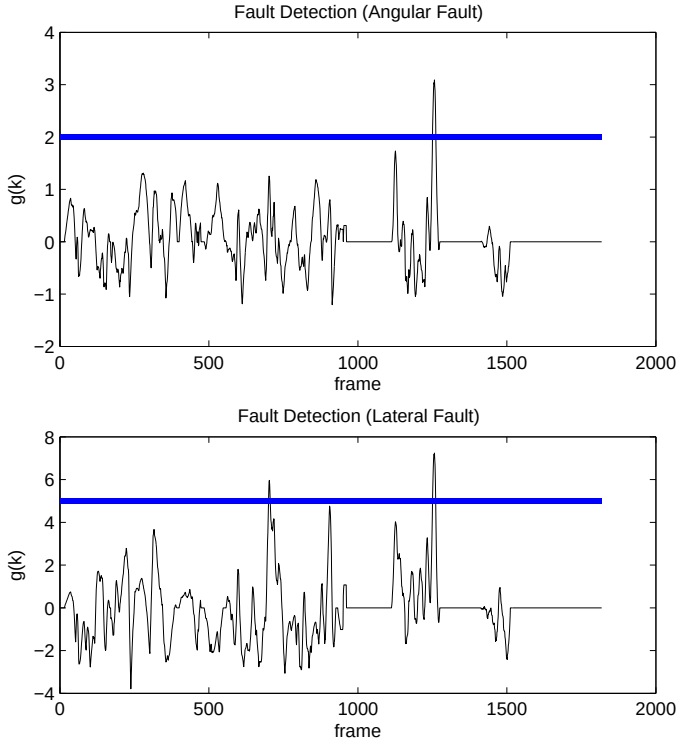


Figure F.13: Fault detection in angular and lateral deviations using a CUSUM change detector. The thick line defines the threshold for a fault.

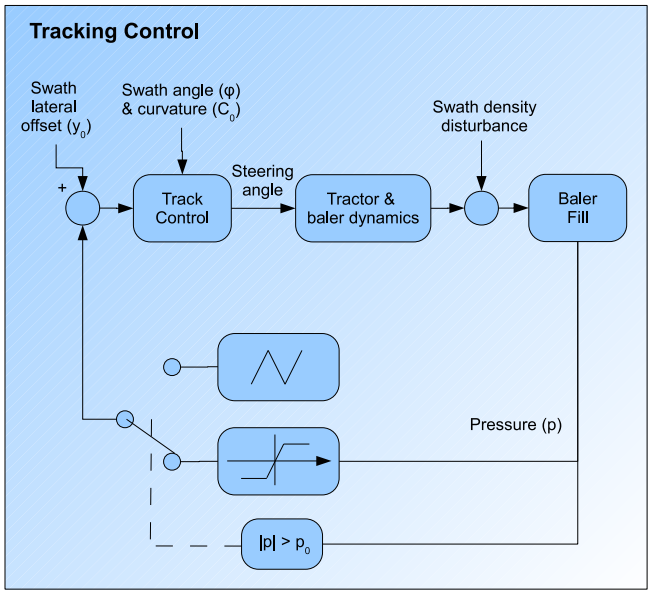


Figure F.14: Block diagram of tracking control for baling. An inner loop provides cone correction of the bale, an outer loop uses the swath tracking vision system as tracking error.

a certain size before the pressure sensors give usable feedback. To compensate for this lack of feedback the controller has two states. In the initial state where pressure has not yet built up an open-loop steering pattern is followed where the vehicle changes between being positioned on the left edge of the swath and then the right edge of the swath. This motion is parameterised as how far out this edge is relative to the centre of the swath; how long it should follow an edge, and how sharply it should change sides. The steering system changes mode when the bale size reaches a minimal level and sensor feedbacks can be used. The sensor feedback provides a signal in the range -1 to 1 indicating how cone shaped the bale is. An adequately amplified version of this signal is added to the measured lateral offset. The swath parameters y_0, ϕ, C_0 are finally fed to the tracking controller.

F.8.1 Results

Results for the system operating live are provided in Fig.F.16. The results span a period of about 800 s and involves creating 7 bales. The system steers

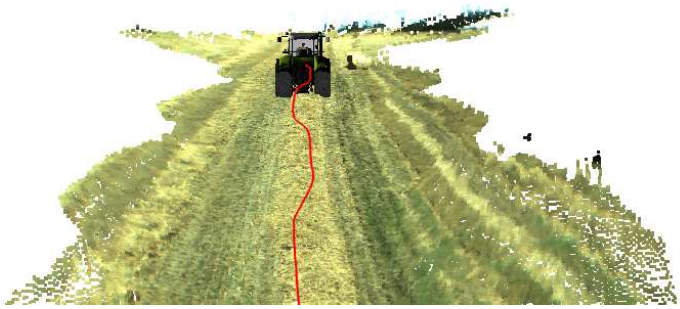


Figure F.15: The open-loop zig-zag motion of the vehicle is here clearly illustrated by the red line. The vehicle in turn drives on the left edge of the swath and then on the right edge.

autonomously. There were 3 drops in the match score which were where the swath ended in the headland and manual steering was required to bring the vehicle to the next length of swath. During the turning periods the lateral deviation measurements were ignored. The match scores shown have been normalised from $0 - 1$. The vehicle needed to stop when ejecting a finished bale, which can be observed from the velocity plot. Similar results have been obtained at speeds up to $20km/h$, which was the legal restriction for maximal speed of the autonomous vehicle.

F.9 Conclusion

The results presented in this paper showed that textures present in outdoor agricultural environments can be learnt and tracked robustly. 3D data provided by the stereo camera facilitated such learning. Geometric shape constraints allowed an initial sorting of false positives from a swath from analysis of width, position, and orientation. Supervision provided further detection of faults by comparing the learnt model to 3D tracking. Future work for the texture classifier will involve eliminating the need for online learning and thus make it more

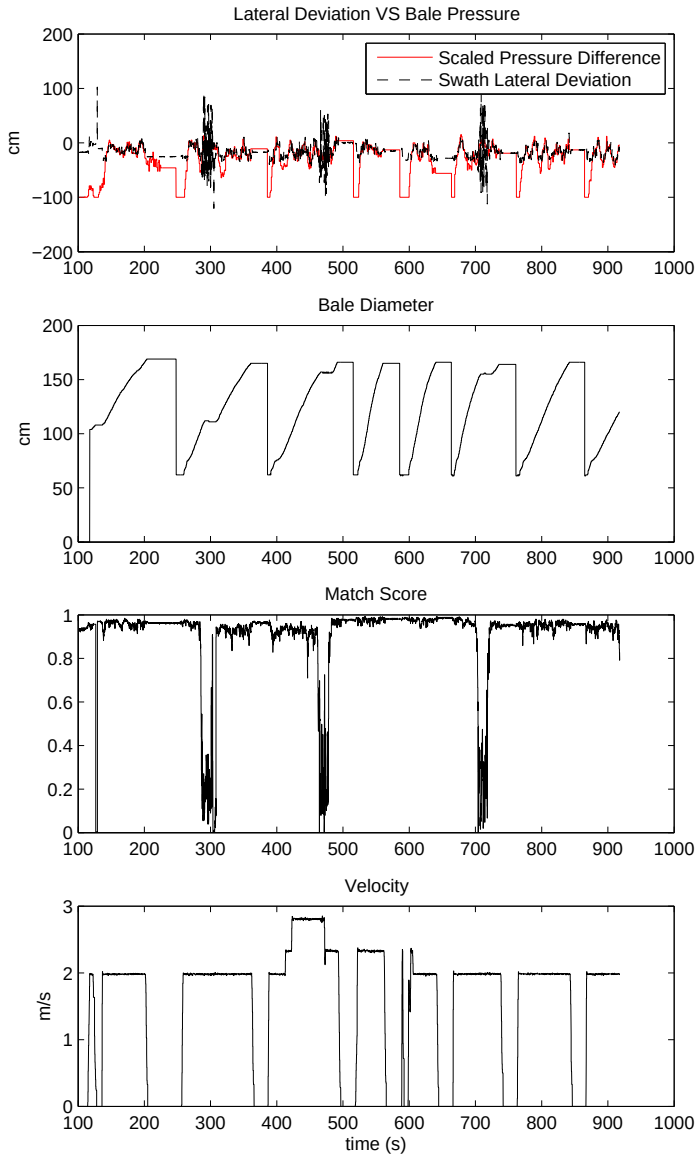


Figure F.16: Results for a 13 minute run where 7 bales are created. The controller is using the feedback signal from the baler.

capable to perform independently of 3D data.

The mapping component allowed additional parameters such as the swath curvature to be tracked and increased robustness of the overall system. Extensive tests demonstrated the system to work online on an autonomous vehicle on a variety of swath.

Paper G

Stochastic Automata with Optimal Signal Quantisation for Classification of Outdoor Environments

Fabio Caponetti, Morten Rufus Blas and Mogens Blanke. Stochastic Automata with Optimal Signal Quantisation for Classification of Outdoor Environments. *Control Engineering Practice*, 2009. Submitted.[1]

[1]This work was supported by The Danish Food Industry Agency under contract 3412-06-01729.

Abstract:
A stochastic automaton is introduced for the purpose of classifying different types of environments of a mobile robot. Perception uncertainties are managed by applying a probabilistic model to quantised signals. An automaton models the spatial relationships between environments and is in turn used for classification. Quantisation is designed based on cumulative probability densities of sensor signals for particular environments. Learning algorithms provide automatic tuning of the modeling and quantisation process to assure a low level of misclassification by the automaton. Data recorded on an autonomous agricultural robot are used to compare the proposed quantisation method and automata-based classification against other state-of-the-art classification techniques. Considering methods that can assess the confidence of classification, an essential feature for use in supervision, the new method is shown to compare very favourably to those existing. The states of the stochastic automaton are shown to be intuitively linked with behaviour modeling and with the optimal signal quantisation, the method provides fast learning and very good saleability. The classification ability in the natural test environment is close to the best of comparable algorithms.

G.1 Introduction

The ability to classify is a fundamental requirement for robots to be able to perform a large range of tasks autonomously. Classification is a subproblem in robotic perception. Robot systems are typically composed of modules for perception, planning, and control. For many applications, perception is usually the weakest link. Thus it is very important to come up with robust methods for amongst other things recognising objects and scenes.

What makes perception difficult is that it often involves high dimensional data such as those coming from a video stream. Sensor signals may often be noisy and have a number of shortcomings in terms of accuracy and range. The classification problem in turn involves classifying objects with very high complexity that can be difficult to model.

Robotic research has turned to learning algorithms to automatically create such models from sensor data. Basic statistics is typically extracted from sensor data and fed into learning algorithms. Currently, these statistics have to be designed by hand and act as a first step in reducing the dimensionality of the input data. Learning is often supervised using hand-labeled training data. An issue here

is that training data can only be a small subset of all expected variation. A system that is to recognise a tree would, as example, only train on a subset of different trees recorded from a subset of different sensor positions relative to the tree. A learning algorithm must hence be good at generalising about objects. With high dimensionality on a limited set of training data, it means that the data will generally be very sparse which in turn makes it hard to generalise. Dimensionality reduction is a way of generalising. Dimensionality reduction is also necessary to make classification problems tractable to compute.

This paper demonstrates a novel method of applying stochastic automata to the task of classification. Data recorded from an autonomous agricultural robot is used as the basis for classifying different types of environments that the robot frequents. The states of the automata are modeled a-priori and represent the different environment types. The state transitions model how the different environment types are connected and provide a topological map of environments.

A stochastic automaton extends the notion of non deterministic discrete-event systems in such a way that the frequency of occurrence of the different events can be addressed. Perception and structure uncertainties are managed using quantised signal spaces [79] and by modeling the spacial relations between semantic places with probabilities. Quantisers for continuous perception signals will be designed in order to maximise the classification capability of the automaton.

Fig. G.1 illustrates modeling of the environment with a stochastic automaton. The real valued input and output are quantised and fed to the model. Through abstraction, the model is tuned to fit the current environment. Knowing the current discrete input and output, the current state probability is estimated through an observation algorithm. Having the state estimate, the robot plan execution can be supervised, controllers be redefined or the localisation process can be boosted while assuring high levels of safety and reliability.

G.2 Background and Related Research

State-of-art for environment classification include methods like Support Vector Machines (SVM), Adaboost classifiers and Gaussian mixture emitting Hidden Markov Model (GHMM) [142].

SVM is a general class of supervised learning techniques based on statistical learning theory used for classification and regression problems.

Adaboost is a boosting technique which linearly combine simple weak classifiers

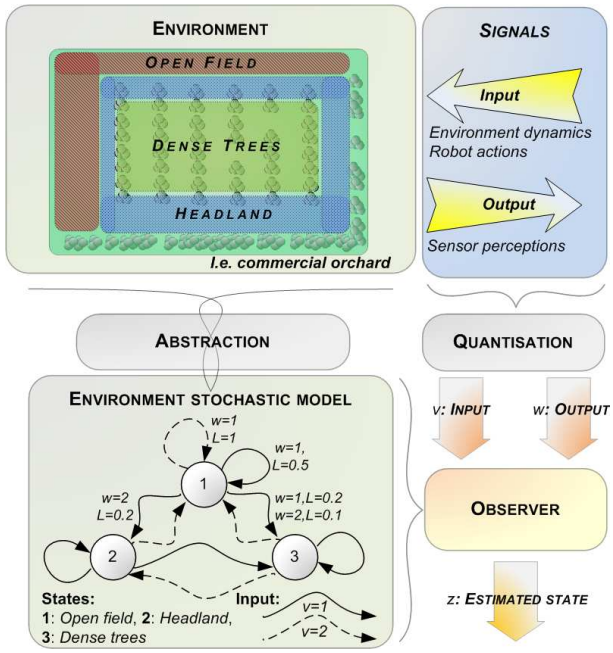


Figure G.1: Modeling the environment by a stochastic automaton with quantised input and an observer to estimate state probability

on the basis of the classification performances on a training set. Such classifiers have been used previously by [94] to classify indoor places. Several approaches were proposed in the literature to improve performance by taking advantage of object recognition [99] and probabilistic environment models [95].

A GHMM consists of a discrete time and discrete space Markov process that contains some hidden parameters and emits observable outputs [107]. A GHMM is built for each possible state by using labeled observations to train the Gaussian mixtures characterising the emissions.

A finite state machine was used to detect and classify film scenes, [143]. Structural information of the scenes together with low and mid-level features were used to classify the scenes for a better content retrieval. Such approach is here extended and ported to the problem of classification and diagnosis of the behaviour of an autonomous mobile robotic system.

G.2.1 Stochastic automaton

The supervised system is modeled as a discrete-event system subject to input with tailored output. The states in which the model evolves are the locations in which the hybrid system works. The dynamic behaviour of the model is described by changes in the discrete signals values, referred to as events. The system's discrete input, state and output are denoted by v,z and w. Their discrete value sets are enumerated as,

$$\begin{aligned} \text{Input:} \quad & v \in \mathcal{N}_v \subset \mathbb{Q},\ \mathcal{N}_v = \{1, 2, \ldots, M\}, \\ \text{State:} \quad & z \in \mathcal{N}_z \subset \mathbb{Q},\ \mathcal{N}_z = \{1, 2, \ldots, N\}, \\ \text{Output:} \quad & w \in \mathcal{N}_w \subset \mathbb{Q},\ \mathcal{N}_w = \{1, 2, \ldots, R\}. \end{aligned}$$

Using the notation of [118], an initialised stochastic automaton is described by the $five-tuple$:

$$\mathcal{S} = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, L, P(z_k)) \tag{G.1}$$

L is the behavioural function, the law that governs the stochastic process underlying the automaton and is defined as,

$$L : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \rightarrow [0, 1] \subset \mathbb{R}$$

$$L(z', w, z, v) = P(z_{k+1} = z', w_k = w | z_k = z, v_k = v).$$

G.2.2 Classification

Being able to classify the environment state using the available information is equivalent to solve an observation problem for the stochastic automaton.

Given an input and output sequence and an initialised automaton $\mathcal{S}$, the solution to the observation problem is obtained by determining the conditional probability distribution [118],

$$P(z_k|k) = P(z_k|V(0...k), W(0...k)). \tag{G.2}$$

The solution of the observation problem is given by the set of all the states z_k to which the automaton may move with non zero probability while accepting the input sequence and generating the output sequence specified. The a-posterior state probability distribution can be evaluated on-line by iterative application of a predict and correct schema discussed in [118, 15].

G.2.3 Quantisation of the signal spaces

Through quantisation the real-valued signals can be fed to the automaton. Let $x(t)$, $\mathbb{R} \rightarrow \mathbb{R}$ be continuous. A quantiser splits the signal space $\mathbb{R}$ into a finite number of disjoint sets $\mathcal{Q}_x(\xi)$ where $\xi \in \mathcal{N}_x \subset \mathbb{Q}$. With $\mathcal{Q}_x(\xi)$ denoting the set of values in x associated with the quantised value ξ, the quantiser function reads, in terms of intervals,

$$\mathcal{Q}_x(\xi) = (x_\xi^{low},\ x_\xi^{up}],\ \xi \in \mathcal{N}_x = \{1, \ldots, N_x\} \tag{G.3}$$

where x_ξ^{low} and x_ξ^{up} are the lower and upper bound of the quantisation respectively.

The index ξ of the partition $\mathcal{Q}_x(\xi)$ to which the current value of x belongs, represents the qualitative level of the signal. [118] points out that such levels can be chosen arbitrarily since no grounded methods are given. In this work, a more concrete procedure driven by examples will be exploited to define discretisation levels.

Let $x \subset \mathbb{R}$ be a generic continuous signal sampled in the time with uniform frequency. N samples are drawn independently from the acquired signal to define the subset $\gamma = \{x_i, i = 1...N\}$. Each sample is associated to the corresponding class z to define the training data $(x_i, \theta_i), i = 1, \ldots, N$.

Since all x_i are independent and identically-distributed samples of a random variable, the probability density function $\hat{f}(\rho)$ can be estimated by kernel density

estimation, [101].

$$\hat{f}(\rho) \approx \frac{1}{Nh} \sum_{i=1}^{N} K(\frac{\rho - x_i}{h}), \tag{G.4}$$

where K is a kernel function, N is the number of samples and h is a smoothing parameter. Given a Gaussian kernel in the form of Eq. G.5, the value of h can be chosen to maximise the reconstruction performance ([21]),

$$K(\varphi) = \frac{1}{2\pi} e^{-\frac{1}{2}\varphi^2}. \tag{G.5}$$

Given the approximated probability density function the cumulative density function $\hat{F}(x)$ is,

$$\hat{F}(x) = \int_{-inf}^{x} \hat{f}(\rho) d\rho \approx \sum_{\rho=-inf}^{x} \hat{f}(\rho). \tag{G.6}$$

Defining

$$\gamma_z = \{(x_j, \theta_j);\ \forall j : \theta_j = z\} \tag{G.7}$$

as the subset of samples related to the state z, it is possible to use the above results to estimate $\hat{F}(x|\theta = z)$, for each $z \in \mathcal{N}_z$. Fig. G.2 shows the results of the procedure when applied to a synthetic dataset.

To simplify the notation, define

$$\hat{F}(x)_z = \hat{F}((x_j, \theta_j); \forall j : \theta_j = z),\ z \in \mathcal{N}_z.$$

Reverting to classification, the quantiser defined in Eq. G.3 can be interpreted as a linear machine, which splits the continuous time signal x into segments. A simple classifier would select the state by looking at the discrete level in which the signal falls. The probability that the robot is in state z while observing a discrete output equal to ξ is,

$$P(z|\mathcal{Q}_x(\xi)) = \frac{P(\mathcal{Q}_x(\xi)|z)P(z)}{\sum_{\zeta \in \mathcal{N}_z} P(\mathcal{Q}_x(\xi)|\zeta)P(\zeta)}. \tag{G.8}$$

Rewriting Eq. G.8 using Eq. G.3,

$$\begin{aligned} &P(\theta = z | x_\xi^{low} < x \leq x_\xi^{up}) \\ &= \frac{P(x_\xi^{low} < x \leq x_\xi^{up} | \theta = z) P(\theta = z)}{\sum_{\zeta \in \mathcal{N}_z} P(x_\xi^{low} < x \leq x_\xi^{up} | \theta = \zeta) P(\theta = \zeta)}. \end{aligned}$$

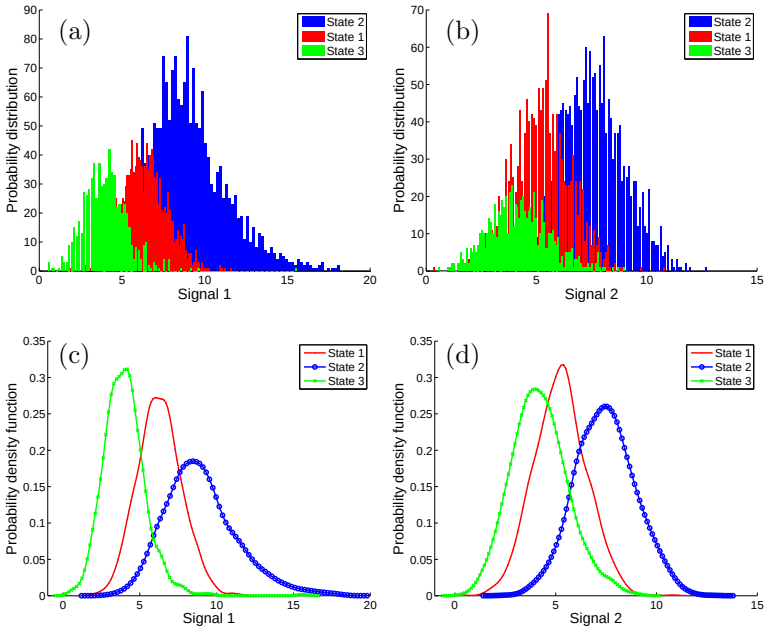


Figure G.2: Synthetic dataset composed of two signals generated by three different models. (a)(b) Two continuous independent signals are uniformly sampled and grouped in states according to a hand-labeled classification. (c)(d) State conditioned probability distribution estimated via kernel density estimation

Further, from Eq. G.8, the conditional probability is a function of the conditional cumulative density function,

$$P(\mathcal{Q}_x(\xi)|\theta = z) = P(x_\xi^{low} < x \leq x_\xi^{up}|\theta = z) = \hat{F}(x_\xi^{up})_z - \hat{F}(x_\xi^{low})_z.$$

Hence,

$$\begin{aligned} P(z|\mathcal{Q}_x(\xi)) &= \frac{P(\mathcal{Q}_x(\xi)|z)P(z)}{\sum_{\zeta \in \mathcal{N}_z} P(\mathcal{Q}_x(\xi)|\zeta)P(\zeta)} \\ &= \frac{\left(\hat{F}(x_\xi^{up})_z - \hat{F}(x_\xi^{low})_z\right) P(z)}{\sum_{\zeta \in \mathcal{N}_z} \left(\hat{F}(x_\xi^{up})_\zeta - \hat{F}(x_\xi^{low})_\zeta\right) P(\zeta)}. \end{aligned} \tag{G.9}$$

Eq. G.9 describes the probability that the system is in the semantic state z while the continuous signal x is contained in the quantised level ξ. This information

is used to define the quantisation levels by maximising the probability that the robot is in a state z while $x \in \mathcal{Q}_x(\xi)$ and minimising the number of levels ξ, $(|\mathcal{N}_x|)$.

$$\min_{|\mathcal{N}_x|} \max_{\mathcal{Q}_x(\xi)} P(z|\mathcal{Q}_x(\xi)) \ , \xi \in \mathcal{N}_x, z \in \mathcal{N}_z. \tag{G.10}$$

Non informative intervals, with low discriminative performance are merged with the confining interval.

An approximation is used for the problem in Eq. G.10. Since the variability space of continuous variables could not be known exactly, the quantisation levels are defined in the probability space. Since the space is delimited to the set $[0, 1] \subset \mathbb{R}$ it is possible to cover the whole space:

$$\{p_j\}, p_j \in [0, 1] \subset \mathbb{R};\ j = 1, \ldots, N_p$$

For each p_j a value in the signal space can be found from the state conditional cumulative density function. An algorithm to find initial estimates of the quantisation levels is,

$$\forall\, p_j \text{ find } x_j \text{ such that } \hat{F}(x_j)_z = p_j,\ z \in \mathcal{N}_z$$

$$\mathcal{X} = \{x_i : \hat{F}(x_i)_z = p_j,\ \forall z \in \mathcal{N}_z,\ j = 1...N_p,\ i = 1...N_x\}$$

Fig. G.2.3 visually explains the procedure for one signal in Fig. G.2.

Supposing the model composed by N states, $\mathcal{X}$ contains $N_x = N \cdot N_p$ points.

By sorting $\mathcal{X}$, the initial quantisation intervals are defined as:

$$\begin{aligned}
\mathcal{Q}_x(1) &= (-\infty,\ x_1] \\
\mathcal{Q}_x(2) &= (x_1,\ x_2] \\
&\vdots \\
\mathcal{Q}_x(\xi) &= (x_i,\ x_{i+1}],\ \xi \in \mathcal{N}_x \\
&\vdots \\
\mathcal{Q}_x(N_x) &= (x_{N_x},\ \infty)
\end{aligned} \tag{G.11}$$

Each quantisation level $\xi \in \mathcal{N}_x$ is associated to the state z for which the following condition holds,

$$\max_z P(\theta_\xi = z|\mathcal{Q}_x(\xi)). \tag{G.12}$$

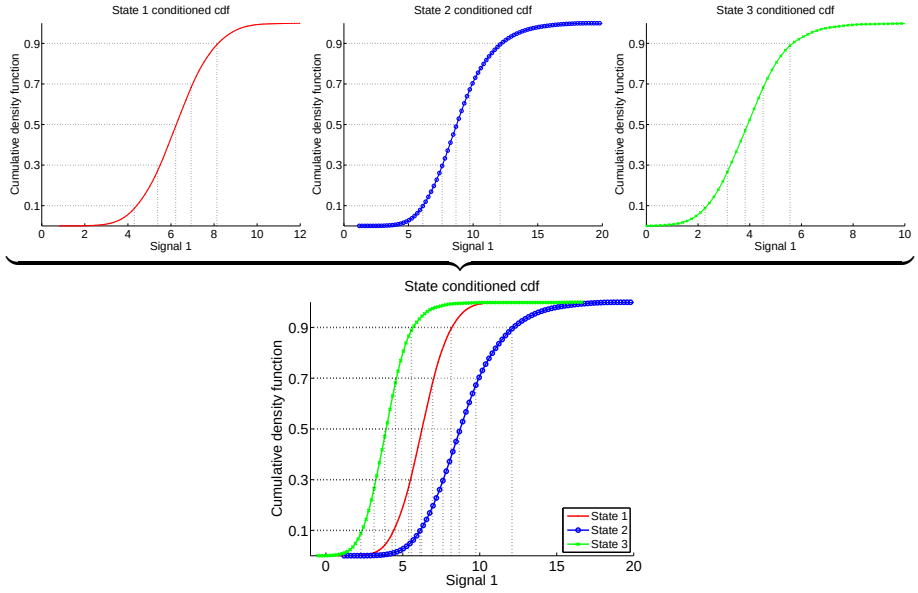


Figure G.3: The probability space is uniformly sampled between 0.1 and 0.9 with step 0.1. For each level, the corresponding signal value is mapped on the state conditional cumulative distribution function.

Two quantisers, $\mathcal{Q}_x(\xi)$ and $\mathcal{Q}_x(\xi+1)$ are merged if:

$$P(\theta_{\xi+1} = \theta_\xi | \mathcal{Q}_x(\xi+1)) \geq P(\theta_{\xi+1} = \zeta | \mathcal{Q}_x(\xi+1)), \zeta \neq \theta_\xi, \forall \zeta \in \mathcal{N}_x, \quad \text{(G.13)}$$

The conditional probability of the resulting merged quantisation interval is re-evaluated before being compared to the succeeding interval $\mathcal{Q}_x(\xi+2)$. By this procedure the number of levels decreases drastically while fulfilling Eq. G.12, as shown in Fig. G.4 for the example dataset.

G.2.4 Model abstraction

The estimation of the behavioural relation for each transition can be done using the abstraction algorithm reported in [15]. From a hand-classified dataset the transition probability is approximated by frequency count. For large sample sizes not all state transition for all the possible input/output couples are found, yielding to $L(z', w|z, f, v) = 0$ even if the transition $z \rightarrow z'$ is feasible for the real system.

To overcome these limitations, the same proportion of training points were se-

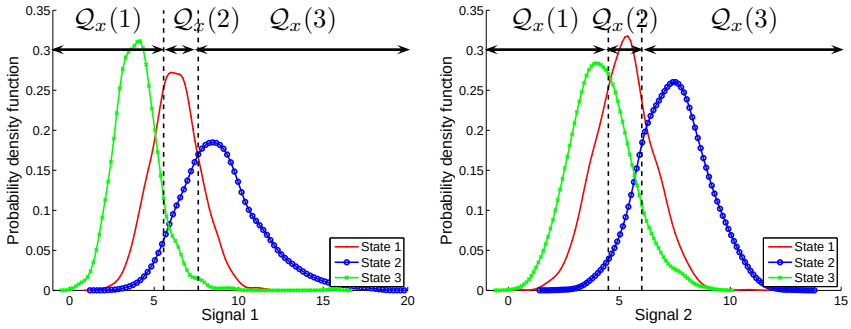


Figure G.4: Estimated discretisation levels resulting by the application of the proposed procedure on the synthetic data. The initial quantisation levels estimated shown in Fig. G.2.3 is reduced by the application of Eq. G.13.

lected for each state while a bias in the state transition matrix is added for transitions that might not be represented in the sample sequence but physically feasible.

G.3 Case study

The data used in this section has been recorded during autonomous operations of a tractor in an experimental orchard owned by Copenhagen University visible in Fig. G.5 in summer time. Typical operations performed by the tractor includes spraying and mowing and are executed according to a task plan.

The tractor is a standard orchard tractor that has been retrofitted with additional sensors and computing power [39]. For environment classification a stereo camera and a laser scanner are used as sensor input.

G.3.1 States and Perception

A state is an environment type that the tractor should be able to recognise. For the orchard it has been chosen to model four different types of environments.

Figure G.5: A typical tour covers the track shown in the image as GPS route on Google Earth background. 1. Open field (Road) 2. Headland 3. Dense trees 4. Sparse trees

To each state is associated the set of allowed actions which the tractor is allowed to execute. Referring to Fig. G.5 and G.6 the states are:

1. *Open field* Few obstacles in view and freely traversable space in the field of view, i.e. a road or a low vegetation field.
2. *Headland* Defines the start/end of the orchard area. It is usually delimited by fences, markers or open space.
3. *Dense trees* The distance between trees limits sensibly the field of view and the manoeuvring possibilities.
4. *Sparse trees* Sparse vegetation, manoeuvres constrained to the layout of the plants.

The perception system is designed to provide a compact set of signals that can discriminate between the semantic environments.

The signals from the perception system have been setup to detect: (a) The amount of visible ground. (b) Space occupied by obstacles. (c) Linear features such as fences. (d) Estimation of free space around the tractor. (e) Vegetation permeability.

It is important to note that visible ground plane and the space occupied by obstacles are not mutually exclusive. In the 3D data it is possible to observe

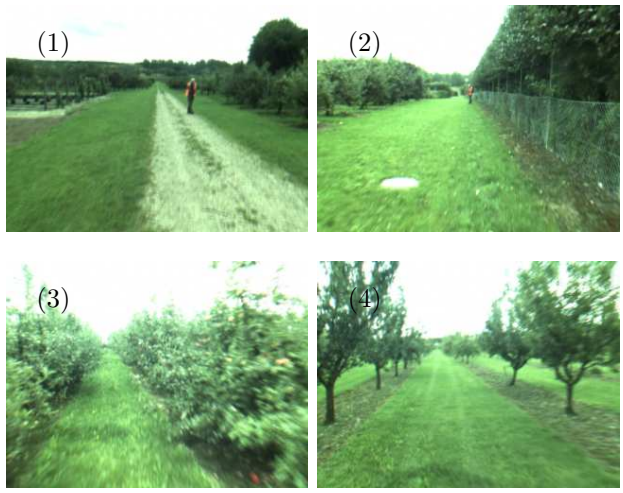


Figure G.6: Pictures from the left stereo-camera for the orchard semantic places in summer time. These images show the resolution available from the sensor. 1. Open field 2. Headland 3. Dense trees 4. Sparse trees

large amounts of ground while observing obstacles. For example sparse trees do not prevent observing the ground plane even if the crown of the trees form large obstacles in view.

G.3.1.1 Signal y_{gp}: Visible Ground plane

The approach for extracting ground plane is similar to [66]. Given a 3D point cloud a RANSAC technique [32] is used to construct ground plane hypotheses. This is done by: (a) choosing three non-collinear points at random from the point cloud; (b) constructing a plane estimate from the three points; (c) ranking the plane estimates based on number of inliers.

A 2D grid map is then constructed. The set of grid cells that get assigned to the ground plane can then be summed to give the amount of ground plane visible.

Let the found ground plane be written in the Hessian normal form of a plane with unit normal vector $\hat{\mathbf{n}}$ and distance p along the line. Let $\mathcal{P}$ be the set of points in the point cloud considered. If a point falls within a grid cell c_{xy}, defined by a quantisation interval $\mathcal{Q}_{xy}$, Eq. G.3 and its distance to the plane is

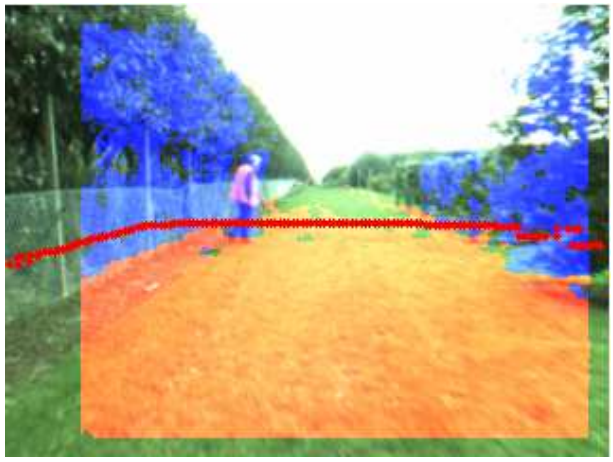


Figure G.7: A graphical representation of the perception output. Blue transparency shows obstacles. Red transparency shows ground plane. The laser measurements are overlaid as red dots

less than $D_{\max}$ then the cell belongs to the ground. For $\mathbf{x} \in \mathcal{P}$,

$$c_{x,y} = \left|\left\{\mathbf{x} \,\middle|\, \mathbf{x}_x \in \mathcal{Q}_{xy} \wedge \hat{\mathbf{n}} \cdot \mathbf{x} + p < D_{\max} \right\}\right| > 0$$
$$y_{gp} = \sum_{x=n_0}^{n_1} \sum_{y=m_0}^{m_1} c_{x,y}. \tag{G.14}$$

G.3.1.2 Signal y_{fs}: Laser free space

The free space observed by the laser is taken as the area spanned by the measurements. Given two adjacent range measurements s_i, s_{i+1} $i = 1, \ldots$ a triangle is formed and its area can be easily evaluated using the Heron's formula,

$$d_i = ||l_i - l_{i+1}||_2,$$
$$s_i = \frac{l_i + l_{i+1} + d_i}{2},$$
$$A_i = \sqrt{s_i(s - l_i)(s - l_{i+1})(s - d_i)}.$$

Having a single laser scan, by summing the area of each triangle defined by adjacent readings, an estimate of the free space is,

$$y_{fs} = \sum_{i=0}^{n_l - 1} A_i. \tag{G.15}$$

G.3.1.3 Signal y_o: Obstacles

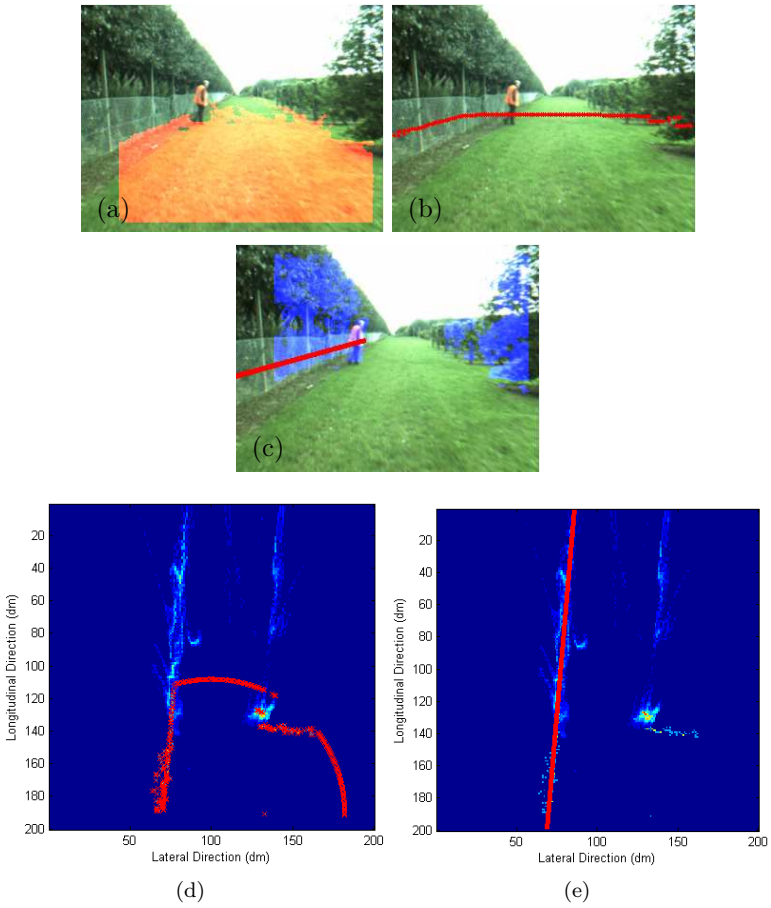


Figure G.8: Images illustrating the information used in detecting obstacles from stereo and laser. (a) Red overlay of ground plane detection. (b) Red shows laser range measurements. (c) Blue overlay of detected obstacles. Red shows position of detected wall segment. (d) Top-down view from 3D stereo points and laser that was used for obstacle detection. (e) Top-down view from 3D stereo points and laser. Red line shows detected wall segment

An obstacle signal y_o is constructed by creating a 3D grid map. Each point from the stereo and laser are projected into a grid map. A grid cell is then labeled as occupied if a stereo point or laser point lands in.

A $3D$ grid cell c_{xyz} is given the value of 1 if a point falls inside the cell and its

height above the ground plane is larger than $D_{\max}$. The number of occupied grid cells is then counted and used as a measure of the amount of obstacles seen from the pose,

$$c_{xyz} = \left| \left\{ \mathbf{x} \,\middle|\, \mathbf{x} \in \mathcal{Q}_{xyz} \wedge \hat{\mathbf{n}} \cdot \mathbf{x} + p \geq D_{\max} \right\} \right| > 0$$
$$y_o = \sum_{x=n_0}^{n_1} \sum_{y=m_0}^{m_1} \sum_{z=k_0}^{k_1} c_{xyz}. \tag{G.16}$$

G.3.1.4 Signal y_{ls}: Linear structures

This signal detects the presence of linear structures in the environment such as walls, fences, or hedges. The 3D grid map of obstacles is collapsed into a 2D grid map, g_{xy}, on the ground plane by summing the number of occupied cells along the vertical component, see Fig. G.8. A RANSAC line-fitting algorithm is then run on the 2D grid map to extract the strongest line.

$$g_{x,y} = \sum_{z=k_0}^{k_1} c_{x,y,z}.$$

A function $l(a, b)$ returns the grid cells that intersect with the line parametrised by a and b,

$$l\left(a, b\right) = \left\{ j \,|\, j \in g_{x,y} \,, j_y = a j_x + b \right\}.$$

The RANSAC algorithm finally attempts to maximise the sum of grid cells that intersect with the line,

$$y_{ls} = \max_{a,b} \sum_{i=1}^{|w|} l\left(a, b\right). \tag{G.17}$$

G.3.1.5 Signal y_{vp}: Vegetation permeability

Considering a region of interest, the local spatial laser range distribution can be captured by the principal components of the spatial covariance matrix [70]. By means of singular value decomposition the covariance matrix can be decomposed into principal components ordered by decreasing eigenvalues. In case of no dominant direction it is likely to be tree foliage, a wall or structured obstacles otherwise. A maximum likelihood strategy for point wise classification is used to distinguish between hypothesis that a point belongs to a tree and the null hypothesis [22]. The vegetation is modeled according to a Gaussian Mixture Model fitted by expectation maximisation.

G.3.2 Model design

The stochastic automata is composed by $N = 4$ states,

$$\mathcal{N}_z = \{\text{Open field, Headland, Dense trees, Sparse trees}\} = \{1, 2, 3, 4\}.$$

The signals used by the robot to construct the semantic map are divided in two sets with respect to the automaton: input $\mathbf{u} \in \mathbb{R}^m$ and output $\mathbf{y} \in \mathbb{R}^r$.

G.3.2.1 Discrete-valued input

Inputs are used to model signals that influence the behaviour of the automaton. In the case study, robot motion was the only input considered $\mathbf{u} = [u_m]$, hence $\mathcal{N}_v = \{\text{moving, stand still}\} = \{1, 2\}$. In order to not allow state transitions when the tractor is not moving.

G.3.2.2 Discrete-valued output

The perception system consists of a vector

$$\mathbf{y} = [y_{gp}, y_{fs}, y_{ls}, y_o, y_{vp}]$$

of continuous signals. The procedure introduced in section G.2.3 is here used to design quantisators for the experimental data. The γ set of Eq. G.7 is populated by randomly picking samples from a training set. Fig. G.9 shows the estimated conditional probability functions for the experimental data sampled as training set.

Through application of Eq. G.4 the state conditioned probability distribution function can be estimated. Fig. G.9 shows the computed thresholds (vertical dotted lines) obtained by applying the quantisation technique proposed to the estimated distribution functions.

Perceptual aliasing is recognisable in the probability space as an overlap of the distribution curves. The problem is handled by creating a unique quantisation level for the interested region. The automaton, through the observer has the duty to overcome to the problem by combining all the signals with the state transition model.

Each quantisation level is associated to an integer code $\xi \in \mathbb{Q}$.

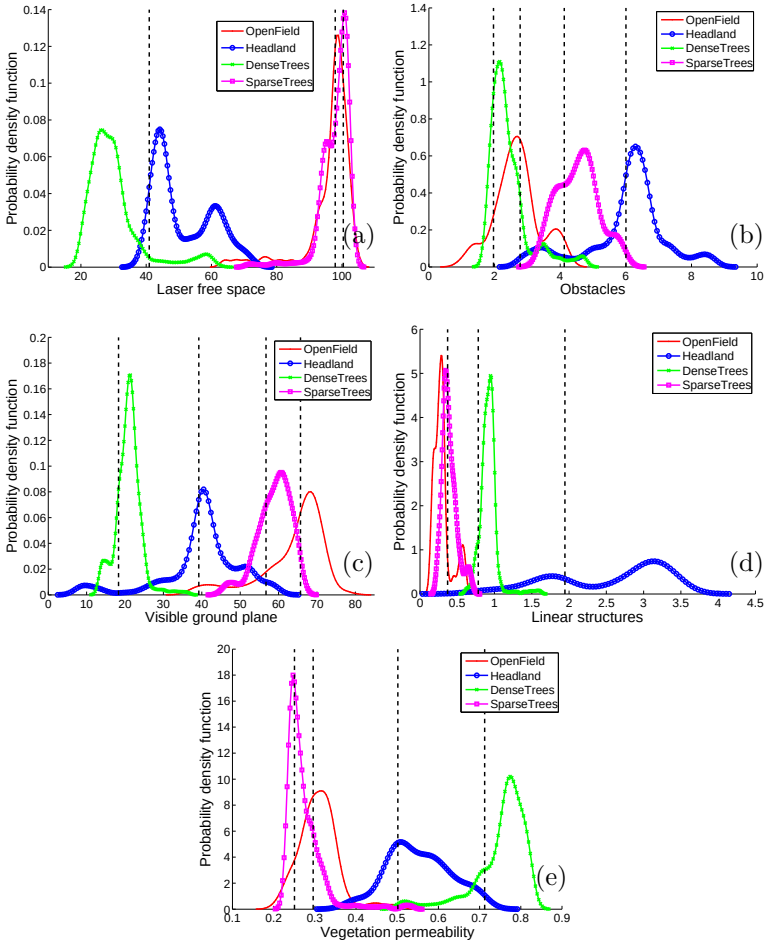


Figure G.9: Refined quantisation levels over the state conditional probability density function. The quantisation levels are individuated as vertical dotted lines whereas the conditional PDFs are coded among the states as colours (red = open field, blue = headland, green = dense trees, magenta = sparse trees). (a) Laser free space (b) Obstacles (c) Visible ground space (d) Linear structures (e) Vegetation permeability

G.3.3 Results

The validation dataset was recorded during a run which covered the track shown in picture G.5. The path first passed through an apple orchard (dense trees), then followed the back fence (headland) to a pear orchard (sparse trees), the path returned along the wine, crossed over behind some nut-trees and took the back route home to the garage.

The run was made in summer time to catch one extreme of the natural scenario. Full grown foliage, bushes and grown tree branches hanging increase the variability of each semantic state. To stress more the robustness of the methods, humans were moving or standing in the perception field of view during the trip.

The automata Matlab implementation was compared to open source library implementations of SVM, Adaboost and GHMM.

Four standard SVM kernel types were used during the experiments: linear, polynomial (degree 3), radial basis function (RBF), and sigmoid. Kernel parameters were fine tuned by an iterative procedure, which finds the best set using the training data. The package LIBSVM [23] was used for SVM learning and classification.

The upper bound for maximum number of Adaboost weak classifiers was set to 10. The implementation used was based on Matlab using tree stumps as weak classifiers.

The Bayes Net Toolbox for Matlab was been used to produce the results presented regarding the GHMM.

To produce and compare the results among different methods a K-fold cross validation procedure was performed. The data collected was split by random sampling in K disjoint sets. $K-1$ sets were then merged and used as the training base for the classifiers while the remaining data were used as a testbed. In this case, the data collected was composed by 2281 synchronised laser and vision observations. Due to the lack of a large amount of data, K was chosen equal to 2. In this way the generalisation capabilities together with the robustness could be stressed by using half of the available dataset for training and half for test. The features described in section G.3.1 were extracted from the raw data.

The performances were evaluated by collecting the classification results from 5 independent runs of the 2-fold validation. In figure G.10 the aggregated classification performances are shown.

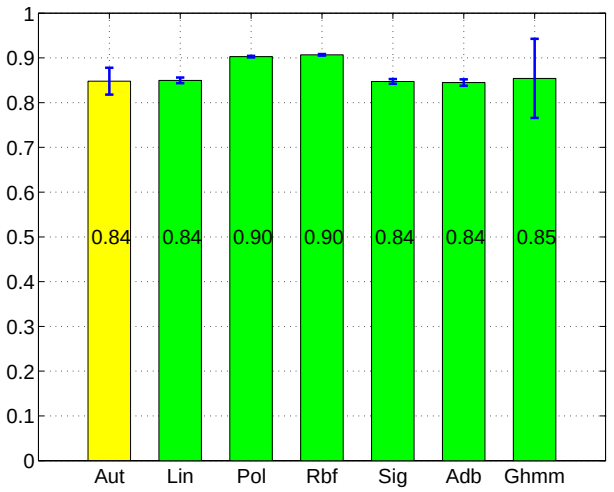


Figure G.10: Correct classification rate for the different approaches considered. The performances were evaluated by collecting the results from 5 runs of a 2-fold cross validation process (Both training and validation were done using two disjoint sets of 1141 samples of 5 features). The standard deviation of the results has been represented as a error line on top of each bar.

In the automaton all the state transitions were allowed to make the comparison fair with the other methods. In addition to the classification rates the confusion matrices of each method has been evaluated and shown in figure G.3.3.

Polynomial and RBF kernel based SVMs show the best classification rate. The GHMM shows the worse performance and is due to singularity problems which can be observed from the classification rate variance. All the methods have problems distinguishing state 1 and 4. Sparse orchards, are characterised by spaced trees, letting either stereovision and laser perceive only few obstacles. Missing trees in sparse orchard are then labeled as open areas. The automata shows similar performance to the other state-of-art methods even though it employs quantised signals. This shows that the information loss in quantising the signals is minimal. The design of the raw signals has been made in order to have as much as possible independent signals. This is further demonstrated by the fact that the automata can achieve the same performances as the other methods. The advantage of working with quantised signals is that it simplifies the classification problem by reducing the amount of data that goes in.

In Fig. G.3.3 the timings for both training and classification are shown for each method. The SVM clearly requires the longest training time and shows the

worst scalability. This is due to the fact that the signals are not bounded which the used implementation has problems in handling.

For the classification times the GHMMs are the slowest. The SVMs again seems to scale the worst. The linear SVM is faster than the automata but the confusion matrix shows the worst performance in terms of discrimination.

Adaboost performs remarkably well for both training and classification timings. However the output of Adaboost does not give information about the confidence of the estimate. This makes the method unsuitable for supervision and diagnosis tasks where low confidence estimates should not trigger false alarms.

The automata trains faster than SVM and classifies faster than the GHMMs. It shows good scalability properties and outputs a confidence estimate unlike Adaboost. Automata based classifiers are easy to implement compared to the other methods, which makes them suitable for robotic hardware.

G.4 Conclusion

The paper presented a novel approach to classifying outdoor environments by means of a stochastic automaton. A main advantage of the automata compared to other classification methods was a straightforward inclusion of how the dynamical system evolves over time, in the case study, the tractor motion in the orchard and the spatial connection of environments. Spurious observations were effectively dealt with in the updating method for state belief but had a penalty in the form of of lower adherence to the ground truth during transitions. This behaviour could be fine-tuned according to the needs of a particular use of the algorithm.

A case study with an autonomous vehicle in an orchard was used to assess the properties of the automata-based diagnosis with optimised signal quantisation. A comparison with state-of-the-art classifiers were made on the same data. Results showed that the automata trains faster than SVM and classifies faster than the GHMMs. The automata approach shows good scalability properties and outputs a confidence estimate, an essential feature to avoid false alerts from low-confidence hypothesis results. Automata based classifiers were shown to be easy to implement compared to the other methods, and were found suitable for implementation in robotic environments with hard real-time requirements.

The method used to optimise quantisation is general and could well apply it to other classification problems.

G.5 Acknowledgements

The support from the Danish Ministry of Food Agriculture and Fisheries, under contract 3412-06-01729 is gratefully acknowledged. Our colleagues Dr. J.C. Andersen from the Technical University of Denmark, Dr. H-W.Griepentrog and Mr. J. Resting-Jeppesen, both from Copenhagen University, Department of Life Sciences, are gratefully acknowledged for access to equipment and data. Hako Werke is acknowledged for providing the tractor used for orchard experiments.

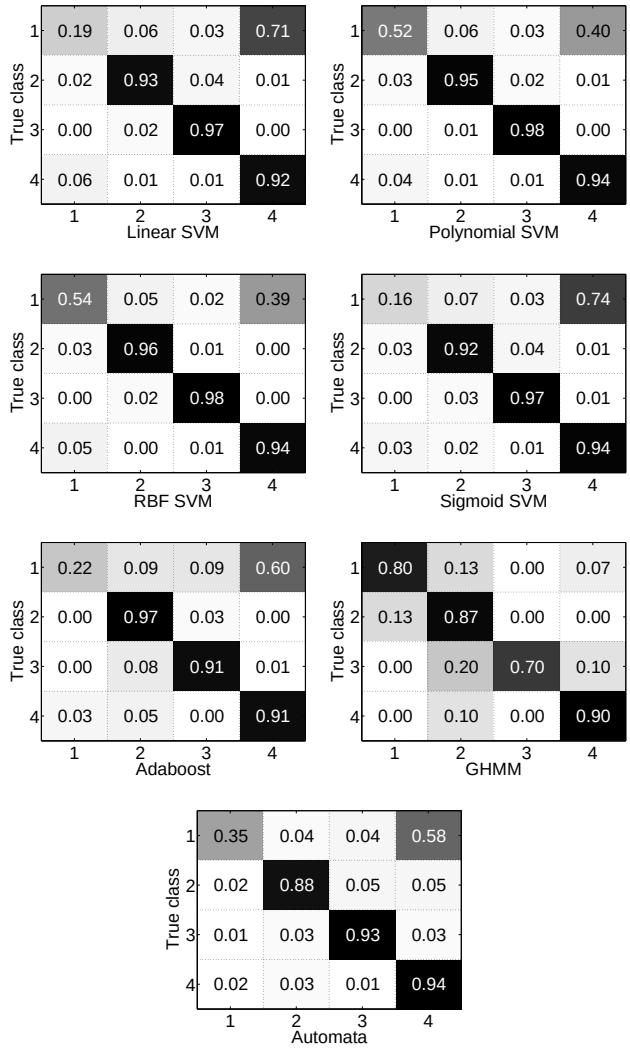


Figure G.11: Mean confusion matrices collected evaluating the performances from 5 trials of a 2-fold cross validation validation. The darker is the square the higher is the related classification probability. All the methods have problems to distinguish open field and sparse trees due to the strong perception aliasing which characterise the states.

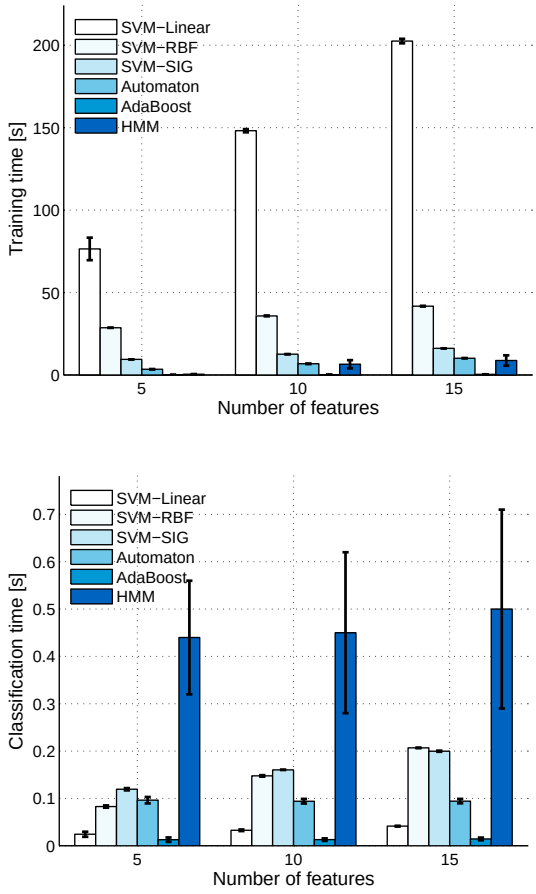


Figure G.12: Comparison of training and classification time for evaluated methods. The test dataset is composed of 1141 samples with 5 features per sample. For 10 and 15 features the existing 5 features have been replicated 2 and 3 times in order to see how the algorithms scale.

Bibliography

[1] Motilal Agrawal and Kurt Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *ICPR*, August 2006.

[2] Motilal Agrawal and Kurt Konolige. Rough terrain visual odometry. In *Proc. International Conference on Advanced Robotics (ICAR)*, August 2007.

[3] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5305 LNCS(PART 4):102–115, 2008.

[4] H. J. Andersen, T. Bak, and M. Christensen. Fusion of gps and visual motion estimates for robust outdoor open field localization. *Second International Conference on Computer Vision Theory and Applications, VISAPP. INSTICC.*, pages 413–418, 2007.

[5] A. Angelova, L. Matthies, D. Helmick, and Pietro Perona. Slip prediction using visual information. *RSS*, 2006.

[6] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona. Learning to predict slip for ground robots. In *ICRA*, pages 3324–3331, 2006.

[7] Anelia Angelova, Larry Matthies, Daniel Helmick, and Pietro Perona. Learning and prediction of slip from visual information. *J. of Field Robotics*, 24(3):205–231, 2007.

[8] Tijmen Bakker, Hendrik Wouters, Kees van Asselt, Jan Bontsema, Lie Tang, Joachim Müller, and Gerrit van Straten. Original paper: A vision based row detection system for sugar beet. *Comput. Electron. Agric.*, 60(1):87–95, 2008.

[9] M. Basseville and I. Nikiforov. Detection of abrupt changes: Theory and applications. *Prentice-Hall*, 1993.

[10] R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[11] M.G. Bello. A combined markov random field and wave-packet transform-based approach for image segmentation. *IEEE Trans. Image Process*, 3 6:834–846, 1994.

[12] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for DEMO III. In *Proc. of the IEEE Intelligent Vehicles Symp.*, 2000.

[13] M. Blanke. Fault-tolerant sensor fusion for marine navigation. *Proc. 7th IFAC Conf. on Manoeuvring and Control of Marine Craft*, Elsevier IFAC, sep 2006.

[14] M. Blanke, R. Izadi-Zamanabadi, S. A. Bøgh, and C. P. Lunau. Fault-tolerant control systems - a holistic view. *Control Engineering Practice*, 5(5):693–702, 1997.

[15] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control 2nd Edition*. Springer, 2006.

[16] M. Blanke and T. Lorentzen. Satool - a software tool for structural analysis of complex automation systems. *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 673–678, 2006.

[17] M. R. Blas, M. Agrawal, K. Konolige, and A. Sundaresan. Fast color/texture segmentation for outdoor robots. In *IROS*, 2008.

[18] M. R. Blas and M. Blanke. Natural environment modeling and fault-diagnosis for automated agricultural vehicle. In *Proceedings 17th IFAC World Congress, Seoul, Korea*, pages 1590–1595, 2008.

[19] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[20] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12 1:55–73, 1990.

[21] A. W. Bowman and A. Azzalini. *Applied Smoothing Techniques for Data Analysis*. Oxford University Press, 1997.

[22] F. Caponetti and M. Blanke. Combining stochastic automata and classification techniques for supervision and safe orchard navigation. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, 2009.

[23] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] T. Coen, A. Vanrenterghem, W. Saeys, and J. De Baerdemaeker. Autopilot for a combine harvester. *Comput. Electron. Agric.*, 63(1):57–64, 2008.

[25] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[26] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R. Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: Science and Systems*, 2006.

[27] D. Demirdjian and T. Darrell. Motion estimation from disparity images. In *Proc. of the Intl. Conf. on Computer Vision*, volume 1, pages 213–218, 2001.

[28] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[29] Chris Engels, Henrik Stewénius, and David Nister. Bundle adjustment rules. *Photogrammetric Computer Vision*, September 2006.

[30] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, Inc., New York, NY, USA, 2008.

[31] D. Fernandez and A. Price. Visual detection and tracking of poorly structured dirt roads. In *12th International Conference on Advanced Robotics, 2005*, pages 553–560, 18-20 July 2005.

[32] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. ACM.*, 24:381–395, 1981.

[33] T. I. Fossen. *Marine Control Systems*. Marine Cybernetics, 2002.

[34] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

[35] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Computer and System Sciences*, 55(1):119–139, 1997.

[36] A. A. Goshtasby. *2-D and 3-D Image Registration: for Medical, Remote Sensing, and Industrial Applications*. Wiley, 2005.

[37] M. Grabner, H. Grabner, and H. Bischof. Fast approximated SIFT. In *Proc. ACCV*, volume 1, pages 918–927, 2006.

[38] P. E. Greenwood and M. S. Nikulin. *A Guide to Chi-Squared Testing*. Wiley, New York, NY, 1996.

[39] H. W. Griepentrog, N. A. Andersen, J.C. Andersen, M. Blanke, O. Heinemann, T.E. Madsen, S.M. Pedersen, O. Ravn, and D. Wulfsohn. Safe and reliable - further development of a field robot. In *Proc. 7th European Conference on Precision Agriculture (ECPA)*, Wageningen, July 2009. Academic Publishers.

[40] J. Guivant, E. Nebot, and S. Baiker. High accuracy navigation using laser range sensors in outdoor applications. In *ICRA*, pages 3817–3822, 2000.

[41] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, California, November 1999.

[42] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *J. Field Robot.*, 26(2):120–144, 2009.

[43] M Happold, M Ollis, and N Johnson. Enhancing supervised terrain classification with predictive unsupervised learning. In *Robotics: Science and Systems Conference*, 2006.

[44] R.M. Haralick, K. Shanmugan, and I. Dinstein. Textural features for image classification. *IEEE Trans. Syst., Man. Cybern.*, SMC-3 6:610–621, 1973.

[45] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[46] Tinne Tuytelaars Herbert Bay and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, May 2006.

[47] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, September 2008.

[48] Thomas Howard, Colin Green, and Alonzo Kelly. State space sampling of feasible motions for high performance mobile robot navigation in highly constrained environments. In *Proc. of the Intl. Conf. on Field and Service Robotics*, July 2007.

[49] K. Iagnemma, F. Genot, and S. Dubowsky. Rapid physics-based rough-terrain rover planning with sensor and control uncertainty. In *ICRA*, 1999.

[50] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[51] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, 24 12:1167–1186, 1991.

[52] Jian Jin and Lie Tang. Corn plant sensing using real-time stereo vision. *J. Field Robot.*, 26(6-7):591–608, 2009.

[53] A. E. Johnson, S. B. Goldberg, Yang Cheng, and L. H. Matthies. Robust and efficient stereo feature tracking for visual odometry. In *ICRA*, pages 39–46, May 2008.

[54] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transac. on Communication Technology*, 15(1):52–60, 1967.

[55] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[56] Steven M. Kay. *Fundamentals of Statistical Signal Processing, Volume 2*. Prentice Hall, 1998.

[57] Alonzo Kelly. A feedforward control approach to the local navigation problem for autonomous vehicles. Technical Report CMU-RI-TR-94-17, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1994.

[58] C. Kervrann and F. Heitz. A markov random field model-based approach to unsupervised texture segmentation using local and global spatial statistics. *IEEE Trans. Image Process*, 4 6:856–862, 1995.

[59] Uwe Kiencke and Lars Nielsen. *Automotive Control Systems: For Engine, Driveline and Vehicle*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[60] R. Kimmel and J. A. Sethian. Computing Geodesic Paths on Manifolds. *Proc. of the National Academy of Science*, 95:8431–8435, July 1998.

[61] M. Kise and Q. Zhang. Creating a panoramic field image using multi-spectral stereovision system. *Comput. Electron. Agric.*, 60(1):67–75, 2008.

[62] M. Kise, Q. Zhang, and F. Rovira-Más. A stereovision-based crop row detection method for tractor-automated guidance. *Biosystems Engineering*, pages 357–367, 2005.

[63] K. Konolige. A gradient method for realtime robot control. In *IROS*, 2000.

[64] K. Konolige and M. Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[65] Kurt Konolige. Small vision systems: hardware and implementation. In *Eighth International Symposium on Robotics Research*, pages 111–116, 1997.

[66] Kurt Konolige, Motilal Agrawal, Morten Rufus Blas, Robert C. Bolles, Brian Gerkey, Joan Solà, and Aravind Sundaresan. Mapping, navigation, and learning for off-road traversal. *J. Field Robot.*, 26(1):88–113, 2009.

[67] Kurt Konolige, Motilal Agrawal, and Joan Solà. Large scale visual odometry for rough terrain. In *Proc. International Symposium on Robotics Research*, page To appear, November 2007.

[68] Kurt Konolige and David Beymer. SVS Reference Manual. Technical report, SRI International, 2007. `http://www.videredesign.com/docs/smallv4.4d.pdf`; accessed July, 2007.

[69] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951.

[70] J.F. Lalonde, N. Vandapel, D.F. Huber, and M. Hebert. Natural terrain classification using three dimensional ladar data for ground robot mobility. *Journal of field robotics*, 23(10):839–861, October 2006.

[71] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, Massachusetts, 1991.

[72] Steve LaValle. *Planning Algorithms*. Cambridge University Press, New York, New York, 2006.

[73] J. J. Leonard and P. Newman. Consistent, convergent, and constant-time slam. In *IJCAI*, 2003.

[74] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43 (1), 2001.

[75] S. Liapis, E. Sifakis, and G. Tziritas. Color and/or texture segmentation using deterministic relaxation and fast marching algorithms. *Journal of Visual Communication and Image Representation*, 15:1–26, 2004.

[76] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *IEEE Conference on Image Processing (ICIP)*, 2002.

[77] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2), 1998.

[78] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[79] J. Lunze. Diagnosis of quantised systems. *Fault Detection, Supervision and Safety for Technical Processes 2000*, 1(1):29–40, June 2001.

[80] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems (2nd ed.), 2004.

[81] T. Mäenpää and M. Pietikäinen. Texture analysis with local binary patterns. In C.H. Chen and P.S.P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision, 3rd Edition*, pages 197–216. World Scientific, 2005.

[82] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of visual odometry on the mars exploration rovers. *J. of Field Robotics*, 24(3):169–186, March 2007.

[83] B.S. Manjunath and R. Chellappa. Unsupervised texture segmentation using markov random field models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13 5:478–482, 1991.

[84] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26, 2004.

[85] J. Martin-Herrero, M. Ferreiro-Arman, and J.L. Alba-Castro. Grading textured surfaces with automated soft clustering in a supervised som. *Proceedings International Conference on Image Analysis and Recognition (ICIAR)*, pages 323–330, 2004.

[86] Larry Matthies, Mark Maimone, Andrew Johnson, Yang Cheng, Reg Willson, Carlos Villalpando, Steve Goldberg, Andres Huertas, Andrew Stein, and Anelia Angelova. Computer vision on mars. *Intl. J. of Computer Vision*, 75(1):67–92, 2007.

[87] K. Matusita. A distance and related statistics in multivariate analysis. In P. R. Krishnaiah, editor, *Multivariate Analysis*, pages 187–200. Academic Press, 1966.

[88] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision (ICCV)*, 2001.

[89] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision (ECCV)*, 2002.

[90] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, pages 43–72, 2005.

[91] M. Montemerlo and S. Thrun. Large-scale robotic 3-d mapping of urban structures. In *ISER*, 2004.

[92] H. Moravec and A. Elfes. High resolution maps for wide angles sonar. In *ICRA*, 1985.

[93] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *CVPR*, volume 1, pages 363 – 370, June 2006.

[94] O.M. Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. *Proceedings of the 2005 IEEE*, International Conference on Robotics and Automation, April 2005.

[95] Oscar MartÃnez Mozos, Patric Jensfelt, Hendrik Zender, Geert Jan Kruijff, and Wolfram Burgard. From labels to semantics: An integrated system for conceptual spatial representations of indoor environments for mobile robots. In *Workshop "Semantic information in robotics" at the IEEE International Conference on Robotics and Automation*, April 2007.

[96] David Nister. An efficient solution to the five-point relative pose problem. *IEEE PAMI*, 26(6):756–770, June 2004.

[97] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *J. of Field Robotics*, 23(1):3–20, January 2006.

[98] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d slam—3d mapping outdoor environments: Research articles. *J. Field Robot.*, 24(8-9):699–722, 2007.

[99] Andreas Nuchter, Oliver Wulf, Kai Lingemann, Joachim Hertzberg, Bernado Wagner, and Hartmut Surmann. 3d mapping with semantic knowledge. *RoboCup International Symposium*, pages 335–346, 2005.

[100] Christine M. Onyango and John A. Marchant. Physics-based colour image segmentation for scenes containing vegetation and soil. *Image Vision Comput.*, 19(8):523–538, 2001.

[101] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 33:1065–1076, 1962.

[102] S. C. Pei and J. H. Horng. Design of FIR bilevel Laplacian-of-Gaussian filter. *Signal Processing*, 82:677–691, 2002.

[103] R. Philippsen and R. Siegwart. An interpolated dynamic navigation function. In *ICRA*, 2005.

[104] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *Int. J. Comput. Vision*, 78(2-3):143–167, 2008.

[105] Ingmar Posner, Mark Cummins, and Paul Newman. A generative framework for fast urban labeling using spatial and temporal context. *Auton. Robots*, 26(2-3):153–170, 2009.

[106] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[107] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, February 1989.

[108] A. Rankin, A. Huertas, and L. Matthies. Evaluation of stereo vision obstacle detection algorithms for off-road autonomous navigation. In *AUVSI Symp. on Unmanned Systems*, 2005.

[109] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1508–1515, Washington, DC, USA, 2005. IEEE Computer Society.

[110] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, 2006.

[111] F. Rovira-Más and S. Han. Kalman filter for sensor fusion of gps and machine vision. Technical report, 2006 ASABE Meeting Presentation - Paper Number 063034, 2006.

[112] F. Rovira-Más, S. Han, J. Wei, and J. F. Reid. Autonomous guidance of a corn harvester using stereo vision. *Agricultural Engineering International: the CIGR Ejournal.*, 9, 2007.

[113] Francisco Rovira-Más, Shufeng Han, Jiantao Wei, and John F. Reid. Fuzzy logic model for sensor fusion of machine vision and gps in autonomous navigation. *2005 ASAE Annual Meeting*, 051156, 2005.

[114] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 59, Washington, DC, USA, 1998. IEEE Computer Society.

[115] R. B. Rusu, Z. C. Marton, N. B., M. Dolha, and M. Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems Journal*, 2008.

[116] Radu Bogdan Rusu, Aravind Sundaresan, Benoit Morisset, Motilal Agrawal, and Michael Beetz. Leaving flatland: Realtime 3d stereo semantic reconstruction. In *ICIRA '08: Proceedings of the First International Conference on Intelligent Robotics and Applications*, pages 921–932, Berlin, Heidelberg, 2008. Springer-Verlag.

[117] T. Sato, M. Kanbara, N. Yokoya, and H. Takemura. Dense 3-d reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held video camera. In *SMBV '01: Proc. IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV'01)*, page 57, Washington, DC, USA, 2001. IEEE Computer Society.

[118] Jochen Schroder. *Modeling, state observation and diagnosis of quantised systems*. Number 282 in Lecture notes in control and information science. Springer, 2003.

[119] Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotic Research*, 21:735–758, August 2002.

[120] J. Shi and C. Tomasi. Good features to track. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 1994.

[121] Dong Hun Shin and Sanjiv Singh. Path generation for robot vehicles using composite clothoid segments. Technical Report CMU-RI-TR-90-31, Robotics Institute, Pittsburgh, PA, December 1990.

[122] N. Soquet, D. Aubert, and N. Hautiere. Road segmentation supervised by an extended v-disparity algorithm for autonomous navigation. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 160–165, 13-15 June 2007.

[123] B. Southall and C. J. Taylor. Stochastic road shape estimation. *Computer Vision, IEEE International Conference on*, 1:205, 2001.

[124] D. J. Spero and R. A. Jarvis. 3D vision for large-scale outdoor environments. In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, 2002.

[125] M. Staroswiecki and G. Comet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37(5):687–699, 2001.

[126] M. Staroswiecki and P. Declerck. Analytical redundancy in nonlinear interconnected systems by means of structural analysis. In *Proc. IFAC AIPAC'89 Symposium.*, volume 2, pages 23–27. Elsevier - IFAC, 1989.

[127] A. Stentz. Optimal and efficient path planning for partially-known environments. In *ICRA*, volume 4, pages 3310–3317, 1994.

[128] N. Sunderhauf, K. Konolige, S. Lacroix, and P. Protzel. Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In *Tagungsband Autonome Mobile Systeme*. Springer Verlag, 2005.

[129] Niko Sunderhauf and Peter Protzel. Towards using sparse bundle adjustment for robust stereo odometry in outdoor terrain. *Towards Autonomous Robotic Systems TAROS06*, pages 206–213, 2006.

[130] L. Tang, L. F. Tian, B. L. Stward, and J. F. Reid. Texture-based weed classification using gabor wavelets and neural network for real-time selective herbicide application. In *ASAE Paper No. 99-3036*, 1999.

[131] Alberto Tellaeche, Xavier P. BurgosArtizzu, Gonzalo Pajares, Angela Ribeiro, and Cesar Fernandez-Quintanilla. A new vision-based approach to differential spraying in precision agriculture. *Computers and Electronics in Agriculture*, 60(2):144–155, 2008.

[132] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, September 2006.

[133] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[134] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 5555.

[135] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[136] S. Tzafestas and K. Watanabe. Modern approaches to system/sensor fault detection and diagnosis. *Journal A.*, 31(4):42–57, 1990.

[137] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(4), April 1991.

[138] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Trans. Image Process.*, 4 11:1549–1560, 1995.

[139] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:691–696, 2003.

[140] P. Viola and M. Jones. Robust real-time face detection. In *ICCV01*, 2001.

[141] M. D. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *ICCV '98: Proc. Sixth Int. Conf. on Computer Vision*, page 917, Washington, DC, USA, 1998.

[142] Denis F. Wolf and Gaurav S. Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258, April 2008.

[143] Yun Zhai, Zeeshan Rasheed, and Mubarak Shah. A framework for semantic classification of scenes using finite state machines. *Lecture notes in computer science*, 3115:279–288, 2004.

[144] Jinyou Zhang and H.-H. Nagel. Texture-based segmentation of road images. In *Proceedings of the Intelligent Vehicles '94 Symposium*, pages 260–265, October 1994.

www.elektro.dtu.dk
Department of Electrical Engineering
Automation and Control
Technical University of Denmark
Ørsteds Plads
Building 348
DK-2800 Kgs. Lyngby
Denmark
Tel: (+45) 45 25 38 00
Fax: (+45) 45 93 16 34
Email: info@elektro.dtu.dk

ISBN 978-87-92465-22-1